Contributions for the automatic description of multimodal scenes

Luís Filipe Pinto de Almeida Teixeira

Universidade do Porto

Faculdade de Engenharia

FEUP

November 2009

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores

Contributions for the automatic description of multimodal scenes

Luís Filipe Pinto de Almeida Teixeira

Dissertação submetida para a obtenção do grau de Doutor
em
Engenharia Electrotécnica e de Computadores

Dissertação realizada sob a supervisão do
Professor Doutor Luís António Pereira de Meneses Corte-Real,
Professor Associado do
Departamento de Engenharia Electrotécnica e de Computadores,
Faculdade de Engenharia, Universidade do Porto
e do
Professor Doutor Aníbal João de Sousa Ferreira
Professor Auxiliar do
Departamento de Engenharia Electrotécnica e de Computadores,
Faculdade de Engenharia, Universidade do Porto

Novembro de 2009

Email: lfpt@fe.up.pt

URL: http://www.fe.up.pt/~lfpt

This thesis was partially supported by the Portuguese Science and Technology Foundation (FCT), INESC Porto, and the European Commission under the IST research network of excellence VISNET II of the 6th Framework Programme.

It was developed at INESC Porto, Faculdade de Engenharia da Universidade do Porto, and during an internship at University of Victoria.

Doctoral Dissertation Committee

Artur Pimenta Alves

Full Professor at Faculdade de Engenharia da Universidade do Porto

Jorge Manuel Moreira de Campos Pereira Batista

Assistant Professor at Universidade de Coimbra

José Santos-Victor

Associate Professor at Instituto Superior Técnico

Ana Maria Rodrigues de Sousa Faria de Mendonça

Associate Professor at Faculdade de Engenharia da Universidade do Porto

Luís António Pereira de Meneses Corte-Real

Associate Professor at Faculdade de Engenharia da Universidade do Porto

Aníbal João de Sousa Ferreira

Assistant Professor at Faculdade de Engenharia da Universidade do Porto

to my parents Manuela and Armínio,
and to Ana

Abstract

With the emergence of an information-oriented society it soon became clear that the massive amount of information that was generated required effective ways for indexing and searching. From as early as the 50s in the 20th century, researchers have sought ways to implement *information retrieval systems*. These systems, and in particular text retrieval systems, have evolved considerably and became a part of our daily life. How we have now virtually the whole internet text content searchable and accessible in less than half a second is paradigmatic of this. The next natural step was indexing also multimedia content besides text content. However, multimedia content introduces additional problems to the indexing task. The large amount of information and the complexity of its relations are factors that dramatically increase the difficulty in achieving highly successful indexing and searching results. For instance, until recently, devising a system that could automatically detect and identify persons in a complex scene, track them across multiple cameras and analyse their behaviour in real-time would be too much of an arduous task. Though such a system is not yet fully accomplished, many recent successful advances, mostly in computer vision and machine learning, take us much nearer to that technological milestone.

In this dissertation we approach the issue of indexing content obtained from real-world scenes. We define "real-world scene" as any scene captured continuously in public or private spaces by automated and often passive sensors. These scenes are usually captured by multiple sensors of multiple types. The actions portrayed in the captured sequences consist of everyday actions, like people walking or running, cars passing by or parking, etc. An example of application is a surveillance system. Most of the information in surveillance scenarios is conveyed by a sequence of images but, more often than not, there is important information that can be obtained from analysing other types of data, or modalities – *multimodal scene analysis* relies on that premiss. We start by analysing the concepts and challenges that are part of multimodal analysis, having in mind real-world scenes. Three processing areas are considered: object detection, object recognition, and event analysis. With object detection we separate both

in space and in time each object and associate a label to them. This label distinguishes objects from one another but does not associate any semantic knowledge. That is the goal of object recognition with which we associate an identity to the object from a set of known classes. With event analysis on the other hand we identify relevant activities and events that are defined by the context of the scene under analysis. For each area we survey relevant algorithms and systems, and present original contributions.

The original contributions we propose in this dissertation follow a line of work that have an emphasis on visual-based algorithms. In particular, we propose a *visual segmentation algorithm* for the detection of generic objects. It uses a cascade of change detection tests, including noise-induced changes, illumination variation and structural changes. For the detection of structural changes the algorithm is based on other commonly used per-pixel modelling algorithms. Additionally we introduce an *object matching system* applicable to scenes captured with multiple views. Objects are first tracked by a common single-view tracker and then the individual results are given as inputs to the matching system. The system can possibly associate a known identity to those tracks. Since typical scenes may present changes to the objects' appearance and new object may appear, the matching system is scalable in two distinct dimensions: an undetermined number of new objects can be added at any given instant and, existing object representations can be updated to reflect changes in time. This object matching system is later extended to an *event navigation system* based on timelines representing person appearances. From an application point of view, the complete system can be seen as a summarisation system. Combining all contributions we obtain a system that takes as inputs raw sequences, segments them into relevant objects (in this case, persons), tracks the detected objects and processes the tracks. Multiple appearances of persons are identified whether these are captured by the same camera, or by different cameras in independent locations. This way it is possible to know the path followed by a given person.

Finally, we also propose a *multimodal framework* that serves as a foundation for the development of algorithms in the scope of this dissertation. Visual and audio analysis algorithms are usually devised and tested using generic mathematical tools. However, if the task involves integrating different types of data, this process can easily become burdensome and inefficient. The framework follows a dataflow architecture where complex networks of processing modules can be assembled and handle multiple and different types of multimedia flows in a meaningful and efficient way.

Keywords – multimodal analysis, content indexing, object detection and tracking, event analysis, ambient intelligence, surveillance systems

Resumo

A afirmação da sociedade de informação tornou clara que a enorme quantidade de dados gerados requeria formas eficientes de indexação e de pesquisa. Desde tão cedo quanto os anos 50 do século XX, a comunidade científica procurou desenvolver formas de implementação de *sistemas de pesquisa* de informação. Estes sistemas, e em particular os sistemas de pesquisa de texto, evoluíram consideravelmente e tornaram-se parte do nosso dia-a-dia. A forma como actualmente temos virtualmente todo o conteúdo textual pesquisável e acessível em menos de meio segundo é exemplificativo disto. O passo seguinte foi indexar também conteúdo multimédia, para além de texto. No entanto, o conteúdo multimédia origina problemas adicionais à tarefa de indexação. A grande quantidade de informação e complexidade das suas relações são factores que dificultam consideravelmente a indexação e pesquisa. Por exemplo, até recentemente, desenvolver um sistema capaz de automaticamente detectar e identificar pessoas numa cena complexa, acompanhá-las através de multiplas câmaras e analisar o seu comportamento em tempo real seria uma tarefa demasiado complexa. Apesar desse sistema não ser ainda uma realidade, muitos avanços recentes, essencialmente em visão computacional e aprendizagem máquina, aproxima-nos desse marco tecnológico.

Nesta dissertação abordamos a indexação de conteúdos obtidos a partir de cenas do mundo real. Definimos "cenas do mundo real" como sendo aquelas que são continuamente capturadas em público ou espaços privados por sensores automatizados e frequentemente passivos. Estas cenas são tipicamente capturadas por múltiplos sensores de múltiplos tipos. As cenas retratadas nas sequências capturadas consistem em acções do quotidiano, como pessoas a andar ou correr, carros a passar ou estacionar, etc. Um exemplo de aplicação será um sistema de vigilância. A maior parte da informação em cenários deste tipo é veiculada por sequências de imagens mas frequentemente existe informação importante que pode ser obtida a partir da análise de outros tipos de dados – a *análise multimodal* baseia-se nesta premissa. Começamos por analisar os conceitos e desafios que são parte da análise multimodal, tendo em mente as cenas do mundo

real. São consideradas três áreas de processamento: detecção de objectos, reconhecimento de objectos e análise de eventos. A detecção de objectos consiste na separação destes tanto no espaço como no tempo. A cada objecto é atribuída uma etiqueta que o diferencia dos outros objectos mas não lhe associa uma semântica. Este é o objectivo do reconhecimento de objectos com o qual associamos um identidade ao objecto a partir de um conjunto conhecido de classes de identidades. Por outro lado, através da análise de eventos identificamos actividades e eventos relevantes, definidos de acordo com o contexto da cena. Para cada área analisamos os algoritmos e sistemas relevantes e apresentamos contribuições originais.

Foi definida uma abordagem com ênfase em algoritmos baseados em informação visual para o desenvolvimento de contribuições originais. Em particular, propomos um *algoritmo de segmentação visual* para a detecção de objectos genéricos. O algoritmo baseia-se na detecção em cascata de alterações, incluindo as provocadas por ruído, por variação de iluminação e alterações estruturais. A detecção de alterações estruturais é feita com base em métodos comummente usados para a modelização *pixel-a-pixel*. Introduzimos ainda um *sistema de matching de objectos* aplicável a cenas capturadas a partir de múltiplas vistas. Os objectos são seguidos por um comum sistema de *tracking*. Os resultados individuais são posteriormente passados ao sistema de *matching* que associa uma identidade aos objectos. Uma vez que em cenas típicas novos objectos podem aparecer e outros objectos podem sofrer alterações na aparência dos objectos, o sistema de *matching* é "escalável" em duas dimensões distintas: um número indeterminado de objectos pode ser adicionado em qualquer instante e a representação dos objectos conhecidos podem ser actualizadas para reflectir as respectivas alterações ao longo do tempo. Este sistema de *matching* de objectos é também usado num *sistema de navegação de eventos* baseado em *timelines* que representam a presença de pessoas. Do ponto de vista da aplicação, o sistema completo pode ser visto com um sistema de sumarização. O resultado final de todas as contribuições é um sistema que recebe sequências não tratadas, segmenta-as em objectos relevantes – neste caso pessoas – segue os objectos detectados e analisa-os. As múltiplas presenç de pessoas são identificadas, mesmo que capturadas por diferentes câmaras em localizações independentes. Desta forma é possível saber qual o percurso seguido por uma dada pessoa.

Finalmente, também apresentamos uma *plataforma multimodal* que foi usada com uma base para o desenvolvimento de algoritmos no âmbito desta dissertação. Os algoritmos de análise visual ou de áudio são geralmente desenvolvidos e testados usando ferramentas matemáticas genéricas. No entanto, se a tarefa envolve a integração de diferentes tipos de dados, este processo pode tornar-se complexo e ineficiente. A plataforma baseia-se numa arquitectura onde redes complexas de processamento de objectos podem ser construídas, formando sistemas que podem manipular múltiplos e diferentes tipos de fluxos multimédia de uma forma representativa e eficiente.

Palavras-chave – análise multimodal, indexação de conteúdos, detecção e *tracking* de objectos, análise de eventos, inteligência ambiente, sistemas de vigilância

Résumé

L'émergence de la société de l'information a clairement indiqué que l'énorme quantité de données générées nécessite de moyens efficaces d'indexation et de recherche. Depuis les années 1950, la communauté scientifique a cherché à développer des *systèmes de recherche d'information*. Ces systèmes, en particulier les systèmes de recherche de texte, ont considérablement évolué et sont devenus une partie intégrante de notre quotidien. Le fait que la quasi-totalité des contenus textuels d'internet soient actuellement indexables et accessibles en moins d'une demi-seconde en est un exemple. L'étape postérieure au texte a été de s'intéresser à l'indexation de contenu multimédia. Cependant, le contenu multimédia a créé des problèmes supplémentaires à la tâche d' indexation. La grande quantité d'informations et la complexité de leurs relations sont des facteurs qui compliquent considérablement l'indexation et la recherche. Par exemple, jusqu'à récemment, mettre au point un système capable de détecter et d'identifier automatiquement des personnes dans une scène complexe, de les suivre grâce à de multiples caméras et d'examiner leur comportement en temps réel était une tâche trop complexe. Bien que ce système ne soit pas encore une réalité, beaucoup de progrès récents, principalement dans la vision par ordinateur et l'apprentissage automatique, nous rapproche de cette étape technologique.

Dans cette thèse nous nous intéressons à l'indexation de contenu obtenu à partir de scènes du monde réel. Nous définissons les "scènes du monde réel" comme prises dans des espaces publics ou privés, par des capteurs automatiques et souvent passifs. Ces scènes sont généralement captées par de multiples capteurs de types divers. Les scènes représentées dans les séquences capturées consistent en des actions quotidiennes, comme des personnes en train de marcher ou de courir, ou des voitures en train de passer ou de se garer, etc. Un exemple d'application est un système de surveillance. La plupart de l'information dans ces scénarios se traduit par une séquence d'images, mais des informations importantes peuvent être obtenu à partir de l'analyse d'autres types de données – *l'analyse multimodale* est basée sur ce prémisse. Nous commençons

par examiner les concepts et les défis de l'analyse multimodale, en gardant à l'esprit les scènes du monde réel. Trois domaines de traitement sont envisagés: la détection d'objets, la reconnaissance d'objets et l'analyse d'événements. La détection d'objets est la séparation d'objets dans l'espace et dans le temps. A chaque objet une étiquette unique est attribuée qui la différencie des autres objets, mais à laquelle aucune sémantique n'est associée. Tel est l'objectif de la reconnaissance d'objets par laquelle nous associons une identité à l'objet à partir d'un ensemble de classes d'identités. Grâce à l'analyse des événements, nous identifions des activités et des événements qui sont définis dans le contexte de la scène. Dans chacun de ces domaine, nous analysons les algorithmes et systèmes pertinents et présentons des contributions originales.

Nos contributions suivent une approche qui met l'accent sur les algorithmes basés sur l'information visuelle pour le développement des contributions originales. En particulier, nous proposons un *algorithme de segmentation visuelle* pour la détection d'objets génériques. L'algorithme est basé sur la détection de changements en cascade, y compris ceux causés par du bruit, par des changements d'éclairage et des changements structurels. La détection de changements structurels est basée sur les méthodes courantes de modélisation pixel par pixel. Nous introduisons également un *système de matching d'objets* applicable à des scènes capturées à partir de plusieurs points de vue. Les objets sont suivis par un système commun de *tracking*. Les résultats individuels sont ensuite transmis au système qui associe une identité aux objets. étant donné que, dans des scènes typiques, des objets peuvent changer d'apparence et de nouveaux objets peuvent apparaître, le système de *matching* est extensible en deux dimensions distinctes: un nombre illimité d'objets peut être ajouté à tout moment et la représentation des objets connus peuvent être mis à jour pour refléter des changements dans le temps. Ce système de *matching* des objets est également utilisé dans un *système de navigation* basé sur les *timelines* d'événements qui constituent la présence de personnes. D'un point de vue applicatif, le système complet peut être vu comme un système de résumé de scénes. Le résultat final de l'ensemble des contributions est un système qui prend en entrée les séquences, les sépare en objets (dans ce cas, les personnes), suit les objets détectés et traite les objects séparemment. La présence de personnes sont identifiées même lorsqu'ils sont capturés par différentes caméras dans des lieux indépendants. Il est donc possible de connaître le chemin suivi par une personne.

Finalement, nous présentons aussi une *plate-forme multimodale* qui a été utilisé comme une base pour le développement d'algorithmes dans cette thèse. Les algorithmes pour l'analyse visuelle ou audio sont généralement conçus et testés à l'aide d' outils mathématiques génériques. Toutefois, si la tâche implique l'intégration de différents types de données, ce processus peut devenir complexe et inefficace. La plate-forme est basée sur une architecture où des réseaux complexes de modules de traitement peuvent être assemblés pour former des systèmes qui peuvent gérer de multiples et différents types de flux multimédia dans une manière représentative et efficace.

Mots-clés - analyse multimodale, indexation de contenu, détection et *tracking* d'objets, analyse d'événements, intelligence ambiante, systèmes de surveillance

Preface

When I timidly entered INESC Porto in a distant September morning in 2001 I was completely oblivious of what the next seven years would reserve me. I had just finished my ECE course and was starting a new stage in my life. Amid good and "not-so-good" moments, many of what represented those years until now is intrinsically associated with these two hundred-some pages. This particular endeavour began in late 2004. I was finishing my M.Sc. and felt I should proceed further with my research work. Prof. Luís Corte-Real was kind enough to accept again the task of supervising me.

During those years INESC Porto was my second home. I can not honestly think of a more straightforward way of expressing how important both the institution and all the people I met there were; from the earlier times with the MOG group to our more recent efforts to consolidate a multimedia processing team. My thanks to them, expressed in Prof. José Ruela, our hard-working coordinator. This work would not be possible without them. Most of the work developed during that period would also not be possible without the kind support provided by FCT – Fundação para a Ciência e a Tecnologia (Foundation for the Science and the Technology). Many thanks to them for supporting and promoting scientific research in Portugal.

The patience and support with which Prof. Corte-Real accompanied the work throughout the last four years has allowed me to experiment, make mistakes, and learn with them. Thank you for believing in me and my work. A particular thank-you goes also to Prof. Anibal Ferreira, my co-supervisor.

Working with image processing was not something I pictured doing. But eventually that was exactly what I did when I started at INESC. Jaime Cardoso had the unfortunate task of supervising my work then – I was not particularly fond of what I was doing, I must admit. That changed over time. A lot. And, for the most part, Jaime was responsible for that. He helped me evolve and become a better researcher. For a long time *the team* was basically us from video and Gustavo Martins from audio. Gustavo has been a

great colleague and friend and it has been great working with him. We can find a lot of common interests and ideas and there is something to talk about, whether about Marsyas or something else – something to disturb the rest of the team with. The team that grew with Fabien Gouyon (thank you for correcting the abstract in french) and Pedro Carvalho, and that now shrunk with my departure. I wish all the best to them.

And talking about Marsyas, around the summer of 2006 I started playing with Marsyas. Gustavo had recently praised the new Marsyas architecture since he was starting to migrate his applications from the first version. And soon after, emerged the question – why not going some time to Victoria, and to UVic? George Tzanetakis, the creator and mentor of Marsyas, was kind enough to welcome us there. Those couple of months were truly a great experience [2]. My thanks to George for being such a great advisor and friend and to all the MISTIC/UVic team at the time – Mathieu, Jen, Adam, Randy, Manj, Ajay, Neil and Graham. A particular thanks goes to Mathieu Lagrange with whom we kept collaborating thereafter. While we were there, the ideas behind MarsyasX flowed in long discussions whether in the UVic Lab or The Beagle Pub.

Articles are part of our daily life. They can be in the same measure as frustrating while we write them and as fulfilling when we finally see them published. I would like to acknowledge all the anonymous reviewers that surely helped making this a better work. A particular thank-you to Kevin Smith – his thorough review of one of our submissions and pertinent comments in such a short notice were really helpful.

I would also like to thank Pedro Quelhas for his support and friendship from day one. Our discussions and his suggestions were fundamental to improve this thesis. Pedro was the first person I knew when I started the ECE course. We followed different paths after graduating but, in essence, our paths were always entwined. I hope we have the opportunity to join efforts in the future and really put forth *those* projects ideas.

A word also to the colleagues and friends with whom I've worked with in UTM. Especially Gustavo Carneiro, Rui Campos, Jaime Dias, and Tânia Calçada that undertook this same endeavour. And of course, Filipe Abrantes and Ricardo Duarte whom I joined in this new adventure we are now in, called FhP. A special thanks also to Filipe Sousa. We worked together in a difficult time for UTM and got through. Filipe's support and uplifting comments are a constant. Thanks for being such a great friend.

When this thesis was still far from being a reality I had the opportunity to experience teaching, in particular in the Programming class of the Informatics course. I miss the thrill of teaching and trying to find answers to the unpredictable questions popped by the students. Thanks to them and to my colleagues I have worked with at the time.

Finally, my deepest thanks to my loving parents, to Ana for her endless love, to my sister, to my cousins, and to all my true friends. Today's not different, I keep thinking of you. Maybe a bit more.

Luis F. Teixeira
Porto, January 11, 2009

Contents

List of figures **xiii**

List of tables **xvii**

Acronyms **xix**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 The surveillance scenario . . . 3
1.3 Thesis scope . . . 5
1.4 Main contributions . . . 6
1.4.1 Object segmentation . . . 7
1.4.2 Object matching . . . 7
1.4.3 Activity analysis . . . 7
1.4.4 Multimodal analysis framework . . . 7
1.4.5 Related publications . . . 7
1.5 Structure of the dissertation . . . 12

2 Multimodal scene analysis **15**
2.1 Concepts . . . 15
2.2 Information fusion . . . 18
2.2.1 Early fusion . . . 19
2.2.2 Late fusion . . . 20
2.2.3 Temporal fusion . . . 22
2.2.4 Adaptive fusion . . . 23
2.3 AV-based multimodal scene analysis . . . 23
2.3.1 Representing AV signals . . . 24
2.3.2 Measuring AV synchrony . . . 30

2.3.3 Applications of AV-based multimodal scene analysis 32
2.3.4 A multimodal application: speaker identification 33
2.4 Summary 38

3 Object detection **41**
3.1 Overview 41
3.2 Visual object segmentation 42
3.2.1 Background modelling and subtraction 44
3.2.2 Comparative evaluation of background modelling methods . . 46
3.3 Cascaded change detection 52
3.3.1 Identification of noise-induced changes 53
3.3.2 Identification of illumination variation-induced changes . . . 54
3.3.3 Identification of dynamic background behaviour 55
3.3.4 Building the system 56
3.3.5 Results 56
3.4 Audio object segmentation 64
3.5 Audio object localisation using AV features 68
3.6 Person detection 72
3.6.1 Face detection 74
3.6.2 Speaker segmentation 76
3.7 Summary 77

4 Object recognition **79**
4.1 Overview 79
4.2 Visual object tracking 81
4.2.1 Multi-camera object tracking 84
4.3 Scalable object recognition using local features 88
4.3.1 Describing visual objects 90
4.3.2 Building the vocabulary 91
4.3.3 Adaptively updating and learning object models 93
4.3.4 Classifying objects 93
4.3.5 Building the system 94
4.3.6 Validation 94
4.4 Person recognition 103
4.4.1 Face recognition 103
4.4.2 Gait recognition 107
4.4.3 Speaker recognition 108
4.4.4 Multimodal recognition 109
4.5 Summary 110

5 Event analysis **113**
5.1 Overview 113
5.2 Event detection and recognition 114
5.3 Event discovery 116

5.4 Activity analysis . . . 116
5.5 Detecting appearances of persons . . . 117
5.5.1 System overview . . . 117
5.5.2 Object segmentation . . . 119
5.5.3 Object tracking . . . 119
5.5.4 Object matching . . . 120
5.5.5 Identity verification . . . 122
5.5.6 Experimental setup and results . . . 124
5.5.7 Interaction with tracking . . . 131
5.6 Visual trajectory representation . . . 133
5.6.1 Application scenario . . . 134
5.6.2 Trajectory representation . . . 134
5.6.3 Trajectory classification . . . 135
5.6.4 Using latent aspect modelling . . . 136
5.7 Audio events detection . . . 140
5.8 Summary . . . 140

6 Multimodal analysis framework 143
6.1 Overview . . . 143
6.2 Marsyas architecture . . . 145
6.2.1 Dataflow programming . . . 145
6.2.2 Implicit patching . . . 146
6.2.3 Basic processing units and networks . . . 146
6.2.4 Dynamic access to modules and controls . . . 147
6.2.5 Dynamic linking of controls . . . 148
6.2.6 Interoperability . . . 149
6.3 MarsyasX – a step toward cross-modal analysis . . . 152
6.3.1 Payloads . . . 153
6.3.2 Data flows . . . 154
6.3.3 Timing and synchronisation . . . 154
6.3.4 Memory management . . . 155
6.3.5 Initial configuration and propagation of flow control updates . 156
6.3.6 Events . . . 157
6.3.7 Legacy interface with Marsyas 0.2 . . . 158
6.3.8 Modules . . . 160
6.3.9 Scripting with Python . . . 163
6.4 Example applications . . . 164
6.4.1 Simple motion detection . . . 164
6.4.2 Segmentation of audio streams . . . 165
6.4.3 Speaker segmentation using audio and video . . . 168
6.4.4 Simple surveillance with multiple inputs . . . 168
6.4.5 Person detection system . . . 171
6.5 Summary . . . 172

7 Conclusions **175**
7.1 Discussion . . . 175
7.2 Current work . . . 177
7.3 Future work . . . 177
7.4 Final thoughts . . . 179

A Datasets **183**
A.1 Segmentation dataset $\mathcal{D}_s$ . . . 183
A.2 4-class audio dataset $\mathcal{D}_4^a$ . . . 184
A.3 Object matching dataset $\mathcal{D}_{30}$ . . . 186
A.4 Independent views dataset $\mathcal{D}_5$ . . . 188
A.5 Event detection dataset $\mathcal{D}_e$ . . . 188

B Evaluation metrics **191**
B.1 Performance metrics . . . 191
B.1.1 Precision and recall . . . 191
B.1.2 False Acceptance Rate (FAR) and Miss Detection Rate (MDR) . 191
B.1.3 F-measure . . . 192
B.2 A metric to compare foreground segmentations . . . 192

C Local descriptors **195**
C.1 Overview . . . 195
C.2 Scale-Invariant Feature Transform (SIFT) . . . 196

Bibliography **199**

Resources **227**
Thesis . . . 227
Surveillance projects and applications . . . 227
Multimodal-based projects . . . 228
Generic libraries and frameworks . . . 228
Visual . . . 228
Audio . . . 229
Multimodal . . . 229
Learning . . . 229
Other . . . 230
Datasets and competitions . . . 230
Visual . . . 230
Audio . . . 231
Multimodal . . . 231

Bio **233**
Curriculum Vitae . . . 235

List of figures

1.1 Generic architecture of a real-world scene analysis system. 2
1.2 Visnet II surveillance scenario general architecture. 4
1.3 Scope of the thesis. 5
1.4 Line of work of the thesis. 6

2.1 The multimodal processing and analysis paradigm. 16
2.2 A multimodal processing network. 16
2.3 A multi-feature system design space. 17
2.4 Definition of visual regions where the possible speakers are located. . . . 35
2.5 Example of the result obtained with combined speaker boundaries. 37

3.1 Examples of possible segmentations for a given scene. 43
3.2 Examples of segmentations obtained using different well-known algorithms. 43
3.3 Evolution of d^c_{sym} using different methods for background modelling. . . 51
3.4 Effects of illumination variation and noise over a reference colour vector. 53
3.5 Proposed algorithm in pseudo-code. 55
3.6 Block representation of the proposed cascaded algorithm. 57
3.7 Evolution of d^c_{sym} for MoG and CMoG implementations. 58
3.8 Complete results with CMoG for the SH sequence at different frames. . . . 59
3.9 Evolution of d^c_{sym} for KDE and CKDE implementations. 60
3.10 Evolution of d^c_{sym} for each sequence of the test set (part 1). 61
3.11 Evolution of d^c_{sym} for each sequence of the test set (part 2). 62
3.12 Segmentation masks obtained using different methods. 64
3.13 Simple decomposition of an audio sequence in four classes. 65
3.14 Temporal segmentation of an audio sequence. 65
3.15 Algorithm for audio segmentation based on frame classification. 66
3.16 Discriminative capability of features. 67
3.17 Hierarchical segmentation and recognition of an audio sequence. 69

3.18 Problems with simple foreground segmentation. 72
3.19 Person detection based on the detection of persons' heads. 73
3.20 Person model used to define the foreground pixels belonging to a person. 73
3.21 Example results of a face detection algorithm. 74

4.1 Example of the output of a person tracker. 81
4.2 Typical scenario handled by the visual object matching system. 89
4.3 Example frames with superimposed points defined by the random point selector. 90
4.4 Block representation of the proposed method. 95
4.5 Confusion matrices of the object classification. 97
4.6 Incorrect segmentation example. 100
4.7 Face recognition task. 104
4.8 Elastic Bunch Graph Matching (EBGM). 107
4.9 Block representation of the multimodal person recognition system. 109

5.1 Problem characterisation and system output. 118
5.2 Block diagram of the system architecture. 119
5.3 Algorithm to verify the identity of objects present in a given track. 123
5.4 Confusion matrices of the object classification. 125
5.5 Examples of automatically generated timelines. 127
5.6 Examples of timelines generated with updated models for previously unknown objects. 128
5.7 Generated timelines for two different cameras capturing the same scene from different views. 129
5.8 Scenarios where typical tracking methods often fail to correctly track/identify an object. 132
5.9 Common tracking error using a state-of-the-art particle filtering tracker. 133
5.10 Tracks for `Person29` and `Person30` in the sequence ShopAssistant2. 133
5.11 Characteristic trajectories to classify/index. 135
5.12 Trajectory representation concepts. 136
5.13 Example of how the trajectory representation is created. 137
5.14 Highest ranking 8 trajectories for 6 aspects from the PLSA model. 139

6.1 Building blocks in Marsyas 0.2. 146
6.2 Processing chunks of data in Marsyas. 148
6.3 Payload architecture in MarsyasX. 153
6.4 Timing information associated to payloads. 155
6.5 Comparison of performance between the slice and payload architectures. 157
6.6 Evolution of the performance using the payload architecture. 158
6.7 Propagation of flow controls updates. 159
6.8 Legacy interface with Marsyas 0.2. 159
6.9 Network used for motion detection. 165
6.10 Network used for the music/speech segmentation scenario. 166

6.11 Network used for the speaker segmentation scenario. 167
6.12 Network used for the surveillance scenario. 169
6.13 Network used for the person detection system. 170

A.1 Segmentation dataset (original frames). 184
A.2 Segmentation dataset (masks). 185
A.3 Images of all 30 visual objects used in the dataset. 186
A.4 Same visual object captured at different instants by different cameras. . . 187
A.5 Different perspectives of the same person are present in the dataset. . . . 187
A.6 Subsets $\mathcal{D}^i, i \in \{1, 2, \ldots, 5\}$ created to test incremental learning of objects. 188
A.7 Two additional independent locations. 189
A.8 Image of the area in front of a store where the sequences were recorded. . 189
A.9 Aglomerated plot of all tracks in the database. 190

B.1 Intersection-graph for two segmentations. 193

C.1 Illustration of the SIFT feature extraction process. 197
C.2 SIFT feature extraction process. 198

List of tables

2.1 Performance results using different combinations of detectors. 37

3.1 Summary of the background modelling methods. 46
3.2 Average d^c_{sym} and frames per second for each method over the evaluated frames of each sequence. 52
3.3 Average d^c_{sym} over the test set sequences. 63
3.4 Audio features for frame classification. 68
3.5 Confusion matrix using k-NN and Gaussian classifiers for audio segmentation for two classes. 68
3.6 Summary of audio localisation algorithms. 71

4.1 Summary of visual object tracking algorithms. 85
4.2 Summary of visual object multi-camera tracking algorithms. 87
4.3 Average classification rate and standard deviations. 96
4.4 Average classification rate and standard deviations. 98
4.5 Classification rate of different tracks using $\mathcal{M}_{30}$. 99
4.6 Confusion matrix of the tracks segmented using a background subtraction algorithm. 100
4.7 Average classification rates and standard deviations. 101
4.8 Classification rate for the simplified model update. 101
4.9 Classification performance rate based on incremental learning. 102
4.10 Summary of face recognition methods. 106

5.1 Algorithm: AdaBoost. 122
5.2 Results for the sequences under evaluation. 130
5.3 Results of trajectory classification. 136
5.4 Results of trajectory classification using PLSA. 138

6.1 List of metadata fields contained by a payload. 153

Acronyms

AV Audio-Visual

API Application Programming Interface

BIC Bayesian Information Criterion

BOV Bag-Of-Visterms

CASA Computational Auditory Scene Analysis

CC Cepstral Coefficients

CCA Canonical Correlation Analysis

CKDE Cascade Kernel Density Estimation

CMoG Cascade Mixture of Gaussians

CVSA Computational Visual Scene Analysis

CV Computer Vision

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DVC Digital Video Coding

DWT Discrete Wavelet Transform

EBGM Elastic Bunch Graph Matching

FAR False Acceptance Rate

FC Frequency Centroid

GMM Gaussian Mixture Model

GPL General Public Licence

FLD Fisher Linear Discriminant

FOV Field of View

GUI Graphical User Interfaces

HBOV Hierarchical Bag-Of-Visterms

HCI Human-Computer Interaction

HMM Hidden Markov Model

ICA Independent Component Analysis

JPDAF Joint Probability Data Association Filtering

KDE Kernel Density Estimation

k-NN k-Nearest Neighbour

LDA Linear Discriminant Analysis

LFCC Log Frequency Cepstral Coefficient

LLR Log Likelihood Ratio

LoG Laplacian of Gaussians

LPC Linear Predictor Coefficient

LSP Linear Spectral Pairs

MCMC Markov Chain Monte Carlo

MCSHR Major Colour Spectrum Histogram Representation

MDR Miss Detection Rate

MFCC Mel Frequency Cepstrum Coefficients

MHT Multiple Hypothesis Tracking

MI Mutual Information

ML Maximum Likelihood

MoG Mixture of Gaussians

NN Neural Network

MRF Markov Random Field

OOP Object-Oriented Programming

PCA Principal Component Analysis

pdf probability density function

PLSA Probabilistic Latent Semantic Analysis

PSNR Peak Signal-to-Noise Ratio

ROI Region Of Interest

SIFT Scale-Invariant Feature Transform

SF Spectrum Flux

SMC Sequential Monte Carlo

SOM Self-Organizing Map

SVD Singular Value Decomposition

SVM Support Vector Machine

TDNN Time-Delay Neural Network

TOC Time Of Capture

TTS Time To Schedule

VQ Vector Quantization

ZCR Zero Cross Rate

1 Introduction

The huge amount of data generated when capturing real-world scenes demands efficient tools that could allow accessing, compiling and searching content. This data needs to be summarised in semantic events to allow efficient searches in applications such as surveillance for security, ambient intelligence, nature monitoring or health care monitoring. The goal is to identify relevant objects and events that characterise the captured scene. The semantics of content can be often found in multiple forms or modalities, which are usually complementary, such as visual and audio data. The methods for extracting additional information from visual and audio data have become generalised but their integration has been limited.

1.1 Motivation

Until recently, devising a system that could automatically detect and identify persons in a complex scene, track them across multiple cameras and analyse their behaviour in real-time would be too much of an arduous task. Though such a system is not yet fully accomplished, many recent successful advances, mostly in computer vision and machine learning, take us much nearer to that technological milestone.

We have witnessed to the maturation of algorithms and techniques for multimedia analysis, high-level representation of multimedia data and search and retrieval in multimedia databases. Allied with the growing evolution of processing capabilities this has provided promising results in real-time semantic analysis applications. Moreover, with the proliferation of affordable sensor devices it became possible to easily create networks of these devices in large scale to collect distributed information. In this type of networks, different types of sensors, with or without spacial and temporal overlapping, generate multimodal signals. These signals need to be summarised in semantic events to allow efficient searches in applications such as surveillance for security.

Most of the security surveillance systems based in video are supported by centralised architectures which collect all information in real time for storing, processing and interpretation, usually by a human operator. This process implies very often a high quantity of resources for storage because all information is recorded, even when no relevant events are taking place. More recent systems try to overcome this problem by reducing spacial and temporal resolutions to the minimum necessary. For example, by using motion detection to detect activity, we store only temporal segments instead of recording everything. However this is done without analysing what is happening and important events are still treated and stored the same way irrelevant events are.

The correct identification of events may sometimes be jeopardised when insufficient resolution is available. If the identification has real-time requirements, it is done "manually" by the system operator, having to span its attention by several surveillance points. The distributed processing of this information with the objective of automatic event detection could help the operator, generating alarms when some meaningful event is detected. Moreover the storage could be performed with different resolutions, based on the detected events. Relevant time segments could be stored with maximum spacial and temporal resolutions and the others with inferior resolutions. These could also be indexed based on information retrieved in real time. In an a posteriori analysis scenario, operators could easily find past events. The surveillance network scalability is therefore improved, due to better resource utilisation.

"Big brother is watching you" (Orwell, 1949) – without disregarding ethical concerns with invasive and abusive surveillance, it is fairly consensual to state that, besides security purposes, a scenario where a person's behaviour is automatically interpreted, can serve a useful and benevolent purpose; examples of such a scenario include nature monitoring or health care monitoring. Considering the broader concept of ambient intelligence, understanding a person behaviour is essential to react to it or adapt the environment appropriately. With the continuous ageing of the population, an effective way to aid the elderly is a growing need, and ambient intelligence can have an important role.

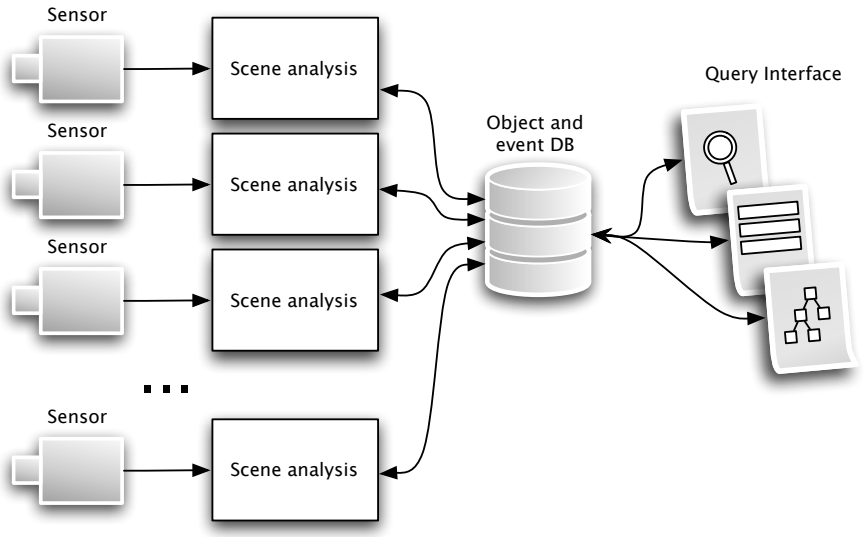


Figure 1.1: *Generic architecture of a real-world scene analysis system. Multiple sensors capture scene activity which is analysed and decomposed in detected objects and events. These are stored for posterior querying.*

In this dissertation we address the problem of automatically describing *real-world scenes*

captured by multiple sensors and multiple types of sensors. We define "real-world scene" as any scene captured continuously in public or private spaces by automated and often passive sensors. The actions portrayed in the captured sequences consist of everyday actions, like people walking or running, cars passing by or parking, etc. The typical scenario is of surveillance cameras in public spaces, but other scenarios are conceivable. The ultimate goal is allowing faster browsing and searching of the content. For critical areas a complementary goal could also be automatic triggering of alarms to counter or prevent imminent peril. A representation of a system with these goals is depicted in Figure 1.1.

1.2 The surveillance scenario

Part of the development of this thesis was done in parallel with VISNET II, a Network of Excellence (NoE) funded by the sixth Framework Programme of the European Commission. The focus of the NoE is on Networked Audiovisual Media Technologies. Our contributions related to this thesis are in the scope of the Work Package WP2.2 – Audiovisual Analysis – and in particular in Activity 2.2.3 – Video segmentation and tracking. In that activity, the scenario of visual surveillance using static cameras was considered as the target. Visual surveillance systems are widely applied in transport scenarios, such as airports, railways, underground, and motorways as well as other public spaces, such as banks and shopping malls.

Surveillance systems can be broadly classified into three generations, which are related to the evolution in video communications, processing and storage (Regazzoni *et al.*, 2001). First generation surveillance systems are basically extensions of the human visual perception capabilities in a spatial sense. Video cameras are used to capture visual signals from multiple remote locations and deliver them to a single physical location. Video signals are captured, transmitted and stored based on analogue techniques and the analysis is solely based on human operators. Second generation surveillance systems benefited from early advances in digital video communications and processing that provide assistance to the human operator by detecting important events. The goal of third generation surveillance systems is to provide digital solutions for the whole architecture, starting at the sensor level, up to the presentation level. Current research efforts focus on these third generation systems and aim to devise smart surveillance systems (Hampapur *et al.*, 2003; Javed *et al.*, 2007; Pavlidis *et al.*, 2001; Valera and Velastin, 2005).

The main goal of smart surveillance systems is to support the human operator by automatically analysing the large amount of data and providing additional data and functionality. Generally the analysis results can be used in two different ways (Hampapur *et al.*, 2003). For online use the automatic analysis can provide real time alerts or enhancement of the data to focus the attention of the operator to an important event. In case an incident occurs, the automatically generated index can provide an efficient

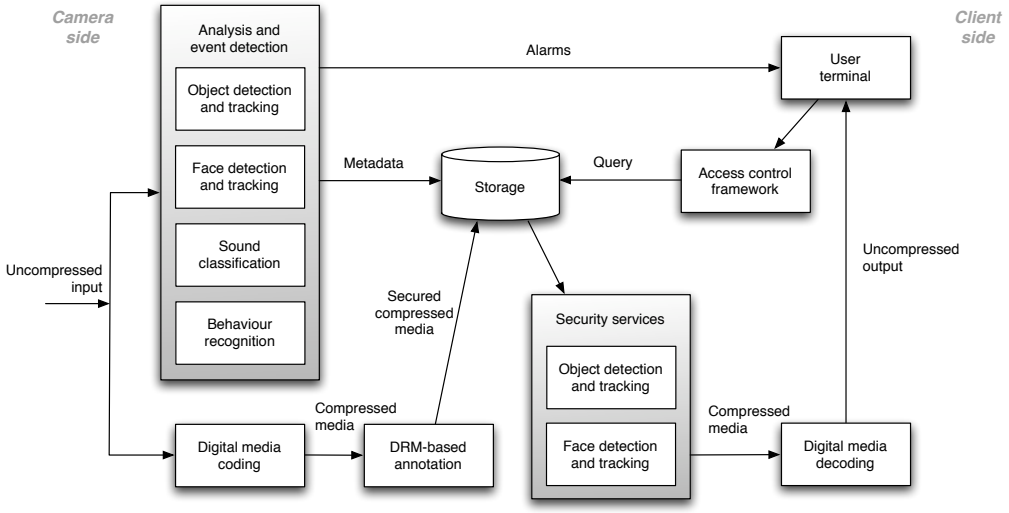


Figure 1.2: *Visnet II surveillance scenario general architecture. This architecture incorporates detection and tracking capabilities, digital media coding and particularly Digital Video Coding (DVC), audio-visual analysis including speaker detection and location, and surveillance-based segmentation and tracking (adapted from Ha-Minh* et al.*, 2007).*

way for an offline forensic search and retrieval of relevant data. In Figure 1.2 the surveillance architecture developed within VISNET II is represented. It consists of a generic architecture that embodies different data analysis, coding and management methods.

Many research projects dedicated to surveillance systems have been conducted in the past due to the large number of potential applications and the increased need for security systems, including the DARPA-funded projects VSAM (1997–1999) and HumanID (2002–2004), as well as the EU-funded projects PRISMATICA (2000–2003), ADVISOR (2000–2002), and AVITRACK (2004–2006) and its follow-up CO-FRIEND (2008–2011). Another research project that had some media projection was IBM's Smart Surveillance System (S3) [7][1]. Also, a new generation of systems for surveillance are starting to be commercialised. Some examples (non-exhaustive list) of commercial surveillance systems are the ones developed by iOmniscient [8], Keeneo [9], DETEC [10], 3VR [11], and BRS Labs (AISight) [12].

Although we considered the surveillance scenario as the potential target benefiting from the contributions of this thesis, other scenarios can also be considered. Ambient intelligence and health care monitoring are examples of alternative scenarios where

[1]This reference notation indicates a relevant resource (typically a web page) that is compiled in a list of resources at the end of the dissertation. More information about this is given in Section 1.5

the algorithms and tools described in this dissertation can be directly applied.

1.3 Thesis scope

This work is largely based on biologically inspired systems. The human sensorial and cognitive systems were looked as models since the earlier stages of artificial and machine intelligence. The work developed in cognitive psychology (Neisser, 1967) which developed on the foundations laid by Gestalt psychology, has been prolific. The main research areas in cognitive psychology are perception, categorisation, memory, knowledge representation, language, and thinking. The foundations of perception – attention, pattern recognition, and object recognition – specifically in vision (De Valois, 2000; Findlay and Gilchrist, 2003; Gregory, 1978; Marr, 1982) and audition (Bregman, 1990; McAdams and Bigand, 1993) are in large part the base of the work presented in this thesis. New fields of research applied to computer science have emerged from the knowledge acquired from biological systems, and particularly from the human perception system. Computer vision and computer audition are examples of such new research fields. Both these extensive fields have spawned multiple areas, and in particular (Computer Visual) Scene Analysis (CVSA) (Duda and Hart, 1973) and Computational Auditory Scene Analysis (CASA) (Rosenthal and Okuno, 1998).

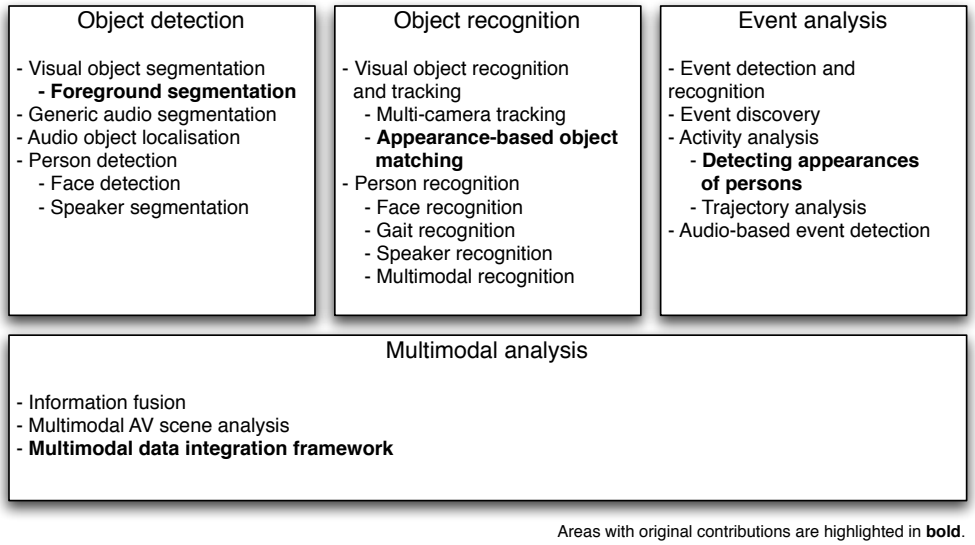


Figure 1.3: *Scope of the thesis. Although several areas are covered in this thesis, original contributions have only been done to a smaller set of topics.*

The scope of areas related to multimodal scene analysis are very broad. In Figure 1.3

we summarise all areas that are covered by this dissertation. These research areas can be divided in three larger areas, namely object detection, object recognition, and event analysis. A transversal larger area that we refer as multimodal analysis area deals with topics common to the top-level three. Each area represents a research challenge on its own and addressing all would be unfeasible. As such, we propose original contributions in specific research topics. The line of work followed is represented in Figure 1.4. Three layers of development were defined: framework layer, analysis layer, and application layer. The former included devising a multimodal framework (Teixeira *et al.*, 2008; Tzanetakis *et al.*, 2008) that could be used as a foundation for all analysis methods and generic multimodal application (Lagrange *et al.*, 2007). For the development of analysis methods we defined a coherent line of work that focused essentially on visual object analysis. New methods were proposed for object segmentation (Cardoso *et al.*, 2009; Teixeira and Corte-Real, 2007; Teixeira *et al.*, 2007), object matching (Teixeira and Corte-Real, 2009) and activity analysis (Teixeira *et al.*, 2009). Regarding the application layer, surveillance was targeted as the application where analysis methods would be applied (Duraes *et al.*, 2008).

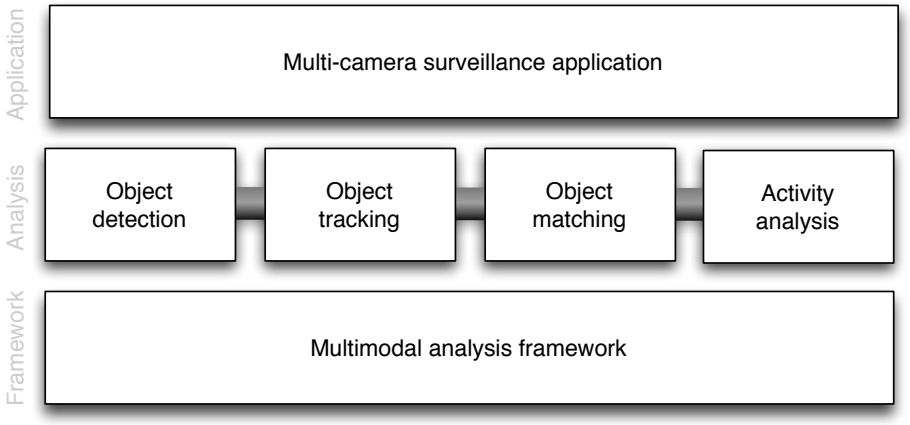


Figure 1.4: *Line of work of the thesis. Work evolved in three layers of development: framework layer, analysis layer, and application layer.*

1.4 Main contributions

Following the line of work defined in Figure 1.4 we developed in the course of this thesis a set of relevant algorithms and tools. These represent the main contributions of this thesis and can be summarised as follows.

1.4.1 Object segmentation

The first contribution is a *foreground segmentation algorithm* and is described in detail in Section 3.3. It consists of a method that models typical dynamic elements like acquisition noise, illumination variation and slow or repetitive structural changes. This model is then used to extract the foreground by testing if each pixel belongs to the background or the foreground. It presents improved performance and results when compared to commonly used methods. An objective comparison of background modelling and subtraction methods was also done using a metric devised for this purpose.

1.4.2 Object matching

The second contribution is an *appearance-based object matching algorithm* which is described in Section 4.3. This algorithm has the goal of being as an efficient method of matching visual objects tracked across independent views using local descriptors and the visual word paradigm. To accomplish that goal we defined a scalable representation and learning of generic visual objects. A direct result of this method is having a base for a surveillance system interface browsable by object tracks.

1.4.3 Activity analysis

The third contribution of this thesis is a *system to detect appearances of persons* which is detailed in Section 5.5. We developed an integrated segmentation/tracking/matching system for detecting appearances of persons over multiple uncalibrated cameras without overlapping field of view. An algorithm for the verification a person identity was defined for this purpose. This system also contributed to show that the performance of person tracking can be aided by the identity verification of each tracked person.

1.4.4 Multimodal analysis framework

In parallel with the other contributions, a forth contribution is the creation of an efficient *multimodal data integration framework* that provides the ability to use or develop new tools and algorithms handling different types of data. This framework is introduced in Section 6.3. It follows a dataflow architecture where complex network of processing objects can be assembled to form systems that can handle multiple and different types of multimedia flows with expressiveness and efficiency.

1.4.5 Related publications

The work presented in this thesis has been presented in several publications, which are available in PDF format at the author's page [1]. The publications are divided in two

groups. Specifically, the first group includes those with results directly related to this thesis and consists of 3 international conference papers and 3 international journal papers.

1. Luis F. Teixeira and Luis Corte-Real. Cascaded change detection for foreground segmentation. In *Proceedings of IEEE Winter Vision Meeting – Motion and Video Computing*, Austin, TX, February 2007. (Teixeira and Corte-Real, 2007)

   Abstract | *The extraction of relevant objects (foreground) from a background is an important first step in many applications. We propose a technique that tackles this problem using a cascade of change detection tests, including noise-induced, illumination variation and structural changes. An objective comparison of pixel-wise modelling methods is first presented. Given its best relation performance/complexity, the mixture of Gaussians was chosen to be used in the proposed method to detect structural changes. Experimental results show that the cascade technique consistently outperforms the commonly used mixture of Gaussians, without additional post-processing and without the expense of processing overheads.*

2. Luis F. Teixeira, Jaime S. Cardoso, and Luis Corte-Real. Object segmentation using background modelling and cascaded change detection. *Journal of Multimedia*, 2(5):55–64, September 2007. (Teixeira *et al.*, 2007)

   Abstract | *The automatic extraction and analysis of visual information is becoming generalised. The first step in this processing chain is usually separating or segmenting the captured visual scene in individual objects. Obtaining a perceptually correct segmentation is however a cumbersome task. Moreover, typical applications relying on object segmentation, such as visual surveillance, introduce two additional requirements: (1) it should represent only a small fraction of the total amount of processing time and (2) real-time overall processing. We propose a technique that tackles these problems using a cascade of change detection tests, including noise-induced, illumination variation and structural changes. An objective comparison of common pixel-wise modelling methods is first done. A cost-based partition-distance between segmentation masks is introduced and used to evaluate the methods. Both the mixture of Gaussians and the kernel density estimation are used as a base to detect structural changes in the proposed algorithm. Experimental results show that the cascade technique consistently outperforms the base methods, without additional post-processing and without additional processing overheads.*

3. Mathieu Lagrange, Luis G. Martins, Luis F. Teixeira, and George Tzanetakis. Speaker segmentation of interviews using integrated video and audio change detections. In *Proceedings of International Workshop on Content-Based Multimedia Indexing*, pages 219–226, Bordeaux, France, June 2007. (Lagrange *et al.*, 2007)

   Abstract | *In this paper, we study the use of audio and visual cues to perform speaker segmentation of audiovisual recordings of formal meetings such as interviews, lectures, or courtroom sessions. The sole use of audio cues for such recordings can be ineffective due*

to low recording quality and high level of background noise. We propose to use additional cues from the video stream by exploiting the relative static locations of speakers among the scene. The experiments show that the combination of those multiple cues helps to identify more robustly the transitions among speakers.

4. Luis F. Teixeira and Luis Corte-Real. Video object matching across multiple independent views using local descriptors and adaptive learning. *Pattern Recognition Letters*, 30(2):157–167, January 2009. (Teixeira and Corte-Real, 2009)

   Abstract | *Object detection and tracking is an essential preliminary task in event analysis systems (e.g. visual surveillance). Typically objects are extracted and tagged, forming representative tracks of their activity. Tagging is usually performed by probabilistic data association, however, in systems capturing disjoint areas it is often not possible to establish such associations, as data may have been collected at different times or in different locations. In this case, appearance matching is a valuable aid. We propose using bag-of-visterms, i.e., an histogram of quantised local feature descriptors, to represent and match tracked objects. This method has proven to be effective for object matching and classification in image retrieval applications, where descriptors can be extracted a priori. An important difference in event analysis systems is that relevant information is typically restricted to the foreground. Descriptors can therefore be extracted faster, approaching real time requirements. Also, unlike image retrieval, objects can change over time and therefore their model needs to be updated continuously. Incremental or adaptive learning is used to tackle this problem. Using independent tracks of 30 different persons, we show that the bag-of-visterms representation effectively discriminates visual object tracks and that it presents high resilience to incorrect object segmentation. Additionally, this methodology allows the construction of scalable object models that can be used to match tracks across independent views.*

5. Luis F. Teixeira, Luis G. Martins, Mathieu Lagrange, and George Tzanetakis. MarsyasX: multimedia dataflow processing with implicit patching. In *Proceedings of ACM International Conference on Multimedia*, pages 873–876, Vancouver, BC, October 2008. (Teixeira *et al.*, 2008)

   Abstract | *The design and implementation of multimedia signal processing systems is challenging especially when efficiency and real-time performance is desired. In many modern applications, software systems must be able to handle multiple flows of various types of multimedia data such as audio and video. Researchers frequently have to rely on a combination of different software tools for each modality to assemble proof-of-concept systems that are inefficient, brittle and hard to maintain. Marsyas is a software framework originally developed to address these issues in the domain of audio processing. In this paper we describe MarsyasX, a new open-source cross-modal analysis framework that aims at a broader score of applications. It follows a dataflow architecture where complex networks of processing objects can be assembled to form systems that can handle multiple and different types of multimedia flows with expressiveness and efficiency.*

6. Luis F. Teixeira, Pedro Carvalho, Jaime S. Cardoso, and Luis Corte-Real. Automatic description of object appearances in a wide-area surveillance scenario. *IEEE Transactions on Circuits and Systems for Video Technology*, 2009. (submitted). (Teixeira *et al.*, 2009)

   Abstract | *Segmentation and tracking of objects in a 2D video sequence is an important and challenging research area with many applications, among which automatic monitoring of installations has gained vital importance. In these situations, algorithms or tracking systems must deal with several factors such as coverage of large areas, humans moving in a group, partial or total occlusion, fast changes in direction, illumination changes and shadows, among others. To track people successfully we need to establish correspondences between objects captured in different cameras. Therefore, trackers are hardly ever the last link in the chain. Multi-camera systems demand for an additional element in the chain, responsible for collecting information from each track and establishing correspondences between objects detected in individual tracks. The output of this matching module is then the basis of several higher-level algorithms featuring event detection or semantic analysis. In this paper we present a complete system for object tracking over multiple uncalibrated cameras without overlapping field of view. The system is able to discover correspondences between different views of the same object. We employ an approach based on the bag-of-visterms to represent and match tracked objects. The tracks are compared with a global object model based on an ensemble of individual object models. If an object is recognised, the corresponding track is labelled. Moreover, if multiple objects are detected in a single track, the track is split and the partial tracks are labelled accordingly. The output is a timeline representing the objects present in a given scene. These timelines are compared with a ground-truth to evaluate the system's performance. The methods employed in the system are online and can be optimised to operate in real-time.*

The second group refers to publications with results partly related to this thesis. This group includes 2 international conference papers and 1 journal paper.

7. Daniel Duraes, Luis F. Teixeira, and Luis Corte-Real. Building modular surveillance systems based on multiple sources of information – architecture and requirements. In *Proceedings of International Conference on Signal Processing and Multimedia Applications*, Porto, Portugal, July 2008. (Duraes *et al.*, 2008)

   Abstract | *Intelligent surveillance is becoming increasingly important for the enhanced protection of facilities such as airports and power stations from various types of threats. We propose a surveillance system architecture based on multiple sources of information to apply on large scale surveillance networks. The main contribution of this paper is the definition of the requirements for a flexible and scalable architecture that supports intelligent surveillance using, alongside video, different sources of information, such as audio or other sensors.*

8. George Tzanetakis, Luis G. Martins, Luis F. Teixeira, Carlos Castillo, Randy Jones, and Mathieu Lagrange. Interoperability and the Marsyas 0.2 runtime. In *Proceedings of International Computer Music Conference*, Belfast, Ireland, August 2008. (Tzanetakis *et al.*, 2008)

   Abstract | *Marsyas is a software framework for building efficient complex audio processing systems and applications. Although originally designed for Music Information Retrieval (MIR) tasks in the past few years it has been expanded to include any type of audio analysis or synthesis. Complex Audio processing systems are defined hierarchically through composition using implicit patching. Both the specification of the processing network and the control of it while data is flowing through can be performed at runtime without requiring recompilation. Compilation is required only when new processing objects need to be defined. Therefore the Marsyas runtime provides considerable functionality and flexibility. In this paper we demonstrate how the Marsyas runtime can be accessed using a variety of different ways allowing non-trivial interactions with common software frameworks and environments.*

9. Jaime S. Cardoso, Pedro Carvalho, Luis F. Teixeira, and Luis Corte-Real. Partition-distance methods for assessing spatial segmentations of images and videos. *Computer Vision and Image Understanding*, 113(7):811–823, July, 2009. (Cardoso *et al.*, 2009)

   Abstract | *The primary goal of the research on image segmentation is to produce better segmentation algorithms. In spite of almost 50 years of research and development in this field, the general problem of splitting an image into meaningful regions remains unsolved. New and emerging techniques are constantly being applied with reduced success. The design of each of these new segmentation algorithms requires spending careful attention judging the effectiveness of the technique. This paper demonstrates how the proposed methodology is well suited to perform a quantitative comparison between image segmentation algorithms using a ground-truth segmentation. It consists of a general framework already partially proposed in the literature, but dispersed over several works. The framework is based on the principle of eliminating the minimum number of elements such that a specified condition is met. This rule translates directly into a global optimisation procedure and the intersection-graph between two partitions emerges as the natural tool to solve it. The objective of this paper is to summarise, aggregate and extend the dispersed work. The principle is clarified, presented striped of unnecessary supports and extended to sequences of images. Our study shows that the proposed framework for segmentation performance evaluation is simple, general and mathematically sound.*

Parts of this thesis also appears in two deliverables produced within the scope of the VISNET II [6] network of excellence.

10. Thien Ha-Minh, Alessandro Tortelli, Luis F. Teixeira, Lutz Goldmann, Mustafa Karaman, Stewart Worrall, Tim Masterton, and Charles Attwood. First set of

contributions and evaluation of tools for video segmentation and tracking. Deliverable D2.2.2, Visnet II consortium, November 2007. (Ha-Minh *et al.*, 2007)

Abstract | This report aims at describing the progress of VISNET II activities in video segmentation and tracking in the first 18 months of the project.

11. Thien Ha-Minh, Luis F. Teixeira, Pedro Carvalho, Lutz Goldmann, Mustafa Karaman, Stewart Worrall, Tim Masterton, Charles Attwood, and Krystian Ignasiak. Update set of developments and evaluation of tools for video segmentation and tracking. Deliverable D2.2.7, Visnet II consortium, August 2008. (Ha-Minh *et al.*, 2008)

    Abstract | This report aims at describing the progress of VISNET II activities in video segmentation and tracking during the second year of the project.

1.5 Structure of the dissertation

Given the broad scope of the topics covered in this thesis, it follows a "non-typical" structure. Throughout the text we present overviews of thesis-related areas, the often called state of the art. In parallel we introduce the original contributions that were developed in scope of this thesis. These novel contributions and respective evaluations are present in all chapters and are signalled with §. Also, the sections signalled with ¶ are the outcome of preliminary collaborations that have not yet resulted in a publication. Another particularity is how references are mentioned in the text. We considered two types of references: bibliographical references and resource references. The former type of references follow the traditional structure of bibliographical references – e.g. (Teixeira and Corte-Real, 2009); the latter type of references are used to identify web resources, such as the webpage of an open-source project or other relevant resources – e.g. [4]. Both types of references are compiled at the end of the dissertation in chapters called Bibliography and Resources, respectively. The remainder of the dissertation comprises the following chapters.

In Chapter 2 we start by presenting some concepts related to *multimodal scene analysis*. The methods associated to multimodal analysis are part of a broader research field, namely *information fusion*. We also present relevant concepts related to information fusion, such as *early fusion*, *late fusion*, *temporal fusion*, and *adaptive fusion*. In this dissertation we focus on multimodal analysis applied to Audio-Visual (AV) streams. With that in mind we discuss how AV signal are typically represented, as well as the types of applications of AV-based multimodal scene analysis. To illustrate some of the concepts introduced in the chapter, a proposal of a simple speaker identification application based on AV analysis is explained.

Chapter 3 consists of a review of *object detection*, including audio and visual-based segmentation. With segmentation we intend to separate complex scenes in its composing objects. We start by discussing *visual object segmentation*, and particularly segmentation

applied for real-world visual scenes. The most commonly used method in this case is *background modelling and subtraction*. We present an overview of techniques with that purpose, as well as a more in-depth comparison of methods. Furthermore, we propose of a background modelling and subtraction method based on the *cascaded detection of changes*. A particular case of object detection occurs when the goal is to detect persons. We discuss specific methods that can be applied, including *face detection*. Regarding *audio object segmentation*, we consider four types of generic classes (silence, noise, music, and speech) and use a method that is commonly applied for that purpose. Finally, we present some methods that can be used to *localise audio objects* (or sources) relying on AV joint analysis.

After object detection, in Chapter 4 we focus on *object recognition*. *Object tracking* combines object detection and recognition with an additional temporal element. We present an overview of visual object tracking methods using only one camera or multiple cameras. Object recognition is the task of associating an identity to a given object from a set of known classes, task closely related to *object matching*. We present afterwards a proposal for a *scalable object recognition method* which relies on the matching of *appearance models* based on local invariant features. An extensive evaluation of this method is presented. In parallel with the previous chapter, we also consider in this chapter the particular case of *person recognition*, including *face recognition*, *gait recognition*, and *speaker recognition*. Combining these methods with the appearance-based recognition system, we discuss how a multimodal person recognition system is possible.

Chapter 5 concludes the line of work regarding the methods employed in multimodal scene analysis. *Events* can be defined as meaningful real-world occurrences and can be valuable cues to index content. The scope of this chapter is *event analysis* and how it can be used to enhance for instance browsing of very long captured scenes, which is the case of real-world scenes (e.g. surveillance). We present an integrated multicamera-based method comprising object segmentation, tracking, and matching to detect when and where persons appear. The result is a timeline that aids the navigation of content. Also, we analyse a method that can be used to retrieve events based on detected person trajectories. A brief overview of audio event detection wraps up this chapter.

In Chapter 6 we present *MarsyasX*, a *multimodal analysis framework* that was used to implement the methods proposed in this thesis. The framework's goal is provide a common tool to implement algorithms based on different modalities. Usually we need to rely on a combination of different software tools for each modality to assemble proof-of-concept systems. We start by presenting the architecture of *Marsyas*, the audio-focused ancestral of MarsyasX. The new concepts introduced by the new framework are discussed afterwards. Finally, examples of applications developed with MarsyasX, including methods described in the previous chapters are shown.

The dissertation is summarised in Chapter 7. We also present possible future lines of work, including improvements to the methods proposed in the previous chapters and possible new methods. The chapter ends with some concluding remarks.

Additionally, three appendices complement the main part of the thesis. In Appendices A and B we describe the datasets and the evaluation methods, respectively, that were used in the development of visual and audio analysis methods. The latter appendix includes a short description of a new metric to compare foreground segmentations. Finally, in Appendix C we give an overview on local interest point detectors and local descriptors, in particular SIFT that we use for object matching in Chapter 4.

2 Multimodal scene analysis

Human perception is in essence based in multimodality. Complex cross-modal interactions occur when the human brain receives stimuli from multiple sources. It is thus appropriate to establish a parallel, and perform computational analysis based on the interaction between modalities. By combining information conveyed by different modalities we aim to extract additional knowledge or to complement the knowledge obtained analysing each modality separately. However, new challenges emerge, namely how can we combine data having different representations, and how can we make such combination effective.

2.1 Concepts

The term *multimodal analysis* is somewhat ambiguous since it can have different interpretations, depending on the area it refers to. We can find references to multimodal analysis in disparate areas, such as linguistics, cognitive psychology and computer science. It is therefore important to define the underlying concepts and the context of application.

A confusion may arise with the terms *multimodal system* and *multimedia system*. Buxton (1994) refers that multimedia focuses on the *medium* or *technology* rather than the *application* or *user*. Conversely, a multimodal system focuses on the way a user interacts with information. In a multimodal system, information is modelled at a higher level of abstraction, more closely related to its meaning.

Considering Human-Computer Interaction (HCI), Nigay and Coutaz (1993) defined that a multimodal system supports communication with the user through different modalities such as voice, gesture, and typing. The term *modal* may cover the notion of *modality* as well as that of *mode*. The modality defines the type of data exchanged, whereas the mode determines the context in which the data is interpreted. Applying the same reasoning to automatic scene analysis, modality can be seen as the type of input data that is analysed. It represents distinct ways of conveying information. In this case, computational models detect and recognise objects and events by analysing the different available modalities, in a similar way the human brain would do.

modality

multimodal scene analysis

The main goal of multimodal scene analysis is to extract knowledge from different signals, potentially representing multiple modalities. Information often does not appear in single modalities, so we can explore the synergy existing across the different modalities. By exploring interactions between modalities, new knowledge is obtained

that would not be possible to obtain or would be incomplete otherwise. The additional knowledge can be in the form of processed signals or descriptive information. Figure 2.1 depicts this paradigm.

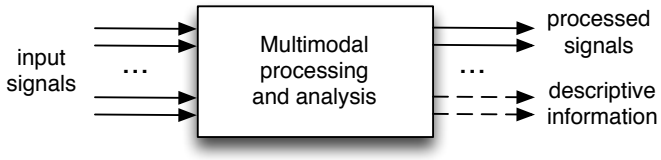


Figure 2.1: *The multimodal processing and analysis paradigm. From multiple input signals representing multiple modalities, obtain additional knowledge in the form of processed signals and descriptive information.*

Arbitrarily complex configurations of multimodal processing and analysis relations are possible. Depending on the application the output could be translated into decision or actions taken by an intelligent system, or structured in descriptive information for posterior analysis. In Figure 2.2 an example of such configuration is shown.

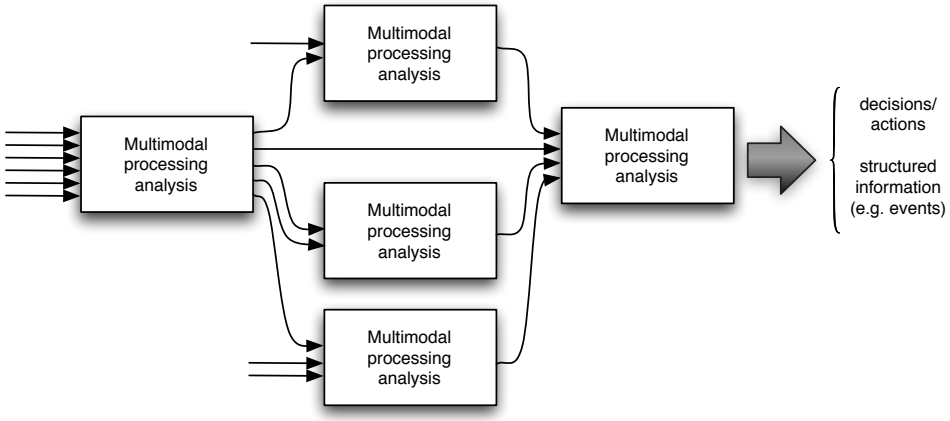


Figure 2.2: *A multimodal processing network. Arbitrarily complex configurations of multimodal processing and analysis relations are possible.*

Nigay and Coutaz (1993) proposed a design space to systematise the description of multi-feature systems, such as a multimodal system. It considers three dimensions: levels of abstraction, use of modalities and fusion. Figure 2.3 shows a three-dimensional graphical representation of this model. As previously, we follow closely the authors' definition applied to multimodal interaction but transpose it to the context of multimodal scene analysis.

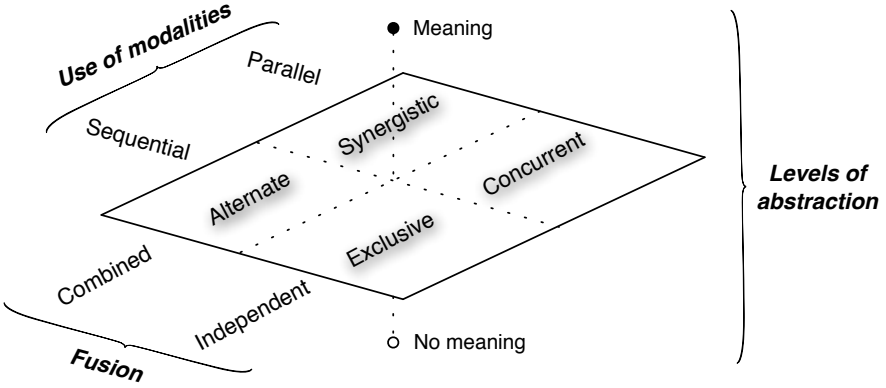


Figure 2.3: *A multi-feature system design space. Three dimensions are considered: levels of abstraction, use of modalities and fusion. (adapted from Nigay and Coutaz, 1993)*

levels of abstraction Data may be processed at multiple levels of abstraction. For example, speech input may be recorded as a signal, described as a sequence of phonemes, or interpreted as a meaningful parsed sentence. Two levels of abstraction are defined: "Meaning" and "No Meaning".

use of modalities The second dimension, use of modalities, expresses the temporal availability of multiple modalities. Two type of use modalities are considered: "Sequential" and "Parallel". A system that supports parallel employs multiple modalities simultaneously, while a system with sequential use does not.

fusion Finally, fusion describes if the combination of different types of data is done or not. If no combination is done, the system is characterised as being "Independent", and if it is done, the system is referred to as "Combined". According to the design space, fusion may be performed with or without knowledge about the meaning of the data exchanged. Fusion based on meaning mixes modalities to process information results in an interpretation at a high level of abstraction.

Despite the efforts to study the theory supporting multimodal systems such as Nigay's, traditionally the work on multimodal integration has been mostly based in the definition of heuristics. The lack of fundamental theory originates two important questions when building a multimodal system: how can we choose the most suitable modalities? how can we optimally combine information from multiple modalities? There is no systematic way to determine an answer to these questions. Additionally, three trade-offs need to be considered: *modality independence*, *curse of dimensionality*, and *fusion model complexity*.

2.2 Information fusion

The methods associated with multimodal analysis are included into the broader research field of information fusion. In general, information fusion includes any area that combines different information sources, either to generate one representational format, or to reach a decision.

By performing information fusion, a decision or an action is generated that is expected to be in some sense better than the one originated when only one source of information was used. The acquisition of information by multiple sensors, capturing the same reality, can be thought as the decomposition of the event into its components by the sensors. By fusing, as optimally as possible, the information provided by the different sensors one expects to reverse this inevitable decomposition. Ideally, the sensor fusion would be able to restore all inherent information of interest. The evaluation of the action improvement may depend on the domain of application, and can be qualitative or quantitative – for example, accuracy, robustness or error resilience.

The huge volume of information generated by multi-sensor systems is one of the main challenges faced by information fusion, especially when real-time constraints are imposed by the application in question. Moreover, this information is continuously corrupted by noise and other disturbances which vary throughout its acquisition by the multiple sensors making it even more challenging.

Information fusion is applied in areas like team decision theory, integration of multiple sensors, distributed detection and distributed decision making. Dasarathy (2000) refer different industrial applications areas where information fusion can be applied, including: machining related problems, diagnostic and non-destructive testing, robot navigation, human identity verification and manufacturing quality control. The goal of fusion in the real-world scenarios we are considering is combining information from different sensors to detect the presence of an object or environmental condition, identify objects or events, and track an object or event over a period.

Traditionally, information fusion has been characterised in different hierarchical levels according to the stage of processing at which the integration is performed. In this sense, information fusion techniques are often classified in three main categories (Dasarathy, 1997): *data fusion*, *feature fusion* and *decision fusion*. This classification is further expanded into five fusion I/O-dependent categories, i.e., whether the fusion takes as input sensor data, features or decisions as well as the output. Additionally, *temporal fusion* is also considered when information is acquired and integrated over a period of time. Temporal fusion can occur at any of the three previous levels and can be characterised as orthonormal to the three-level categorisation. Sanderson (2002) propose a two-category classification: *pre-mapping fusion*, or input-level fusion, when information is combined before any use of classifiers; and *post-mapping fusion*, or output-level fusion, when information is combined after mapping from sensor-data/feature space into opinion/decision space. A combination of both categorisation proposals will be

used in the remainder of this chapter. Also, the terms *early fusion* and *late fusion* will be used instead of pre-mapping and post-mapping fusion, respectively.

2.2.1 Early fusion

Early fusion can be further sub-divided in three categories: *data in-data out fusion*, *data in-feature out fusion* and *feature in-feature out fusion*.

□ Data in-data out fusion

Data in-data out fusion is at the lower level according to this classification. At this level, data captured by sensors, is combined directly using one data merging method. It is thus performed at the front-end of the processing stream, immediately after acquisition. Fusion at data level requires compatible sensors and both spatial and temporal registration of data are very important to this effect.

Typically two methods are used for data level fusion: *weighted summation* or *mosaic reconstruction*. The former can be used to combine, for example, visual and infrared images into one image, or, acoustic data from two or more redundant microphones, reducing noise. A more detailed description of weighted summation will be presented next. Mosaic construction can be used, for example, to create one image out of images captured by several cameras, where each camera captures part of the same scene or object. Other examples of application areas include the processing techniques used in multispectral data analysis applications. In this case pixel intensities are acquired by different sensors and combined in a multispectral approach, which can be useful in the extraction of fine details. Multidimensional data fusion can also be accomplished through Principal Component Analysis (PCA) or other transform techniques.

The level of detail is highest at this data level, however the corruption of information due to noise is also highest. The process of extracting relevant information at this level, in the form of features or decisions, may result in the loss of some information in levels of details, but can also reduce the noise that degrades the decision system's quality. In fact when fusion is performed at any level of information abstraction, there is a inherent process of compactation, resulting in the loss of information that will only be available at this level and will not be carried to the next level.

□ Data in-feature out fusion

With data in-feature out fusion, data from different sensors are combined to derive some form of a feature of the object in the environment or a descriptor of the event under observation. Techniques based in this type of fusion were studied for machine perception of depth in robotic systems. In some cases this may be the first step in a processing chain with the previous mode being totally absent.

□ **Feature in-feature out fusion**

In this mode, both input and output are features. An initial processing is therefore applied to data to extract sets of features from one or more sources of raw data. These sets of features are then combined and processed either quantitatively, in a multidimensional feature space processing, or qualitatively, within a heuristic decision logic process, or through a combination of both. The latter two are particularly true when sensors have different data structures and features obtainable from one are not derivable from the other. If the features have a common measure then the fusion can be performed using *weighted summation*. Otherwise *feature vector concatenation* can be applied and an aggregate feature vector is constructed by concatenating the feature vectors obtained from each source. Some issues may however arise when using this method. There is no explicit control over how much is the contribution of each individual feature vector in the final decision. Moreover, the aggregate feature vector may suffer from the "curse of dimensionality" due to its usual large size. Applying methods in order to reduce dimensionality may however obviate this problem. Finally, the feature extraction must be synchronous, i.e., separate feature vectors must be available at the same rate.

2.2.2 Late fusion

The late fusion approaches may be sub-divided into two other categories: *feature in-decision out fusion* and *decision in-decision out fusion*. The latter can have different interpretations if we are dealing with the combination of hard decisions provided by a set of *classifiers*, or soft decisions (or opinions), given from a set of *experts*. Often the terms decision fusion and opinion fusion are used interchangeably. However, since each expert provides an opinion and not a hard decision, the term opinion is more appropriate when an ensemble of experts is used.

□ **Feature in-decision out fusion**

In this case, the inputs are features from different sensors and the output of the fusion process is a decision, for example, a target class recognition. It is probably the most common of the fusion modes since pattern recognition systems with inputs from multiple sensors have long been performing this type of fusion. A feature vector is classified according with a priori knowledge or a model obtained in a training phase. Therefore all techniques used for pattern recognition are directly applied when the type of fusion is performed.

□ Decision in-decision out fusion

Both inputs and output are decisions in this type of fusion. The fusion of multiple classifiers, was first proposed as a possible solution to the problems posed by the traditional pattern classification approach which involved selecting the best classifier from a set of candidates based on their experimental evaluation – no classifier is known to be the best for all cases. It was also motivated by requirements of many sensor fusion applications, that induce natural decompositions of the information processing tasks and demand for complex decision fusion architectures. Moreover, fusion at the lower levels is not always possible. For example, if incompatible sensors are used and data registration is not feasible, local decisions have to be performed and passed on to a fusion processor where the individual decisions are integrated. Although fusion at the decision level is not always necessary or the most interesting approach, it is always at least feasible.

The classifiers used for decision fusion can be of the same type but working with different set of features (e.g. audio and video data). Other configurations include non-homogeneous classifiers evaluating the same set of features or a hybrid configuration using both of the previous. The main motivation to use non-homogeneous classifiers with the same set of features is to take advantage of the classifier specialisation – i.e., a classifier can be good at evaluating a particular set of classes while being bad evaluating another different set of classes. This way it can be possible to overcome the downsides of each classifier. Decisions can be combined by majority voting, using AND & OR operators or combination of ranked lists.

majority voting

With majority voting, the final decision is the one with the higher number of classifiers agreeing on that decision. An odd number of classifiers is required in order to avoid ties. Moreover, the number of classifiers must be greater than the number of decisions to ensure that a decision is reached. For verification problems this last condition is not a limitation.

AND & OR operators

With AND operator fusion, a class is only accepted when all classifiers agree – thus, this classification is very restrictive. For multi-class problems a no-decision situation may be reached and is therefore mainly used when a low false acceptance is required. On the other hand, with OR operator fusion, a class is accepted when at least one classifier makes a positive decision. In opposition to the AND operator fusion, this type of fusion is very relaxed and may lead to multiple possible decisions in multi-class problems. It is mainly used when a low false rejection rate is required.

combination of ranked lists

The final decision using combination of ranked list is reached by combining ranked list of class labels provided by each classifier. The lists of classes are sorted according the degree of preference for each class and the top entry usually corresponds to the most preferred. The final decision is reached by selecting the top ranked class of a combined list. The way this combined list is created depends on the application and can take into account the reliability and discrimination ability of each classifier.

□ Decision vs. opinion fusion

The main advantage of opinion fusion when compared to decision fusion is that the decisions can be weighted. These weights can reflect the reliability and discrimination ability of the individual experts. The opinions can be combined using *weighted summation* or *weighted product* approaches before using any classification criterion. Different classification criteria to reach a decision are possible including hard-level combination techniques such as the max the min and median rules. The max and min operators rely on the classifier with the highest an the lowest best opinion scores, respectively, and disregarding the decisions of the other classifiers. The max operator tends to have high false accept rate, whereas the min operator is best suited for high security applications. Both methods rely solely in opinion scores and do not employ additional reliability measures. It is also possible to use soft-level combination techniques as classification criteria which, unlike the hard-level techniques, regard each coefficient as a measure of reliability of each classifier. Reliability values R_p can assume fixed values defined using a priori knowledge about the performance of each expert. Alternatively it can be estimated adaptively for each decision instant.

Different types of experts can be used in opinion fusion, however the opinions are usually required to have a common measure. This can be accomplished by mapping the output of each expert to the $[0, 1]$ interval.

Alternatively to weighted summation and product fusion techniques, a post-classifier can be used to reach a decision using the opinions provided by the experts. In this case opinions can be considered as features in the likelihood space. The opinions from N_E experts regarding N_C classes form a $N_E \times (N_C - 1)$ dimensional opinion vector, which is used by a post-classifier to make a final decision.

An important advantage of using a post-classifier approach is that the opinions do not necessarily need to have a common measure as in the two previous approaches. In this case, the post-classifier makes adequate mapping from the likelihood space to the class label space. Notice that in a verification scenario, the dimensionality of the opinion vector is only dependent on the number of experts since only two classes (accept / reject) exist.

2.2.3 Temporal fusion

Temporal fusion happens typically when data is integrated over a period of time, such as averaging of samples. An example of temporal fusion is the tracking function of an objects or a person. It is applicable in any level of fusion previously presented and it assumes an orthonormal dimension of the fusion process.

2.2.4 Adaptive fusion

The first developments in adaptive fusion used several different expert networks and a gating network that decided which of the experts should be used for each specific case. Later work extended the system in order that the system could learn how to allocate new cases to experts by checking the output – if the output is incorrect, the weights for the selected experts change in the gating network. Hence, there is no interference with the weights of other experts that specialise to quite different cases. The experts are therefore local in the sense that the weights in one expert are decoupled from the weights in other experts. An error measure is usually used to compare the desired output with a combination of local experts and adjust the weights to cancel this error. When the weights in one expert change, the same happens to the residual error, and the error derivatives for all the other local experts. The strong coupling between experts causes them to cooperate – *associative learning* – but tends to lead to solutions where too many experts are necessary. An alternative is the *competitive learning* where only one expert is used for each case instead of combining the outputs of all experts.

2.3 AV-based multimodal scene analysis

During the last decades many studies have been made about the human perception of visual and audio stimuli and how they interact. There are multiple known influences between vision and audition. The McGurk effect (McGurk and MacDonald, 1976), and the ventriloquist effect (Howard and Tempelton, 1966) are examples of what we see influences what we hear. In the same manner, several phenomena have been discovered that show influences from audition onto vision. For example, a sound may influence the perceived direction of a bistable visual motion display (Sekuler *et al.*, 1997). Besides effects on bistable visual stimuli, audition can affect even unambiguous visual stimuli. A brief flash accompanied by two beeps is mostly perceived as two flashes. Speech perception is also significantly influenced by seeing and hearing a speaker, compared to just hearing. This is especially prominent with noisy signals, showing that the additional visual information enhances speech perception (Sumby and Pollack, 1954).

In general, for a complete scene understanding, it is important to analyse all the available information and identify not only the objects that compose the scene but also its nature and their relationship. In the context of audiovisual scene analysis, the nature of the sources of information is usually restricted to the audio and the video signals. Although visual and audio processing and analysis research areas had a considerable amount of attention in the past two decades, work has progressed mostly in separate paths. Some effort has been put more recently to understand how the integration of these areas can help get a better understanding of the dynamics being analysed or simply improve performance of already proven algorithms in either area. Some examples of the latter are the audiovisual speech processing, person authentication, document retrieval and tracking of objects and humans, whereas for the former, examples

of research applications are multimodal Human-Computer Interaction (HCI), emotion recognition and intelligent systems.

Visual and audio sensors, or any other sensors capturing reality for that matter, carry information about events that when combined correctly can provide complementary perception of the same event. Multimodal contingency can therefore be used to help determine which signals in different modalities share the same a common origin. This allows signals to help locate and interpret each other. The integration of visual and audio data can be performed at earlier or later processing stages. When performing early integration all streams are merged and considered as one. However, we are assuming there are dependencies at the lower level – at the frame level – but some higher information dependencies might be missed – feature or semantic levels. The dependencies can be taken into account at later stages, performing late integration.

The main challenges when integrating audiovisual information arise from the increased dimensionality and complexity (Wu *et al.*, 2004). Often heuristics are needed to limit the information space. Capturing long-term dependencies is naturally more difficult than when analysing only one stream. Another challenge is the lack of largely available benchmark datasets that could provide common ground for multimodal analysis research.

2.3.1 Representing AV signals

□ Visual features

Many visual feature sets have been proposed throughout the years and, in general, each set can be grouped in four possible categories: appearance-based, shape-based, mixed-based, and motion-based features. We present a small overview of these categories, but for more in-depth information, refer to (del Bimbo, 1999; Gupta and Jain, 1997; Ojala *et al.*, 1996; Wang *et al.*, 2000).

⇀ **Appearance-based features** This group of features are generally directly related to the pixel-level structure of the images forming the visual signal. The features may be analysed for the whole image or just a Region Of Interest (ROI). The ROI can assume many forms, depending on the prior knowledge of the dynamics and is therefore application-specific - it may have irregular forms and may not be spatially connected, for example, the result of an object segmentation.

An important discriminative visual feature is *colour*. Colour histograms, which represent colour distribution in an image, are a widely used and simple approach, yet very effective (Swain and Ballard, 1991). Histograms are robust to change in viewpoint and scale and to occlusion but depend highly on the used colour space. The RGB representations are widely used but usually are not suited to capture the invariant characteristics. A representation often used is HSV because of its proximity to the

human perceptual organisation of colours. The main problem regarding colour histograms is that spatial configuration of pixels is not considered, leading to completely different images to have the same colour histogram. Several approaches have been proposed to overcome this problem, like augmented colour histograms containing, besides colour probability, also the mean, variance and entropy of pair-wise distances among pixels with this colour. Other common colour-based features are colour moments (Stricker and Dimai, 1996) and colour correlograms (Huang *et al.*, 1997), which can salso be used to characterise the colour information. To overcome the problem of high-dimensionality, Singular Value Decomposition (SVD) (Hafner *et al.*, 1995), dominant colour regions (Ravishankar *et al.*, 1999) and colour clustering (Wan and Kuo, 1998) have also been proposed.

The object's *texture* can also be used as an identifying characteristic. Texture models are concerned with representing regular patterns in an image. We can generally classify texture models into two groups: statistical models and spectral models. Statistical models collect statistics directly from the image. The most simple texture models include 1D grey-level histograms, co-occurrence matrices, and grey-level differences. Also, the Haralick feature (Haralick *et al.*, 1973), derived from co-occurrence matrixes, models the spatial relationship of each pixel pair in the image. Spectral models, on the other hand, collect statistics related to features computed in the frequency domain from the responses of filters applied to the image. Filtering methods are the most commonly used techniques for the extraction of the texture features. One advantage of spectral models is that the filters are selective: they can enhance certain features while suppressing others. For instance, Lindeberg and Lowe have shown that the Laplacian and Laplacian of Gaussian filters (Lindeberg, 1998; Lowe, 2004) are good at detecting edges in an image. Low-pass Gaussian filters are useful for blob detection. Two-dimensional Gabor filters (Bovik *et al.*, 1990) have proven to be popular for modelling texture, due to their efficiency in detecting dominant frequency and orientation in texture patterns.

⇀ **Shape-based features** Unlike appearance-based, shape-based features consider that the relevant information is contained in shapes or parametric models. These models are usually application specific and require prior knowledge of its dynamics. For example, for the specific case of visual speech recognition, different models can be used such as the contours of the speaker's lips or the face contours.

Many techniques, including moment invariants, Fourier descriptors, autoregressive models and geometry attributes have been proposed for measuring shape. These features can be further classified in global and local features. Global features represent properties derived from the entire shape. Examples include roundness or circularity, central moments, eccentricity and major axis orientation. Local features derive from partial processing of a shape and do not depend on the entire shape; size and orientation of consecutive boundary segments, points of curvature, corners and turning angle are some examples.

Shape representation can be based in different techniques. Jain and Vailaya (1996) proposed a shape representation based on the use of a histogram of edge directions. Curvature scale-space has been proposed by Mokhtarian *et al.* (1996) to characterise shape features of an object or region based on its contour. This representation can be robust to non-rigid motion, partial occlusion, and perspective transformations due to camera motion.

⇀ **Mixed Features** Although appearance- and shape-based features can be seen as two different ways of looking to the information – low-level and high-level information, respectively – it is possible to combine both in order to achieve better performance. Typically, features of both type are simply concatenated and evaluated as such, but more complex combined framework can be devised.

⇀ **Motion-based features** An important component of the human vision system is motion detection. The most common representation of motion is optical flow. Typically, optical flow represents motion as vectors originating or terminating at pixels in the video, though other representations exist as well. The seminal work on optical flow was by Horn and Schunck (1981), followed by the popular Lucas-Kanade method (Lucas and Kanade, 1981). Later methods computed optical flow more robustly over multiple scales using a pyramid scheme. These methods, however, only compute motion between subsequent frames. More recently, methods have been proposed to compute optical flow over longer time scales (Brox *et al.*, 2004). While optical flow is concerned with the motion of pixels, others have looked at the motion of low-level image features, such as corners and the endpoints of lines, for finding motion of the scene or the camera. In scene analysis, the most important motion features are: motion activity, camera motion, and motion trajectory. Motion activity gives an idea about the intensity of actions (Divakaran and Sun, 2000). The camera motion expresses the intention of the viewer's focus of attention, and can thus be used to discriminate events (Jeannin *et al.*, 2000). The motion trajectory of an object is a simple high-level description, and is defined as the localisation, in time and space, of one representative point of this object.

□ Audio features

Audio features used for scene analysis include typical measures of the audio waveform already used in other well-known problems such as speech recognition, but also specific features. The audio features can be extracted in two levels: short-term frame level and long-term clip level. The frame level is defined as a group of neighbouring samples with duration between 10 and 40 ms, so that a stationary signal can be assumed. For a feature to reveal the semantic meaning of an audio signal, analysis over a much longer period is necessary, usually from one second to several tens of second. This interval is called an audio clip and it consists of a sequence of audio frames. The clip boundaries

may be the result of audio segmentation such that the frame features within each clip are similar. In fact, some authors proposed a direct segmentation of audio signals into regions, based on temporal changes of selected features, without trying to classify the content (Tzanetakis and Cook, 1999).

These two main divisions can be further divided according to their processing domain into time-domain features and frequency-domain features. Time domain features are computed directly from the audio waveform, and reflect temporal properties of the audio signal. However, some differences between distinct classes of audio signals become easier to identify in the frequency-domain than in the time-domain. The spectrum of an audio frame is a representation of its frequency content, but using the spectrum itself as a frame-level feature is not practical due to its high dimensionality. Consequently, more succinct descriptors may be computed from the spectrum, resulting in highly discriminative frequency domain features.

A brief description of the most relevant audio features used for scene analysis is presented next. For a more detailed overview of audio features for multimedia content analysis, refer to Liu *et al.* (1998); Lu *et al.* (2002); Saunders (1996); Scheirer and Slaney (1997); Wang *et al.* (2000).

⇀ **Frame-level features** Audio features extracted at the frame level capture the short-term characteristics of an audio signal.

volume The easiest frame feature to compute frame is volume. It is a reliable indicator for silence detection, which may help to segment an audio sequence and to determine clip boundaries. Volume is also known as loudness. Volume is approximated by the root mean square (rms) of the mean energy of the signal within a frame and its squared version, is also known as Short Time Energy (STE).

zero cross rate The Zero Cross Rate (ZCR) of a frame is computed by counting the number of times the audio waveform crosses the zero axis per time unit after DC removal. It is a very useful measure to discern between voiced/unvoiced speech, because typically unvoiced speech has a low volume, but a high ZCR. By combining ZCR and volume together, it is possible to classify low volume and unvoiced frames as silence.

frequency centroid Frequency Centroid (FC) is related to the human sensation of the brightness of a sound and gives discriminating results for music and speech, as well as for voiced and unvoiced speech. This feature has a high correlation with the ZCR feature.

spectral flatness Flatness-oriented spectral features describe the flatness properties of the short-term power spectrum of an audio signal. This family of features expresses the deviation of the signal's power spectrum over frequency from a flat shape (corresponding to a noise-like or impulse-like signal). A high deviation from a flat shape may indicate the presence of tonal components. Since the desired characteristics (tone vs noise-likeness) are attributed to specific frequency bands rather than the entire spectrum, these features will be applied on a frequency band basis.

cepstral coefficients In speech processing, the Cepstral Coefficients (CC) are used to obtain the formants from voiced phonemes. The information relevant to the formants is contained in the first few coefficients of the cepstrum. The cepstrum can be computed as the inverse Fourier transform of the logarithm of the spectrum.

MFCC Mel Frequency Cepstrum Coefficientss (MFCCs) are based on the human auditory system model of critical bands. Linearly spaced filters at low frequencies (below 1000 Hz) and logarithmically at high frequencies (above 1000 Hz) have been used to capture the phonetically important characteristics of speech (mel-frequency scale). The speech signal is divided into frames of N samples, with adjacent frames being separated by M ($M < N$) samples. A Hamming window is used to minimise the signal discontinuities at the borders of each frame. After this step the Fast Fourier Transformation is applied and the absolute value is taken to obtain the spectrum magnitude. The signal is then processed by the Mel-filterbank. Cepstrum is the final step where the log mel spectrum is converted back to time using the Discrete Cosine Transform (DCT).

$\rightharpoonup$ **Clip-level features** If a higher semantic content analysis is to be performed, it is necessary to observe the temporal variation of frame features during longer time intervals. This leads to the development of various clip-level features, which characterise how frame-level features change over a clip.

volume -based Multiple volume-based features are proposed in the literature. Liu *et al.* (1998) did an extensive work on audio feature extraction and analysis. They proposed several audio measures, such as the standard deviation normalised by the maximum volume in a clip, the mean value of the volume within a clip, and the volume dynamic range. Another feature volume-based, more suitable to detect music segments, is the pulse metric proposed by Scheirer and Slaney (1997), which uses long-time band-passed autocorrelations to determine the amount of "rythmicness" in a 5-second window.

energy -based Low Short Time Energy Ratio (LSTER) is defined as the ratio of the number of frames with an STE are less than 50% of the average short time energy in a $1s$ window. LSTER is an effective measure to distinguish between speech and music. Since there are more silence frames in speech than in music, the LSTER measure of speech will be much higher than that of music (Saunders, 1996). Energy entropy is another energy based feature, computed by dividing each audio frame into segments of K samples each. The signal energy is computed over each of these segments and normalised by the overall frame energy.

ZCR -based Some researchers have reported the usefulness of the standard deviation of the ZCR to differentiate between TV program categories. According to (Saunders, 1996), statistics of the ZCR can be used to discriminate between speech and music audio segments with high accuracy classification rate. Lu *et al.* (2002) propose a more discriminative feature

based on the ZCR, known as High Zero Crossing Rate Ratio. This feature is defined as the ratio of the number of frames whose ZCR are above 1.5-fold average Zero Crossing Rate in an $1s$ window.

spectrum flux Spectrum Flux (SF) is defined as the average variation value of spectrum between the adjacent two frames in a window. According to Lu *et al.* (2002), SF values of speech are higher than those of music. In addition, environmental sounds present the highest values with more changes. Therefore, SF is a good feature to discriminate speech, environmental sound and music.

LSP distance Linear Spectral Pairs (LSP) are derived from Linear Predictor Coefficient (LPC), and have shown to have good discriminative power and to be highly robust in noisy conditions (Lu *et al.*, 2002). To measure the LSP dissimilarity between two 1-second audio clips, an abbreviated Kullback-Leibler divergence measure may be used. This abbreviated distance measure is normally called divergence shape distance, and is similar to the cepstral mean subtraction method use in speaker recognition to compensate the effect of environment conditions and transmission channels. This dissimilarity measure provides good results when used to discriminate speech and noisy speech from music. Furthermore, LSP divergence shape also proved to be a good feature for the discrimination of different speakers (Lu *et al.*, 2002).

□ Pre- and post- processing

In most cases, the dimensionality d of the feature vector is too large and inhibits the use of common statistical modelling (e.g. Hidden Markov Model (HMM)). To overcome this problem, transformations to the feature space are often performed. The most common are *image transforms* which are borrowed from the image compression literature and aim to preserve all the important information while reducing dimensionality. Typically, a $D \times d$-dimensional *linear transform* matrix $\mathbf{P}$ is defined in order that the $D \ll d$-dimension transformed data vector $\mathbf{y}_t = \mathbf{P}\mathbf{x}_t$ contains most information necessary.

Principal Component Analysis (PCA) – The PCA minimises the square error between the original vector $\mathbf{x}_t$ and its transformation result vector $\mathbf{y}_t$. Appropriate data scaling constitutes a problem in the classification of the resulting vectors. One proposed implementation of PCA scales data according to the data inverse variance, and computes the correlation matrix $\mathbf{R}$. This matrix is then diagonalised as $\mathbf{R} = \mathbf{A}\Lambda\mathbf{A}^\mathsf{T}$ where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$ has as columns the eigenvectors of $\mathbf{R}$, and Λ is a diagonal matrix containing the eigenvalues of $\mathbf{R}$. If the D highest eigenvalues are located at the $j_1, \ldots, j_D$ diagonal positions, then the data projection matrix is $\mathbf{P}_{\mathsf{PCA}} = [\mathbf{a}_{j1}, \ldots, \mathbf{a}_{jD}]^\mathsf{T}$. In short, the data vector $\mathbf{x}_t$ is first element-wise mean and variance normalised, and then its feature vector is extracted as $\mathbf{y}_t = \mathbf{P}_{\mathsf{PCA}}\mathbf{x}_t$. PCA is especially useful when there is a prior knowledge of the statistics.

Linear Discriminant Analysis (LDA) – LDA assumes two conditions: a set of classes

$\mathcal{C}$ is *a priori* chosen; and the training set data vectors $\mathbf{x}_l, l = 1, \ldots, L$ are labelled as $c(l) \in \mathcal{C}$. The matrix is then computed $\mathbf{P}_{\mathsf{LDA}}$ such that the projected training sample is "well separated" into the set of classes $\mathcal{C}$, according to a function of the training sample within-class scatter matrix $\mathbf{S_W}$ and its between-class scatter matrix $\mathbf{S_B}$. To estimate $\mathbf{P}_{LDA}$, the generalised eigenvalues and right eigenvectors of the matrix pair $(\mathbf{S}_B, \mathbf{S}_W)$, that satisfy $\mathbf{S}_B\mathbf{F} = \mathbf{S}_W\mathbf{F}\Lambda$, are first computed. The columns of matrix $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_d]$ are the generalised eigenvectors. Assuming that the D largest eigenvalues are located at the $j_1, \ldots, j_D$ diagonal positions of $\mathbf{F}$, then $\mathbf{P}_{\mathsf{LDA}} = [\mathbf{f}_{j1}, \ldots, \mathbf{f}_{jd}]^{\mathsf{T}}$. It should be noted that the rank of S_B is at most $|\mathcal{C}| \leq 1$, where $|\mathcal{C}|$ denotes the number of classes (the cardinality of set $\mathcal{C}$); hence $D \leq |\mathcal{C}|-1$ should hold. In addition, the rank of the $d\times$ d-dimensional matrix $\mathbf{S}_W$ cannot exceed $L-|\mathcal{C}|$, therefore having insufficient training data, with respect to the input feature vector dimension d, is a potential problem.

Discrete cosine, wavelet and other image transforms – In place of **P** other linear image transforms are also used. Examples include the *Discrete Cosine Transform (DCT)*, the *Discrete Wavelet Transform (DWT)* and the *Hadamard* and *Haar* transforms. Usually separable transforms are used allowing fast implementations when M and N are powers of 2. In this case, the rows of matrix **P** are the image transform matrix rows that maximise the transformed data energy over the training set or that correspond to a priori chosen locations.

2.3.2 Measuring AV synchrony

In the absence of a formal definition of audio-visual synchrony, Hershey and Movellan (1999) first interpreted synchrony as "the degree of mutual information between audio and spatially localised video signals". More recently other definitions for measures of synchrony between any visual stream and an associated audio stream were proposed. In practice, the most appropriate definition of audio-visual synchrony may vary according with the application.

First, let the acoustic signal be described by the feature vector $a_t \in \Re^n$ and the visual signal be described by the feature vector $v_t \in \Re^m$. The set of T audio and visual vectors can therefore be defined by $\mathcal{S} = ((a_1, v_1), \ldots, (a_T, v_T))$, sampled between instants 1 and T. Whereas the components of the audio feature vector can include cepstral coefficients, pitch measurements, or the output of a filter bank, the visual feature vector can be composed Gabor energy coefficients, RGB colour values, etc. The synchrony between both streams can be evaluated by the measure of synchrony between the vector sequences $\mathcal{A} = a_1, \ldots, a_T$ and $\mathcal{V} = v_1, \ldots, v_T$. Two types of measures are possible: generic and specific.

□ Generic Measures

Consider each feature vector in $\mathcal{S}$ to be an independent sample form the joint distribution $p(A, V)$, instead of explicitly modelling temporal dependence in the individual sequences. As previously said, one possible measure is the *mutual information* $\mathcal{I}(A; V)$ between random variables A and V. In practice, the distributions $p(A)$, $p(V)$ and $p(A, V)$ are unknown and some assumptions must be made to overcome this. One possible assumption is to consider *discrete distributions* (Nock *et al.*, 2002). This requires however a preparation phase which constructs codebooks to quantise a_t, v_t and (a_t, v_t) prior to discrete estimation at test time.

Another assumption is to consider *continuous multivariate Gaussian distributions* (Hershey and Movellan, 1999). Unlike discrete distributions, this assumption allows parameter estimation at test time without prior preparation. If we define synchrony at time t as the estimate of mutual information between acoustic and visual components of the process, we then have a measure of synchrony between audiovisual signals. A generalisation of this approached was done by Slaney and Covell (2000). They use canonical correlation analysis to deduce a relationship between the cepstral representation of the audio and the visual information. These methods assume strong parametric assumptions, limiting the capture of more complex dependencies. Moreover, the audio and video joint densities are estimated by training on audio-video sequences. Fisher III and Darrell (2004) proposed a method that does not make use of any previous model training. Their method is based on a probabilistic generation model that is used to learn audio and video linear features that maximise the mutual information between the different modalities.

□ Specific Measures

In this case the measures are application-specific and are determined using some *a priori* knowledge of the environment constraints. An example of this is the usage of face and speech signals, where $\mathcal{A}$ and $\mathcal{V}$ correspond to speech audio and images containing faces. Assume that the word sequence $\mathcal{W}$ spoken in the audio is represented by $\mathcal{A}$. The likelihood $p(\mathcal{S}|\mathcal{W})$ defines a measure of synchrony. In practice $\mathcal{W}$ may be unknown – audio-only speech recognition provides reasonable approximation – as well as $p(\mathcal{S}|\mathcal{W})$. Possible implementations use Hidden Markov Model (HMM) or Time-Delay Neural Network (TDNN) trained on joint audio and visual data.

2.3.3 Applications of AV-based multimodal scene analysis

It is possible to find in the literature multiple applications where the interaction of audio and visual information is explored to achieve better results, compared to approaches based only in one modality. Possible fields of application include automatic multimedia content annotation, surveillance, immersive and interactive environments, ambient intelligence and life logging (e.g. MyLifeBits project – Bell and Gemmell, 2007). We can broadly divide the AV-based multimodal scene analysis applications according to their scope, i.e., oriented toward: semantic analysis, intelligent systems, and human-computer interaction and affective computing.

□ Oriented toward semantic analysis

Content-based video indexing is the task of tagging semantic video units obtained from content analysis to enable convenient and efficient content *browsing*, *retrieval* and *adaptation*. The goal is to automatically extract low- and mid-level features then partially derive or understand the video semantics by analysing and integrating these features. Current work extracts video events and speaker identity at the semantic level, based on the integration of audio and visual knowledge. It is usually focused on specific applications and environments such as movies or news where it is possible to establish some "rules of thumb". More generically, multimodal-based analysis tries to fully exploit multimodal features and inferred concepts. Each concept relies on many different features and associations, drawn from different input media sources – following the paradigm depicted in Figures 2.1 and 2.2.

Semantic learning and inference uses automatic and semi-automatic (e.g. using relevance feedback) approaches to detect and recognise semantically meaningful scenes, objects and events present in the content. This process requires the association of low-level and mid-level automatically extracted features with higher-level semantic concepts. Examples of work in this area include automatic video annotation, video indexing and summarisation (Adams *et al.*, 2003; Li *et al.*, 2004b; Tsekeridou and Pitas, 2001; Wang *et al.*, 2000). Also, the EU-funded COST 292 action [13] has the goal of developing interoperable, semantic-based, multimodal analysis of digital content.

□ Oriented toward intelligent systems

Unlike the previous, in this case data arises from real-world events and processes in a less controlled way. The data is generated mainly in the form of data-streams from sensors of all types and in particular from visual and/or audio sources, with the objective of acquiring knowledge of the environment. The system uses the acquired information to react or learn accordingly. In a simpler system a set of detectable events are previously defined but in more complex systems, through learning algorithms it can learn new events using automatic or semi-automatic approaches.

The scope of this thesis is therefore based on this type of multimodal applications – oriented toward intelligent systems. Related work include automatic meeting logging, surveillance event detection and ambient intelligence. The past EU IST projects M4 [14] and AMI [15] developed systems to enable structuring, browsing and querying of automatically analysed meetings captured in a room equipped with multimodal sensors.

□ Oriented toward HCI and affective computing

This scope of applications is a specialisation of the previous, using a human-centric analysis. Human-computer interaction environments detect and track the user's emotional, motivational, cognitive and task states, and initiates communications based on this knowledge. Sensing and tracking human body motion is a key technology for developing such interfaces. Processing is therefore oriented to human action, including primarily human (based on face and body) detection, tracking and recognition as well as gesture and emotion recognition.

In this case, multimodal analysis can be used, for example, to enhance gesture recognition in human-computer interfaces, behaviour analysis, and emotion recognition. In general, the applications focus on improved and natural interfaces (Kollias and Karpouzis, 2005), art and performance (Camurri *et al.*, 2004), pervasive computing (MIT Oxygen Project [16]), and ambient intelligence.

2.3.4 A multimodal application: speaker identification[§]

In Section 2.3 we noted that speech perception is regarded as an integration of visual and auditory speech signals. As such, it should be instinctive to develop computational models for speech recognition based on both visual and audio analysis and, in fact, many relevant developments were made in this area (Dupont and Luttin, 2000; Kaynak *et al.*, 2004; Potamianos *et al.*, 2003; Tomlinson *et al.*, 1996).

A distinct application, yet closely related, is speaker identification; the goal is to identify the temporal segments that correspond to a given speaker. A simple method using both audio and visual information was proposed in (Lagrange *et al.*, 2007); more details about this method can be found in that article. We assume that only one person is speaking at a time. Moreover, we also assume that one camera and one microphone are

[§]This section is based on the article *Speaker Segmentation of Interviews Using Integrated Video and Audio Change Detections* published in the International Workshop on Content-Based Multimedia Indexing (Lagrange *et al.*, 2007).

used and the speakers are always visible, with a relatively static location. The acoustical cues are processed by an algorithm based on a causal two-stage approach, where both metric and model-based criteria are used for unsupervised speaker turn detection. Concerning the visual cues, we exploit the above described assumptions. This allows us to set up the video acquisition device in fixed and predefined positions and to use a simple segmentation scheme to efficiently detect changes among speakers. Moreover, the number of speakers is usually known beforehand. These constraints render this method useful only to very specific scenarios, but allows to instantiate the concepts presented thus far in this chapter.

□ Acoustic speaker segmentation

The algorithm used for the speaker segmentation based on the acoustic speech signal, assumes no prior knowledge about the number of speakers or their identities. It presumes that the audio input data contains only speech, or that non-speech audio segments were already filtered by a previous audio segmentation module. It was introduced by Lu and Zhang (2002) and a comparison of this method with other speaker segmentation methods was done by Kotti *et al.* (2006).

The algorithm starts by downsampling the input speech audio to 8 kHz, 16 bits mono audio format, and applies a pre-emphasis filter (using a first order FIR filter). The speech stream is then divided into analysis frames of about 16 ms duration, without overlap. From each audio frame 10-order LSP features are extracted. In a first stage, speaker change detection is coarsely performed using a metric-based approach to calculate the Kullback-Leibler divergence shape between consecutive and non-overlapping speech segments. A potential speaker turn point is detected between two consecutive segments whenever their divergence distance is a prominent local maximum. This process is repeated over the incoming audio signal, using the circular buffer for achieving the sliding of the speech segments of about 27 frames at each iteration.

As an attempt to reduce the false alarm rate, Bayesian Information Criterion (BIC) is additionally used in order to validate any potential speaker change point detected at the coarse segmentation step. Since BIC is well known for suffering from insufficient model estimation traits when dealing with small amounts of data (due to just using data from two consecutive speaker segments stored in the circular buffer), as new speech segments are received, the arriving data is used to incrementally update an internal speaker model. When in the presence of a potential speaker change, this will allow better model estimates, potentially increasing the accuracy of the BIC validation.

□ Visual scene segmentation

Most of existing research for the segmentation of video content using visual cues uses intensity or colorimetric changes (Wang *et al.*, 2000). In our work such cues are not

useful as the camera is supposed to be static. A very common approach relies on face detection to identify potential speakers (Rehg *et al.*, 1999). Alternatively, we explore the relative static locations of speakers within the scene avoiding complex face and mouth/lips detection algorithms, limiting to scenarios with only two speakers, such as interviews or lectures. However, the proposed video scene segmentation can easily be generalised to a larger number of speakers, provided that the locations of each speaker are previously given.

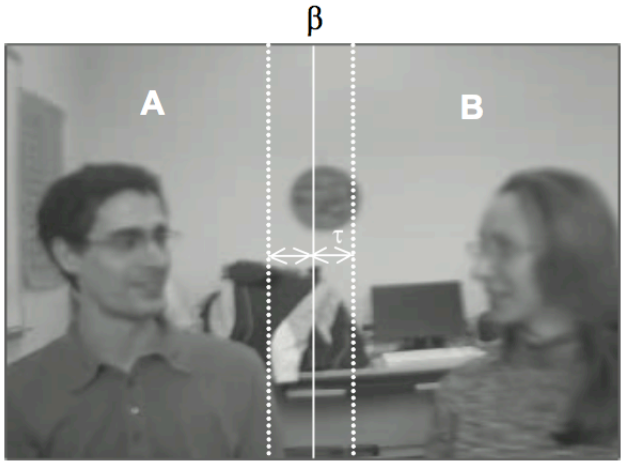


Figure 2.4: *Definition of visual regions where the possible speakers are located.*

In order to identify the person speaking we will be using motion estimation. Moreover, since we are considering only sequences with two persons facing the camera it should be safe to assume that there are two distinct visual regions associated with each person (Figure 2.4). The separation of these regions is defined by a boundary β. For simplicity this is kept as a vertical straight line splitting the image in two halves. It should however be noted that the two regions could be arbitrarily complex and defined, for example by binary masks. This scheme can be generalised to several speakers, by previously selecting several areas of interest. The motion centroid is calculated using the absolute value of each component in each pixel as a weight. To avoid excessive jitter, a median filter of order 5, i.e., the current and the past four estimates, is applied to estimate a more accurate centroid. In a first approach, given the motion centroid for each new frame, a simple classifying algorithm can be applied. It consists of a threshold matching the previously defined boundary but with added hysteresis as shown in Figure 2.4. Instead of marking a frame as a speaker change when the centroid crosses the boundary, it is only marked as such if it crosses a new boundary displaced by a tolerance set, for all sequences, as 5% of the frame width. Even if the speaker is always moving while speaking, the amount of movement depends on each speaker, and may change across time.

□ Multimodal identification algorithm

While the acoustic and video speaker change detectors could be used independently, performing a combination will help improve the performance. We perform "decision in-decision out" fusion of the audio and video cues by combining the speaker changes boundaries detected using the two algorithms presented in the previous sections. The audio and video modalities can be considered to observe the same behaviour of the audiovisual scene, but in a different way. Indeed, people tend to move their bodies, arms and lips before producing any sounds and the first sounds produced are usually non speech vocalisations such as breath, etc. The speech segments happen lastly, usually with half a second delay. The visual change detector is then more likely to be fired before the audio one. Also, this last audio detector is more likely to detect the correct boundary but with a higher false alarm rate due to the presence of non speech sounds and background noise. We aim at designing a causal combination algorithm that takes these two constraints into account. At a given time t advanced at video frame rate (30 fps), we seek for audio and video speaker change boundaries, respectively noted $s_a(k)$ and $s_v(k)$, such that it is below a given threshold δ, set to 1.6 seconds. Only those boundaries are now considered. We then seek for the combination of an audio boundary $s_a(k_{min})$ such that abs$(s_a(k))$ is minimal and a video boundary such that

$$s_v(k) - s_a(k_{min} < \delta/2) \tag{2.1}$$

and

$$\text{abs}(s_v(k) - s_a(k_{min} + \delta/2)) < \text{abs}(s_v(k) - s_a(k_{min} + 1)). \tag{2.2}$$

If those two conditions are met, $s_a(k_{min})$ is used as a combined boundary and the two boundaries are discarded. An example of the resulting boundaries is plotted on Figure 2.5.

□ Evaluation

In order to assess the performance of the proposed system, an audiovisual dataset was created by recording two-speaker conversations using a consumer web camera equipped with a microphone. The dataset comprises 14 audiovisual sequences of 5 different speakers, with a mean duration of 60 seconds. This dataset is divided in two smaller datasets with different scenarios, one formal and the other closer to a real interview setting. In the first scenario, the speakers are asked to alternatively read 8 poetry sentences. The total number of speaker changes for each sequence is therefore 15, plus the 2 edges corresponding to the start and end of the speech sequence. The speaker segments in this dataset have a mean duration of 4.14 seconds, with a standard deviation of 1.21 seconds. In the second one, the speakers are asked to improvise

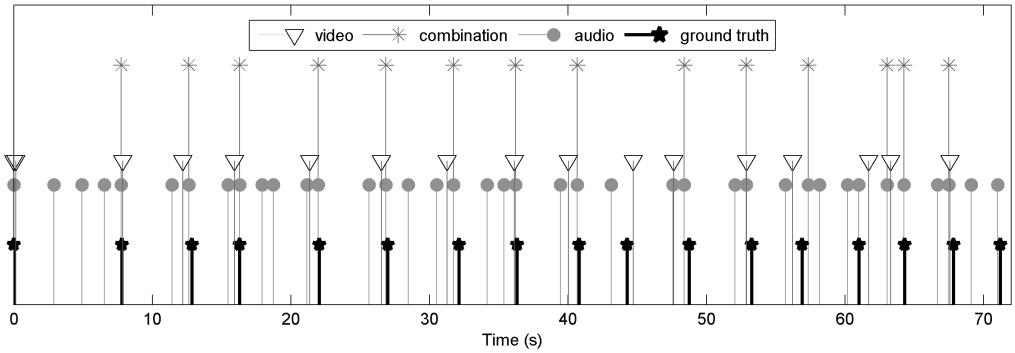


Figure 2.5: *Example of the result obtained with combined speaker boundaries.*

an interview without any prior preparation. The total number of speaker changes for each sequence is around 8, plus the 2 edges. The speaker segments in this dataset have a mean duration of 3.53 seconds, with a standard deviation of 6.68 seconds. We are interested in detecting speaker turn points, based on the acoustical speaker changes, so the boundaries that we are interested in correspond to those of the audio. The two datasets have therefore been manually labelled using the audio stream as the reference.

		FAR	MDR	recall	precision	F1
poetry	audio	63.52 (4.43)	7.63 (10.18)	92.37 (10.18)	34.6 (5.36)	50.21 (6.75)
	video	43.62 (6.41)	43.82 (14.64)	56.18 (14.64)	41.94 (10.57)	47.83 (11.83)
	a+v	36.68 (6.84)	30.02 (7.65)	69.98 (7.65)	54.72 (8.06)	61.27 (6.95)
interview	audio	53.38 (12.18)	20.71 (24.06)	79.29 (24.06)	39.63 (10.16)	51.4 (12.63)
	video	40.3 (15.89)	24.54 (22.06)	75.46 (22.06)	52.18 (12.92)	59.23 (12.83)
	a+v	36.26 (10.77)	35.26 (24.99)	64.74 (24.99)	51.84 (12.08)	56.04 (15.76)

Table 2.1: *Performance results using different combinations of detectors.*

The experimental results for the poetry reading dataset are summarised on Table 2.1. The audio segmentation algorithm performs as expected, with a low Miss Detection Rate (MDR) and a high False Acceptance Rate (FAR), leading to an F1-Measure around 50% (refer to Appendix B for details on these evaluation metrics). Concerning the use of the video cues, two detectors are considered. The first uses a constant threshold (presented as "video" in the tables). Their performance characteristics are of a balanced FAR and MDR. The adaptive scheme outperforms the constant one and leads to an improvement of 7% in terms of F1-Measure. Compared to the respective performance of the two detectors, the combined one significantly decreases the FAR, while averaging the MDR, leading to an improvement of 20% over the audio performance

in terms of F1-Measure. In this formal setting, the proposed detection scheme performs reasonably well. For the informal interview, the audio detector achieves similar performances with a higher MDR rate.

From this, we can conclude that the proposed combination scheme takes advantage of the complementary characteristics of the performance statistics of the two combined detectors. Despite that, and due to the simplicity of the method, the performance results are only fair. We perform a late fusion of audiovisual cues by combining two speaker change detectors, but alternatively better results can be achieved with an early fusion approach. Early fusion, or signal-level fusion, could be performed measuring for example the synchrony of the audio and visual signals (refer to Section 2.3.2). Cutler and Davis (2000a) propose exploiting the correlation between video and audio, Fisher III and Darrell (2004) perform speaker motion and speech association using signal-level fusion, and Monaci *et al.* (2006) consider that the primitive used to represent visual information is correlated with the audio signal over an observation time slot, if the scalar product between the corresponding activation vectors is large.

Other localisation applications using AV combined analysis are possible. Kidron *et al.* (2007) localise visual events associated with sound. Their method is based on canonical correlation analysis, where inherent ill-posedness is removed by exploiting sparsity of cross-modal events, to detect the pixels that are associated with sound, while filtering out other dynamic pixels. Bhanu and Zou (2004) propose a method to detect moving objects in a cluttered environment. Their approach is based on a TDNN to fuse the audio and video data at the feature level for detecting the walker with multiple persons in the scene.

2.4 Summary

The concept of multimodal scene analysis relies on the premiss that, if adequate methods are applied, additional information can be extracted from a given scene when using a combination of different modalities. Nevertheless, some challenges need to be overcome, namely, know which modalities are really relevant and how can the multiple modalities be combined. It is possible to perform this combination at different levels; before or after performing classification, commonly designated early and late fusion.

The concepts of Computer Vision (CV) and CASA are largely based on the mechanisms of the human perception system. It is commonly known that the human perception system relies on interactions between the different senses. However, with few exceptions both CV and CASA followed non-crossing paths despite this common ground. In this thesis we focus on the automatic description of scenes essentially from AV information. Despite the less-than-desirable dissemination between both research communities, in the recent years the combination of AV techniques has garnered some attention. One of the most prominent areas is speaker recognition, but also localisation of a speaker or other audio sources, emotion recognition, among others, have been very

active. We presented a simple example of a speaker segmentation application combining an acoustic speaker segmentation method and a motion-based visual attention method to perform speaker localisation. A late fusion is performed on the decisions outputted by both methods to generate the combined decision. An early fusion approach is also possible, applying a common decision framework with the inputs from both modalities being considered simultaneously.

In the next chapters we approach the description of real-world scenes by considering two semantic entities: *objects* and *events*. It will be discussed how these entities can be detected and assigned a meaning. In both cases an AV-based multimodal scene analysis is performed, however, for object detection, this analysis is not based in a tight integration of audio and visual cues.

3 Object detection

Before associating higher-level meaning to a audiovisual scene it is important to identify parts that are indeed relevant. The first step for interpreting a scene is thus to decompose it into meaningful objects. This is a complex process that may comprise different ways of accomplishing a significant decomposition. In particular, both bottom-up and top-down information can be used. An example of the former is the use of a saliency characteristic like colour for visual scenes and intensity for auditory scenes, to distinguish relevant objects. In contrast, examples of the use of top-down information include typical human characteristics to find persons in a scene, or rhythmic information to detect music.

3.1 Overview

An important initial step toward efficient audiovisual processing is the separation of a complex scene in its composing objects. Each object can then be separately analysed, identified, or classified. However, detecting or segmenting objects is a very complex problem since, to attain results comparable to human-performed segmentations, semantic or high-level a priori knowledge is also required.

Let us first consider an important perception mechanism, visual attention (refer to Wolfe, 2000). When looking, for example, to a field and our attention is drawn to a flower, we may be relying on our bottom-up mechanism of attention. In this case the attention was not dependent of a priori knowledge about the flower and its characteristics, it could be triggered solely by the fact that it is more visually salient then the rest of the field. Conversely, when we look for a flower in the field, we base this on a representation of the flower in a top-down fashion. A limited set of basic features were found to be used by the bottom-up mechanisms of attention. Neisser (1967) introduced the idea of a *preattentive* stage of visual processing, i.e., vision before attention. At this stage, everything could be processed at once across the entire visual field. Evidence suggests that visual attention can be guided by the preattentive processing of colour, orientation, motion, size, curvature, various cues to depth, and several aspects of form. The case for preattentive processing of more complex properties like object shape, letters, or faces is weak though there is some evidence for efficient search. In our work the preattentive stage will be modelled by a set of algorithms for object detection. Our goal is to define areas of the field of view that may contain relevant areas that will be further analysed and recognised. Paraphrasing Wolfe (2000), rather than saying that attention (object detection) somehow identifies an object, we would say that attention enables object recognition processes to work on a single item at a time. With auditory

attention a similar rationale can be followed. In this case the features most relevant to detect salient auditory elements are intensity, feature contrast, and temporal contrast (Kayser *et al.*, 2005).

This chapter presents current methods to detect both visual and auditory objects[1]. The goal is to separate both in space and in time each object and associate to them a label. This label distinguishes objects from one another but does not associate any semantic knowledge.

The different nature of audio and visual signals imply different approaches for analysing each one of them. The detection of visual objects comprises segmentation which partitions a given image, i.e. a frame in a video sequence, into coherent objects. On the other hand, detecting audio objects consists of separating different objects in two dimensions: separate simultaneous audio objects – a process typically called sound source separation – and define time intervals in which a given audio object is detected. Both types of detection commonly take very different approaches, since the latter is a simpler process to characterise. The spatial detection of sound objects – or sound source localisation – can be achieved using both visual and audio cues by determining the synchronisation between them or can more generically rely on multiple microphones with known responses and configuration. We will only discuss the former approach.

3.2 Visual object segmentation

Real-world scenes are composed of a very large number of visual patterns. In general, these patterns form objects that are associated to a concept (e.g. person, car, bicycle, cup, cat, ...). As discussed previously, without knowing prior information about the objects, computationally grouping visual patterns and associating high-level concepts is often not possible. In fact, it is known that humans may segment an image differently. Particularly, the same scene may be distinctively perceived by different persons, or may attend to different parts of the scene, resulting in image segmentations of different granularities. Additionally, the analysis of objects in images is dependent on how we can distinguish between the objects of interest (foreground) and "the others" (background). In Figure 3.1 two examples of segmentations of a manually partitioned scene is shown. While the segmentation in the left comprises a larger number of partitions (objects), the segmentation in the right consists solely of the person walking. Both segmentations can be considered valid partitions of the original scene, but which one is more adequate? The degree of "adequateness" depends on how we define background and foreground. Numerous other valid segmentations are also possible if we consider more or less detailed segmentations of the cars and the houses. For the type

[1] In the literature related to auditory analysis, the term *source* is often used instead of *object* using only content-level knowledge. We opted for the latter to strengthen the parallel between audio and visual analysis.

of scenarios we are interested, the most adequate segmentation is the one in the right, since we consider as background the elements that remain "static" or that suffer gradual changes over time.

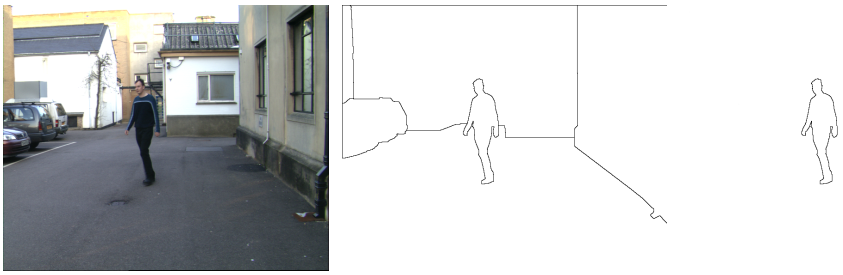


Figure 3.1: *Examples of possible segmentations for a given scene. From left to right, original image, more detailed segmentation, less detailed segmentation (from Cardoso, 2006).*

In general, the segmentation task can be characterised by the partitioning of an image into different meaningful regions with homogeneous characteristics, using discontinuities or similarities of image components. A wide range of computational vision problems can make use of segmented images if these segmentations can be reliably and efficiently estimated. However, there is no universally applicable segmentation technique that will work for all images, and no segmentation technique is perfect – Figure 3.2 shows the results of three segmentation algorithms applied to the scene depicted in Figure 3.1.

Figure 3.2: *Examples of segmentations obtained using different well-known algorithms. From left to right the algorithms are JSEG (Deng and Manjunath, 2001), Mean shift (Comaniciu and Meer, 2002), and NCut (Shi and Malik, 2000).*

For typical real-time applications oriented to the analysis of visual scenes in order to identify events and actions – such as intelligent surveillance systems and human-machine interface systems – simplifications are needed. For these, motion is a key factor helping the segmentation process. Objects that are moving or performing some

action, are considered the foreground, and need to be separated from the other elements in the scene, the background.

Probably due to its simplicity, the most common approach for discriminating a moving object from the background is *background subtraction*. The rationale is the subtraction of the current image from a reference image, which is somehow acquired in a step prior to subtraction. More generically, a *background model* is defined which is usually based in a priori knowledge. Non-changing segments of the image are considered as being part of the background, whereas the foreground consists of the changing segments – including moving and new objects.

3.2.1 Background modelling and subtraction

If the background model is not modelled or updated adequately, background subtraction can be highly susceptible to environment conditions like illumination changes. For example, a straightforward way of acquiring a reference image would be using the previous "history" by obtaining a *background model* based on the statistical representation of the previous N frames, for example a pixel-wise average image. After estimating the reference image, the segmentation can be obtained from an efficient thresholded subtraction operation. This simple approach, although efficient, may not perform well in real-world, non-controlled environments. Changes in illumination conditions and dynamic behaviour in the background may cause unacceptable rates of false positives. To achieve robust background modelling, techniques that can better adapt to dynamic behaviour are needed. Ideally, the performance should not depend on the camera placement, nor should it be sensible to what happens in its visual field or to lighting effects. It should also be capable of dealing with movement through cluttered areas, objects overlapping in the visual field, shadows, lighting changes, effects of moving objects in the scene, slow-moving objects and objects being introduced or removed from the scene. However, it is also important to stress that this operation is often required to perform as fast as possible, since it is usually the first step in the processing chain. Overly complex modelling schemes may reveal themselves unfeasible despite performing at low error rates.

predictive methods

non-predictive methods

A possible classification of existing algorithms for background modelling divides them in *predictive* and *non-predictive* methods. Predictive methods model the scene as a time series and develop a dynamic model to recover the current input based on past observations. Usually Kalman filters (Koller *et al.*, 1994; Ridder *et al.*, 1995; Toyama *et al.*, 1999) are employed to update slow and gradual changes in the background. In general, these methods are mainly applicable to backgrounds consisting of stationary objects. On the other hand, non-predictive methods for background modelling do not consider the order of input observations and build a probabilistic representation (p.d.f.) of the observations at a particular pixel. These are by far the most common methods and many different proposals and adaptations can be found in the literature. In (Wren *et al.*, 1997) a unimodal distribution was proposed – a Gaussian distribution is used to

model the background. If each pixel resulted from a particular surface under particular lighting, a single Gaussian would be sufficient to model the pixel value accounting for acquisition noise. Moreover, if only lighting changed over time, a single, adaptive Gaussian model per pixel would be sufficient. In practice this does not happen and for that purpose a model based in the mixture of a fixed number of Gaussians distributions has been frequently used (Lee, 2005; Stauffer and Grimson, 1999). In (Elgammal *et al.*, 2000), a non-parametric model is proposed, where a kernel-based function is used to represent each pixel's colour distribution. The kernel-based distribution is a generalisation of the mixture of Gaussians which does not require parameter estimation. In (Haritaoglu *et al.*, 2000), a similar approach is followed, where the distribution of temporal variations in colour at each pixel is used to model the background. Motion-based adaptive kernel density estimation is also proposed in (Mittal and Paragios, 2004). More recent approaches to background modelling include the principal features approximation (Li *et al.*, 2004a) that considers only the more relevant features to create a model and the mean-shift method (Han *et al.*, 2004; Piccardi and Jan, 2004), which was also previously applied to image segmentation.

All of the previous methods are based in pixel-wise background modelling. Other techniques combine temporal and spatial modelling. In (Paragios and Ramesh, 2001), a mixture model (Gaussians or Laplacians) is used to represent the distributions of background differences for static background points. A Markov Random Field (MRF) model incorporates the spatial coherence for robust foreground segmentation. Another approach to detect moving objects is to extract groups of motion, either by accumulating consistent flows in terms of direction over successive frames (Wixson, 2000) or by using layered approaches to fit a collection of motion models to the image data (Pundlik and Birchfield, 2006; Wang and Adelson, 1994). Recently, an approach that defines foreground objects as clusters of pixels salient with respect to both motion and colour was presented (Bugeau and Pérez, 2007).

Additionally other authors propose a different approach that employs information about the object's structure to segment them. In (Dubuisson and Jain, 1995), the contour of moving objects is estimated by fusing motion with colour segmentation and edge detection. Active contours were employed in (Malladi *et al.*, 1995) and, besides edge information, prior models on the image intensity values inside and outside the contour were also proposed (Chakraborty and Duncan, 1999). These methods estimate the object's contour by minimising a global cost function and, although robust, the estimates are achieved at a high computational cost. The main drawbacks are the complexity and the need to generally assume some prior knowledge in order to avoid ill-posed problems. However, this knowledge is not always available. Moreover, the priors assumed by the techniques can be unsuitable to the problem in hands. More recent work in this line of thought includes the use of layered models (Jojic and Frey, 2001) and rigid objects modelling (Aguiar and Moura, 2005).

A summary of background modelling methods for foreground segmentation is presented in Table 3.1.

	Method	References
pixel-wise predictive	Kalman filters	Koller *et al.* (1994), Ridder *et al.* (1995) Toyama *et al.* (1999)
pixel-wise non-predictive	unimodal Gaussian	Wren *et al.* (1997)
	Mixture of Gaussians (MoG)	Stauffer and Grimson (1999), Lee (2005)
	Kernel Density Estimation (KDE)	Elgammal *et al.* (2000), Haritaoglu *et al.* (2000)
	adaptive KDE	Mittal and Paragios (2004)
	principal features	Li *et al.* (2004a)
	mean shift	Han *et al.* (2004), Piccardi and Jan (2004)
combining other features	spatial and temporal modelling	Paragios and Ramesh (2001)
	accumulation of consistent flows	Wang and Adelson (1994), Wixson (2000), Pundlik and Birchfield (2006)
	clusters of motion and colour	(Bugeau and Pérez, 2007)
other approaches	active contours	Malladi *et al.* (1995)
	fusion of motion with colour segmentation and contours	Dubuisson and Jain (1995)
	contours with prior models	Chakraborty and Duncan (1999)
	layered models	Jojic and Frey (2001)
	rigid objects modelling	Aguiar and Moura (2005)

Table 3.1: *Summary of the background modelling methods.*

3.2.2 Comparative evaluation of background modelling methods[§]

In this section we present a small overview of some of background and modelling algorithms mentioned in the previous section. A representative set of state of the art techniques were implemented and tested. The Running Average background modelling algorithm is used as a base performance index. To compare the segmentation results of each technique, we used an objective metric detailed in (Cardoso *et al.*, 2009) and summarised in Appendix B.2.

[§]This section is based on the article *Cascaded change detection for foreground segmentation* published in the Proceedings of IEEE Winter Vision Meeting – Motion and Video Computing (Teixeira and Corte-Real, 2007)

□ Running Average (RAvg)

The background can be modelled as the average of the previous frames but, in order to avoid expensive memory requirements, this average is approximated by an adaptive filter with a learning rate α. Each background pixel value at position (i, j) and time instant t is given by:

$$B_{(i,j)}(t) = \alpha I_{(i,j)}(t) + (1 - \alpha) B_{(i,j)}(t - 1).$$

Foreground is then estimated using a thresholded subtraction of the current frame and the estimated background. This technique is probably the most naive but has a very simple and very fast implementation. The results are therefore far from good in particular with complex backgrounds. Since we are considering only a static representation of the background to perform the subtraction, whenever some kind of dynamic behaviour in the background happens, it will be incorrectly classified as foreground. Nevertheless, the running average should represent the base performance for these types of algorithms. Each pixel is classified as foreground if its value exceeds the estimated background by a threshold T_{RAvg} of 15.

□ Mixture of Gaussians (MoG)

Instead of estimating the background representation directly, another and more effective approach is to estimate a background model that can predict the behaviour in each pixel, using the pixel's "history". By estimating the background probability density function (p.d.f.) we are able to do just that. Assuming that any structural changes affecting the value of the pixel are caused by several processes, each modelled by a Gaussian, we can therefore define the probability of observing its value as:

$$P(v_t) = \sum_{k=1}^{K} P(G_k) P(v_t | G_k) = \sum_{k=1}^{K} \omega_k \cdot \eta(v_t, \mu_k, \sigma_k). \tag{3.1}$$

where G_k is the k-th Gaussian of K distributions, ω_k, μ_k and σ_k are, respectively, an estimate of the weight, the mean value and the variance of the k-th Gaussian in the mixture; η is the normal density function. Moreover, it can be easily shown (Lee, 2005) that, given the current colour vector v_t in a pixel, the probability that the pixel belongs to the background is:

$$P(B | v_t) = \frac{\sum_{k=1}^{K} P(v_t | G_k) P(G_k) P(B | G_k)}{\sum_{k=1}^{K} P(v_t | G_k) P(G_k)}. \tag{3.2}$$

If $P(B|v_t) > T_{MoG}$ the pixel is estimated as being part of the background. However, two density estimation problems are left to resolve: firstly, estimating the distribu-

tion of all observations, within a period of time, at each pixel location using the Gaussian mixture in equation (3.1), which provides estimates of both $P(G_k)$ and $P(\upsilon_t|G_k)$; and secondly, evaluating how likely each Gaussian in the mixture represents the background, i.e., $P(B|G_k)$. To accomplish the first estimation, Stauffer and Grimson (1999) proposed an online K-means approximation in order to model pixel variation over time by a mixture of Gaussians, as given by equation (3.2). It uses a fixed learning rate to update each Gaussian's parameters over time and a Gaussian substitution algorithm whenever no match is possible. However, using a fixed learning rate can often result in slow convergence. Following the same rationale, and in order to improve the convergence speed, Lee (2005) proposed an adaptive learning rate schedule for each Gaussian defined by a parameter α; we will be using this last approach. The estimation of $P(B|G_k)$ is based on application-specific heuristics (Lee, 2005).

The background image representation can be defined as the expected value of the background process. Thus, the background pixel at (i, j) and time t is defined by $E[\upsilon_{i,j,t}|B]$ which is evaluated by a weighted average of the Gaussian means.

$$E[\upsilon_{i,j,t}|B] = \frac{\sum_{k=1}^{K} \mu_k P(B|G_k)P(G_k)}{\sum_{k=1}^{K} P(B|G_k)P(G_k)} \tag{3.3}$$

The tests with MoG were done with the following parameters: $K = 3$, $\alpha = 0.005$ and $T_{MoG} = 0.05$.

□ Kernel Density Estimation (KDE)

It is possible to approximate each background's pixel p.d.f. by the histogram of the most recent values classified as background. This approach has however some problems namely, being the histogram of a step function, the p.d.f. modelling can reveal itself erroneous. A non-parametric model based on KDE was proposed by Elgammal *et al.* (2000). KDE guarantees a smoothed, continuous representation of the histogram. The background p.d.f. is given by Equation (3.4) as a sum of Gaussian kernels centred in the most recent N background values.

$$P(\upsilon_t) = \frac{1}{N} \sum_{k=1}^{N} \eta(\upsilon_t - \upsilon_k, \Sigma_k) \tag{3.4}$$

Even if background values are not known, unclassified sample data can be used instead. This inaccuracy will be recovered along model updates. Given equation 3.4, the pixel with the colour vector υ_t is classified as foreground if $P(\upsilon_t) < T_{KDE}$, where T_{KDE} is a global threshold. An important issue in KDE is the estimation of Σ_k – the kernel bandwidth. For simplicity, a diagonal matrix is considered and each variance is esti-

mated in the time domain by analysing the set of differences between two consecutive values

Model update consists in selectively updating the vector of the previous N background values. The model proposed by Elgammal *et al.* (2000) also considers the use of two concurrent similar models, one for long-term and the other for short-term memory. In addition, spatial correlation is taken into consideration by the model. However, we will not be considering both these modifications since we are comparing pixel-wise modelling techniques. These types of considerations are transversal to all algorithms we are evaluating. The tests executed with this algorithm used the following parameters: $N = 50$ and $T_{KDE} = 10^{-6}$.

□ Principal Features (PF)

More recently, other approaches were proposed to estimate the background p.d.f.. For instance, the background can be represented at each pixel by the most frequent features, or *principal features* (Li *et al.*, 2004a).

The classification is done using a Bayesian framework, and it is shown that a pixel represented by $\boldsymbol{v}$ is classified as belonging to the background if:

$$2P(\boldsymbol{v}|B)P(B) > P(\boldsymbol{v}). \tag{3.5}$$

Otherwise, it is classified as belonging to the foreground. We need however to know *a* priori or estimate the probabilities $P(\boldsymbol{v}|B)$, $P(B)$ and $P(\boldsymbol{v})$. As stated previously, one way to estimate these probabilities is to use a histogram of features. The important contribution of Li *et al.* (2004a) is that they propose that these probabilities can be estimated using solely the most representative features in the histogram, given that these can represent the background effectively. Therefore, for a proper selection of features, there would be a small value N of features (the principal features) that can approximate well the background by $\sum_{k=1}^{N} P(v_k|B)$.

The learning and update process is done using a table of statistics for the possible principal features of the background. The update of estimated probabilities through time is done using a simple adaptive filter according to the type of change that occurred (gradual or "once-off").

Note also that the algorithm proposed by Li *et al.* (2004a) uses several types of features, namely: spectral, spatial and temporal features. In our implementation we used only the spectral features, i.e. colour information. Otherwise, the results for this algorithm would be biased. The tests executed with principal features used the following parameters: $\alpha = \beta = 0.04$ (rate for probability and background learning, respectively), $M = 50$, $N = 20$ and $M1 = 0.75$ (for "once-off" detection).

□ Mean Shift (MS)

Finally, yet another way of estimating the background p.d.f. is to use the mean-shift (Han *et al.*, 2004; Piccardi and Jan, 2004) which is an iterative gradient-ascent method to detect modes of a multimodal distribution and their covariance matrix. The only parameter needed is the bandwidth range that is application-specific. The mean shift algorithm states that, for a given set of points $x_i, i = 1, \ldots, n$, the mean shift vector in the one-dimensional case can be expressed as:

$$m(x) = \frac{\sum_{i=1}^{n} x_i g\left(\frac{x-x_i}{h}\right)^2}{\sum_{i=1}^{n} g\left(\frac{x-x_i}{h}\right)^2} - x.$$

where x is an arbitrary point in the data space, h is a positive value called the analysis bandwidth and $g(u)$ is a bounded support function, first derivative of another bounded support function, $k(u)$, or kernel profile. It can be proven that, for a kernel with a convex and monotonically decreasing profile, the iterative procedure $x^{l+1} = m(x^l) + x^l$ converges. Piccardi and Jan (2004) introduced some optimisations in order to reduce processing time, namely histogram-base mean shift computation – the mean shift vector is calculated with respect to the number N of histogram bins. This was the method implemented and tested.

All points $x_i, i = 1, \ldots, t_u$ belonging to a mode will converge to the same point, the mode centre, or mean μ_u. Moreover, if we assume Gaussian modes, for each feature – in this case the components of the colour vector $\boldsymbol{v}$ – the p.d.f consists of a weighted sum of the U modes modelled by a Gaussian distribution. A threshold test can simply be applied to the estimated p.d.f.:

$$\sum_{u=1}^{U} \prod_{f=1}^{F} \omega_{(u,f)} \eta(x_f, \mu_{(u,f)}, \sigma^2_{(u,f)}) < T_{MS}. \tag{3.6}$$

Note that we are assuming that the features $f, f = 1, \ldots, F$ are independent. The weights $\omega_{(u,f)}$ also need to be estimated, and are generally defined by heuristics (Piccardi and Jan, 2004). If the probability estimated for a given pixel value $\boldsymbol{v}$ is smaller than the threshold T, the pixel is classified as foreground. The mean shift algorithm was tested with the parameters: $N = 50$, $h = 3$ and $T_{MS} = 10^{-20}$.

□ Comparative evaluation

A comparative study of background modelling techniques was previously presented in (Piccardi, 2004), however this study consists of a theoretical comparison of several algorithms and no qualitative tests are presented. In order to get a better understanding of the algorithms, we tested them in several sequences. The results for some of the

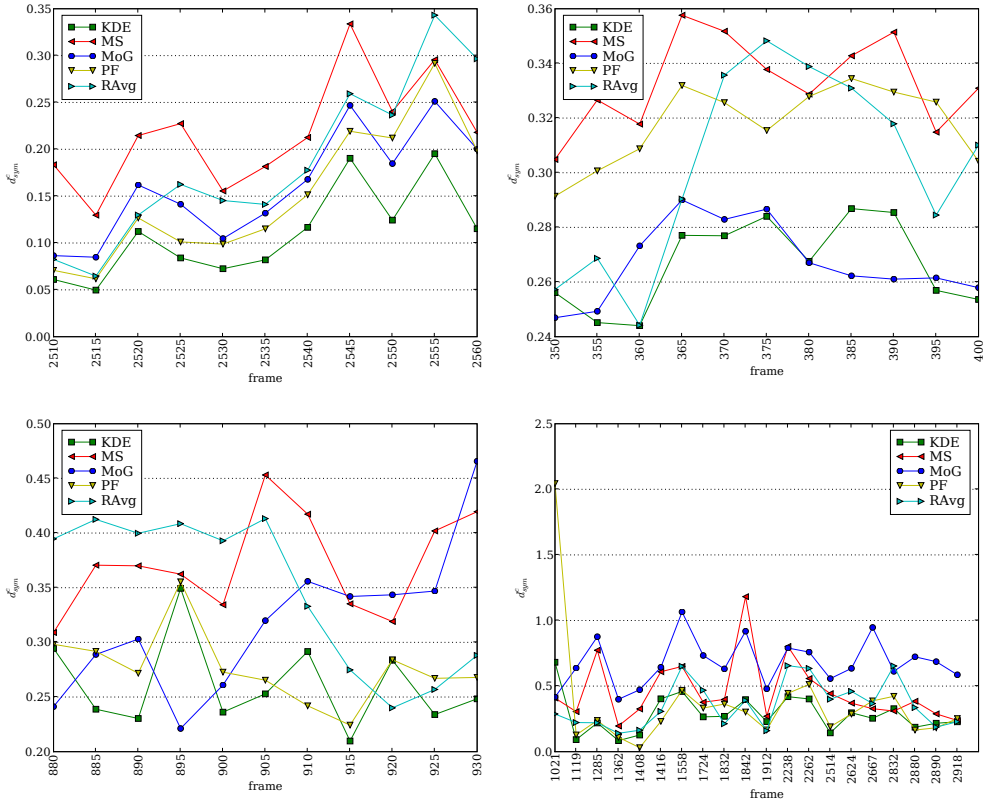


Figure 3.3: *Evolution of d^c_{sym} using different methods for background modelling. From left to right and top to bottom, results for the sequences SW, SH, OD and BR are presented. We compare implementations based on kernel density estimation (KDE), mean-shift (MS), mixture of Gaussians (MoG), principal features (PF) and running average (RAvg).*

test set sequences are presented next – refer to appendix A.1 for more details on the test sequences. All tests were executed using only colour vectors as features; the YUV colour space was used.

In Appendix B.2 we introduce a cost-based partition distance d^c_{sym} between segmentation masks to evaluate the foreground segmentation methods.

Table 3.2 summarises the results obtained for each algorithm in each sequence; for each algorithm-sequence combination the average distance to the "ground-truth" and the throughput in frames per second (fps) are presented. No post-processing was employed on each algorithm's output segmentations. Figure 3.3 shows the evolution of the distance over time in the SW, SH, OD and BR sequences. All algorithms were tested in a Pentium 4 3.4GHz with 1GB of RAM.

Results show that KDE perform better than the other methods. The principal features

Table 3.2: *Average d^c_{sym} and frames per second for each method over the evaluated frames of each sequence.*

		SW	SH	OD	BR	FT
RAvg	d^c_{sym}	0.185	0.302	0.347	0.357	0.336
	fps	134	152	156	571	526
MoG	d^c_{sym}	0.160	0.267	0.317	0.678	0.307
	fps	6.7	6.1	6.7	31.5	29.2
KDE	d^c_{sym}	0.109	0.267	0.261	0.285	0.093
	fps	1.3	1.1	1.4	5.6	7.5
PF	d^c_{sym}	0.150	0.318	0.276	0.362	0.126
	fps	2.9	2.2	2.9	13.7	14.7
MS	d^c_{sym}	0.218	0.333	0.372	0.460	0.157
	fps	0.04	0.04	0.05	0.2	0.2

and the mixture of Gaussians modelling methods follow closely. Note that the results obtained with MoG in the BR sequence are severely degraded by the initial state which includes foreground objects. Since the analysed time window is not sufficient for the model to recover, the results are continuously affected by it. Also, note that the mean shift approach processing time is extremely high, invalidating its use for real-time applications. On the other hand the MoG method has a considerably better processing frame rate than the other methods and is therefore better suited for real-time applications.

In summary, despite having the less average distance to the "ground-truth", KDE's throughput makes it less interesting than MoG, which has the best relation performance/complexity. Moreover, the segmentations produced by the MoG method are prone to be improved even further since they are less fragmented than the ones produced by the other methods – this will become more evident in section 3.3.5.

3.3 An object segmentation algorithm using cascaded change detection[§]

Consider a typical visual surveillance scene being capture by a static camera; most of the background pixel values change slowly in time and can be caused by phenomena of different nature. If it is possible to model these independently, better results can be achieved. Before detailing the proposed algorithm let us first consider the following:

[§]This section is based on the article *Object Segmentation Using Background Modelling and Cascaded Change Detection* published in the Journal of Multimedia (Teixeira *et al.*, 2007)

when no structural change occurs, the difference between a pixel value, represented by a colour vector $\boldsymbol{v}$, in the current frame $\mathcal{F}_c$ and a reference frame $\mathcal{F}_r$ can essentially result from two factors – illumination variation or noise. Illumination variation can be accounted by a positive multiplicative factor k which modulates the signal, while noise can be accounted by a superimposed vector $\boldsymbol{v}_N$, modelled by a Gaussian or Laplacian distribution – Figure 3.4 represents this in a two-dimensional space.

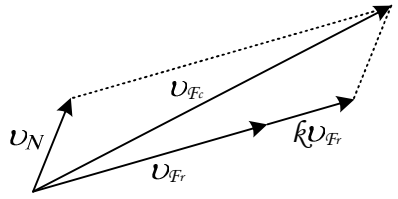


Figure 3.4: *Effects of illumination variation and noise over a reference colour vector. The latter is modelled by a positive multiplicative factor which modulates the signal, while the former is modelled by a superimposed vector, modelled by a Gaussian or Laplacian distribution.*

Considering that the sophomore cause can be successfully eliminated, we are left with the first. Hence, when a pixel change results solely from illumination variation, its colour vector is necessarily collinear with the reference colour vector and a simple test can be used to identify illumination variation-induced changes. Therefore, we will first address how we can effectively remove typical noise introduced by the capture process in order to guarantee colour vector collinearity.

3.3.1 Identification of noise-induced changes

Assuming that we know the reference frame $\mathcal{F}_r$, we will first address how we can effectively remove typical noise introduced by the capture process. For that purpose we use a method proposed by Aach *et al.* (1993) and previously used for simple background subtraction (Cavallaro and Ebrahimi, 2001). It states that it is possible to assess what is the probability that a value at a given position, in a given image, is due to noise instead of other causes when compared to another image. It is assumed that the additive noise affecting each image results from a Gaussian process with mean μ_N and standard deviation σ_N. Also, noise affecting successive images in the sequence is considered as uncorrelated. The standard deviation σ_N can be obtained by computing the statistics of the difference $d_{(i,j)}$ for each pixel (i, j) between the reference image and the current image. Now, consider a window W^n containing n pixels around the pixel under evaluation with $\Delta^2_{(i,j)} = \sum_{(k,l)\in W^n_{i,j}} d^2_{(k,l)}$. It can be shown that the corresponding random variable Δ^2 follows a χ^2 distribution. Given the hypothesis H_0 that $\Delta^2_{(i,j)}$ results from noise and not from other factor, the probability that hypothesis H_0 is satisfied is given

by equation (3.7).

$$P(\Delta^2 > \Delta^2_{(i,j)}|H_0) = \frac{\Gamma(\frac{n}{2}, \frac{\Delta^2_{(i,j)}}{2\sigma_c^2})}{\Gamma(\frac{n}{2})} \tag{3.7}$$

with $\sigma_c^2 = 2\sigma_N^2$ and where $\Gamma(n/2)$ is the Gamma function. The choice for the window size n must take into consideration the trade-off between noise sensitivity and foreground edge definition. Nevertheless, all experiments were performed using a window size of $n = 25$. When the estimated probability in equation (3.7) is smaller than a threshold T_N we consider that H_0 is not satisfied at the pixel position (i, j).

Whereas for pixels that validate the hypothesis H_0 we guarantee that changes were originated solely by camera noise, for others we can safely assume that the effect of noise is negligible when compared to any other change. In other words, if a pixel's colour vector is being modified by illumination variation and no structural change, we have $\boldsymbol{v}_{\mathcal{F}_c} \simeq k\boldsymbol{v}_{\mathcal{F}_r}$.

This test defines a first set of pixels that can potentially be part of the foreground because all pixels that satisfy H_0 are necessarily part of the background and are marked as such for the current frame; all others need further analysis.

3.3.2 Identification of illumination variation-induced changes

After discarding noise-induced changes, a simple collinearity test is performed. As previously stated, with this test any modification introduced by illumination variation is discarded. The test consists in evaluating the angle between the current pixel colour vector $\boldsymbol{v}^c$ and the reference colour vector $\boldsymbol{v}^r$.

$$\cos\theta = \frac{\boldsymbol{v}^c \cdot \boldsymbol{v}^r}{\|\boldsymbol{v}^c\| \, \|\boldsymbol{v}^r\|} \tag{3.8}$$

If $\cos\theta$ is smaller than a threshold T_I very close to 1, the vectors are not considered to be collinear and the test is not validated. In practice, this test can become unstable and to overcome this problem we consider a second threshold in the noise-identification test; the second threshold usually is much smaller than T_N. The identification of illumination variation-induced changes is therefore only applied to pixels that fall between both these thresholds.

The set of potential foreground pixels is refined by marking as background the pixels that validate this second test. Pixels that have a probability less than the second threshold in the previous test are maintained as foreground.

3.3.3 Identification of dynamic background behaviour

The final estimation of whether a pixel is part of the background can be performed only in the pixels that do not validate the statistical (3.7) nor the collinearity (3.8) tests, resulting in a considerable reduction in execution time of the algorithm. This is especially true for typical surveillance streams where the relevant moving objects occupy a small fraction of the entire field of view. Another positive effect is the drastic reduction of small artefacts which often need to be removed in typical modelling by applying morphological filtering (e.g. connected operators).

To model and identify the dynamic background behaviour we will be using pixel-wise background estimation approaches, namely the ones presented in section 3.2.2. The head-to-head comparison revealed that the Mixture of Gaussians (MoG) modelling approach had the best performance/efficiency relation. We will therefore be using mainly MoG to identify dynamic background behaviour. In this case, if the probability defined by equation (3.2) for a given pixel is larger than a threshold T_S, the pixel is considered to be part of the background. Also, background representation defined by the MoG model and described by (3.3) is used as the reference image $\mathcal{F}_r$. Figure 3.5(a) summarises the algorithm in pseudo-code, called cascaded mixture of Gaussians (CMoG).

```
Input: α, K, T_N, T_I and T_S
begin
  for t = 0, ...
    get current frame Fc
    foreach pixel in Fc
      // classification
      if (3.7) > T_N
        classify pixel as background
      else if (3.7) > 10^-10 T_N
        if (3.8) > T_I
          classify pixel as background
      else if (3.2) > T_S
          classify pixel as background
      else
          classify pixel as foreground
      // update background
      update ω_k, μ_k and σ_K in (3.2)
      update reference using (3.3)
    end foreach
  t = t + 1
  end for
end
```

(a) CMoG

```
Input: N, T_N, T_I and T_S
begin
  for t = 0, ...
    get current frame Fc
    foreach pixel in Fc
      // classification
      if (3.7) > T_N
        classify pixel as background
      else if (3.7) > 10^-10 T_N
        if (3.8) > T_I
          classify pixel as background
      else if (3.4) > T_S
          classify pixel as background
      else
          classify pixel as foreground
      // update background
      update reference using running average
      update previous background sample
    end foreach
  t = t + 1
  end for
end
```

(b) CKDE

Figure 3.5: *Proposed algorithm in pseudo-code. Both are very similar with the exception of the way dynamic background behaviour is modelled.*

Alongside MoG, cascaded kernel density estimation (KDE) was also tested. For KDE, if

the probability defined by equation (3.4) for a given pixel is larger than a threshold T_S, that pixel is classified as background. The background representation can be defined as the weighted average of the N most recent background values. The weights are defined, for each previous background and for each pixel, by the probability in (3.4) when a sample is tested against the other $N-1$ previous backgrounds. For efficiency reasons a simple running average of the incoming frames was used to estimate a background without significant errors in the final foreground estimation. In accordance with CMoG, the KDE-based cascaded algorithm will be named CKDE hereafter and is summarised in Figure 3.5(b).

3.3.4 Building the system

The proposed algorithm can be seen as a series of different techniques resulting in the refinement of the segmentation provided by the previous as shown in Figure 3.6. First we use the statistical test (3.7) to determine the set $\mathcal{S}_a$ of pixels that are identified as being changed by any phenomenon other than noise – in our model a structural change or illumination variation. Then, on that set of pixels a simple collinearity test (3.8) is performed in order to assert that the modification was not due to some modification in illumination conditions, resulting in a new set $\mathcal{S}_b$ of candidate pixels. Finally, the set is further refined eliminating any structural change that resulted from repetitive dynamic behaviour of the background using pixel-wise background estimation. The result is the final set of pixels $\mathcal{S}_c$ which are labelled as belonging to the foreground, i.e., any relevant moving object. Although the different classification tests happen in a cascade configuration, some interaction happens between them. Namely, if the change results from camera noise and not from other factor, the model is updated with the current background value instead of the new frame value. This way we effectively reduce model error by only introducing modifications to the model when any other phenomenon than noise changes the pixel value.

3.3.5 Results

Results are presented next and a comparison is first drawn between CMoG and MoG as well as CKDE and KDE. An overall comparison between all is done last. Also, note that all experiments used the YUV colour space as input features.

□ CMoG

The CMoG configuration consisted of a MoG model composed by a mixture of 3 Gaussians, a learning rate α of 0.005 and a threshold T_S of 0.05. For the statistical test it was used a significance threshold T_N of 10^{-4}. Finally, for the collinearity test a threshold T_I of 0.995 was used. All thresholds were found through empirical testing to be fairly

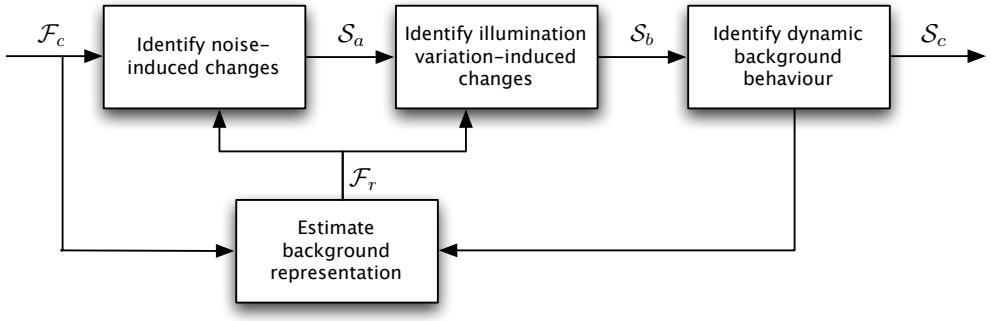


Figure 3.6: *Block representation of the proposed cascaded algorithm. The modules are connected in cascade configuration with the exception of the background estimation feedback.*

stable and can be used without modification for typical real-world scenes. The MoG algorithm was tested with the same common parameters. Moreover, to reduce the implementation complexity, for MoG implementation, a diagonal covariance matrix was considered.

Figure 3.7 shows the evolution of the cost-based partition distance (d^c_{sym}) for four sequences of the test set, namely OD, AP, BR and FT. It is clear that the proposed method outperforms the mixture of Gaussians modelling method in all test sequences. Even without post-processing the CMoG method performs significantly better than the regular MoG with post-processing. In fact, for the test set sequences, CMoG's results with and without post-processing do not differ much. In Table 3.3 the overall results are presented for every sequence in the test set. Note that for noisy and highly dynamic scenes, like OD, the regular method without post-processing has the worst results. Also note that for the AP and FT sequences a noticeable difference between the segmentations and the "ground-truth" occurs at some point in time. This is due to sudden changes of global illumination, that are not properly handled by pixel-wise estimation of the background – in this case higher level input would be needed in order to quickly adapt to the changes. Nevertheless, CMoG easily returns to the normal classification performance. Additionally, the proposed method is faster than the original MoG: for 176×144 sequences, MoG performs at 26fps, while CMoG performs at 32fps; for 352×288 sequences, MoG performs at 7fps, while CMoG performs at 9fps.

In Figure 3.8 the results obtained with CMoG for the SH sequence at different frames are shown. From top to bottom, the original frames, the intermediate results with the different sets being represented by different colours, the final mask, and the foreground are shown. The sets in the intermediate results are the outputs of the each module of the algorithm, as shown in Figure 3.6. The representation in different sets depicts the role of each module, and what is the respective influence on the refinement

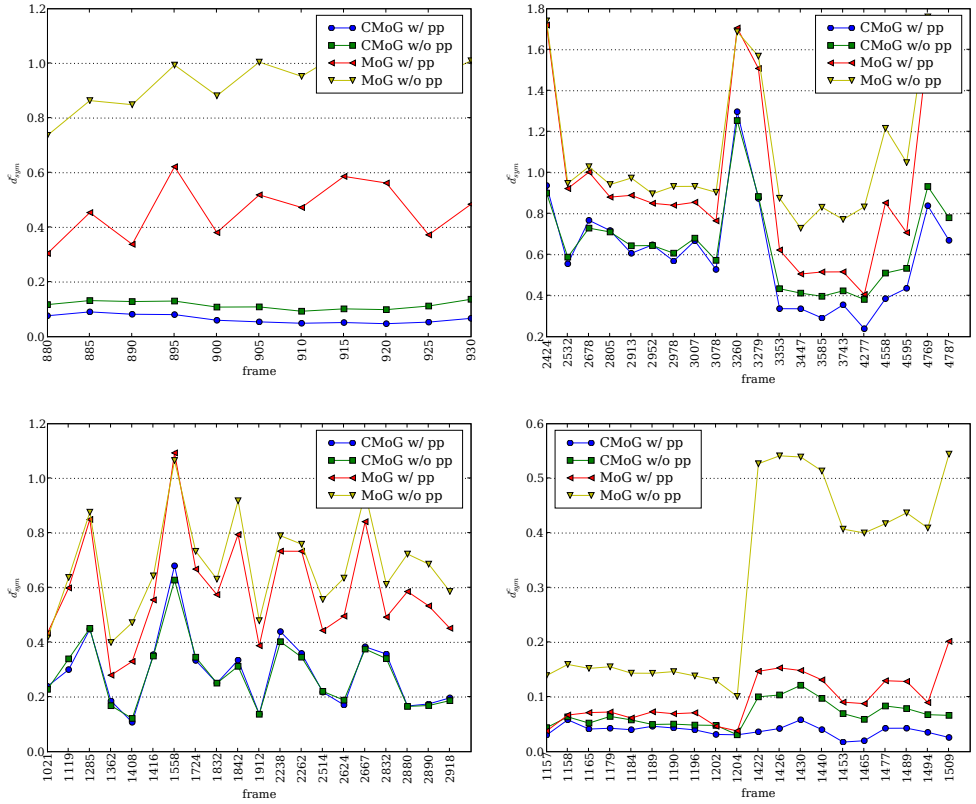


Figure 3.7: *Evolution of d^c_{sym} for MoG and CMoG implementations. From left to right, the results are presented for the sequences OD, AP, BR and FT, with and without post-processing.*

of the foreground mask. In fact, some particular events are worth noting. In frame 403 there is a global illumination change that is detected by the illumination-change detection module. This way we reduce significantly the portion of the frame that needs to be analysed by the more computationally expensive dynamic change detection module. Also, from frames 707 to 903, the person in front of the store is "absorbed" by the background since he stand still for some time. However, it is clear that in 1218 this is successfully corrected, despite the area where the person stood still being marked as potential foreground by the first two detection modules.

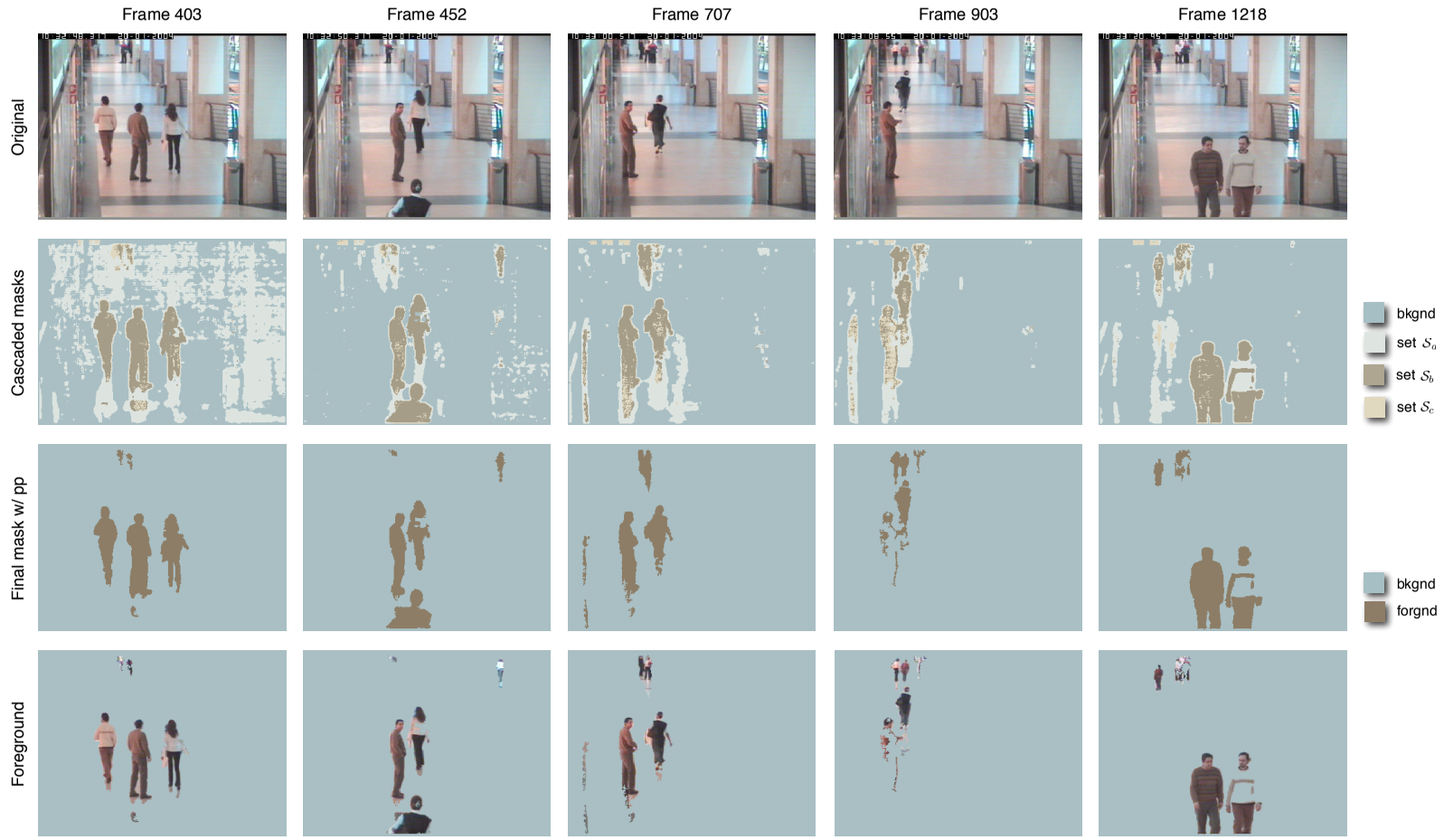


Figure 3.8: *Complete results with CMoG for the SH sequence at different frames. From top to bottom, the original frames, the intermediate results with the different sets being represents by different colours, the final mask, and the foreground are shown. The sets in the intermediate results are the outputs of the each module of the algorithm, as shown in Figure 3.6. In this representation it is clear the role played by each module. Two relevant events are visible: in frame 403 there is a global illumination change that is detected by the illumination-change detection module; and, from frames 707 to 903, the person in front of the store is "absorbed" by the background since he stands still for some time, however, in 1218 this is successfully corrected.*

□ CKDE

CKDE was tested with the same significance threshold T_N and collinearity threshold T_I as CMoG and a sample size N of 50 and a threshold T_S of 10^{-6}. The KDE algorithm was tested with the same common parameters.

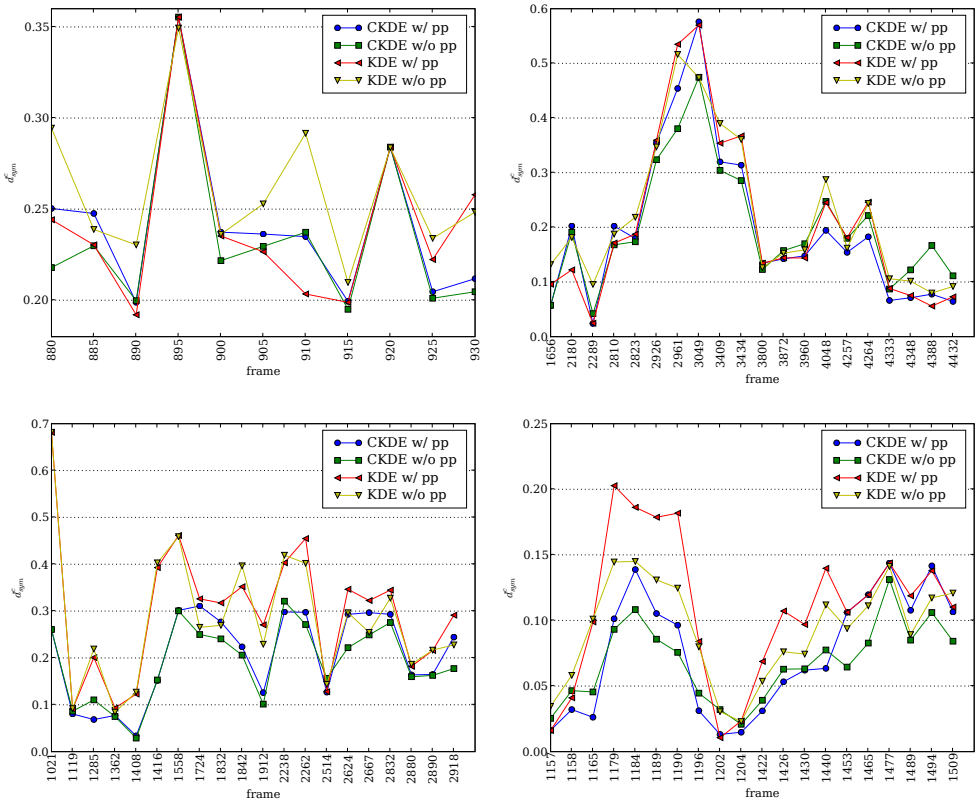


Figure 3.9: *Evolution of d^c_{sym} for KDE and CKDE implementations. From left to right, the results are presented for the sequences OD, AP, BR and FT, with and without post-processing.*

Figure 3.9 shows the evolution of d^c_{sym} for the same four sequences tested with CMoG. CKDE performs better than KDE but in general the gains are not as significant as compared to CMoG. Also, the difference between the results obtained with and without post-processing is small and, in some sequences, the results are worse with post-processing. This is due to the fragmented masks typically produced by KDE. Some isolated fragments, if small enough, will be incorrectly removed by post-processing. Speed-wise KDE is very slow compared to MoG as already noted in Table 3.2. Also, CKDE is faster than KDE: for 176×144 sequences, KDE performs at 5fps, while CKDE performs at 6fps; for 352×288 sequences, KDE performs at 1fps, while CKDE performs at 2fps.

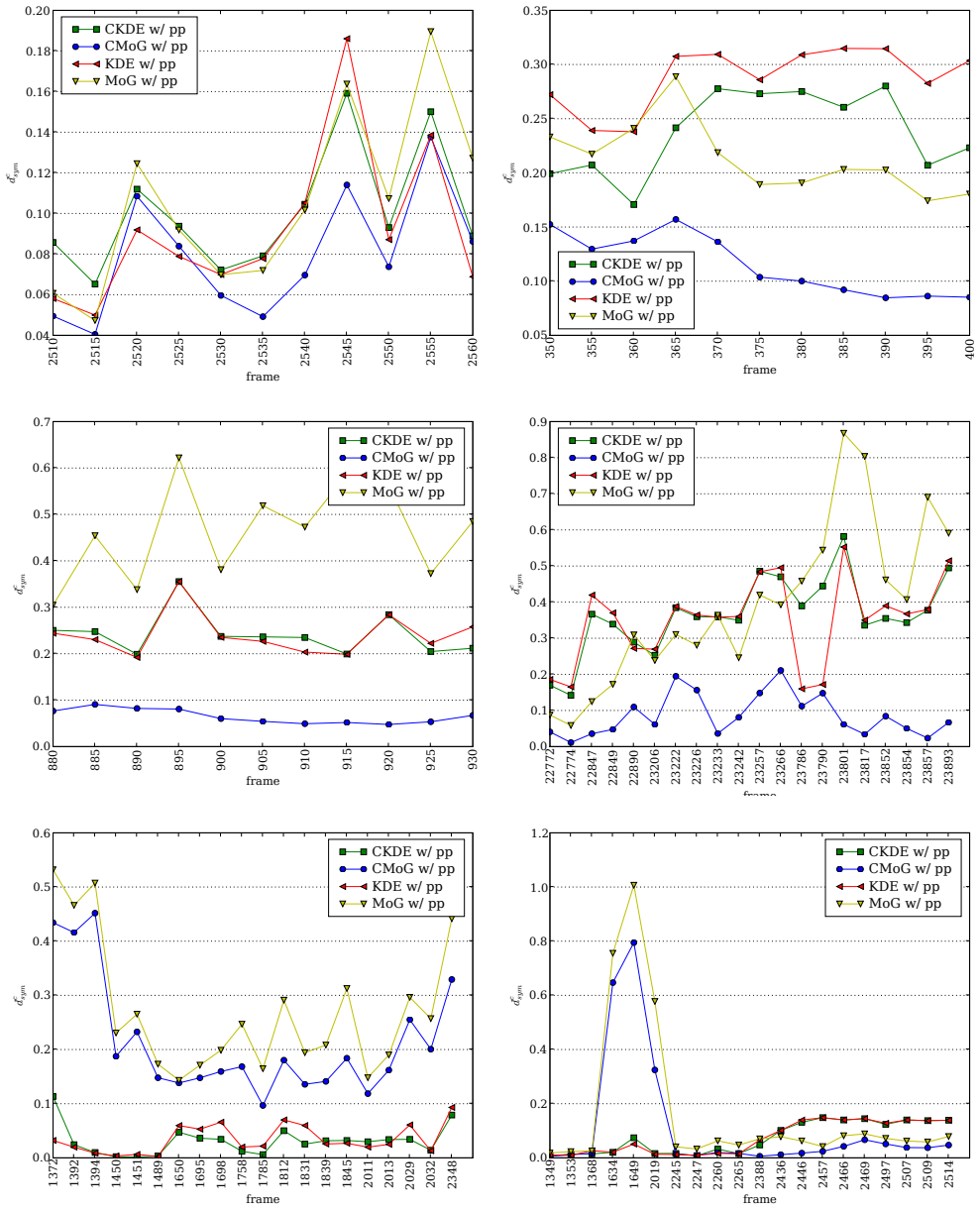


Figure 3.10: *Evolution of d^c_{sym} for each sequence of the test set (part 1). From left to right and from top to bottom, the graphs are from the sequences SW, SH, OD, MR, CAM and LB.*

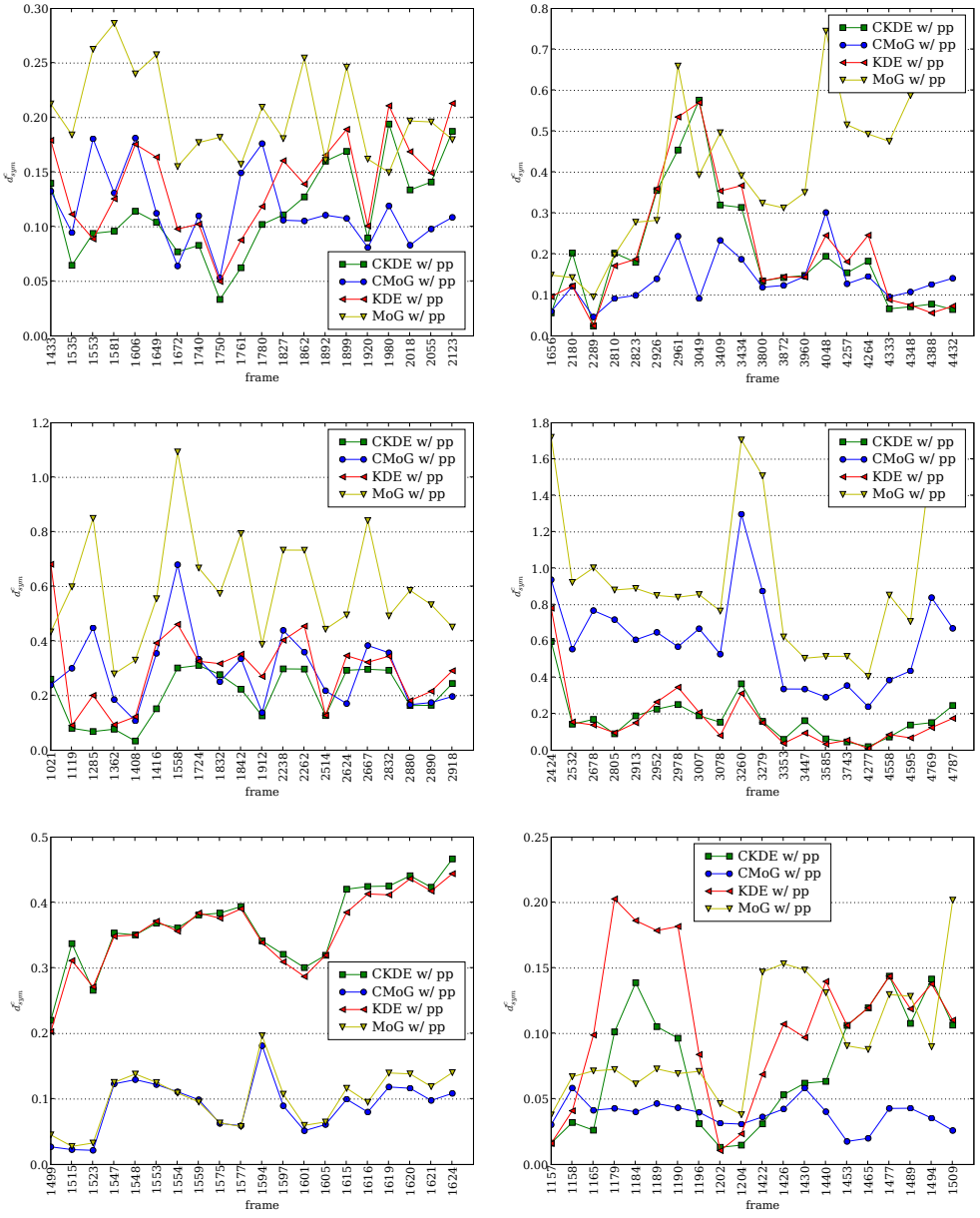


Figure 3.11: *Evolution of d^c_{sym} for each sequence of the test set (part 2). From left to right and from top to bottom, the graphs are from the sequences SC, AP, BR, SS, WS and FT.*

	MoG		CMoG		KDE		CKDE	
	w/o pp	w/ pp	w/o pp	w/ pp	w/o pp	w/ pp	w/o pp	w/ pp
SW	0.160	0.105	0.090	0.079	0.109	0.092	0.112	0.100
SH	0.267	0.213	0.119	0.115	0.267	0.289	0.235	0.238
OD	0.938	0.463	0.115	0.065	0.261	0.241	0.234	0.242
MR	0.808	0.391	0.118	0.086	0.325	0.351	0.325	0.365
CAM	0.782	0.272	0.480	0.214	0.080	0.034	0.151	0.031
LB	0.453	0.164	0.190	0.110	0.064	0.073	0.063	0.074
SC	0.355	0.203	0.111	0.115	0.133	0.140	0.108	0.114
AP	0.691	0.409	0.166	0.137	0.221	0.209	0.199	0.196
BR	0.678	0.594	0.286	0.292	0.285	0.299	0.190	0.204
SS	1.117	0.962	0.651	0.603	0.226	0.168	0.253	0.175
WS	0.223	0.100	0.102	0.089	0.322	0.356	0.329	0.365
FT	0.307	0.096	0.068	0.038	0.093	0.109	0.069	0.076

Table 3.3: *Average d^c_{sym} over the test set sequences. CMoG has the best performance with an overall average distance of 0.162 compared to 0.331 for MoG, 0.182 for CKDE and 0.199 for KDE.*

□ **Overall**

Figure 3.12 shows, from left to right, the segmentation results for frames 365 and 415 of the SH sequence, frame 600 of the OD sequence and frame 2550 of the SW sequence. The top row shows the original frame, the second row shows the results from the MoG implementation (Lee, 2005) and the third and fourth rows show the results obtained with CMoG and CKDE, respectively.

Table 3.3 shows the average distance for all the sequences in the data set and Figures 3.10 and 3.11 depicts the evolution of d^c_{sym} in each sequence. In both figures only the results with post-processing are presented to avoid excessive graph cluttering Comparing CMoG and CKDE as well as their respective base algorithms, it can observed that the CMoG has the best performance with an overall average distance of 0.162 compared to 0.331 for MoG, 0.182 for CKDE and 0.199 for KDE. Again, it is clear that adapting KDE with a cascaded change detection configuration is not as advantageous as adapting MoG. For the former an approximate gain of 8.5% is achieved while for the latter the average distance between outputted segmentations and the "ground-truth" is more than halved.

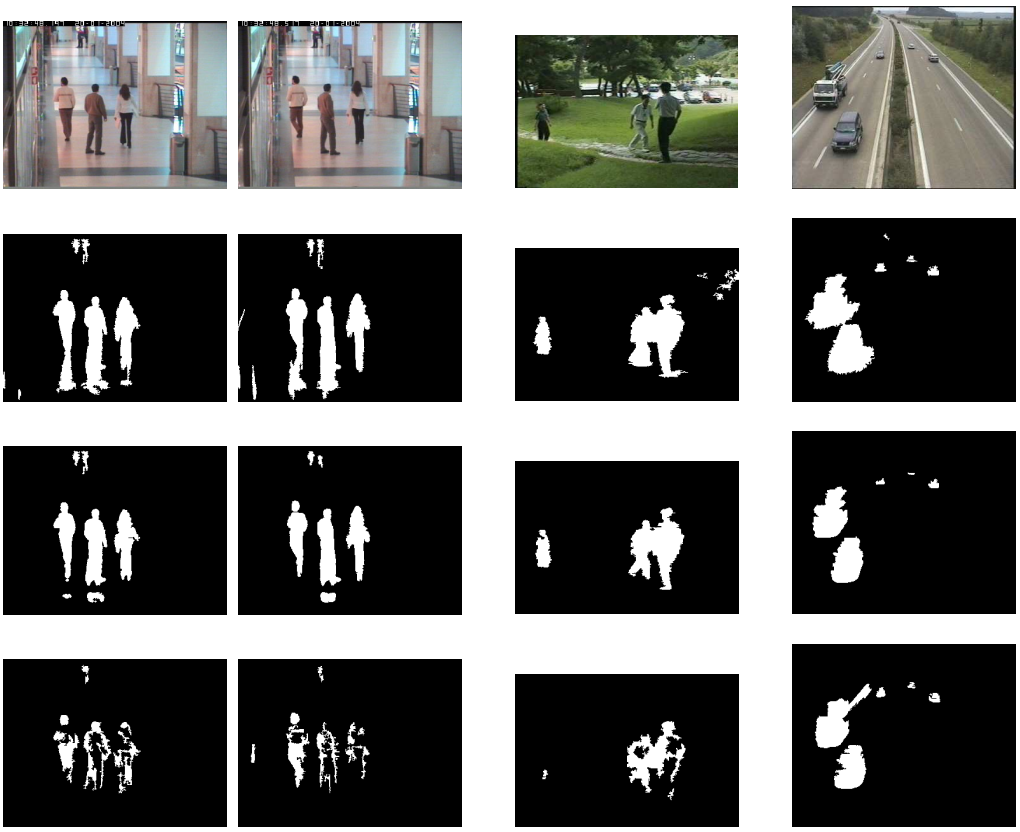


Figure 3.12: *Segmentation masks obtained using different methods. The top row shows the original frame. The second row shows the results from the MoG implementation (Lee, 2005). The third row shows the results obtained with CMoG. Finally, the fourth row shows the results with CKDE.*

3.4 Audio object segmentation ¶

Following the same rationale used for visual sequences, we can define audio segmentation as the partitioning of an audio sequence in coherent or homogeneous partitions. The partitions refer to temporal sections of the sequence that comply to a given homogeneity criterion. This criterion depends on the context of application and therefore no generic rule of segmentation is feasible.

An indirect way of segmenting an audio stream is by classifying individual small time windows, or frames. By grouping the classified frames we obtain the audio homogeneous segments. Frequently, the number of classes is defined beforehand and is kept

¶This section is based on a collaboration with Gustavo Martins as an extension to his Ph.D. work (Martins, 2009).

unchanged. For most real-world scenarios we can consider four classes: silence, noise, speech, and music. Hence, for a given audio signal, for example the one in Figure 3.13, the segmentation algorithm should be able to decompose it into segments according to the four known classes.

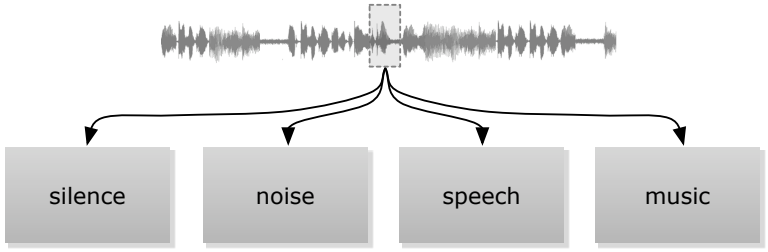


Figure 3.13: *Simple decomposition of an audio sequence in four classes. For real-world scenarios we assume that four sound classes can occur: silence, noise, speech, and music.*

The goal is to obtain a set of time segments for an input audio signal, similarly to Figure 3.14. In that simple example, segments of different classes are concatenated in a single stream and no overlap between classes is present. Ideally our audio object segmentation algorithm should output the temporal segments with the assigned labels as shown in that figure.

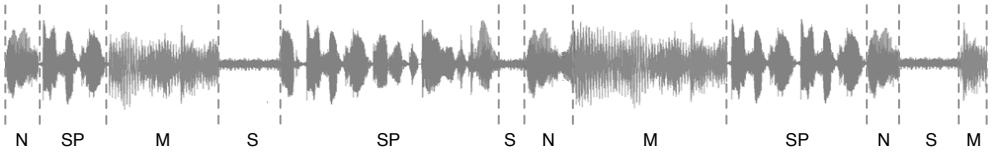


Figure 3.14: *Temporal segmentation of an audio sequence. The audio object segmentation algorithm should be able to output a define the segments shown in the figure, taking into account the four known classes, i.e. silence (S), noise (N), speech (Sp), and music (M).*

In order to accomplish this goal we will be using the algorithm depicted in Figure 3.15. The audio stream is analysed separately for frames containing N samples (step 1). For each frame a set of features are extracted and, according to a previously trained model, that frame is classified as one of the four possible classes $\{S, N, Sp, M\}$ (step 2). The frames are grouped into audio buffers containing a total of M samples. A majority voting is performed taking as votes the classification for the frames comprising an audio buffer (task 3). Finally, adjacent buffers with the same assigned label are merged to form the segments shown in Figure 3.14.

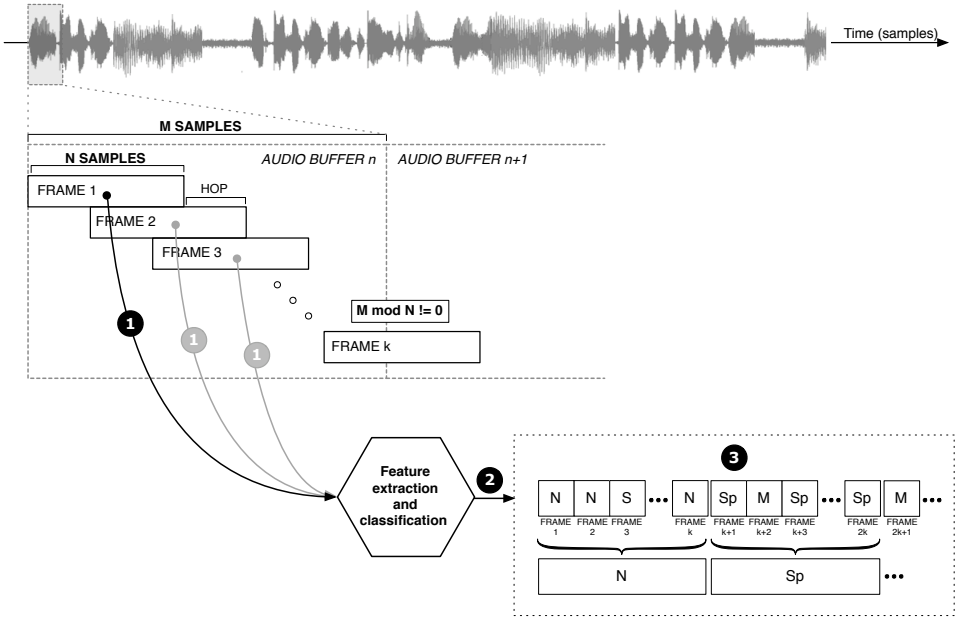
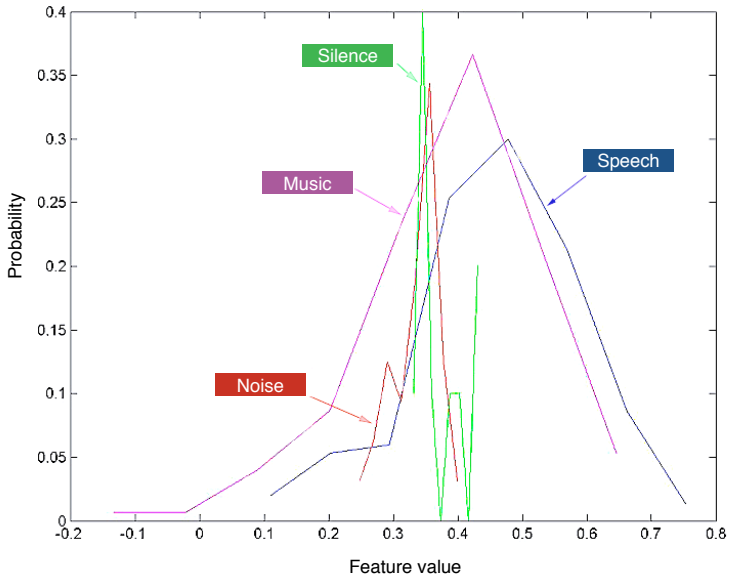


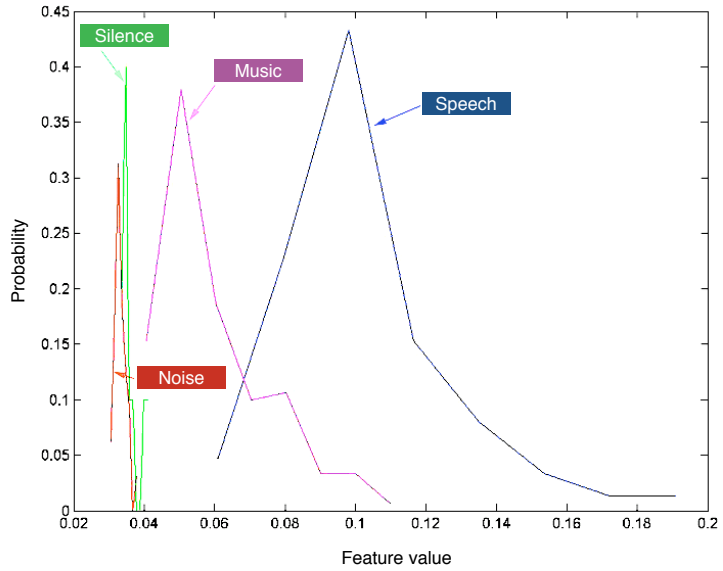
Figure 3.15: *Algorithm for audio segmentation based on frame classification. In step 1, the audio stream is analysed separately for frames containing N samples; in step 2, for each frame a set of features are extracted and classified, according to a previously trained model; in step 3, the frames are grouped in audio buffers containing a total of M samples using a majority voting scheme.*

Defining suitable features as inputs to classify the segments is critical for the performance of the overall algorithm (Scheirer and Slaney, 1997). As an example, Figure 3.16 depicts the probability of the 8^{th} Mel Frequency Cepstrum Coefficients (MFCC) coefficient (refer to Section 2.3.1 for a description of MFCCs and other audio features) taking a given value. While the average of that feature for an audio frame, shown in Figure 3.16(a), does not allow a clear distinction of the four classes, the variance offers a better discriminative capability, especially between structured sounds (speech and music) and the other classes (silence and noise). The features that were used to build the model to classify the audio segments are summarised in Table 3.4.

The dataset $\mathcal{D}_4^a$ described in Section A.2 was used for the tests and two classifiers were tested: k-Nearest Neighbour (k-NN) and Gaussian classifiers. The k-NN algorithm is a simple machine learning algorithms that classifies an object by a majority vote of its neighbours. The object is assigned to the class most common amongst its k nearest neighbours. The parameter k is a typically small positive integer; in our experiments we used $k = 3$. To determine the distance between objects the Mahalanobis distance was used. The Gaussian classifier is a statistical classifier that assumes a Gaussian distribution of the data. The results using a 5-fold cross-validation test are summarised



(a) Average of the 8^{th} MFCC coefficient.



(b) Variance of the 8^{th} MFCC coefficient.

Figure 3.16: *Discriminative capability of features. The average of the 8^{th} MFCC coefficient presents a poor discriminative capability between the four defined classes. On the other hand, the variance of the same MFCC coefficient presents a better discriminative capability, especially between structured sounds (speech and music) and other classes (silence and noise).*

Feature	Statistic
Spectral flatness	average
Spectral flatness	variance
Low energy perc. (25%)	—
MFCC 2^{nd} coef.	variance
MFCC 3^{rd} coef.	variance
MFCC 4^{th} coef.	variance
MFCC 5^{th} coef.	variance
MFCC 6^{th} coef.	variance
MFCC 7^{th} coef.	variance
MFCC 8^{th} coef.	variance

Table 3.4: *Audio features for frame classification.*

in Table 3.5 only for the speech/music segmentation.

		k-NN estimated class		Gaussian estimated class	
		Speech	Music	Speech	Music
true class	Speech	94.7	5.3	94.4	5.6
	Music	3.9	96.1	12.4	87.6

Table 3.5: *Confusion matrix using k-NN and Gaussian classifiers for audio segmentation for two classes. The overall successful detection rate is of 96% for k-NN and 91.6 for the Gaussian classifier.*

After classifying a segment as a given class, that segment can be further processed in a recognition step. For example, specific algorithms can be used to process noise, speech and music segments. For music segments, the recognition task can include genre recognition or music fingerprinting. The system introduced in Figure 3.13 can therefore be expanded to incorporate these tasks, resulting the system in Figure 3.17.

3.5 Audio object localisation using AV features

Hershey and Movellan (1999) first tried to develop a system that uses audio visual synchrony to locate sound sources (refer to Section 2.3.2 for a description about how to measure AV synchrony). The system searches for regions of the visual field that correlate highly with acoustic signals using an estimate of the Mutual Information (MI). Only an audio feature, the average acoustic energy over a time interval corresponding to interval between visual frames, and a visual feature, the pixel intensity were

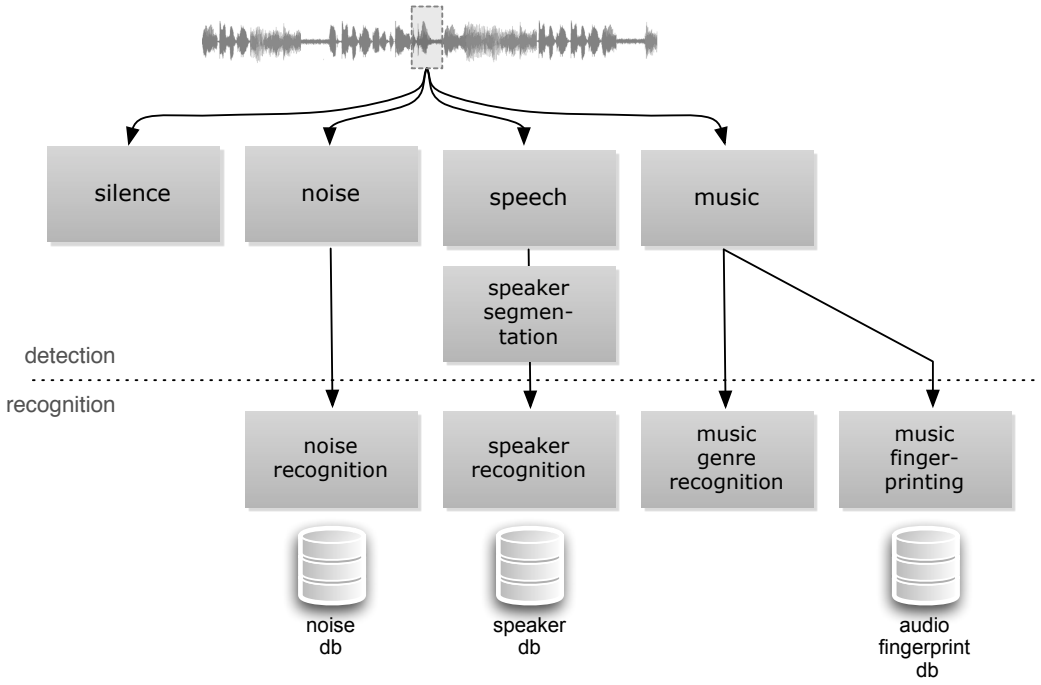


Figure 3.17: *Hierarchical segmentation and recognition of an audio sequence. After classifying a segment as a given class, that segment can be further processed. For example, specific recognition algorithms can be used to process noise, speech and music segments. For music segments, the recognition task can include genre recognition or music fingerprinting.*

computed and analysed. A centroid was then calculated, corresponding to the estimated acoustic source position. Alongside the assumption that the joint statistics are Gaussian, another limitation of this method arises from the per-pixel measure which invalidates the integration of the measure over an image region without making unrealistic assumptions – e.g. pixel are independent of each other, conditioned by the audio signal. Slaney and Covell (2000) proposed a more general approach, aiming specifically at optimizing temporal alignment between audio and visual signals. The authors use a face-recognition algorithm and Canonical Correlation Analysis (CCA) to measure audio-visual synchrony. The approach consists in deducing a relationship between the cepstral representation of the audio and the video pixels by obtaining a canonical correlation model. This is equivalent to maximum mutual information projection in the jointly Gaussian case. The algorithm is processed in two stages: training of the model and evaluating the fit of the model to the data. Like the Hershey and Movellan algorithm, the Pearson's correlation is also used to measure synchronisation. However, in this case the optimal combination of audio and visual data is computed, unlike Her-

shey's approach which does not consider the mutual information between adjacent pixels. The immediate consequence of computing the canonical correlation model is the need for training data which may influence the performance of the algorithm and condition its use to application-specific scenarios.

Nock *et al.* (2003) evaluate three different approaches to assess audio visual synchrony in a speaker localisation application – two based in MI (histogram-based estimate over vector quantised code-books and Gaussian estimate over feature vectors) and a third one is based in HMMs. Several features were tested in all approaches, which included MFCCs for audio and DCT and pixel intensity change for visual characterisation. The Gaussian approach is reported as the one with the best results when using MFCC as audio features and DCT coefficients as visual features. Fisher III and Darrell (2004) proposed a methodology for testing audio-visual association in the absence of a prior model and without requiring a training stage to estimate one. The methodology is based on the probabilistic generation model, used to define projection rules on maximally informative subspaces. Densities are learnt using a non-parametric Parzen density estimator which can be used to represent complex joint densities of projected signals, and to successfully estimate MI. Butz and Thiran (2005) proposed an approach based on Markov chains to model audio and visual signals. The audiovisual consistency is assessed by maximising the efficiency coefficient, i.e. the ratio between audiovisual MI and the audio-video joint entropy. The distributions of audio-video features are again estimated using Parzen density estimators. Regarding the audio signal, a linear combination of power spectrum coefficients that present the higher entropy, and on the other hand, pixel intensity change is analysed, in order to characterise the visual signal. Gurban and Thiran (2006) have recently proposed learning the parameters of a Gaussian Mixture Model (GMM) that is used to estimate the joint density of audio-video features. A video feature specific to the difference between the average optical flow on the top and bottom halves of the central part of the mouth region is used. Audio sources are localised on the video by finding the image regions where samples have highest likelihood to have been generated by the learned joint density function.

While the former methods have a common rationale, the estimation of MI between audio and visual signals, other approaches following different paths are also possible. Cutler and Davis (2000a) followed an approach based on a TDNN. Audio-video correlations are learned on positive and negative examples using the neural network, which is then used to find in time and space a speaking person on the input data. Normalised cross correlation between consecutive images is employed as video feature, while cepstral representation is used for the audio signal. Monaci *et al.* (2006) on the other hand represent audio-visual signals using decompositions on over-complete dictionaries. Activation vectors are built from audio energy peaks and the displacement peaks for each video element (atom) are extracted. The synchronisation scores between the audio activation vector and the video activation vectors are computed as the scalar product between those signals. A summary of the audio localisation algorithms can be found in Table 3.6.

Authors	Measure of synchrony	Features	Requires training
Hershey and Movellan (1999)	MI using Pearson's correlation	pixel intensity and average acoustic energy over a time interval	yes
Slaney and Covell (2000)	CCA	MFCC and whole video frame	yes
Fisher III and Darrell (2004)	MI estimated using Parzen estimator	pixel intensity for the video and audio periodograms	no
Butz and Thiran (2005)	efficiency coefficient	pixel intensity change and linear combination of the audio power spectrum coefficients	no
Gurban and Thiran (2006)	GMM	average optical flow on the top and bottom halves of the central part of the mouth region and MFCC	yes
Cutler and Davis (2000a)	TDNN	normalised cross correlation between consecutive images and cepstral representation	yes
Monaci *et al.* (2006)	synchronisation scores between activation vectors	visual atoms and estimation of audio energy	no

Table 3.6: *Summary of audio localisation algorithms.*

3.6 Person detection

The segmentation obtained with background modelling and subtraction returns a set of candidate regions, but an identity still needs to be assigned to one or more of these regions to form a relevant object. This is typically done at the recognition level, but if the type of object is known, a priori information can be used to further improve the detection result. For example, we know that humans have a particular shape, hence shape models can be used. Also, for the subsequent tracking algorithm specific motion models can be used, since a person's movement follows certain rules. The periodicity of the human locomotion can be an importance factor when designing a human detection and tracking system (Cutler and Davis, 2000b). In systems focused on people, like surveillance systems, it is often desirable to use additional information to exclude moving objects other than people from the tracking algorithm. Another difference when approaching specific person tracking is that, while generic object tracking algorithms typically only focus on tracking the object as a whole, person tracking algorithms may also decompose the motion into global motion and limb motion. This allows to infer additional information, such as posture or locomotion modes – standing, walking or running. Object tracking is analysed in more detail in Section 4.2.

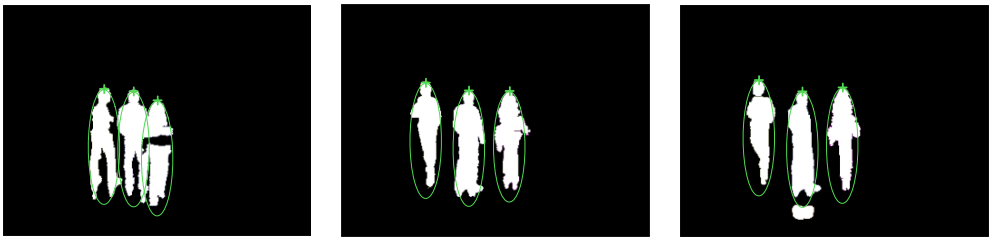
Considering only the detection task, let us consider the scenario of people detection in Figure 3.18, where the segmentation results for three frames of the test sequence SH are shown. If we look for isolated connected regions, we define a set of objects, shown with different colours in each image. While in frame 397 the three objects (persons) are correctly separated, the same does not occur for the two other frames. In frame 365, the segmentations of the two persons on the right merge, forming a single object, and, in frame 407, errors in the segmentation algorithm originates incorrect objects.

(a) Frame 365 – 2 objects (b) Frame 397 – 3 objects (correct) (c) Frame 407 – 5 objects

Figure 3.18: *Problems with simple foreground segmentation. The figures show three frames from the SH sequence where each isolated region is labelled as an object. The inconsistencies that occur between frames can be avoided if specific models for persons are used.*

It is possible to take advantage of the camera being placed several meters above the ground in typical scenarios, to avoid some situations of occlusion and reduce the probability of people heads being occluded. For this reason, head detection is used in the

generation of human hypothesis. Head detection has been used in several systems such as (Denman *et al.*, 2005; Haritaoglu *et al.*, 2000; Siebel and Maybank, 2002). In Figure 3.19 the same example shown in Figure 3.18 is applied to a person detection algorithm based on head detection.



(a) Frame 365 – 3 persons (b) Frame 397 – 3 persons (c) Frame 407 – 3 persons

Figure 3.19: *Person detection based on the detection of persons' heads. This assumption is possible since in typical real-world scenes are captured using cameras located several meters above the ground. Each person is represented by an ellipse with the respective major axis following the person's body orientation.*

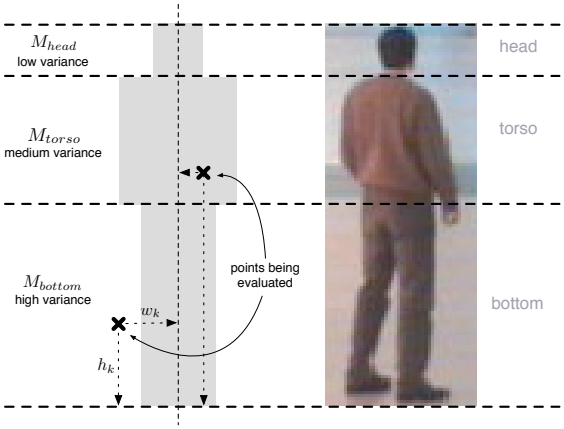


Figure 3.20: *Person model used to define the foreground pixels belonging to a person. The models $M_{\{head,torso,bottom\}}$ define an occupancy probability such that it is higher for points nearer to the centre line and smaller for points farther from the centre line. (adapted from Kim, 2005)*

A more complex way of assigning a foreground pixel to a person is to use the model shown in Figure 3.20 which is based on the shape of a person. Depending on the height h_k of the point being analysed, one of three probabilistic models $M_{\{head,torso,bottom\}}$ is

used. With these models an occupancy probability is defined such that it is higher for points nearer (smaller values of w_k) to the centre line and smaller for points farther from the centre line (larger values of w_k). Different variances are used for each model since, for example, there are less variations for head heights between persons than for bottom heights.

3.6.1 Face detection

The goal of face detection is to locate an unknown number of faces in an image or sequence of images. An example of successful results using a state-of-the-art algorithm is shown in Figure 3.21. This is however not an easy task since there is a high variability of size, shape, colour, and texture that result from differences in pose, presence or absence of structural components, facial expressions, occlusions, different imaging conditions, among other factors. Techniques for face detection can be divided in two categories, and each into two sub-categories: knowledge-based, divided in bottom-up or feature-based, and top-down; template-based, divided in template matching, and appearance-based. For more in-depth information about face detection refer to the surveys by Hjelmas and Low (2001) and Yang *et al.* (2002).

Figure 3.21: *Example results of a face detection algorithm. These results were obtained with the AdaBoost algorithm using images from the MIT + CMU test-set (taken from Viola and Jones, 2002)*

.

□ **Knowledge-based**

⇀ **Top-down** A priori knowledge about a face is expressed in terms of rules. Typically, these rules are based on the relationships between the facial features (Lv *et al.*, 2000; Yang and Huang, 1994). These methods perform well in frontal faces in uncluttered scenes but translating face knowledge into rules can be a difficult task. Also, it is

difficult to extend this approach to detect faces in different poses since enumerating all the possible cases is impractical.

⇀ **Bottom-up or feature-based** Face features, including position and relationships between them, can be extracted regardless of the lighting conditions and pose variations. Different features can be used for this task, namely facial features (Yow and Cipolla, 1997), texture, skin colour (Hsu *et al.*, 2002; Jones and Rehg, 2002), or a combination of multiple features. The main difficulty is to locate facial features in complex backgrounds and due to changes caused by illumination, noise, and occlusion.

□ **Template-based**

⇀ **Template matching** A standard face pattern is manually predefined or parameterized by a function. The correlations between an input image and the pattern are computed to detect a face. In general, templates need to be initialised near face images. Rigid templates (Govindaraju *et al.*, 1989; Samal and Iyengar, 1995) were initially proposed, but these can not deal with variations in scale, pose, and shape. Alternatively deformable templates were also proposed (Kwon and da Vitoria Lobo, 1994; Yuille *et al.*, 1992).

⇀ **Appearance-based** A priori information of the face is implicitly incorporated into the system through training schemes, learning "template" characteristics. This is the most successful approach to face detection but can be relatively complex to implement. Machine learning techniques have been used, like neural networks (Rowley *et al.*, 1998) and Support Vector Machines (SVMs) (Osuna *et al.*, 1997), as well as statistical analysis (Turk and Pentland, 1991; Viola and Jones, 2002). The representation of faces used for training can use an holistic approach, where each face image as a whole is represented by a vector of intensity values, or a block-based approach, each face image is decomposed into a set of overlapping or non-overlapping blocks. Multiple scales can be used or further processed can be applied using vector quantisation, PCA, or other methods.

□ **AdaBoost**

boosting

An important breakthrough on object detection, and face detection in particular, resulted from the work by Viola and Jones (2001, 2002). Their method is based on boosting (a tutorial about boosting can be found in Freund and Schapire, 1999), a machine learning meta-algorithm that relies on ensembles of classifiers whose joint decision rule has arbitrarily high accuracy even though the component classifiers have poor performance. The algorithms based on boosting alter the training data distribution before each new bootstrap sample is obtained. The altered distribution ensures that

more informative instances are drawn into the next dataset. Viola and Jones applied a boosting-based algorithm named AdaBoost (Freund and Schapire, 1997) and used local contrast features as inputs. Their method proved to be one of the most effective state of the art methods for face detection. For this reason a more detailed description of AdaBoost applied to face detection is presented next.

Local contrast features are extracted using Haar filters at different positions of an image sub-window. These features are evaluated using weak classifiers that evaluate if a given sub-window corresponds to face or not. While each of the individual classifiers have a low performance rate, combining some of them to form a strong classifier results a much higher successful detection rate. Using solely this approach it is possible to obtain more than 99% detection rates, but at the same time the false detection rate reaches unacceptable values, more than 30%. The workaround proposed by Viola and Jones was a cascade of strong classifiers, or stages, trained using an adaptive training scheme. Each stage is trained with all the examples that the previous stage has misclassified plus some new ones. This leads to an optimal selection of features in each cascade which are able to detect always harder examples. In other words, the first stages can discard sub-windows which are very different from faces, whereas the latter stages could reject more difficult examples like round objects.

3.6.2 Speaker segmentation

The goal of speaker segmentation is to find speaker change points in an audio stream. It usually precedes audio recognition tasks such as audio indexing, speaker recognition (including identification, verification and tracking), automatic transcription. The various segmentation algorithms that have been proposed in the literature, can be categorised as: metric-based segmentation, model-based segmentation, and hybrid segmentation.

metric -based In speaker segmentation based in metrics, a distance-like metric is calculated between two neighbouring windows placed at each sample. Different metrics can be used such as Kullback-Leibler (KL) divergence (Siegler *et al.*, 1997), Log Likelihood Ratio (LLR) (Bonastre *et al.*, 2000) and most commonly, Bayesian Information Criterion (BIC) (Chen and Gopalakrishnan, 1998). The local maxima or minima of these metrics are considered to be the change points.

model -based In alternative, model-based segmentation methods were also proposed (Gauvain *et al.*, 2002; Kemp *et al.*, 2000). These methods use different models e.g. GMM or HMM for a fixed set of acoustic classes, i.e. speakers – similarly to the method described in Section 3.4. For every speaker in the audio recording, a model is trained and then Maximum Likelihood (ML) is performed to find the best time-aligned speaker sequence. Speaker change points are identified where there is a change in the acoustic class.

hybrid By combining different techniques, some authors proposed hybrid methods that proved to perform better than other approaches without needing prior data (Kim, 2005).

A commonly used method to achieve real-time performance relies on a two-stage approach, where a coarse segmentation is obtained in the first stage and a refinement of the potential speaker boundaries is done in the second stage (Wu *et al.*, 2003). The acoustic speaker segmentation algorithm that was used in the multimodal speaker identification described in Section 2.3.4 follows a similar approach.

3.7 Summary

The extraction or detection of objects from an audiovisual scene is an important first step of processing chains usually assembled to acquire higher level semantic knowledge. The goal is to partition the visual and audio data in homogeneous segments. These segments can then be further analysed independently using specific methods. The different nature of audio and visual signals imply different detection approaches.

For the scenarios we are considering it is possible to rely on assumptions that simplify the detection task. The segmentation of visual scenes is a difficult task that usually is dependent on the context. In this case, we consider only two visual classes in a first stage: foreground and background. The foreground is composed by the objects considered relevant, that present a dynamic behaviour that somehow differs from the "predictable" behaviour. Conversely, the background includes "static" (including changes of the appearance due natural causes like illumination) components of the scene as well as components that, despite having a dynamic behaviour, present a repetitive pattern. A person detection algorithm using body or face detection algorithms can then be applied to the foreground, subdividing it into visual objects associated to persons or generic objects. Three object classes are thus considered: generic object, person (both comprising the foreground) and background.

The objective of foreground segmentation is therefore to dissociate the regions containing the relevant objects from regions composing the background. Due to typical dynamical behaviour of the background and variations in illumination conditions it is not always a straightforward task. An efficient method of extracting the foreground objects consists of modelling each background pixel evolution, by estimating a probability density function (p.d.f.). Several methods to accomplish this have been proposed in the past but no objective comparison of these methods has been done before. We introduced a new method that takes in consideration that background changes are caused by phenomena of different nature. The method consists of a cascaded evaluation of typical dynamic elements that, although changing in time, are part of the background. These elements include acquisition noise, illumination variation and slow or repetitive structural changes. The latter type of changes is classified using the methods that estimate a p.d.f. to model the background. Also, a statistical test and a simple collinearity test are used to classify changes originated by noise and illumination changes, respectively. Two approaches were tested: one based in MoG and named cascaded MoG (CMoG), and another based in KDE named cascaded KDE (CKDE). The pro-

posed methods CMoG and CKDE perform consistently better than the base methods. While CMoG reduces the average distance to the "ground-truth" by 51%, CKDE has an 8.5% reduction. Overall, the best results are achieved by CMoG which, even without post-processing, has similar or better performance than the base method with post-processing.

The segmentation of audio scenes consists of defining time segments that correspond to a specific audio class that presents homogeneous characteristics. An indirect way of segmenting an audio stream is by classifying individual small time windows, or frames. By grouping the classified frames we obtain the audio homogeneous segments. Assuming that there is no overlapping of the time segments corresponding to audio objects of different classes, the fixed-size frames are individually analysed and classified according to a previously trained model. The number of classes is tipically defined beforehand and is kept unchanged. For typical scenarios we consider four audio object classes: silence, noise, speech, and music. Besides temporal segmentation, audio object detection also comprises spatial localisation. Using solely audio information this can be achieve using two sources, following a similar model to human audition, or using arrays of sources with known responses and configuration. However, visual information can be used to aid audio localisation using only one source of audio. Most methods are based on the search for regions of the visual field that correlate highly with acoustic signals using an estimate of the Mutual Information (MI). Different approaches are also possible such as learning a Time-Delay Neural Network (TDNN) or measuring directly the co-occurrence of audio and visual events.

The step after detecting the objects comprising the scene is associating an identity to them. This is typically done at the recognition level, but if the type of object is known, a priori information can be used to further improve the detection result. In systems focused on detecting people it is often desirable to use additional information to exclude moving objects other than people from the tracking algorithm. Besides shape, another cue to detect persons is through face detection. The goal of face detection is to locate an unknown number of faces in an image or sequence of images. Current state-of-the-art face detection methods can efficiently detect faces in typical scenes. Nevertheless, differences in pose, structural components, facial expressions, occlusions, or imaging conditions are challenges that need to be tackled in more complex scenarios, like in surveillance systems. Audio streams can also be further segmented by detecting changes in the speaker. Segments that were identified as speech are given as input to speaker change detection methods. These methods are commonly based in the distance-like metric that is calculated between neighbouring windows, or in classification changes done by previously trained models. While many of these methods require multiple processing cycles for the whole stream, real-time methods are also possible.

In the next chapter we will focus on the recognition of objects, taking as inputs the results obtained with object detection.

4 Object recognition

In real-world scenes, relevant objects appear in a cluttered environment and their identification comprises detection and recognition. Object recognition is a fundamental task that relates to how the human cognitive system works. In general the recognition process involves comparing an unknown sample with a model — matching. A related task is object tracking that relies on both detection and recognition to follow an object along time. The recognition is nonetheless simplified since the set of samples and models to match are restricted to a time window. This reasoning is applied to both vision and auditory systems, despite following somewhat different cognitive processes.

4.1 Overview

Object recognition is the task of associating an identity to a given object from a set of known classes. Therefore, object recognition can be seen as a problem of matching models from a database with representations of those models. The object representation scheme adopted by the recognition system is crucial for the system's performance. A brief insight into the audio and visual features used for object representation in this work was previously given in Section 2.3.1. Taking as example the case of a visual object, storing in a database all possible object appearances with different views or different lighting conditions is unfeasible. In this case, if we use features that are invariant to transformations in scale, rotation, and size we will be limiting the models complexity. Two approaches are possible to deal with the many possible transformations that an object may undergo: firstly, determine the transformation and try to reverse its effects, and secondly, as noted before, find measurements of the object that are invariant to the typical transformations.

Formally, object recognition is a process of hypothesising an object-to-model correspondence and then verifying that the hypothesis is correct. Generally an hypothesis is considered successful if the error between the projected model features and the corresponding object features is below some threshold. A recognition system comprises two stages: the acquisition stage, where a model library is constructed from certain descriptions of the objects; and the recognition stage, where an unknown object is given to the system and its identity is determined. Although typically the recognition task is preceded by the object detection task (detailed in the previous chapter), the detection (or localisation) of the object is not a requirement for object recognition. If this is the case, the output of the object recognition system will simply be if a given class is shown in the scene or not.

A task that combines object detection and recognition with an additional temporal element is object tracking. The target objects must be detected and identified in each frame, and correspondences must be drawn between objects from frame to frame so that they have consistent identities. The matching between objects is thus limited to the set of objects detected in each frame and the previous frames. This contrasts with the typical recognition task that matches a given object with a global database of known classes. Regarding tracking, we will focus on visual object tracking. Nevertheless, the concept of tracking in audio has a parallel with that of tracking visual objects. Object tracking, or more commonly source tracking in the audio community, is part, for instance, of harmonic sound separation (a recent framework for predominant melodic source separation using normalized cuts was proposed by Lagrange *et al.*, 2008). In the source separation process, sinusoidal modelling is used, consisting of creating sound models from the summation of sine waves parameterized by time-varying amplitudes, frequencies, and phases. The peaks of the magnitude spectrum in these time-varying quantities are estimated and connected over time to form partials (tracking). The partials are then grouped to form potential sound sources (formation).

A specific task of object recognition is person identification. The identification of a person can be performed using different modalities: appearance, gait, speech, among others. Similarly to generic object recognition, we define the identification task as the process of searching a database for a reference model matching a submitted sample and if found, the corresponding identity is obtained. Two types of identification are possible: open-set and closed-set. Generalising the definition provided by the VoiceXML Forum for the more specific task of speaker identification (Skerpac, 2007),

open-set identification

an identification process is said to be open-set if the test subject does not necessarily have any representative reference model in the database of the recognition engine. In this case, the person recogniser may either return an identity for the test speaker or may reject the speaker as not-present in its database. On the other hand, an identification process is said to be closed-set if the test subject is known to exist in the trained database. In this case, the identification engine returns the identity of the person in its database whose model most closely resembles the test sample.

closed-set identification

In this chapter we present the basis of visual tracking and present a proposal of a method to match unknown objects with an appearance-based model of generic visual objects. This will provide an example of a typical application of object recognition based on a single modality. The attention will then be on the specific task of recognising persons, combining in this case multimodal information, namely appearance (proposed method), face, gait and speech.

4.2 Visual object tracking

Object tracking consists of establishing correspondences between the regions defined by an object detection algorithm for each frame, with other partitions in subsequent frames. The goal is to generate the trajectory of an object over time by locating its position in every frame of the video (Figure 4.1). The result may also include the complete region in the image that is occupied by the object at every time instant. If additional information is available, such as calibration information from the cameras that captured the scene, each frame can be associated with a global coordinate. During recent years a very large community within computer vision has tackled object tracking with a multitude of approaches. (Yilmaz *et al.*, 2006) compiled an extensive survey; more information about object tracking can be found there. We will nevertheless present here an overview of the relevant concepts and approaches.

Figure 4.1: *Example of the output of a person tracker. The output consists of the trajectory of the person over time by locating its position in every frame of the video and the complete region that is occupied by the person at every time instant.*

Different shapes and appearance models can be used to represent the objects being tracked. In the simplest form, an object can be represented by a single point potentially representing that object's centroid. This simple form of representation can be used only with a translational model. However, more points located by an interest point detector (Lowe, 2004; Mikolajczyk and Schmid, 2004) can alternatively be used to describe the object, allowing more complex transformation models. The association of points comprising the individual detected objects in consecutive frames is based on the previous object state which can include object position and motion. The methods

point tracking

following this approach are classified as *point tracking* methods and can be divided into two broad categories: deterministic and statistical methods. The deterministic methods use qualitative motion heuristics (Veenman *et al.*, 2001) to constrain the correspondence problem, and probabilistic methods explicitly take the object measurement and take uncertainties into account to establish correspondences.

Regarding the deterministic methods, Sethi and Jain (1987) solve the correspondence by greedy approach using proximity and rigidity constraints, termed Greedy Exchange (GE), later modified by Salari and Sethi (1990) to handle occlusions, and entries or exits of new objects. More recently, Shafique and Shah (2005) proposed a multiframe approach to preserve temporal coherency of the speed and position; correspondence are formulated as a graph theoretic problem.

The statistical correspondence methods use the state space approach to model the object properties such as position, velocity, and acceleration. In the case where there is a single object in the scene, the state can be simply estimated by two steps: *prediction* (estimate prior pdf) and *correction* (estimate posterior pdf). Bar-Shalom (1988) applied this method for object tracking using Kalman filtering (Kalman, 1960). For linear and Gaussian systems, the optimal state estimate for the single object case is given by the Kalman filter (implementation available at [39]). For non-linear systems, adaptations like the extended Kalman filter and the unscented Kalman filter (implementation available at [40]) are possible. Also, with non-Gaussian systems, the Kalman filter will give poor estimations of state variables. In the general case that object state is not assumed to be a Gaussian, state estimation can be performed using particle filters (Kitagawa, 1987). Particle filter is a generic term for Sequential Monte Carlo (SMC) techniques that are useful for object detection and tracking, among other applications (tutorial by Arulampalam *et al.* (2002) and implementations available at [41]). Isard and Blake (1998) applied particle filtering to the problem of visual tracking using active contours. Other types of particle filters, and extensions to the traditional particle filter were introduced thereafter, including partitioned sampling (MacCormick and Blake, 2000), and Markov Chain Monte Carlo (MCMC) particle filtering (Khan *et al.*, 2005).

Both Kalman and particle filtering assume a single measurement at each time instant, limiting the state estimation to a single object. If there are multiple objects in the scene, measurements need to be associated with the corresponding object states, requiring a joint solution for the data association and state estimation problems. The methods Joint Probability Data Association Filtering (JPDAF) (Fortmann *et al.*, 1983) and Multiple Hypothesis Tracking (MHT) (Reid, 1979) (later applied to visual tracking by Cox and Hingorani, 1996) are commonly used for data association. The MHT algorithm predicts multiple hypotheses for an object at each time step using a predictive filter (often a Kalman filter), and manages the hypotheses by clustering and pruning or deferring difficult data association decisions until more data is received. While JPDAF algorithm can not handle objects entering or leaving the Field of View (FOV), MHT does but exhaustively enumerates all possible association, which is computationally

exponential both in memory and time.

Using a geometric representation like an ellipse (as depicted in Figure 3.19), parametric motion models like affine or projective transformations are possible. This representation form is associated with *kernel tracking* methods. Kernel refers to the object shape and appearance and can be a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion of the kernel in consecutive frames, usually in the form of a parametric transformation. Two types of kernel tracking methods exist: *templates and density-based appearance models*, and *multiview appearance models*.

kernel tracking

Templates and density-based appearance models have been widely used because of their relative simplicity and low computational cost. Template matching is a brute force method of searching the image for a region similar to the object template. As an alternative to a brute force search for locating the object, Comaniciu *et al.* (2003) use the mean-shift procedure (implementation of this method is available in OpenCV as CAMSHIFT [17]). Prior to that Shi and Tomasi (1994) proposed the Kanade-Lucas-Tomasi (KLT) tracker which iteratively computes the translation of a region centred on an interest point (implementation of the method in [42]). All previous are single object tracking methods, but within this type of methods, Tao *et al.* (2002) proposed an object tracking method based on modelling the whole image as a set of layers for multiple object tracking. The appearance models (templates, histograms, etc.) are usually generated online, representing the most recent object information. If, for example, an object appears from a different view this appearance model may become incompatible with the current appearance. Multiview appearance models (learned offline) can therefore be used to overcome this problem. Two examples of this approach includes the one proposed by Black and Jepson (1998) that used a eigenspace to compute the affine transformation from the current image of the object to the image reconstructed using eigenvectors, and Avidan (2004) that used a Support Vector Machine (SVM) classifier.

Both point and kernel representation forms are suitable to approximate the motion of rigid objects. Tracking non-rigid objects requires nonetheless most descriptive representations like the object's silhouette. In this case, tracking is performed by estimating the object region in each frame and describing it using appearance density and shape models, usually in the form of a colour histogram, object edges or the object silhouette. These methods are designated *silhouette tracking* methods, and can either follow a shape matching or silhouette evolution approach. Two types of silhouette tracking are possible: *shape matching* and *contour tracking*.

silhouette tracking

Shape matching approaches search for the object silhouette in the current frame. Huttenlocher *et al.* (1993) performed shape matching using an edge-based representation and the Hausdorff distance, while Sato and Aggarwal (2004) proposed to generating object tracks by applying Hough transform in the velocity space to the object silhouettes in consecutive frames. Terzopoulos and Szeliski (1992) predict the silhouette using

Kalman filtering, with the object state being defined by the dynamics of the silhouette's control points. As mentioned previously, Isard and Blake (1998) and MacCormick and Blake (2000) used particle filters for silhouette prediction. On the other hand, Chen *et al.* (2001) rely on JPDAF to estimate the state transition probabilities of an HMM. In contrast to shape matching approaches, contour tracking approaches, evolve an initial contour to its new position in the current frame by either using the state space models or direct minimisation of an energy functional using greedy methods or gradient descent. The contour energy is defined in terms of temporal information in the form of either optical flow (Bertalmio *et al.*, 2000), or appearance statistics generated from the object and the background regions (Yilmaz and Shah, 2004).

Table 4.1 summarises the most relevant visual tracking approaches found in the literature.

4.2.1 Multi-camera object tracking

In typical visual surveillance or ambient intelligence systems, event analysis is based on the tracking and identification of visual objects across multiple camera views. Objects are often captured in more than one view and it is desirable that these multiple instances of the same visual object can be automatically identified. Consider the following visual surveillance scenario: a system operator identifies a suspicious person crossing an area covered by a given camera and would like to be informed of where and when that person was previously captured by the system. To accomplish this, the surveillance system would have to track that person from the first moment it was captured by a camera and across all cameras whose field of view overlaps the person's path. The captured visual objects can be as diverse as people walking, riding a bicycle or doing other activities, cars crossing a road, etc. The tracking of multiple visual objects in one (usually static) view is a classic vision problem that has received much attention and finds application not only in surveillance systems but also in other machine vision scenarios, such as robotics. Incidentally, multiple-view tracking has only recently received much research activity. The main advantages of using many cameras for tracking in surveillance scenarios (Foresti *et al.*, 2005; Wu *et al.*, 2003) are: an arbitrarily large coverage of any given area since, for most environments, a single camera is not able to provide adequate coverage; and, tracking performance improvement, especially in critical areas and where more robustness against occlusion is desirable (Mittal and Davis, 2003).

While a single-camera tracker searches for correspondences only between frames, the task of a multi-camera tracker is also to establish correspondences between observations of objects across cameras. The ultimate goal is to correctly tag all instances of the same visual object at any given location and at any given time instant.

Specific models can add constraints that simplify this task. For example, for human tracking, Hu *et al.* (2004a) rely on the definition of principal axis. Moreover, for generic

	Method	Type Handling	Occlusion Handling	References
point tracking	· Deterministic			
	Greedy Exchange (GE)	multiple	no	Sethi and Jain (1987)
	Modified GE(MGE)	multiple	yes	Salari and Sethi (1990)
	Multiframe correspondence	multiple	yes	Shafique and Shah (2005)
	· Statistical			
	Kalman filter	single	no	Bar-Shalom (1988)
	JPDAF	multiple	no	citetBarShalom1988
	MHT	multiple	yes	Cox and Hingorani (1996)
kernel tracking	· Templates and density-based appearance models			
	Mean-shift	single	partial	Comaniciu *et al.* (2003)
	KLT	single	partial	Shi and Tomasi (1994)
	Layering	multiple	yes	Tao *et al.* (2002)
	· Multiview app. models			
	EigenTracker	single	partial	Black and Jepson (1998)
	SVM	single	partial	Avidan (2004)
contour tracking	· Contour evolution			
	State-space model			
	Kalman filter	single	no	Terzopoulos and Szeliski (1992)
	Particle filter	multiple	no	Isard and Blake (1998)
		multiple	yes	MacCormick and Blake (2000)
	JPDAF	single	no	Chen *et al.* (2001)
	Gradient descent			
	Temporal gradient	single	no	Bertalmio *et al.* (2000)
	Region statistics	multiple	yes	Yilmaz and Shah (2004)
	· Matching shapes			
	Template matching	single	no	Huttenlocher *et al.* (1993)
	Hough transform	single	yes	Sato and Aggarwal (2004)

Table 4.1: *Summary of visual object tracking algorithms.*

feature matching

object tracking, a commonly used method is to find correspondences using *feature matching*. Matching colour or other features may be performed statistically using, for example, Kalman filtering (Utsumi and Ohya, 2000) or Bayesian network inference (Nillius *et al.*, 2006; Qu *et al.*, 2000). It is also possible to use camera calibration information to learn more about the camera geometry and derive additional constraints. Cai and Aggarwal (1999) use relative calibration between cameras and define correspondences using a set of feature points in a Bayesian probability framework. Different camera properties and illumination variations can contribute to different appearances of the same object in different cameras. To overcome this, Javed *et al.* (2005) proposed learning the subspace of inter-camera brightness transfer functions. Ross *et al.* (2007), on the other hand, presented a method that incrementally learns a subspace representation, adapting itself to appearance changes.

3D model

An alternative (or complement) to the previous method is to use a priori knowledge of the captured area's geometry, building a *3D model*. Multiple cameras are used to recover the homographic relations between each camera view. In this case it is possible to find correspondences by projecting the location of each object in the world coordinate system, and establishing equivalences between objects that project to the same location at the same time. Likewise, equivalences between views are established by linking views that have similar projected 3D location. For example, Black *et al.* (2002) use this method as well as a combined 2D/3D Kalman filter for object tracking. Creating the 3D model is not always a simple task, and therefore these methods are mostly suitable for controlled environments. Alternatively, *alignment*-based approaches rely on recovering the geometric transformation between cameras automatically. This can be done using spatial image alignment methods and incorporating time information (Caspi and Irani, 2000) or by matching motion trajectories in different cameras (Lee *et al.*, 2000). Khan and Shah (2003) propose finding the limits of the field of view of each camera that are visible by other cameras. Also, Zhao *et al.* (2005) propose a ground-based fusion method for camera handover using space-time constraints and stereo segmentation. However, using alignment requires overlapping fields of view which is not always feasible. To avoid using overlapping field of views, cameras are located in non-overlapping locations that nonetheless allow establishing path dependencies between them using probabilistic models (Javed *et al.*, 2003; Kettnaker and Zabih, 1999). On the other hand, Madden *et al.* (2007) relies on feature matching to find track correspondences across multiple independent cameras.

alignment -based

path dependency

In summary, most approaches of multi-camera tracking have one of the following rationales: *different angles of the same area*, to improve tracking performance in cluttered environments; *different but overlapping areas*, simplifying the camera handover of tracked objects; or *non-overlapping areas with some path correlation between them*, that can be previously defined or automatically learned.

	Method	Type	References
feature matching	Kalman filtering	DA	Utsumi and Ohya (2000)
	Bayesian network inference	DA	Nillius *et al.* (2006)
		DA	Qu *et al.* (2000)
	Bayesian probability framework using calibration between cameras	DA	Cai and Aggarwal (1999)
	Learn subspace of inter-camera brightness transfer functions	DA	Javed *et al.* (2005)
	Learn incrementally subspace representation	DA	Ross *et al.* (2007)
	Find track correspondences for independent cameras	I	Madden *et al.* (2007)
3D model	Combined 2D/3D Kalman filter	O	Black *et al.* (2002)
alignment-based	Incorporate image alignment and time information	O	Caspi and Irani (2000)
	Match motion trajectories in different cameras	O	Lee *et al.* (2000)
	Find FOV limits	O	Khan and Shah (2003)
	Fusion method using space-time constraints and stereo segmentation	O	Zhao *et al.* (2005)
path dependency	Probabilistic models	NO	Kettnaker and Zabih (1999)
		NO	Javed *et al.* (2003)

Table 4.2: *Summary of visual object multi-camera tracking algorithms. Another classification of most multi-camera tracking approaches is the underlying rationale: different angles of the same area (DA), to improve tracking performance in cluttered environments; different but overlapping areas (O), simplifying the camera handover of tracked objects; non-overlapping areas with some path correlation between them (NO), that can be previously defined or automatically learned; and, independent areas (I).*

4.3 Scalable object recognition using local features[§]

In many situations it is important to keep track of persons or other video objects across independent areas, regardless of the time of capture, where it is not possible to rely on the techniques mentioned in the previous section. For example, in a surveillance scenario with multiple independent cameras we need a method that can reliably represent and identify the different visual objects. Since the number of objects is unknown and existing objects can change their appearance, an incremental learning scheme is also required. In this section we describe the proposed methodology to match tracked objects across independent views, that can be scalable in two distinct dimensions: an undetermined number of new objects can be added at any given instant and, existing object representations can be updated to reflect changes in time. At the same time, it needs to maintain performance at acceptable levels. As previously stated, when trying to find correspondences between objects in independent views, methods like alignment and path prediction are not suitable. Feature matching is therefore the best alternative.

Our work can be seen as a complement to classic tracking techniques rather than a direct alternative. It takes as inputs the results obtained by an object tracking algorithm and establishes correspondences between objects, independently of their location and the time of capture. Whereas most multi-camera tracking methods rely on geometry models, overlapping fields of view or on correlated paths between cameras, our method does not assume any of these priors. A recent proposal by Madden *et al.* (2007) finds some common points with our work, namely the rationale of relying on feature matching to find track correspondences across multiple cameras. It also considers as inputs the segmentations and tags of each object track detected system-wide but a different representation is used. Scalability issues also are not approached, namely the storage of descriptors of increasing number of objects of tracks. With a very large number of tracks, the matching would require a one-to-one comparison between tracks which may become easily unfeasible; in the article the authors only present results for a small number of objects/tracks.

Object matching using appearance features is a well studied problem, especially in the context of image retrieval. The joint use of interest point detectors and local descriptors for object detection, recognition and classification has grown significantly (Lowe, 2004). Building on top of these descriptors, Sivic and Zisserman (2003), Willamowski *et al.* (2004), and more recently Quelhas *et al.* (2007), showed the usefulness of relating image invariant local descriptors to visual words, or *visterms*. Sivic and Zisserman (2003) successfully applied this approach to retrieve shots from movies. They proposed a description scheme where descriptors are extracted from local affine invariant regions and quantised in *visterms*, reducing noise sensitivity in matching. Objects are then matched and frames with similar content (i.e., visual objects) can be retrieved ef-

[§]This section is based on the article *Video object matching across multiple independent views using local descriptors and adaptive learning* published in the Pattern Recognition Letters. (Teixeira and Corte-Real, 2009)

ficiently using inverted files. Nevertheless, this work still proposes solving a problem closely related to classic image retrieval and does not take into account inherent constraints of real-time video capturing systems, such as scalability – namely storage and computational restrictions. We use local descriptors and the *bag-of-visterms* paradigm for object representation to match tracks detected in a visual surveillance scenario. For this specific case, we are usually interested only in the foreground objects that are captured by the system. Taking this into consideration, two differences arise when comparing this work to others like the one by Sivic and Zisserman (2003): (1) descriptor extraction can be restricted to the foreground, which can greatly increase speed and (2) each object representation can change over time – for instance, a person can have different appearances, depending on the capture angle – which implies updating each object model continuously.

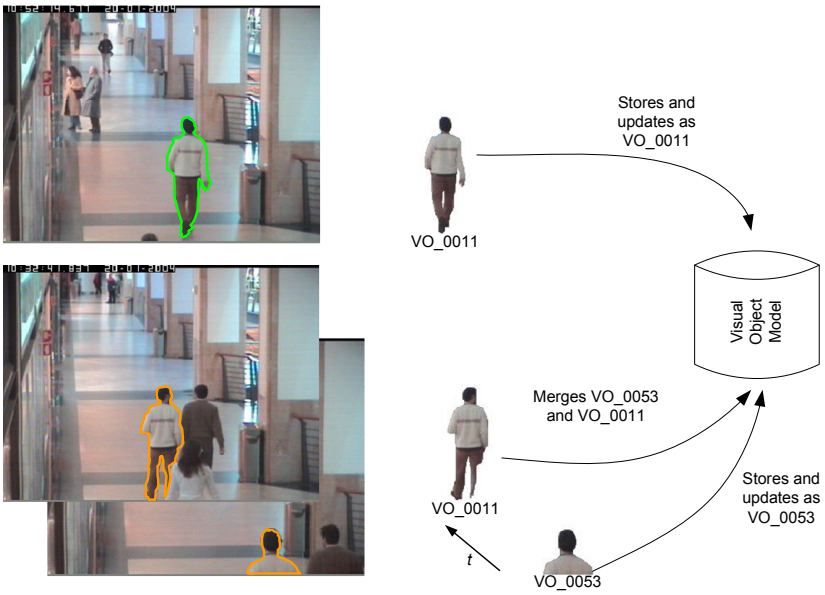
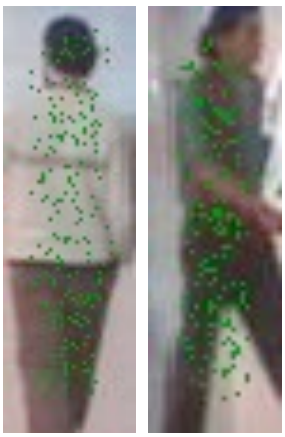


Figure 4.2: *Typical scenario handled by the visual object matching system. The same person is tracked at different instants, although captured by the same camera; due to incomplete appearance information it is labelled incorrectly but as new information is acquired the label is corrected.*

Figure 4.2 depicts an example of a typical scenario we would like to address: one person is detected and tracked; its description is extracted and stored and an identification is assigned; in another instant, the same person is again detected and tracked; even if initially a different identity is assigned, when more information is processed, it is correctly identified. This process involves three distinct steps: (1) obtaining an object description, (2) creating and updating a model of the objects, and (3) classifying the object, given its captured track.

4.3.1 Describing visual objects

Choosing a representation scheme for visual objects which is compact and representative at the same time is important to assure a feasible solution to the object matching problem. The descriptors used to represent objects need to accommodate different appearances – due to different camera characteristics, capture position, illumination variations, etc. Madden *et al.* (2007) addressed this by a compact colour histogram representation dubbed Major Colour Spectrum Histogram Representation (MCSHR) which describes the object's main colours. An incremental MCSHR (IMCSHR) is computed over a period of time to compensate for small, short-term changes in the object's pose. Finally, a transformation is applied to IMCSHR to compensate for illumination variations. The MCSHR description scheme is further discussed and evaluated in Section 4.3.6. As an alternative to the representation scheme used in our method, we also implemented MCSHR and evaluated it on the dataset presented in detail in Section 4.3.6.



(a) `Person01` (b) `Person12`

Figure 4.3: *Example frames with superimposed points defined by the random point selector.*

Local descriptors are widely used for image retrieval applications, and we now propose to apply local descriptors to object track matching. For each object frame comprising the full object track, a set of descriptors is extracted in specific keypoints. These points are defined by an interest point detector (number of points depends on the image content) or by a random point selection (number of points is fixed and pre-defined). For object matching and classification, the number of words extracted from the image is the most important factor influencing performance (Nowak *et al.*, 2006). Since objects' images are usually very small, interest point detectors tend to select an insufficient number of points to efficiently represent them. We used therefore random point selection from a pyramid with regular grids, as suggested by Nowak *et al.* (2006), since

this method provides a dense representation of each object image. In Figure 4.3 we show example frames for two visual objects with superimposed points defined by the random point selector.

A 128-dimensional vector is extracted in each keypoint using the SIFT descriptor developed by Lowe (1999). The descriptor is then quantised to form visual words using a pre-defined vocabulary. Features obtained by SIFT are invariant to image scale, rotation, and robust to changes in viewpoints and illumination. For most types of applications, it compares favourably with other local descriptor schemes (Mikolajczyk and Schmid, 2005). When affine invariance is required, SIFT is more prone to perform worse than more complex affine-invariant descriptors. A more thourough description of SIFT can be found in appendix, Section C.2.

4.3.2 Building the vocabulary

The Bag-Of-Visterms (BOV) concept derives from the bag of words model commonly used in natural language processing to represent documents. It ignores words order and allows a vocabulary-based modelling of a document. Similarly, with BOV, images are modelled by a set of visual words, or visterms, according to a previously defined vocabulary.

Most BOV approaches rely on k-means clustering of local descriptors to create a vocabulary. The descriptor vectors of a training set are extracted and quantized into a pre-defined number of words. Nistér and Stewénius (2006) proposed an adaptation to this approach which, instead of creating a "flat" vocabulary, creates a hierarchical relation of visual words in the form of a *vocabulary tree*. This allows a more efficient search which, in turn, enables the use of large vocabularies. In our work, using a large vocabulary is a key factor since we will not be using any information of the geometric layout of visual words extracted from an image. On the other hand, given the rate at which frames are acquired and analysed, an efficient search of visual words is also required. This trade-off needs to be taken into consideration when building the vocabulary.

To build the vocabulary tree, an initial k-means clustering is first run on the training data, defining k cluster centres. The data is then partitioned into k groups, where each group consists of the descriptor vectors closest to a particular cluster centre, forming quantisation cells, or *words*. The same process is then recursively applied to each group of descriptor vectors, splitting each quantisation cell into k new parts. The vocabulary tree is created level by level, up to a maximum number of L levels.

When a new object's image needs to be classified, each extracted descriptor is propagated down the tree by comparing the descriptor vector at each level to the k candidate cluster centres and choosing the closest one. At each level k dot products are performed, resulting in a total of kL dot products. If k is not too large, the vocabulary tree can be very efficient, compared to an equivalent "flat" vocabulary defined by the

total number nodes M in the tree, which is given by:

$$M = \sum_{i=1}^{L} k^i = \frac{k^{L+1} - 1}{k - 1} - 1. \tag{4.1}$$

To account for different relevancies of tree nodes, a weight w_i is assigned to each node, and is defined by:

$$w_i = \ln \frac{N}{N_i} \tag{4.2}$$

where N is the total number of images in the model and N_i is the number of images having at least one path through node i, i.e., containing the word represented by that node.

The training process of the tree uses a large set of descriptor vectors in an unsupervised fashion. Two data sources have been tested to build the vocabulary tree: (1) the segmented objects from the dataset's sequences (defined in Section A.3) as training data forming a specific vocabulary $\mathcal{V}_s$ and (2) object frames from a different source as training data forming a generic vocabulary $\mathcal{V}_g$. By using both vocabularies it is possible to assess the performance impact of having a generic vocabulary to represent a wide variety of objects.

□ Descriptor vector

All extracted 128-dimensional SIFT vectors are quantised in visual words using the vocabulary tree to define the final descriptor vector. For a given object c, its segmented image I_t^c at instant t is represented by:

$$v(I_t^c) = \{x_1, x_2, \ldots, x_i\} i \in 1, \ldots, M \tag{4.3}$$

where M is the number of words in the vocabulary, given by equation (4.1). Each element x_i of (4.3) is the weighted histogram of the words defined by the vocabulary; x_i is therefore given by:

$$x_i = n_i w_i \tag{4.4}$$

where n_i is the number of input descriptors containing the visual word i and w_i is the weight defined by equation (4.2). Unless stated otherwise, all experiments used a vocabulary with $k = 10$ and $L = 4$, resulting in a representation vector of size $M = 11110$.

4.3.3 Adaptively updating and learning object models

In order to effectively update the previous model as new data becomes available, the model should (1) be compact, that would not imply having all previous history stored, and (2) be scalable, that would imply classifying visual objects that change in time, while also having the ability to learn new ones. Recently, some methods have been proposed to solve this problem, such as Learn++ (Polikar *et al.*, 2001). Learn++ draws its inspiration from AdaBoost, which in turn relies on an ensemble of classifiers trained using adaptive bootstrap techniques. It is iteratively updated by new sets of data, possibly containing new classes. A modification called Learn++.MT, proposed by Muhlbaier *et al.* (2004), improves the performance when new classes are added. Learn++.MT meets the compact and scalable requirements, and was used to create our visual object model. A brief description of it follows.

□ Learn++.MT algorithm

For each new dataset $\mathcal{D}_k$, the inputs to Learn++.MT are: (1) a sequence of training data instances x_i and their correct labels y_i, (2) the classification algorithm BaseClassifier, and (3) T_k, the maximum number of classifiers.

As in typical boosting learning algorithms, data is drawn according to a distribution D_t. For the first set, D_t is initialised as a uniform distribution. From the second set onwards, this distribution is updated according to the performance of the ensemble on the new data. T_k classifiers are added to the ensemble when a new dataset is added. For each new classifier, a subset of $\mathcal{D}_k$ is drawn, according to D_t, and evaluated against the new classifier to obtain the hypothesis h_t. The classifier error is estimated by $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ and if $\epsilon_t > \frac{1}{2}$, a new subset is drawn, discarding the classifier. A dynamic weight voting (DWV) algorithm is then called to obtain a composite hypothesis H_t. It represents the ensemble decision of all classifiers trained until now. The distribution D_t is updated according to the performance of H_t. This process is repeated until all new T_k classifiers have been trained.

The essential difference from Learn++.MT to its parent method is the voting scheme. As in Learn++, voting is based on the weights assigned to each classifier but with DWV these weights are modified according to the classification of the specific testing instance. This is achieved by adjusting weights of classifiers that have not been trained with a given class. The adjustment is proportional to the ensemble's confidence on that class.

4.3.4 Classifying objects

The goal of object classification is to attribute a known label to each input object frame comprising that object's track. This can be seen as a multi-class classification problem,

with a variable number of classes. As described previously, we use the Learn++.MT algorithm to handle class variability, and opted to use SVMs (Burges, 1998) as the respective BaseClassifier.

SVMs are commonly used in machine learning problems, especially with large dimensional input spaces – which is the case for the object classification problem. Standard SVMs rely on margin optimisation to learn a decision function $h(x)$, such that, if x belongs to the target class, $h(x)$ is large and positive; otherwise, $h(x)$ is negative. SVMs are thus binary classifiers but can be adapted to multi-class problems; we opted for the one-against-one approach, where $n(n-1)/2$ models are constructed for a n-class problem. Additionally, we used linear kernel SVMs, mainly for processing speed purposes. Specific kernels could alternatively be used, such as the pyramid match kernel proposed by Grauman and Darrell (2005) for multi-resolution histograms.

4.3.5 Building the system

In summary, the main steps involved in the proposed solution for the surveillance scenario from Figure 4.2 are: (1) detect objects, including segmenting and tracking them for each view, (2) obtain a compact object description, yet sufficiently discriminative, and (3) compare the object with the current visual object model, updating it if appropriate. A block representation of these steps is shown in Figure 4.4. While the first step is performed independently for each camera/view, the third step aggregates inputs from all views. The second step, on the other hand, combines object description which is performed independently for each view, and a common part to all views. This common part consists of obtaining a histogram representation of the *visterms* found in each object. A vocabulary of visterms, with which object descriptors are compared, is obtained a priori and is kept unchanged.

Multiple appearances of an object are detected by matching a newly tracked object with the visual object model. The matching process takes the object tracks (i.e., sequences of segmented images) detected by any given tracking algorithm as inputs and finds multiple object occurrences. From an application point of view, it would act as a linking mechanism between sequences. An operator analysing a sequence containing an object would have the ability to know every other appearance of that object, and the possibility to jump to another sequence, pinpointed by time and location. Numerous other possibilities would than be possible, like using the previous knowledge of camera location (if any) to represent graphically the known path of an object/person.

4.3.6 Validation

In order to assess the performance of our methodology, a dataset $\mathcal{D}_{30}$ was created. More information about this dataset can be found in appendix, Section A.3.

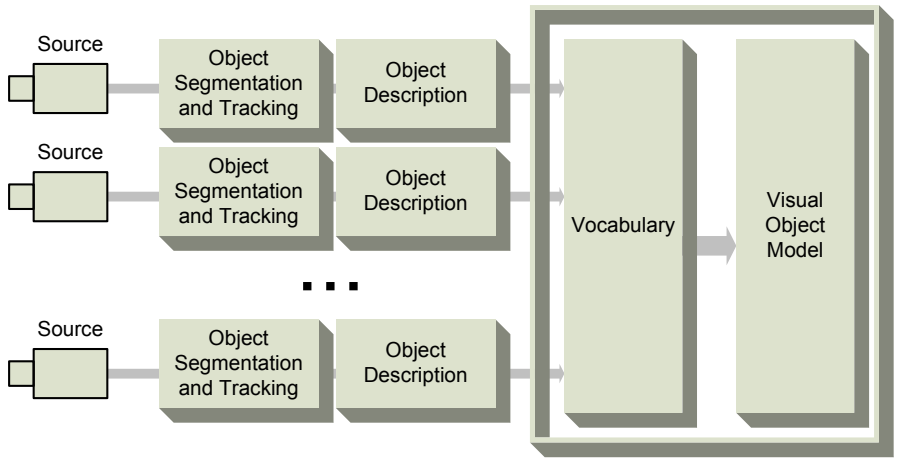


Figure 4.4: *Block representation of the proposed method. Objects detected in each view are first segmented and tracked; a description is extracted and quantised by a common global vocabulary; finally, the visual object model is updated accordingly.*

□ Discriminative capability

To test the discriminative capability of the representation scheme, $\mathcal{D}_{30}$ was divided in random training (fixed size) and validation (remaining) set partitions. We tested two representation schemes: MCSHR (Madden *et al.*, 2007) and bag-of-visterms using a vocabulary tree defined from SIFT descriptors, which we designate hierarchical bag-of-visterms HBOV. Both consist of a descriptor vector for each image but, while Hierarchical Bag-Of-Visterms (HBOV) defines a fixed-size vector, MCSHR's size is variable. Due to this, evaluation consisted of a majority voting classification procedure. Each descriptor vector extracted from the test set images is compared with the training set descriptor vectors and votes are cast for the 10 closest matching, using a 1-norm distance metric. Results are obtained using a 5-fold cross validation. The SVM-based classification method described in section 4.3.4 will only be used later.

The vocabulary for the HBOV was trained with dataset $\mathcal{D}_{30}$, i.e., with the same dataset to be described by the vocabulary. We tested two different vocabulary tree sizes: 125 leaf visual words ($k = 5$ and $L = 3$) and 10000 leaf visual words ($k = 10$ and $L = 4$). The overall results are shown in table 4.3.

The HBOV representation presents much better results, even with a descriptor size similar to the ones obtained with MCSHR. If we increase the vocabulary size, and hence the descriptor size, the results are further improved. Moreover, classification performance with MCSHR is less resilient to similar objects, as denoted in the confusion matrix shown in Figure 4.5(a) in opposition to Figure 4.5(c). With HBOV, and especially with a larger vocabulary, all classes present a very high classification performance, which clearly shows both the discriminative capability and stability of the represen-

Vocabulary	Classification rate	Descriptor size
MCSHR	71.6 (17.7)	variable, ~ 200
HBOV-125	82.9 (13.1)	fixed, 155
HBOV-10000	95.0 (5.2)	fixed, 11110

Table 4.3: *Average classification rate and standard deviations. Results using MCSHR and HBOV with different descriptor sizes are shown.*

tation scheme described in Section 4.3. The MCSHR scheme is used as a comparison baseline for the remaining experiments.

Since it is required that the descriptors are extracted in the shortest amount of time, it is also important that MCSHR evaluate the time cost. For non-optimised implementations, while MCSHR descriptors can be extracted at a rate of approximately 3.7 typical track images per second[1], HBOV-125 can be processed at 2.9 images per second and HBOV-10000 at 2.7 images per second. As expected, due to higher complexity, the HBOV extraction process is slower but nevertheless an optimised real-time implementation is equally feasible. Note that most of the time consumed in obtaining a HBOV description for an image is spent extracting the SIFT descriptor vector ($\approx 2/3$ of the total time). Due to this, the difference between HBOV-10000 and HBOV-125 is not very significant.

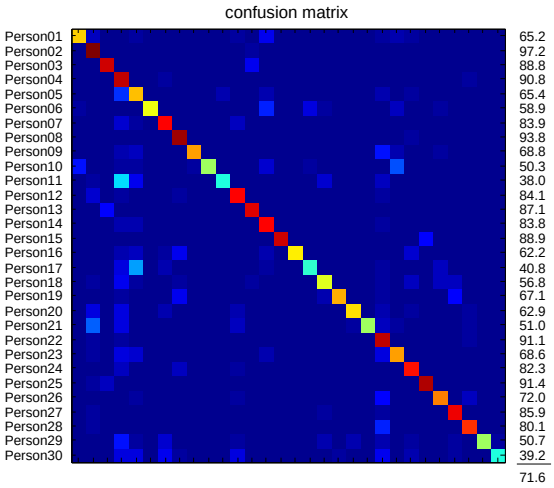
Despite a relatively high time cost, this method meets the needs of a real-time surveillance system since an optimised implementation could explore parallelization, redundancy of track images in short periods of time (< 1s) and inactivity.

□ Impact of using a generic vocabulary

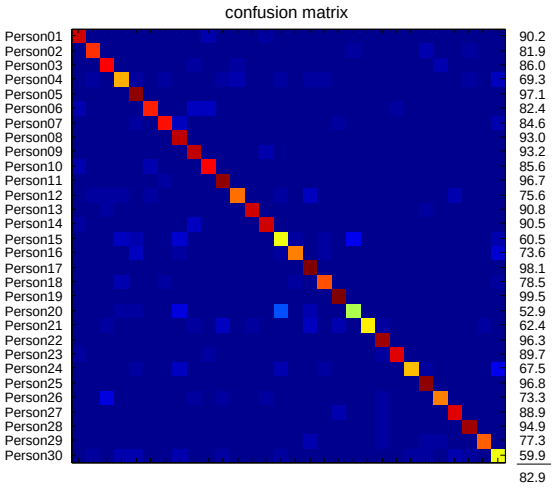
The appearance of objects captured by a surveillance system is typically not known a priori. Hence, it is useful to create a vocabulary from images that are not directly related to the captured visual objects. We analyse the performance of such generic vocabulary with a similar setup as before. For each image of the generic dataset $\mathcal{D}_g$ 128-size SIFT descriptors were extracted in the points defined by a Laplacian of Gaussians (LoG) interest point detector (Lowe, 2004). The input for the vocabulary tree construction algorithm described in Section 4.3.2 consisted of the concatenation of all descriptor vectors from all images.

The dataset $\mathcal{D}_{30}$ was again divided in random training and validation partitions. In Table 4.4 we show the average performance of different types of vocabularies using in each case a single multi-class SVM. While the first four vocabularies were created with dataset $\mathcal{D}_{30}$, the last four were created with the generic dataset $\mathcal{D}_g$. For both groups we tested four different vocabulary sizes: 125 leaf words, with $k = 5$ and $L = 3$; 625

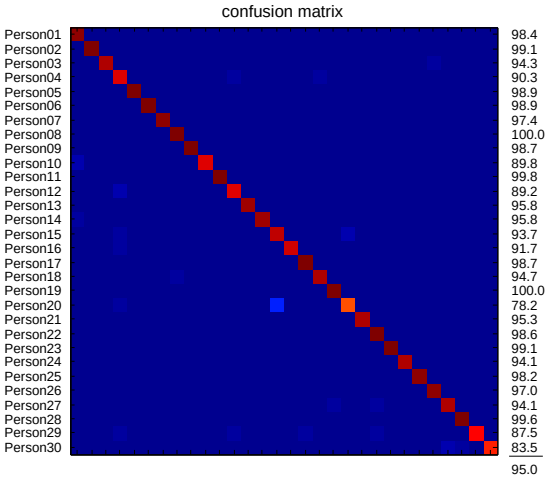
[1]Results obtained with a C++ implementation running on a Pentium IV 3.4GHz with 1GB of RAM



(a) MCSHR



(b) HBOV-125



(c) HBOV-10000

Figure 4.5: *Confusion matrices of the object classification. Visual representation of confusion matrices for different representation schemes.*

leaf words, with $k = 5$ and $L = 4$; 1000 leaf words, with $k = 10$ and $L = 3$; and, 10000 leaf words, with $k = 10$ and $L = 4$. Results show that the use of a generic dataset to train the vocabulary does not affect significantly its performance.

Vocabulary	Classification rate
baseline	71.6 (17.7)
$\mathcal{V}_s^{125}$	82.9 (13.1)
$\mathcal{V}_s^{625}$	92.7 (7.1)
$\mathcal{V}_s^{1000}$	93.6 (6.3)
$\mathcal{V}_s^{10000}$	95.0 (5.2)
$\mathcal{V}_g^{125}$	78.7 (14.8)
$\mathcal{V}_g^{625}$	91.4 (7.9)
$\mathcal{V}_g^{1000}$	91.6 (6.4)
$\mathcal{V}_g^{10000}$	93.3 (6.2)

Table 4.4: *Average classification rate and standard deviations. Results using different vocabularies are shown – specific $\mathcal{V}_s$ and generic $\mathcal{V}_g$ vocabularies, with tree sizes of 125, 625, 1000 and 10000 leaf words.*

All experiments henceforth use $\mathcal{V}_g^{10000}$. Also, the SVM-based model trained with $\mathcal{D}_{30}$, using $\mathcal{V}_g^{10000}$, will be designated $\mathcal{M}_{30}$.

□ Object representation adaptability

In the previous subsection each person's data used to train the model consisted of information collected from a single track. However, it is important to know how the classification performance stands with new captured tracks. For that purpose we used $\mathcal{M}_{30}$ model to classify several different tracks. Table 4.5 shows the results obtained for 8 new different tracks of 6 different persons. The first three new tracks have a similar viewing angle to the tracks used for training; the five other new tracks have a very different viewing angle and most of the time only the person's profile is visible (an example of both camera angles is shown in Figure A.4). The classification performance using the baseline method is also presented. Unsurprisingly, the classification rate using the previous model $\mathcal{M}_{30}$ (second column) is much higher for the first set of new tracks. However, if we update the model with a random sample containing 50 images from each new track and retrain the model also with these samples, the results improve significantly (third column).

These results show that, given a correct model update, the visual object representation effectively adapts to the object's changes in time. Note also that the performance of the baseline method has a high variability, presenting either a high or very low clas-

	person ID	baseline	previous model $\mathcal{M}_{30}$	updated model	# of frames
similar view angle	01	57.5 (5.1)	76.8 (0.8)	93.4 (2.9)	331
	12	84.3 (2.6)	71.6 (1.0)	95.9 (2.3)	293
	20	11.9 (3.9)	43.1 (0.5)	83.4 (2.2)	422
different view angle	01	19.6 (8.4)	50.1 (0.6)	89.6 (3.7)	280
	06	14.1 (1.5)	12.7 (1.7)	97.2 (2.5)	86
	12	70.9 (6.1)	12.0 (1.3)	83.8 (4.7)	272
	29	24.5 (2.5)	17.5 (1.4)	93.1 (3.6)	146
	30	14.7 (1.9)	26.8 (1.4)	94.4 (2.3)	176

Table 4.5: *Classification rate of different tracks using $\mathcal{M}_{30}$. Results are obtained for (1) the baseline, (2) the model $\mathcal{M}_{30}$ trained previously and (3) an updated model, including a sample of the new track.*

sification performance. This confirms the conclusions drawn previously regarding its discriminative capability.

□ Resilience to incorrect segmentation

Until now, all objects were tested with the respective segmentations defined by the ground-truth's bounding box. In a real scenario the only information about the object's location is the segmentation obtained using for example a background subtraction method. However, with such automatic segmentation methods errors occur more often. An example of an incorrect segmentation is shown in Figure 4.6. The segmentation shown in green is very different from the true segmentation. It suffers from false positives errors – the person shadow to the left – as well as false negatives – part of the person is not correctly classified. Also, part of the same person is merged with the segmentation shown in red. It is clear that segmentation errors will inevitably penalise the overall performance. In order to assess the effect of imperfect segmentation on our classification performance, we used the segmentation algorithm proposed in Section 3.3 and classified the segmented objects according to the ground-truth.

In Table 4.6 the confusion matrix of a model trained and tested with the images obtained with background subtraction algorithm is shown. The model consists only of the 4 classes of objects considered in this scenario and in this case a good discrimination is achieved.

If we test the object images extracted with the background segmentation algorithm with $\mathcal{M}_{30}$ model, results deteriorate. In Table 4.7 we summarise the results obtained with this model and the baseline, both using as inputs the segmentation algorithm and

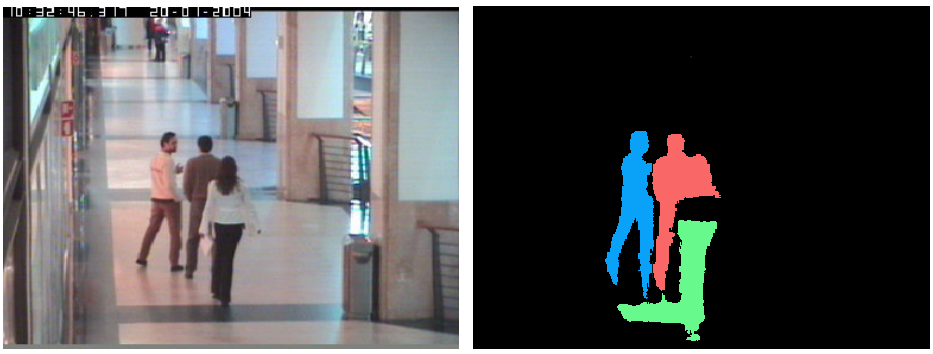


Figure 4.6: *Incorrect segmentation example. From left to right:* Person01 *(blue segmentation),* Person20 *(red segmentation),* Person06 *(green segmentation).*

		estimated class				# of
		01	05	06	20	frames
true class	01	96.8	0.0	3.0	0.1	356
	05	1.7	98.0	0.1	0.2	601
	06	1.1	0.8	96.5	1.6	390
	20	1.1	0.7	0.2	98.0	1246

Table 4.6: *Confusion matrix of the tracks segmented using a background subtraction algorithm.*

the ground-truth. Despite a smaller classification rate of the first 3 visual objects, all are successfully discriminated. The baseline representation method shows a significantly worse performance, probably due to its less resilience to occlusions and incorrect segmentation of objects. Interestingly, the results for Person20 are better probably due to a good overall track segmentation. In theory, with a perfect segmentation, a better object description is achieved since the background is removed while the object is kept, but this is often not the case.

□ Incremental model learning

One of the requirements defined earlier for the object model was that it should be adaptable to changes. In a visual surveillance scenario the objects' appearances can vary, depending on the viewing angle they are captured. We need therefore a way to incorporate these changes in the model. The straightforward approach to accommodate this variability of incoming data is to recreate a new model whenever a new set of data is available (similar to the approach tested in Section 4.3.6). When, for instance, new data becomes available, the previous model is forgotten and a new model is

person ID	segmentation		ground-truth	
	baseline	$\mathcal{M}_{30}$	baseline	$\mathcal{M}_{30}$
01	51.5 (2.4)	95.1 (0.2)	65.2 (2.1)	97.5 (1.5)
05	39.4 (3.5)	94.9 (0.5)	65.4 (1.7)	99.0 (0.1)
06	30.8 (1.7)	96.4 (0.5)	58.9 (1.9)	98.8 (0.8)
20	35.0 (2.1)	76.1 (0.1)	62.9 (2.3)	73.5 (2.5)

Table 4.7: *Average classification rates and standard deviations. Results using the segmented tracks (second and third columns) and the ground-truth (fourth and fifth columns) are shown. For each case, results are shown for the baseline and the $\mathcal{M}_{30}$ model.*

trained with that data, as well as with previous data. This approach, despite being simple, is unfeasible in real systems since it requires ever increasing storage resources, as well as computational resources. Nevertheless, this approach provides a performance reference to which our incremental learning will be compared to. The experimental setup consisted of training a model based on a multi-class SVM for all the data available until that moment. The incoming data consists of the subsets $\mathcal{D}^i$ defined in Section A.3 and represented in Figure A.6.

The incremental learning approach described in Section 4.3.3 was tested in a similar fashion as the model rebuild approach. However, in this case, whenever a new set of data becomes available, only that set's data is provided to the classifier. The model is then updated, providing that new classes that appear are learnt and classification performance of other classes is kept. The number of classifiers added to Learn++.MT when a new set is added was of $T_k = 4$.

subset	type of model update		# of frames
	retrain all	incremental	
$\mathcal{D}^1$	98.9 (2.0)	83.9 (16.8)	1864
$\mathcal{D}^2$	97.5 (5.0)	45.5 (34.2)	2813
$\mathcal{D}^3$	98.1 (2.8)	58.0 (22.4)	3158
$\mathcal{D}^4$	95.9 (6.0)	37.7 (16.9)	2770
$\mathcal{D}^5$	91.0 (8.5)	47.7 (13.1)	1951
test set	98.0 (2.9)	52.2 (20.5)	1500

Table 4.8: *Classification rate for the simplified model update. At each new set a new model is built using all current available data. Classification performance is kept at a very high rate.*

	person ID															
subset	01	02	04	05	08	11	15	17	19	20	23	24	27	27	30	30-class result
$\mathcal{D}^1$	—	81.3	46.2	62.7	—	86.8	61.4	100.0	95.7	74.0	—	85.1	97.7	96.2	100.0	83.9 (16.8)
$\mathcal{D}^2$	4.4	72.4	38.3	49.9	2.8	87.1	50.6	84.1	16.3	39.7	72.3	41.7	91.5	99.6	73.5	45.5 (34.2)
$\mathcal{D}^3$	69.2	27.2	—	37.1	71.4	100.0	—	69.9	—	—	41.4	83.2	65.6	69.1	72.3	58.0 (22.4)
$\mathcal{D}^4$	20.2	—	—	64.3	49.4	83.4	—	63.9	—	—	38.1	37.0	22.2	—	29.8	37.7 (16.9)
$\mathcal{D}^5$	—	—	—	37.4	—	66.7	—	38.5	—	—	21.7	24.4	—	—	23.2	47.7 (13.1)
test set	30.0	74.0	54.0	80.0	42.0	96.0	50.0	82.0	72.0	32.0	30.0	52.0	52.0	82.0	56.0	52.2 (20.5)
track class.	75.0	75.0	100.0	100.0	66.7	100.0	100.0	100.0	50.0	66.7	50.0	60.0	100.0	100.0	100.0	76.2 (24.3)

Table 4.9: *Classification performance rate based on incremental learning. Each subset may not contain images from a given person and, if this is the case, no classification rate presented. The last line shows the results for track classification; track segments are associated to each set and are classified by a simple rule: the most representative visual object among that segment's frames. The overall correct track classification is 76.2%.*

We evaluated the two update methods using a 5-fold cross-validation, and the overall results are shown in Table 4.8. For each subset, the total number of frames is also shown. Comparing the classification rates, it is clear that performance drops significantly if we use an incremental model update instead of the simplified approach. With a large number of new classes appearing in each new incoming data, the model is not able to adapt as promptly. Nevertheless, the overall classification rate of object images reaches around 50%, which is an acceptable value given the number of object classes. Also, an increase of performance in the last subset suggests that with more data the model adapts adequately. Table 4.9 shows the partial results for each subset. Note that only the performance results for half of the classes are shown for brevity (no particular choice).

A better measure of performance of our system is how successfully can the system classify the whole track. In our case, track classification is performed for each time window. The window, or segment, comprises a large number of images – 150 in this case – and a simple classification rule can be applied: a segment is classified as the visual object gathering the largest amount of frames labelled as that object. The last line in Table 4.9 shows the results for track classification, with an overall correct classification rate of 76.2%. Note that this result is for the model that resulted from all incremental updates.

4.4 Person recognition

The appearance analysis proposed in the previous section has some limitations. Although for typical cases heuristics can be used to narrow the number of objects, this type of matching is only effective for a limited time window (maybe a day), since a person's appearance including her/his clothes changes. Alongside appearance specific features may also be used, and humans have distinctive features that distinguish one person from another. In this section we analyse face, gait and speaker recognition methods and discuss a possible multimodal person recognition system.

4.4.1 Face recognition

A distinctive characteristic used by humans to recognise other humans is the face. The recent evolution of automatic recognition of faces raised this technology to a level where it can now be used for both verification and identification (open-set and closed-set) of persons. In Figure 4.7 the typical process of face recognition is depicted.

The first systems followed a semi-automatic approach, where features such as eyes, ears, nose, and mouth were manually located and measures were taken automatically afterwards (an example of an early work in this area is the one by Goldstein *et al.*, 1971). In these earlier stages of automatic face recognition, Kanade (1977) also explored the

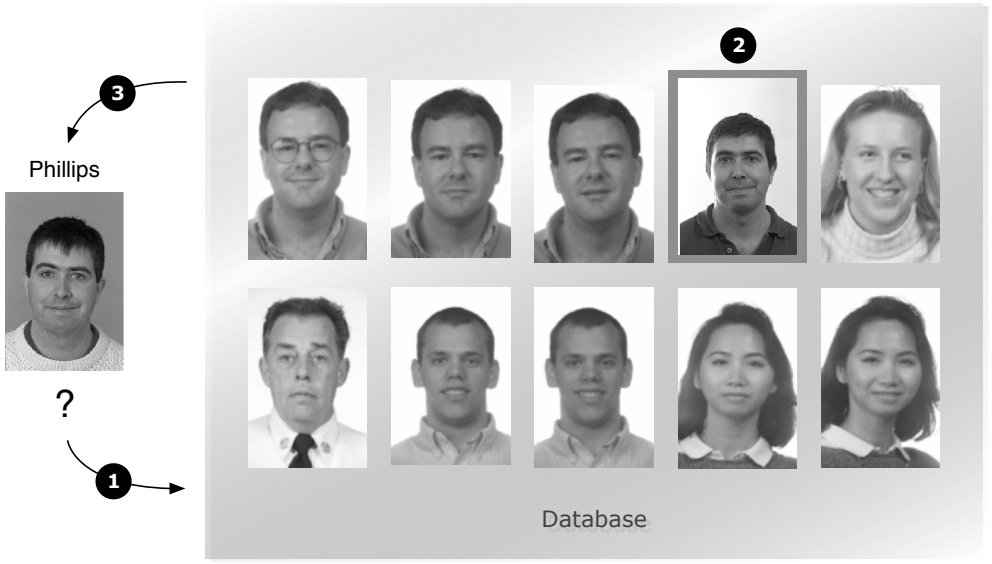


Figure 4.7: *Face recognition task. An image with unknown identity is first compared with the ones in the database using a face recognition algorithm. If the person is in the database, a match occurs and the identity is returned. The database may contain more than one image of each person, with different illumination conditions, expression and pose. (images are from the FERET database Phillips* et al.*, 1998).*

automatic location of these features. Face recognition methods can broadly be classified either as holistic or analytic approaches.

holistic methods

Holistic methods make use of the information derived from the whole face. An example is the application of the principle of Principal Component Analysis (PCA) to face recognition by Kirby and Sirovich (1990). Their work demonstrated that a relatively small number of values were required to accurately code a suitably aligned and normalised face image. The algorithm by Turk and Pentland (1991), although limited by environmental factors, enabled the creation of reliable real-time automatic face recognition systems; image face data is processed by PCA in order to reduce its dimensionality The result is a set of uncorrelated components – *eigenfaces* – which, combined through a weighted sum, represent a face. Each new image to be tested is compared with the known faces by measuring the distance between their respective feature vectors. A limiting requirement of PCA is that face images must be the same size and must first be normalised to line up the eyes and mouth of the subjects within the images. Also, although good results are achieved with full frontal faces, poor performance is obtained otherwise. Many other adaptations of this method were also proposed, namely probabilistic eigenfaces (Moghaddam and Pentland, 1997) and

modular eigenfaces (Pentland *et al.*, 1994). Belhumeur *et al.* (1997) applied Fisher Linear Discriminant (FLD)/Linear Discriminant Analysis (LDA)(Fisher, 1938). LDA defines a projection that maximises between-class variance and minimises within-class variance, defining *fisherfaces*. When dealing with high dimensional face data, this technique faces the small sample size problem that arises where there is a small number of available training samples compared to the dimensionality of the sample. For that reason PCA is usually used to reduce the feature dimension before LDA (Zhao *et al.*, 1998). Other methods were also proposed to solve the small sample size such as regularised LDA (Lu *et al.*, 2003) and direct LDA (Yu and Yang, 2001). Independent Component Analysis (ICA), that uses higher-order statistics, can be used as an alternative to PCA (Bartlett *et al.*, 2002). In order to achieve greater generalisation, learning methods like probabilistic decision-based Neural Network (NN) (Lin *et al.*, 1997) and evolutionary pursuit (Liu and Wechsler, 2000) were proposed.

analytic methods

Alternatively to holistic methods, analytic methods compare only salient facial features. The earlier approaches Kanade (e.g. 1977) described previously can be defined as analytic methods as well as more recent approaches. An example are methods based on Hidden Markov Model (HMM), that do not find the exact locations of facial features (Nefian and Hayes III, 1998). Local features are generally considered to be more robust to complex distortions of face information that can be caused by variations in illumination, facial expression and pose. Wavelet analysis, and in particular wavelets based on Gabor functions (Gabor, 1946), have demonstrated to have these properties (Shen and bai, 2006). The application of Gabor wavelets to face recognition was initially proposed by Lades *et al.* (1993) using a dynamic link architecture. Faces are represented by a rectangular graph with local features based on Gabor wavelets extracted at deformable nodes. Wiskott *et al.* (1997) extended this method to EBGM. A Gabor wavelet transform creates a dynamic link architecture that projects the face onto an elastic grid. The Gabor jet is a node on the elastic grid that results from the convolution of the image with a Gabor filter; it describes the image behaviour around a given pixel. Recognition is based on the similarity of the Gabor filter response at each Gabor node. The crucial point of EBGM for a successful recognition system is to obtain an accurate landmark localisation where Gabor jets are extracted. This is however a difficult task and can be achieved using different methods, for example combining PCA and LDA methods. Similar approaches that extract local information from salient facial features have been explored more recently, namely Line Edge Map (LEM) (Gao and Leung, 2002) and Directional Corner Point (DCP) (Gao and Qi, 2005).

Methods following other types of approaches can also be found in the literature – flexible shape models (Lanitis *et al.*, 1997), recognition based on depth-maps and surface curvature (Gordon, 1991), and 3D morphable model (Huang *et al.*, 2003), just to mention a few examples. Detailed reviews of face recognition methods can be found in (Zhao *et al.*, 2003) and (Abatea *et al.*, 2007).

	Method	References
holistic	Principal Component Analysis (PCA)	
	Eigenfaces	Kirby and Sirovich (1990), Turk and Pentland (1991)
	Probabilistic eigenfaces	Moghaddam and Pentland (1997)
	Modular eigenfaces	Pentland *et al.* (1994)
	Linear Discriminant Analysis (LDA)	
	Fisherfaces	Belhumeur *et al.* (1997),
	(PCA+LDA)	Zhao *et al.* (1998)
	Regularized LDA	Lu *et al.* (2003)
	Direct LDA	Yu and Yang (2001)
	Independent Component Analysis (ICA)	Bartlett *et al.* (2002)
	probabilistic decision-based NN	Lin *et al.* (1997)
	Evolutionary pursuit	Liu and Wechsler (2000)
analytic	Geometry-based methods	
	earlier methods	Kanade (1977)
	recent methods	Cox *et al.* (1996)
	Dynamic link architecture (DLA)	
	original DLA	Lades *et al.* (1993)
	EBGM	Wiskott *et al.* (1997)
	Line edge map (LEM)	(Gao and Leung, 2002)
	Directional corner point (DCP)	(Gao and Qi, 2005)
	Hidden Markov Model (HMM)	Nefian and Hayes III (1998)
other	Flexible shape model	Lanitis *et al.* (1997)
	Depth-maps and surface curvature	Gordon (1991)
	3D morphable model	Huang *et al.* (2003)

Table 4.10: *Summary of face recognition methods.*

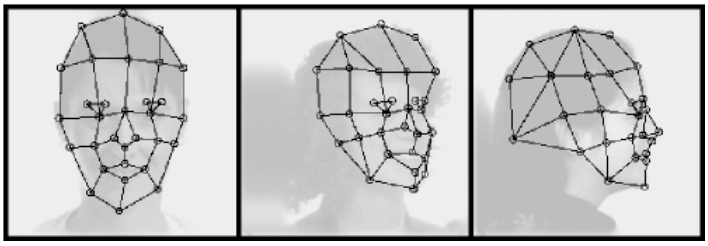


Figure 4.8: *Elastic Bunch Graph Matching (EBGM). The grid nodes are positioned automatically using the EBGM algorithm. Note that the grids adapt to different poses. (image from Wiskott* et al.*, 1997).*

4.4.2 Gait recognition

The way a person usually walks is a distinctive feature that may be used for recognition. Gait is an idiosyncratic characteristic (Johansson, 1975; Troje *et al.*, 2005) and relates to the way a person walks. It is a complex spatiotemporal biometric that can be used to distinguish individuals. The recognition based on gait analysis typically proceeds by extracting the silhouette of the walking person and by analysing that silhouette sequence over time. The interest in using gait as a biometric feature to identify a person has grown recently, especially for automated person identification systems at a distance, where for instance face information is not available in high enough resolution for recognition. Also, gait is more difficult to conceal than other biometric features (Nixon *et al.*, 1999).

The analysis of gait encompasses two components: shape and motion; most methods proposed in the literature use primarily one of them. Regarding the shape features, average silhouette is commonly used (Han and Bhanu, 2004; Veres *et al.*, 2004). Other shape-based gait features include clustered silhouette stances using HMM and dynamics-normalised shape cues (Liu and Sarkar, 2006). On the other hand, motion-based methods include those that use body part moments extracted from a human body silhouette (Lee *et al.*, 2003), eigengait space (BenAbdelkader *et al.*, 2002) and HMMs (Kale *et al.*, 2004). Even though shape-based methods were shown to outperform the motion-based methods (Sarkar *et al.*, 2005), shape suffers from greater variance within the same individual, influenced by external conditions like footwear, clothing and load carrying. Therefore, more robust gait recognition can be achieved by taking into account variations of both types of features. For instance, Collins *et al.* (2002) used shape cues like body height, width, part proportions in combination with motion cues like stride length and the amount of arm swing to improve recognition results. An alternative approach that combines information from shape and temporal variation (in essence, motion) are spatio-temporal patterns. Examples of these patterns are self similarity plot (difference values of all pairs of images) (Cutler and Davis, 2000b), frieze

patterns (projection of silhouette images along horizontal and vertical axis) (Liu *et al.*, 2004), and Gait Energy Image (GEI) (Han and Bhanu, 2006), among others.

4.4.3 Speaker recognition

Speaker recognition is the task of identifying a person from the his or her voice characteristics that are automatically extracted and analysed. A related concept is speech recognition, but whereas in speaker recognition we are interested in recognising who is speaking, in speech recognition we are interested in recognising what is being said.

Speaker recognition can be broadly used in two application areas: speaker verification and speaker identification. In this case the difference between both is more subtle. In the case of speaker verification the goal is to verify if the identity claimed by a given speaker is in fact correct. On the other hand, speaker identification occurs when the goal is to identify an unknown speaker. In our work we are interested in the latter area. Also note that, given the context of use of the speaker verification methods – typically authentication systems – their precision should be very high to avoid unwanted intrusions due to false positives. The same does not apply necessarily to speaker identification systems.

The methods for speaker recognition can be divided into text-dependent and text-independent methods. The difference between both lies in the enforcing of acceptable text corpora. While the former requires the speaker to provide utterances of known words or sentences, the latter does not require that a specific text is spoken. Text-dependent methods can directly exploit voice individuality associated with each phoneme or syllable, which results in a higher recognition performance. Nevertheless, these methods can only to be used in a limited scope of applications, usually authentication-related.

Markel and Davis (1979) introduced text-independent speaker recognition using long-term statistics such as the mean and variance of spectral features over a series of utterance. Later codebook models for Vector Quantization (VQ) (Soong *et al.*, 1985), and statistical models with GMMs (Reynolds and Rose, 1995) were also proposed. VQ approaches have the goal to compress the training data of the otherwise impractical representation of short-term training feature vectors of a speaker. A somewhat dated but comprehensive tutorial about speaker recognition by Campbell, Jr. (1997) provides details on these approaches.

More recently, advances have been made to speaker recognition methods, being the introduction of discriminative techniques and advanced channel compensation methods. The variability of the channel and environment is an important factor affecting the performance of text-independent speaker recognition systems, and techniques model-based for channel compensation are usually used. GMM-based methods are usually used, but other methods have been also proposed, such as Nuisance attribute projection (NAP) (Solomonoff *et al.*, 2005) and factor analysis (FA) (Kenny *et al.*, 2005),

both proving to reduce error rates. Regarding the introduction of discriminative techniques, SVMs in combination with standard GMMs resulted in improved performance, with examples such as Generalized Linear Discriminant Sequence (GLDS) kernels (Campbell *et al.*, 2006b), and GMM Supervector Linear (GSL) kernel (Campbell *et al.*, 2006a).

4.4.4 Multimodal recognition

The different person recognition modules can be combined to form a more robust system. Rather than alternatives, face, gait and speaker recognition methods can be used in cooperation. Whenever the visual data does not carry enough information to identify a person but speech is detected (refer to Section 3.4 for details on audio detection), the speaker recognition module can provide that information. In Figure 4.9 a possible system architecture of a multimodal person recognition system is shown. Initially the visual scene is segmented into visual objects (1a) and tracked along time (2). Also, the time segments containing speech are obtained (1b). The results of appearance analysis (3a), face analysis (3b), gait analysis (3c), and speaker analysis (3d) are matched with the objects stored in the object model (3e). Finally, the combined recognition result is obtained (4) and the object model is updated if necessary (5).

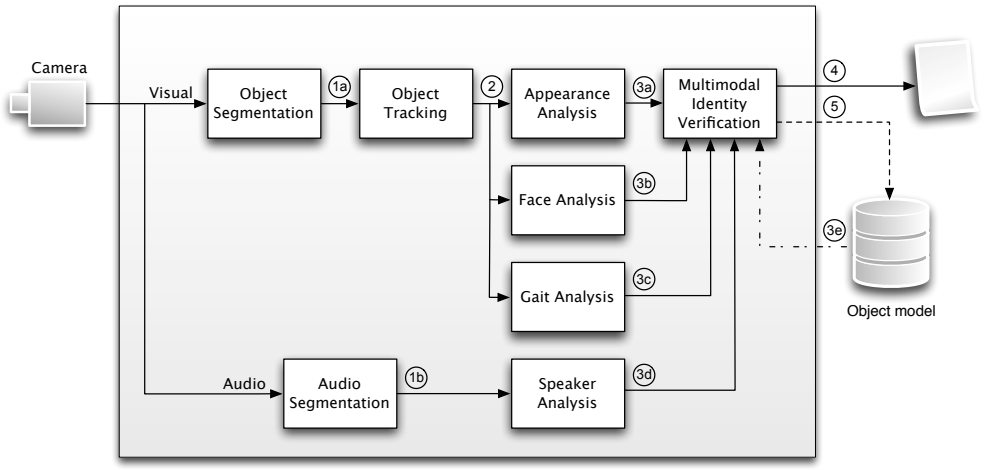


Figure 4.9: *Block representation of the multimodal person recognition system. Initially the segmented visual objects are extracted from the raw data (1a) and the time segments containing speech are obtained (1b); the visual objects are also tracked in time (2); the results of appearance analysis (3a), face analysis (3b), gait analysis (3c), and speaker analysis (3d) are matched with the objects stored in the object model (3e); the combined recognition result is obtained (4) and the object model is updated if appropriate (5).*

In an early work in this area, Shakhnarovich *et al.* (2001) showed that integrated face

and gait recognition provides improved performance over either modality alone. Combining gait and face information can result from an early fusion method, combining data at the feature level. A recent work by Zhou and Bhanu (2008) integrates information from side face and gait at the feature level. The features of face and gait are obtained separately using PCA from enhanced side face image and gait energy image (GEI), respectively. Multiple discriminant analysis is employed on the concatenated features of face and gait to obtain discriminating synthetic features.

Bernardin *et al.* (2008) integrate in a probabilistic model tracking cues as well as face and voice identification cues from several non-intrusive cameras and microphones, whenever these cues can be captured with a sufficient degree of confidence. The main difficulty of using audio cues captured only with far-field sensors (as in the case of surveillance) is that, without audio localisation, no association can be done between speech and speaker. Speaker information can also be combined with the other biometric information using late fusion, due to the different nature of the features. The fusion is thus performed at the decision level, combining identification information from gait/face recognition and speaker recognition modules.

A final reference to Mistral [43], an open-source project that aims to provide an efficient platform. Mistral implements in a single recognition engine support for multiple modalities, primarily voice and face and implements some of the methods described previously.

4.5 Summary

Object recognition is the task of associating an identity to a given object from a set of known classes, which relates to the problem of matching models from a database with an unknown sample. A recognition system comprises two stages: the acquisition stage, where a model library is constructed from certain descriptions of the objects; and the recognition stage, where an unknown object is given to the system and its identity is determined. Object tracking is a task that combines the detection of target objects in each frame, and the identification of correspondences frame to frame between detected objects so that they have consistent identities. Object tracking is in part related to object recognition in the sense that matching between objects is required to establish correspondences. Nevertheless both processes rely on different methods, keeping in mind that tracking has the additional temporal element. We presented an overview of visual object tracking methods in a single camera or across multiple cameras. The concept of tracking can also be applied to audio signals, but was not considered in this work.

In this chapter we also presented a method for appearance-based object recognition. A combined representation scheme and incremental object model was used to find matches between visual object tracks. We relax the assumption of dependence between views, since: (1) no calibration information is used, (2) no spatial registration is

made, and (3) no correlation is assumed between tracks detected by different cameras. The scheme was experimentally validated for a surveillance scenario using a publicly available dataset. The description scheme relies on SIFT local descriptors and a text-like bag-of-words representation. Object images are identified by a histogram of visual words that are identified in the image. Results show that this object representation scheme can be used to aid tracking of generic objects in visual surveillance systems, since it can discriminate a large quantity of different visual objects very well, and can be adapted to reflect object changes. It also presented a good resilience to incorrect segmentations when extracting the visual objects from complex scenes. The object model is updated through incremental learning, avoiding excessive data storage while maintaining performance and allowing new objects to be learnt by the system. This appearance analysis is effective for a limited time window since a person's appearance including her/his clothes changes. A more effective methodology relying on human-specific distinctive features, like face, gait and voice could be used. We analysed the typically used methods for each individual task and discussed how they can be combined to create a multimodal person recognition system.

In the next chapter we will focus on the last area defined in the thesis scope (Figure 1.3) – event analysis.

5 Event analysis

In a library of sequences captured by multiple cameras and covering a large area it is essential to have a way to efficiently browse the content. If we want to obtain expeditious answers of what occurred in a given time interval, direct manual inspection may not be possible. Event analysis methods that identify relevant activities and events are therefore needed. Event analysis comprises event detection and recognition tasks which are usually intrinsically related. While object analysis is essentially related to the content, event analysis has a strong context-dependent component. This means that, in order to understand what is a relevant event, we must know the context of the captured scene.

5.1 Overview

Automatically detected *events* are important cues for indexing and browsing very long audiovisual sequences that otherwise would be difficult and time-consuming to examine, for example for forensic analysis in a surveillance system. In this scenario, events can also generate alarms for an operator if detected in real-time. Xie *et al.* (2008) define events as real-world occurrences that unfold over space and time. Events are based in a more generic concept – *actions*. Zelnik-Manor and Irani defined actions (or behaviours) (Zelnik-Manor and Irani, 2006) as long-term temporal objects spanning several frames. The length of the action, which is directly related to the number of frames that need to be analysed may vary significantly, depending on the type of event, which further increases the difficulty of segmenting them in an arbitrary sequence. These actions can then be mapped into events, depending on the type of application. Note that, whereas actions can be classified regardless of the application, events usually are associated with a context. For example, a door opening will necessarily be detected by a surveillance system and can generate a relevant event if that door was not supposed to be opened; on the other hand, if that door opens very frequently it may not be desirable to trigger events in this case.

A classification was proposed by Polana and Nelson (1993) comprising three types of temporal objects, according to their spatial and temporal uniformity: 1) *temporal textures* have indeterminate spatial and temporal extent, for example wind swinging trees or flowing water, 2) *activities* are temporarily periodic but spatially restricted, for example a person walking or running, and 3) *motion events* are isolated actions that do not repeat either in space or time, for example a door opening. Each of these types are associated with a characteristic approach for modelling and recognition.

Another possible classification of events consists of distinguishing their commonness of occurrence. Events can either be usual or unusual (alternatively, normal or abnormal). This differs from other commonly used classification that considers the event's frequentness – events are defined as regular or sporadic. Note that sporadic events may or may not be considered abnormal, but regular events are almost always normal. As before, these event properties affect how the event analysis is devised. For example, regular and sporadic events can both be detected with a set of trained classifiers, but detecting unusual events often means finding outliers that do not fit the current set of models.

A substantial part of the effort in recent years in event analysis focused on the detection of events in sports videos (Chen *et al.*, 2006; Ekin *et al.*, 2003; Leonardi *et al.*, 2004; Sadlier and O'Connor, 2005; Zhang and Chang, 2002; Zhou *et al.*, 2000). Since the types of events is usually limited and their characterisation well defined, a good performance can be achieved. Event detection is typically performed in two steps. Firstly, low-level descriptors are extracted to represent the content in a compact set; and secondly, a decision is made whether an event was detected, usually based on a machine learning technique. Considering a real-world scenario, for example visual surveillance, typically three distinct modules comprise the automatic surveillance system. The previously mentioned object detection and object tracking, and event detection. While the two former provide a spatial or spatio-temporal segmentation of the scene, event detection segments the video temporally by detecting interesting or unusual events. The definition of these events depends on the context of application and may include detection of abandoned luggage, loitering, unusual movements, abnormal crowd movements, vandalism, etc.

In the literature it is common to find references to event detection. This definition is rather generic and refers in fact to different tasks and diverse goals. The following tasks can be defined in event analysis: detection, recognition, and discovery.

This chapter presents the concepts of event analysis and some of the previous work done in this area. Similarly to other chapters, we will not present an exhaustive study and evaluation of the methodologies used to analyse events. An extensive review of event analysis in multimedia streams was very recently done by Xie *et al.* (2008). In there, references to more work in this area can be found, including some not directly related to the scope of this thesis.

5.2 Event detection and recognition

The task of event detection is commonly performed in event analysis systems, since it is often used as a preliminary step for other analysis tasks. This task consists of comparing the input data with a known event or event model in order to define if that event has occurred. The output for a given sequence is binary, stating if a given event happened or not in that sequence. Event recognition is closely related to event detec-

tion. Both tasks are often associated in a single task. The goal of event recognition is to recover from the data a description of the event. This description can include attributes that identify the event, the location, and the associated object (if any).

Many approaches have been proposed to handle event detection and recognition. A simple method is comparing distances of pixel-based features that represent changes in time and space of the input data and templates. For controlled environments where the camera position is known these methods can be effective. For example, Davis and Bobick (1997) use a motion history image that integrates motion intensity with a memory-related weighting factor. Action recognition is done comparing the distance between these images and templates. Zelnik-Manor and Irani (2001) detect events by clustering the motion data using the local intensity gradient as input features. A view-invariant method proposed by Rao *et al.* (2002) is achieved with the motion curvature method and dynamic segmentation of the object in space and time.

The most common approaches to model actions focus on periodic activities and temporal textures (Polana and Nelson, 1993; Saisan *et al.*, 2001; Szummer and Picard, 1996). A simple approach to model structured activities is to represent objects using distributions of features at multiple scales. While Chomat and Crowley (1999) proposed using distributions of motion features at multiple spatial scales, Zelnik-Manor and Irani (2006) used multiple temporal scales. Other approach proposed by Efros *et al.* (2003) consists in comparing templates of dense optical flow fields. This approach however does not handle correctly objects with different sizes and shapes or with different phases of the motion.

The conceptual similarity between events and a structured aggregation and evolution of tracked objects influenced many authors to use grammars and graphs to encode event relationships. Medioni *et al.* (2001) rely on graphs and represent tracked object parts as the nodes, and the tracking likelihoods as edge weights. Events are then matched with known events using a finite state automaton. An example of an approach using grammars, is the work by Ivanov and Bobick (2000). They use as stochastic context-free grammar (SCFG) to recognise complex action sequences. Low-level objects are initially segmented and tracked and SCFG is used to analyse the sequence of object transitions. Seong-Wook and Chellappa (2006) extended SCFG with an attribute grammar. Shi *et al.* (2004), on the other hand, propose propagation networks to recognise activity, with the inference process being done by particle filtering.

A different approach consists of using statistical models. It is however necessary that enough training data is available. As previously mentioned for other tasks (for example, in Chapter 4, face and gait recognition) HMMs is commonly used for stochastic representation of sequences. In the case of event analysis, extensions of traditional HMM methods, like Coupled HMM (CHMM) (Brand *et al.*, 1997) were proposed. CHMM explicitly models the temporal dependencies among different streams for multi-object action recognition. Chen *et al.* (2004) use a dynamic Bayesian network (DBN) for inferring events in a multi-camera home surveillance scenario. In a recent proposal, Gupta and

Davis (2007) follow a different approach and merge object recognition, object tracking, and event recognition in a single recognition framework. Bayesian networks are also used to accomplish this.

Although generative models, like HMMs, are typically more suited for modelling temporal evolution, some authors also considered discriminative methods. For example, a variant of NNs called functional link network was used by Eng *et al.* (2003) for detecting drowning and distressing events in swimming pools. SVMs can also be used on kernels generated from HMM likelihoods and parameters from input feature streams (Ebadollahi *et al.*, 2006).

5.3 Event discovery

Event discovery aims to find events without previously knowing its semantics. Automatically detecting events that were not learned beforehand, rely on finding regular events and unusual events.

Regular patterns are typically found using clustering operations with various features and models. Clarkson and Pentland (1999) cluster AV streams with HMMs to identify different user locations. Pavan Turaga (2007) build linear dynamic systems on optical flow features for surveillance action events, with temporal, affine, and view invariance. Zhu *et al.* (2005) use association rules on mid-level AV features to discover events.

Unusual events are identified as deviations from usual activity collected previously, and are detected using an underlying distance/similarity metric. Zhou and Kimber (2006) trained a coupled HMM with the usual events in typical surveillance streams and detect unusual events as outliers in the likelihood values. Other methods to find unknown events worth mentioning include analysis of co-occurrence matrix (Zhong *et al.*, 2004), multilevel Self-Organizing Map (SOM) clustering (Petrushin, 2005), and n-grams (usually used in natural language processing and genetic sequence analysis) and suffix trees (Hamid *et al.*, 2005).

5.4 Activity analysis

Activity analysis based on object trajectories is one of the basic problems in event analysis for video surveillance. The term activity analysis is often confused with behaviour analysis. Makris (2004) distinguishes both terms the following way. Activity describes the global motion of an object within a scene, which is defined as the sequence of target positions over time. This sequence is commonly referred to as object trajectory. On the other hand, behaviour is usually more complex and considers human motion, which comprises also the particular motion of the target's subparts. While the de-

scriptions of activity can be verbal phrases as "goes from area A to B", "stops in area C", "moves fast", typical descriptions of behaviour are "walks", "runs", "picks up", "jumps", "leans", etc. Activity is also closely related to the structure of the scene, while behaviour is usually independent of the scene.

5.5 Detecting appearances of persons[§]

In this section we describe a complete system for detecting appearances of persons over multiple uncalibrated cameras without overlapping field of view. The system is able to discover correspondences between different views of the same object. We employ the approach detailed in Section 4.3 to match tracked objects. The tracks are compared with a global object model based on an ensemble of individual object models. If an object is recognised, the corresponding track is labelled. Moreover, if multiple objects are detected in a single track, the track is split and the partial tracks are labelled accordingly. The output is a timeline representing the objects present in a given scene. These timelines are compared with a ground-truth to evaluate the system's performance.

5.5.1 System overview

The system can be transparently applied for wide area coverage with multiple disjoint cameras, for single area coverage with multiple cameras or even a mix of these scenarios since it is agnostic of the camera deployment. The goal is to automatically obtain a description of where and when each person is detected by the system. A person that crosses multiple camera fields of view is tracked independently for each camera and the resulting information collected from each track (sequence of object images) is compared with a global visual object model. For the same visual object, the system establishes links between that object's tracks. No camera calibration information is used in this procedure, only appearance information of the visual objects. This method can be used in generic scenes containing persons or other objects. However, throughout the text we use indistinctly the terms "person" and "object" to refer to the relevant elements in the scene being tracked.

Fig. 5.1 shows an example scenario of application for this system. This scenario consists of a set of disjoint areas covered by a single camera. People move around, crossing one or more areas. Since tracking is to be performed for the full monitored area it is necessary to establish a global identity for each tracked object. This type of scenarios presents several problems.

[§]This section is based on the article *Automatic description of object appearances in a wide-area surveillance scenario* submitted to the IEEE Transactions on Circuits and Systems for Video Technology (Teixeira *et al.*, 2009).

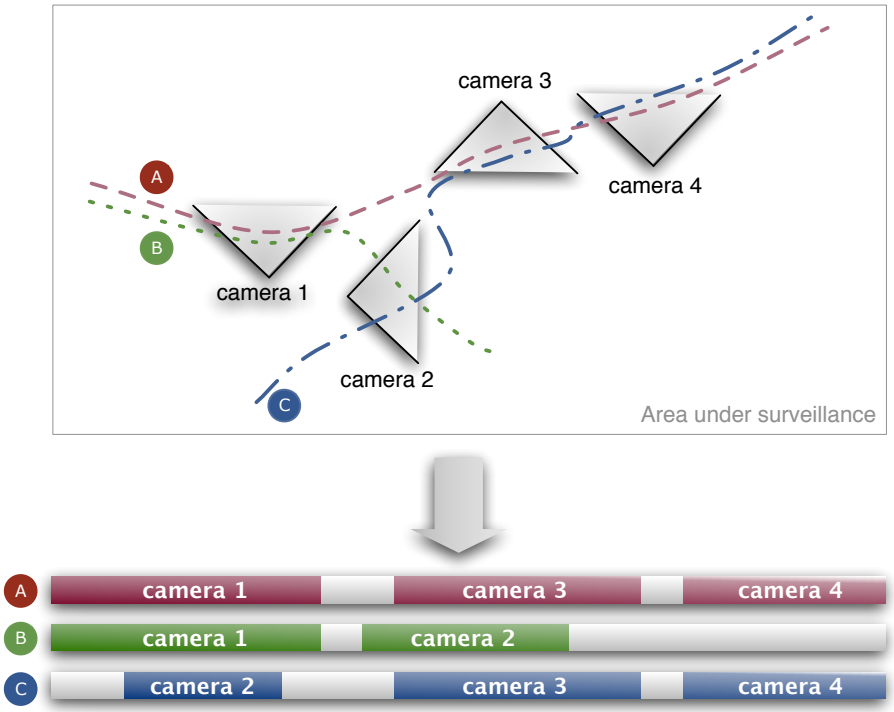


Figure 5.1: *Problem characterisation and system output. On the top a graphic illustration of a possible scenario containing multiple cameras covering disjoint areas. On the bottom a possible output of the system consisting of a timeline for each detected object, where the period tracked with each camera is marked.*

Consider the following situation. Three different persons are detected by the tracking system in a given time interval. Person A and person B walk side by side while they are captured by camera 1. Person C enters camera 2 field of view and meets person B. In camera 3, person A and person C start walking side by side. Finally, both are again captured by camera 4, after switching their relative positions. In this simple scenario, there are potential problems for typical tracking systems, namely: when persons B and C cross mutual occlusion may occur and identities can be switched; when person A is accompanied by person B or C, group movement (both are identified as a single object) and prolonged occlusion may occur leading to track lost or mistaken identities.

The output is a timeline similar to the one shown in the bottom of Fig. 5.1. For each person, time intervals are marked and labelled according to the area where the person was detected. Alternatively, we could create a timeline where, for each camera, time intervals indicate which person was detected.

In Fig. 5.2 the block diagram of the proposed system is shown. The main steps involved in the process are: (1) obtain the location of potential objects using an object segmentation algorithm, (2) track each object using a single-view tracking algorithm, (3) match the track with a global visual object model, which comprises (a) obtaining an appearance description and (b) comparing it with the model to verify its identity, (4) update the model with the new track information, and (5) optionally, use the identity information to give feedback to the tracking algorithm. Each of the steps are described in the next sections. Note that steps 3 and 4 are described as a single object matching operation.

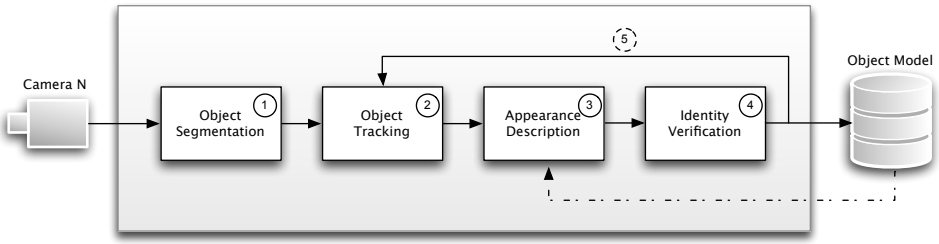


Figure 5.2: *Block diagram of the system architecture.*

5.5.2 Object segmentation

The first step in the processing chain aims to obtain the possible location of relevant visual objects. Many approaches have been proposed, but the most common and widely adopted is obtaining a background model and perform a "subtraction" filtering to incoming frames. An overview about background modelling and subtraction algorithms can be found in Section 3.2.1. The object segmentation algorithm integrated in the system is based on the cascaded detection of common types of changes that is presented in Section 3.3.

5.5.3 Object tracking

This section briefly describes the tracking algorithm used for the purpose of building the system depicted in Figure 5.2. It also presents changes made to the original algorithm and a brief assessment of its behaviour. Although the depicted architecture refers to a generic object tracker, considering a visual surveillance scenario as an example of a study case, we chose the algorithm proposed by Zhao and Nevatia (2004) as it aims to achieve a good compromise between processing performance and robustness, and because it is was developed for the purpose of tracking people. Note that other algorithms, possibly targeting generic objects, could also have been used.

Zhao's algorithm was designed for a single stationary camera placed above people's head in an inclined position looking down. It uses camera calibration information and the assumption that humans move in the ground plane to perform transformations between the physical world and the image; humans are detected by identifying their heads and an ellipsoid is used as a coarse human shape model to constraint the shape of an upright human. Processing is done in two stages: detection, where the shape and camera models are used in boundary analysis of foreground objects to compute human hypothesis; tracking, where each human hypothesis is tracked in subsequent frames using a Kalman filter and object appearance constraints. As in other person tracking algorithms (Denman *et al.*, 2005; Haritaoglu *et al.*, 2000; Siebel and Maybank, 2002), head detection is used in the generation of human hypothesis. This technique takes advantage of the placement of the camera several meters above the ground, which helps to avoid some situations of occlusion and to reduce the probability of people heads being occluded. The camera calibration parameters are an essential factor of the algorithm. Also, the user is required to provide information about the entrances and exits from the scene, which is used to determine when a track should be generated or removed.

The necessity to use camera calibration parameters prevents the algorithm from being used in situations where such information is not available. The texture and probability templates aid in the correct matching of tracked objects. However, in case of gradual and prolonged superimposition of objects track drift may occur, i.e, the label assigned to an object being tracked is assigned to a different object. Both the boundary analysis and the matching process depend on point by point operations leading to a loss of performance with the increase of the image and object size as well as with the number of foreground objects present. For objects far away from the camera, i.e., objects with reduced dimensions in the image, the rate of miss detections and track loss increases.

In our implementation, enhancements have been made to the original algorithm. We are not using camera calibration parameters. We assume that the camera is placed so that the vertical axis of the image corresponds to the vertical axis of the scene and approximate the height variation of objects due to their distance to camera by a linear function. Some information regarding reference measures are still required, but different techniques can be used to obtain the required points. Since all measures are made in the image, unlike the proposed algorithm we do not use a constant velocity model. Also, in the Kalman filter we followed the approach proposed by Weng *et al.* (2006) which uses the occlusion rate to adjust the estimate parameters of the filters: if the occlusion rate is small the measurement is considered more trust worthy; otherwise the prediction is trusted completely.

5.5.4 Object matching

Multiple appearances of an object are detected by matching a newly tracked object with the visual object model. The matching process takes the sequences of segmented object images detected by the tracking algorithm as inputs. The matching method is

based on the one presented in Section 4.3 but a different learning strategy is used.

□ Classifying object images

The goal of object classification is to attribute a known label to an object frame. For this purpose we propose using a model $\mathcal{M}$ that consists of C classifiers. Each individual model $\mathcal{M}_k, k = 1, \ldots, C$ is associated with an object known by the system and outputs one of two results: $+1, -1$. The results correspond to acknowledging or not, respectively, that the frame being tested contains the object k. As new object classes are detected by the system, more of these individual models can be added. The main advantage of such an approach compared to the approaches applied in Section 4.3, is that we only need to train a binary model when a new object class is added.

⇀ **Training the individual models** Each model is trained using AdaBoost (Freund and Schapire, 1997), which is an ensemble-based algorithm (we previously described the application of AdaBoost to face detection in Section 3.6.1). The training process of AdaBoost consists of training B `WeakClassifiers`. When a new `WeakClassifier` is trained, a bootstrap sample is drawn from a distribution. This distribution is iteratively updated between training steps, depending on the classifier performance. By updating the distribution, subsequent classifiers focus on samples causing more errors. The classification is based on the weighted majority voting of each classifier. A short description of the AdaBoost algorithm for the binary case is presented in Table 5.1.

The `WeakClassifiers` comprising the models are based in linear kernel SVMs.

⇀ **Obtaining the global hypothesis** Given the global model $\mathcal{M}$, and the input frame described by x, the global hypothesis $\mathcal{H}(x)$ consists of the combination of the individual hypothesis $\mathcal{H}_k(x), k = 1, \ldots, C$ where C is the total number of objects. For each model $\mathcal{M}_k$, an hypothesis $\mathcal{H}_k(x)$ is defined as

$$\mathcal{H}_k(x) = \begin{cases} \{\mathcal{O}_k\}, & \text{if } H(x) = 1 \\ \emptyset, & \text{otherwise} \end{cases} \tag{5.1}$$

where $\mathcal{O}_k$ is the label defined for object k (for example, `Person01`) and $\emptyset$ is the empty set. The aggregate set of hypothesis $\tilde{\mathcal{H}}$ is thus given by:

$$\tilde{\mathcal{H}}(x) = \bigcup_{k=1}^{C} \mathcal{H}_k(x) \tag{5.2}$$

Table 5.1: *Algorithm: AdaBoost.*

Given $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in X, y_i \in \{-1, +1\}$
Initialize $D_1(i) = \frac{1}{n}, i = 1, \ldots, n$
for $b = 1, 2, \ldots, B$: 1) Draw bootstrap training data S_b^* according to the current distribution D_b 2) Train classifier b with S_b^* 3) Obtain hypothesis h_b and calculate the weighted error ϵ_b $\epsilon_b = \sum_i D_b(i)[h_b(x_i) \neq y_i]$, if $\epsilon_b > \frac{1}{2}$, abort where $[h_b(x_i) \neq y_i] = 1$ if $h_b(x_i) \neq y_i$ and 0 otherwise 4) Calculate classifier weight $\alpha_b = \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon}$ 5) Update distribution D_b $D_{b+1}(i) = \frac{D_b(i) e^{-\alpha_b y_i h_b(x_i)}}{Z_b}$ where Z_b is a normalisation factor such that $\sum_{i=1}^{n} D_{b+1} = 1$
Hypothesis obtained by weighted majority voting $H(x) = \text{sign}(\sum_{b=1}^{B} \alpha_b h_b(x))$ where x is the input data under evaluation

Finally, the global hypothesis is based on the cardinality of the aggregate set of hypothesis, such that:

$$\mathcal{H}(x) = \begin{cases} \mathcal{U}, & \text{if } \#\tilde{\mathcal{H}}(x) = 0 \\ \tilde{\mathcal{H}}(x), & \text{if } \#\tilde{\mathcal{H}}(x) = 1 \\ \text{rand}(\tilde{\mathcal{H}}(x)), & \text{otherwise} \end{cases} \tag{5.3}$$

where # is the set cardinality, rand is a random operator with uniform distribution, and $\mathcal{U}$ represents the label Unknown.

5.5.5 Identity verification

The task of identifying each visual object's identity is done in multiple steps, as depicted in Fig. 5.3. Each track outputted by the tracking algorithm is first divided into segments comprising N frames. The segments are analysed separately by associating a label $\mathcal{L}_i$ to the respective segment i. This analysis consists of comparing each of the N frames with the global object model (refer to Section 4.3.4) and performing a simple majority voting to obtain a single label. When two or more labels are tied for the most voted label, one is chosen randomly between those most voted.

This process is repeated while there are segments in a track to be analysed, and the final estimation for the object label $\hat{\mathcal{L}}$ is again done by majority voting. In this case the votes correspond to the segment labels $\mathcal{L}_i$. A common tracking error is identity drifting thus it is possible that a track outputted by the tracking algorithm contains more than one object. To help correct this we apply the following test: for each segment i, if $\mathcal{L}_{i-P} = \cdots = \mathcal{L}_{i-1} = \mathcal{L}_i \neq \hat{\mathcal{L}}_i$ the track is split, otherwise the analysis proceeds with the current track. The splitting consists of two sub-steps 1) close the current track and attribute a final label $\hat{L}$, 2) start a new track containing the last $P + 1$ frames and continue analysing the remaining segments, associated with this new track. We chose to set P to 2, as shown in Fig 5.3.

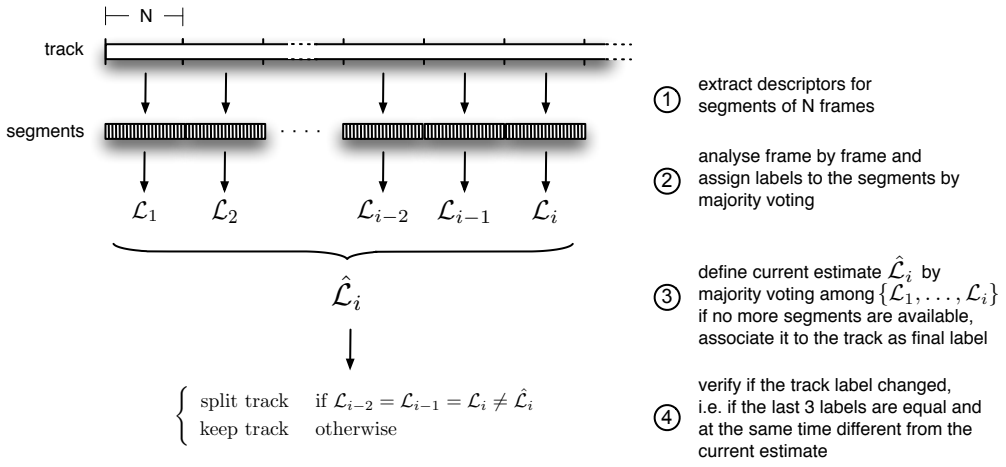


Figure 5.3: *Algorithm to verify the identity of objects present in a given track. The algorithm consists of 4 steps that are done iteratively for the incoming track.*

By dividing the track in segments we are in fact smoothing the errors caused by incorrect object description. It is often the case that the description extracted from a frame is incorrectly classified, essentially due to a deficient segmentation. Using sufficiently large segments these errors can be attenuated. For all experiments we considered segments of $N = 25$ frames.

The identity verification can be performed online, since all tracks are analysed independently and we only need to keep the last 3 segment labels for each track. However, this process introduces a delay of $3N$ frames, which for a frame rate of $25fps$ and $N = 25$ corresponds to a $3s$ delay.

5.5.6 Experimental setup and results

□ Dataset

The training set consists of the $\mathcal{D}_{30}$ dataset, described in Section A.3. The evaluation dataset consists of 11 sequences also from the CAVIAR shopping set [44] with two different views. Each sequence contains more than one person and some persons appear multiple times. However, the ground-truth provided for the sequences does not preserve the labels for persons that leave the field of view and reappear later on. In order to evaluate the labelling process, all tracks contained in the original ground-truth were labelled manually, according to the labels defined in the global object model.

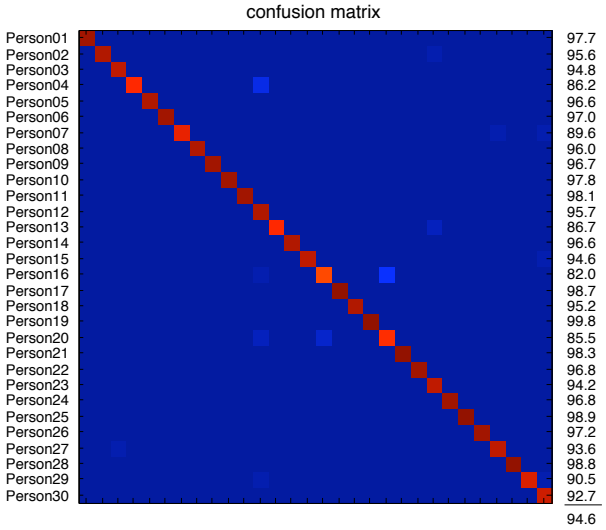
The CAVIAR dataset comprises sequences of the same area captured by two cameras with different fields of view. Nonetheless, the usage of that dataset does not oppose the proposed scope for the system, since the system is agnostic of the camera displacement, treating each camera independently. Hence, it is irrelevant if the two cameras are covering the same area or non-overlapping ones. Nevertheless, to complement the results with the CAVIAR dataset, we captured 3 additionally sequences and formed a dataset containing track images from 5 persons that appear multiple times in these sequences (described in Section A.4). A model representing these 5 persons was created in a similar manner as the model $\mathcal{M}$ described earlier. However, for these sequences the model $\mathcal{M}$ includes the 5 persons alongside the 30 persons of the CAVIAR training dataset.

□ Model evaluation

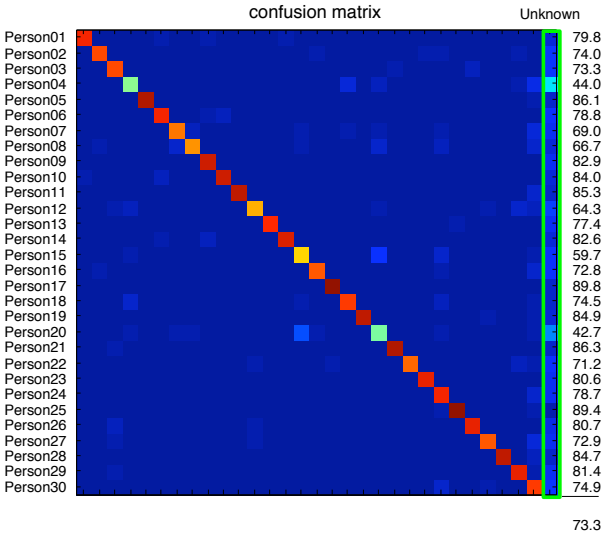
As described previously, the model we will be using consists of an ensemble of binary individual models. Each of these individual models is associated with a given object class. If, for instance, the global model "knows" 30 different objects, it should contain 30 different individual models that classify a given image as being a "known" object or an "unknown" object. We compared this model with the straightforward approach of having a multi-class SVM. To train the individual AdaBoost-based models we used a random sample of 1000 images from other objects to represent the "unknown" object.

The evaluation consisted of randomly dividing the training dataset described previously into train and validation subsets and perform a 5-fold cross-validation. The original sample is partitioned into 5 subsamples, and each of the subsamples is validated individually with the remaining datasets being used as training data. The 5 results from the folds are average to produce a single result that is shown in Fig. 5.4. In that figure we show the visual representation of the confusion matrices where each column of the matrix represents the predicted class and each row represents the actual class.

Using the ensemble of models, the performance drops significantly from an average successful classification rate of 94.6% to 73.3%. By considering the object class "un-



(a) Multi-class model (30 object classes).



(b) Ensemble of binary models with AdaBoost (30 object classes, plus an "unknown" class).

Figure 5.4: *Confusion matrices of the object classification. Visual representation of the confusion matrices using different approaches to build a global object model.*

known", we are contributing to the performance degradation since the sample used to represent it may be insufficient to obtain the best discriminative model between classes. However, in the trade-off between complexity and performance we opted for simpler and more feasible models in detriment of performance. For instance, if a new object needs to be added to the global model we do not need to retrain everything, just the individual object model. The same applies if we want to update an individual model with new information. For the task of track classification it is possible to afford this penalisation, as we show in the next subsections.

□ Timeline generation

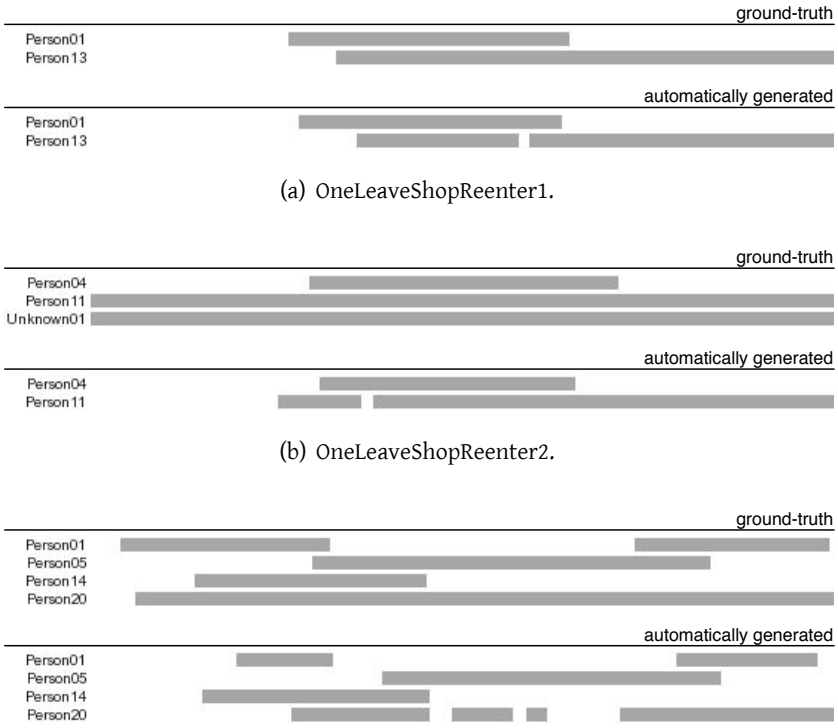
Timelines are generated from the combination of the results produced by the tracking algorithm and the identity verification process. Fig. 5.5 shows some examples of automatically generated timelines and the respective ground-truth. These can be considered successful cases, since all known objects are correctly identified. The main difference to the ground-truth is the length of the detected tracks. The result tracks usually are shorter essentially due to two reasons: when the objects are distant, the tracking results are usually unstable; and, when the objects leave the field of view, the rules defined specifically for person tracking (refer to Section 5.5.3) filter out those object images.

In Fig. 5.6 results for crowded scenes are shown. The larger number of unknown persons introduces errors in the timeline generation. To understand the effect of unknown persons in the system performance, we assigned a label to some of the unknown persons and added them to the global model. For the EnterExitCrossingPaths1 sequence, a new individual model was trained and for the WalkByShop1 sequence, we added two additional individual models of two other unknown persons. Therefore, the global model used in this test consisted of 33 persons, instead of the previous 30. It is possible to observe that the generated timelines with the updated model are closer to the ground-truth.

Timelines from different views can be combined to analyse the appearance of objects throughout the area covered by the camera system. Fig. 5.7 shows an example of that for the WalkByShop1 and WalkByShop1front sequences. In the scene captured by both cameras, `Person01` and `Person12` at some point in time enter a shop, leaving the first field of view and entering the second. These instants are marked with the rounded labels A and B.

□ Objective evaluation of the generated timelines

To evaluate the timeline generation quality we rely on the commonly used precision and recall measures, defined in Section B.1.1.



(a) OneLeaveShopReenter1.

(b) OneLeaveShopReenter2.

(c) OneShopOneWait1.

Figure 5.5: *Examples of automatically generated timelines. Each timeline is compared with the respective ground-truth.*

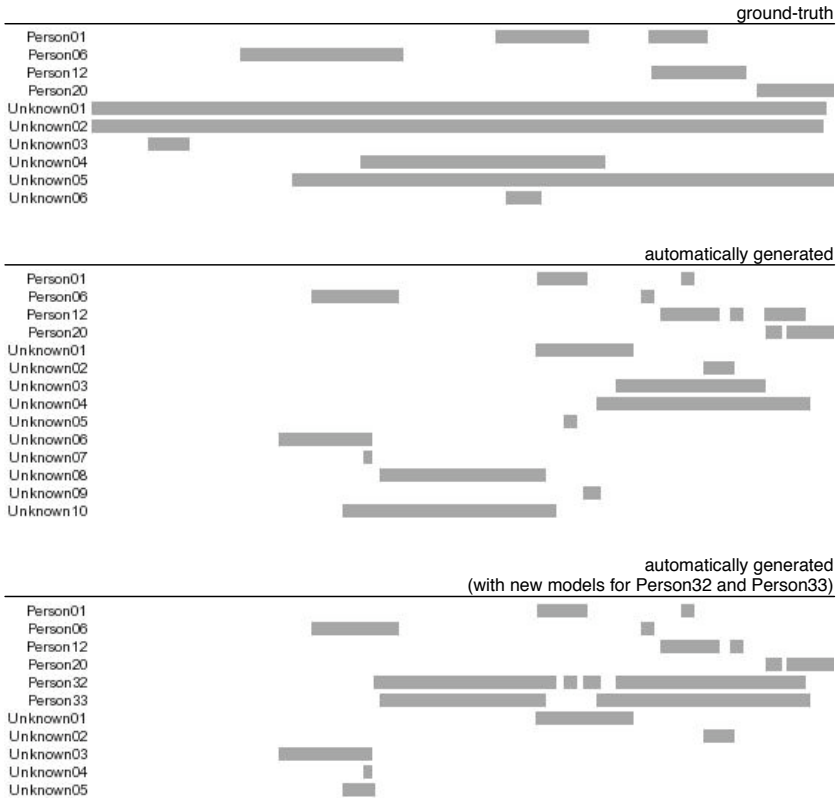
The values of number of true positives N_{tp}, false positives N_{fp}, and false negatives N_{fn} are calculated as follows. Consider $\mathcal{S}_t$ and $\mathcal{R}_t$ the sets containing the labels of the object present at the time instant t (frame) both in the timeline to be evaluated and the reference (ground-truth), respectively. For a sequence containing T frames, N_{tp} and N_{fp} and N_{fn} are given by equations (5.4), (5.5) and (5.6), respectively.

$$N_{tp} = \sum_{t=1}^{T} \#(\mathcal{R}_t \cap \mathcal{S}_t) \tag{5.4}$$

$$N_{fp} = \sum_{t=1}^{T} \#(\mathcal{S}_t \backslash \mathcal{R}_t) \tag{5.5}$$

$$N_{fn} = \sum_{t=1}^{T} \#(\mathcal{R}_t \backslash \mathcal{S}_t) \tag{5.6}$$

(a) EnterExitCrossingPaths1.



(b) WalkByShop1.

Figure 5.6: *Examples of timelines generated with updated models for previously unknown objects. The results are better for both cases.*

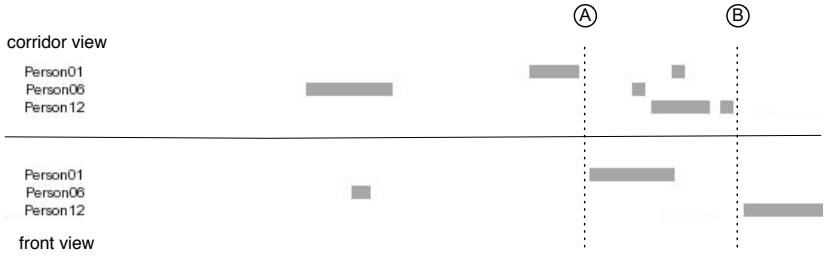


Figure 5.7: *Generated timelines for two different cameras capturing the same scene from different views. The rounded labels A and B mark the instants when* `Person01` *and* `Person12` *enter the shop, leaving one field of view and entering the other. To avoid excessive clutter only the relevant objects present in both views are represented.*

A measure that combines both precision and recall is called F-measure (defined in Section B.1.3). In our case, lower values for the recall should mean that the resulting timeline has many gaps, while lower values of the precision represent incorrect labelling. If the tracking algorithm wrongly ignores objects in the scene, the recall will essentially reflect this. Since we are not evaluating the tracking algorithm itself, we will be using as the main measure F_β with $\beta = 0.5$, i.e., $F_{0.5}$.

The results for the 14 sequences are presented in Table 5.2. As expected, precision is generally higher than the recall. With a better tracking output (including segmentation and tracking algorithms) we can further improve these results. The worst results are for OneStopMoveEnter1 which consists of a crowded scene. The tracking outputs a large number of fragmented tracks that end up being incorrectly classified. With three sequences, marked with $\diamondsuit$, an additional test was made. As previously mentioned, for the sequences EnterExitCrossingPaths1 and WalkByShop1, we updated the global model with information about unknown persons. Using the updated global model the matching corrects some tracking errors that would not be corrected otherwise. Fig. 5.6 shows the improved results, and the objective evaluation supports this. For OneStopMoveEnter1 a similar approach was followed. In all cases, results improve, especially in the WalkByShop1 sequence. This is due to the fact that the fragmented tracks of the previously unknown persons were grouped correctly afterwards (confirmed by the number of detected objects that changed from 14 to 11).

We also observe that in more crowded sequences the performance measures are considerably lower. This is largely due to the high number of unknown objects. To better evaluate the performance of the identity verification algorithm we also present in Table 5.2 the precision, recall and $F_{0.5}$ of the generated timelines considering only the known objects. In other words, all tracks labelled in the ground-truth with an `Unknown` tag are not considered for evaluation.

sequence	# of frames	# of objects gk	rk	gt	rt	all objects precision	recall	$F_{0.5}$	only known objects precision	recall	$F_{0.5}$
EnterExitCrossingPaths1	383	2	2	4	6	0.69	0.69	0.69	0.39	0.83	0.44
EnterExitCrossingPaths1♢	383	3	3	4	4	0.75	0.89	0.77	0.47	0.94	0.52
OneLeaveShopReenter1	390	2	3	2	2	1.00	0.94	0.98	1.00	0.94	0.98
OneLeaveShopReenter2	560	2	2	3	2	1.00	0.45	0.80	1.00	0.86	0.97
OneShopOneWait1	1377	4	4	4	4	0.89	0.67	0.83	0.89	0.67	0.83
OneStopEnter2	2725	3	3	6	6	0.75	0.51	0.68	0.77	0.57	0.71
OneStopMoveEnter1	1590	7	7	18	29	0.13	0.12	0.13	0.27	0.44	0.30
OneStopMoveEnter1♢	1590	9	9	18	25	0.23	0.24	0.23	0.31	0.47	0.33
OneStopMoveNoEnter1	1665	1	1	2	3	1.00	0.68	0.91	0.89	0.68	0.84
OneStopNoEnter1	725	2	3	3	4	0.70	0.39	0.61	0.82	0.77	0.81
ShopAssistant2	3700	8	8	17	18	0.59	0.40	0.54	0.68	0.62	0.67
WalkByShop1	2360	4	4	10	14	0.14	0.11	0.13	0.53	0.84	0.57
WalkByShop1♢	2360	6	6	10	11	0.64	0.37	0.55	0.58	0.88	0.62
WalkByShop1front	2360	3	3	5	4	0.80	0.67	0.77	0.83	0.86	0.84
Hall1	1413	5	5	11	14	0.43	0.26	0.39	0.57	0.46	0.54
Lobby1	1795	5	5	14	23	0.37	0.61	0.40	0.64	0.84	0.73
Lobby2	2343	5	5	12	16	0.53	0.79	0.63	0.76	0.68	0.72
Overall (excluding the respective ♢ sequences)	—	—	—	—	—	0.64 (0.30)	0.52 (0.25)	0.61 (0.26)	0.72 (0.22)	0.72 (0.16)	0.71 (0.19)
Overall (including only the respective ♢ sequences)	—	—	—	—	—	0.69 (0.24)	0.56 (0.22)	0.65 (0.21)	0.73 (0.20)	0.73 (0.16)	0.72 (0.18)

Table 5.2: *Results for the sequences under evaluation. For each sequence, the number of frames, the number of known (gk) and total (gt) objects in the ground-truth, and number of known (rk) and total (rt) objects in the result timelime are presented. Evaluation is done using precision and recall values, as well as the $F_{0.5}$ F-measure. The sequences marked with ♢ use an updated global model. This new model includes one or more of the objects present in the sequence but not in the previous $\mathcal{M}$ model. Two types of evaluation were considered: (1) including all tracks and (2) excluding tracks labelled in the ground-truth with an* Unknown *label.*

5.5.7 Interaction with tracking

Until now we assumed that captured tracks were the inputs for our method and no further interaction happened. However, track matching can be integrated with the tracking algorithm to improve the overall performance. The interaction between both can be two-fold:

1) Tracking helps building the database – the same person can have a different description depending on the view angle. By tracking a person with more than one camera with overlapping fields of viewing, for example, we can provide more information to the database, knowing that it's the same person.

2) Object matching can correct incorrect tagging – matching can help disambiguate two tracks when, for example, two or more objects cross in the field of view.

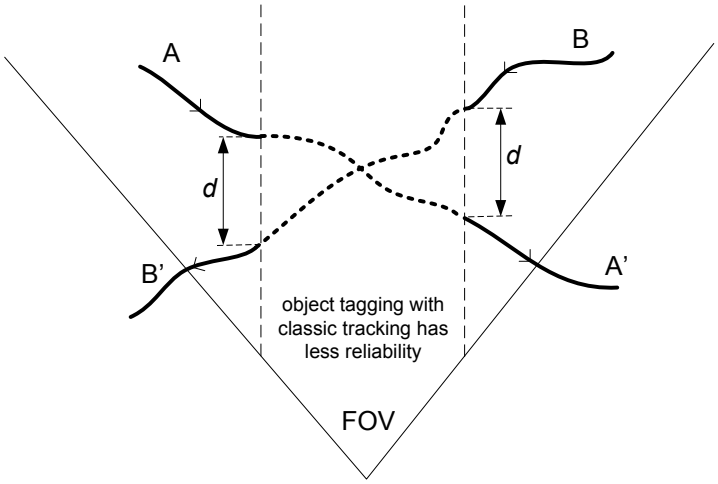
While the first interaction is largely explored in our approach, the second interaction is discussed in this section.

Consider the scenario depicted in Figure 5.8(a). Objects A and B are tracked across a single FOV and at some point in time both tracks cross. Without depth information or another view over the same area, disambiguating the tracks can be difficult. As objects get closer, typical segmentation outputs a single large combined object. When objects eventually are again farther apart, the tracking system should assign A and B to A' and B', respectively, but errors often occur. An example of this using a particle filtering tracker (Chateau *et al.*, 2005) is depicted in Figure 5.9. Using a track matching method, inconsistent assignments could be detected and corrected.

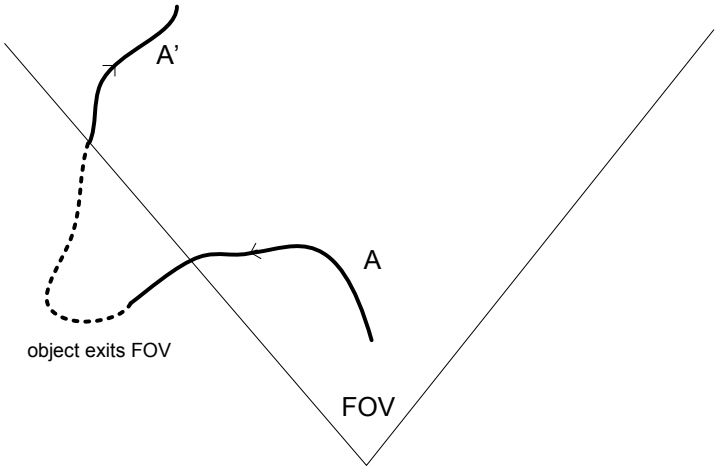
Another similar problem occurs when a tracked object leaves the field of view and re-enters later, as depicted in Figure 5.8(b). In most tracking systems, the track corresponding to the re-entrance of the object would have a new tag assigned to it. The same object would then have two different identities. By comparing the visual representation of the new track with the visual objects model would correct the previous assignment.

The second type of interaction can also be extended to multiple independent views, where typical tracking methods can not be used. A track produced by the tracking algorithm at a given view is matched with a global model. An existing tag is assigned to it, or a new one, if the object is not known. In multi-camera systems with overlapping FOVs, the spatial information can be a useful help in matching. Although this is a possibility, it is not explored in this work as we do not assume any dependence between views.

We are not evaluating the tracking results but it is nevertheless important to observe how the tracking results are improved by the matching step. In Fig. 5.10 two different tracking results are shown, as well as the respective ground-truth. In Fig. 5.10(b) the



(a) Object occludes another object – as objects cross the tracking algorithm can incorrectly assign the label A to B' and B to A'.

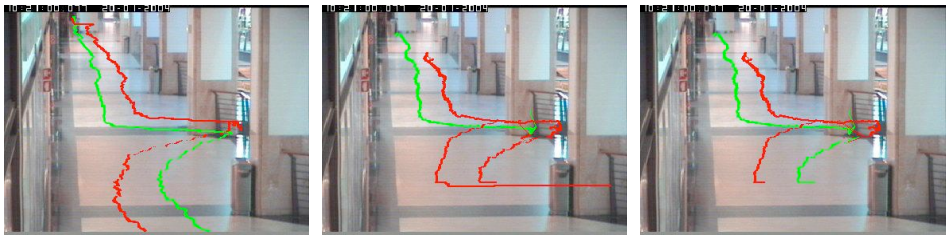


(b) Object exits and re-enters FOV – the track A', corresponding to the re-entrance of the object, would have a new assigned tag.

Figure 5.8: *Scenarios where typical tracking methods often fail to correctly track/identify an object.*

track is obtained without matching. At some point in time there is an identity drift between the two persons, resulting in incorrect labelling afterwards. If we apply both the object matching method and the identity verification algorithm, the drift is detected and the track is split correctly, as shown in Fig. 5.10(c). These results can be further explored in the future. By providing a tighter integration between tracking and matching we expect to obtain better performance results.

Figure 5.9: *Common tracking error using a state-of-the-art particle filtering tracker. The tracker changes targets because the new target has a higher classifier score then the initial one. (Chateau* et al.*, 2005)*



(a) Ground-truth. (b) Without matching. (c) With matching.

Figure 5.10: *Tracks for* `Person29` *and* `Person30` *in the sequence ShopAssistant2. Without the matching step the tracks outputted by the tracking algorithm are wrongly assigned. The matching step corrects these errors.*

5.6 Visual trajectory representation for event retrieval [§]

Many applications such as scene monitoring require tracking for different purposes (refer to Section 4.2 for further details on visual object tracking). Typically, tracking information in monitoring applications is used as an input of decision-making modules (e.g. alarm generation) but is also stored for later analysis. The amount of information stored can easily become overwhelming for a human operator to find a specific event. Hence, the task of automatically detecting and understanding events is an essential task for effective monitoring in surveillance systems. Methods for indexing and retrieval of object trajectories in large surveillance databases are therefore of ma-

[§]This section is based on a collaboration with Pedro Quelhas as an extension to his Ph.D. work (Quelhas, 2006).

jor importance for data accessibility. Also, learning trajectories patterns described by tracked objects can be used for anomaly detection and behaviour prediction.

A large variety of different schemes were proposed in the past to describe object-based trajectories. Most of them rely on polynomial models and other function approximations to create a compact descriptor for the trajectory (Aghbari *et al.*, 2003; Chang *et al.*, 1998; Jin and Mokhtarian, 2004; Jung *et al.*, 2001). Also, Chebyshev polynomials were used by Cai and Ng (2004). Johnson and Hogg (1996) and Hu *et al.* (2004b) alternativelly used discrete point-based flow vectors. Discrete Fourier Transform (DFT) was also shown to improve the clustering quality of trajectories by Naftel and Khalid (2006).

Regarding the techniques applied to identify trajectory patterns, Johnson and Hogg (1996) proposed a statistical model of object trajectories based on a structure with two layers modelling the probability density function (pdf) of possible instantaneous movements and trajectories inside the scene using a neural net of competitive unsupervised Learning Vector Quantisation (LVQ). Owens and Hunter (2000) applied a single layer SOM, incorporating second order information to the flow vectors that code the trajectory, besides position and instantaneous speed. Naftel and Khalid (2006) also relied in SOM for their method with DFT as input features. Distance metrics are also used to identify trajectories, namely the Hausdorff (Lou *et al.*, 2002). A more robust distance metric based on the concept of Longest Common Subsequence (LCSS) between two trajectories was used by Buzan *et al.* (2004).

In the remaining of this section we present a possible representation for object trajectories. Motion trajectories are represented as the location which the object occupies in the image and the motion from each location to the next. To classify trajectories we use latent aspect modelling.

5.6.1 Application scenario

The shopping scene used in previous sections was also used for trajectory analysis. Specifically, we used the dataset $\mathcal{D}_e$ defined in appendix, Section A.5. The dataset consists of 124 trajectories, each associated to one out 5 of trajectories. These trajectories are related to the activities of people in the shopping, such as walking through the corridor or entering the store from different directions; these are illustrated in Figure 5.11. For each characteristic trajectory we also take into account the direction of the movement (for example: entering or leaving the store).

5.6.2 Trajectory representation

Our approach to trajectory representation is based on the quantisation of the trajectory using a grid, and tracking the transition between grid locations of the object being tracked. To simplify the process we consider that from each location the trajectory can move only to its 4 neighbours (city block metric).

Figure 5.11: *Characteristic trajectories to classify/index. Five trajectories were identified: 1) crossing the whole corridor (blue and dot-dot-dashed line), 2) entering/leaving the shop from/to the corridor on the left of the shop (yellow and short dashed line), 3) entering/leaving the shop from/to the corridor on the right of the shop (green and solid line), 4) entering/leaving the shop from/to the corridor in front of the shop (red and dot-dashed line), and 5) entering/leaving the shop from/to the far corridor on the right of the shop (magenta and long-dashed line).*

The trajectory $t = \{(x_i, y_i), ...\}$ is represented by directional elements $t' = \{e_i, ...\}$, which represents the movement of the object being tracked, from a point on the quantisation grid to another (represented by green arrows in Figure 5.12).

The final representation is the histogram of the occurrence of each directional element, whose dimensionality is independent of the trajectory size (Figure 5.13). This representation is compact and easy to compare.

5.6.3 Trajectory classification

In the case of trajectory classification we used a k-NN classifier, with $k = 1$. We used "leave one out" cross-validation, resulting in a total of 124 experimental runs, each one with either success or failure as a result. The results are presented in Table 5.3.

Compared with the polynomial representation, using the grid quantisation method we improved the trajectory classification performance. However, further work can be done in the definition of the base grid in which the representation is based. Making a specific grid for this scenario can lead to improved performance in the trajectory representation. On the other hand, this would increase the complexity and would make

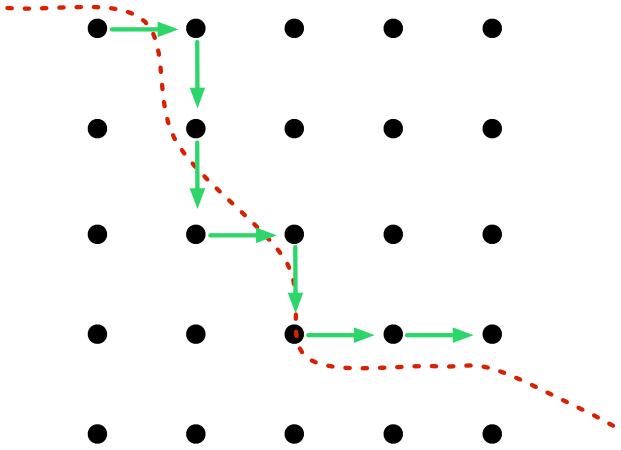


Figure 5.12: *Trajectory representation concepts. The graphical representation of the quantisation grid (black dots), the original trajectory (red dotted line), and the resulting motion elements (green arrows) are shown.*

method	error(%)
polinomial	45.2
grid quantisation (10x10)	29
grid quantisation (10x15)	27.4
grid quantisation (15x15)	24.2

Table 5.3: *Results of trajectory classification. Two representation schemes were used – polinomial and grid quantisation.*

more difficult the portability of the method.

5.6.4 Using latent aspect modelling

The trajectory representation presented in the previous section suffers from sparsity, ambiguities and some remaining sensitivity to translation. To address these issues we introduced latent aspect modelling.

Several latent aspect models such as Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 2001), Latent Dirichlet Allocation (LDA) (Blei *et al.*, 2003), and Multinomial PCA (MPCA) (Buntine, 2002) have been proposed in the literature for discrete components analysis.

In this work, we consider the PLSA model (Hofmann, 2001), that assumes each occur-

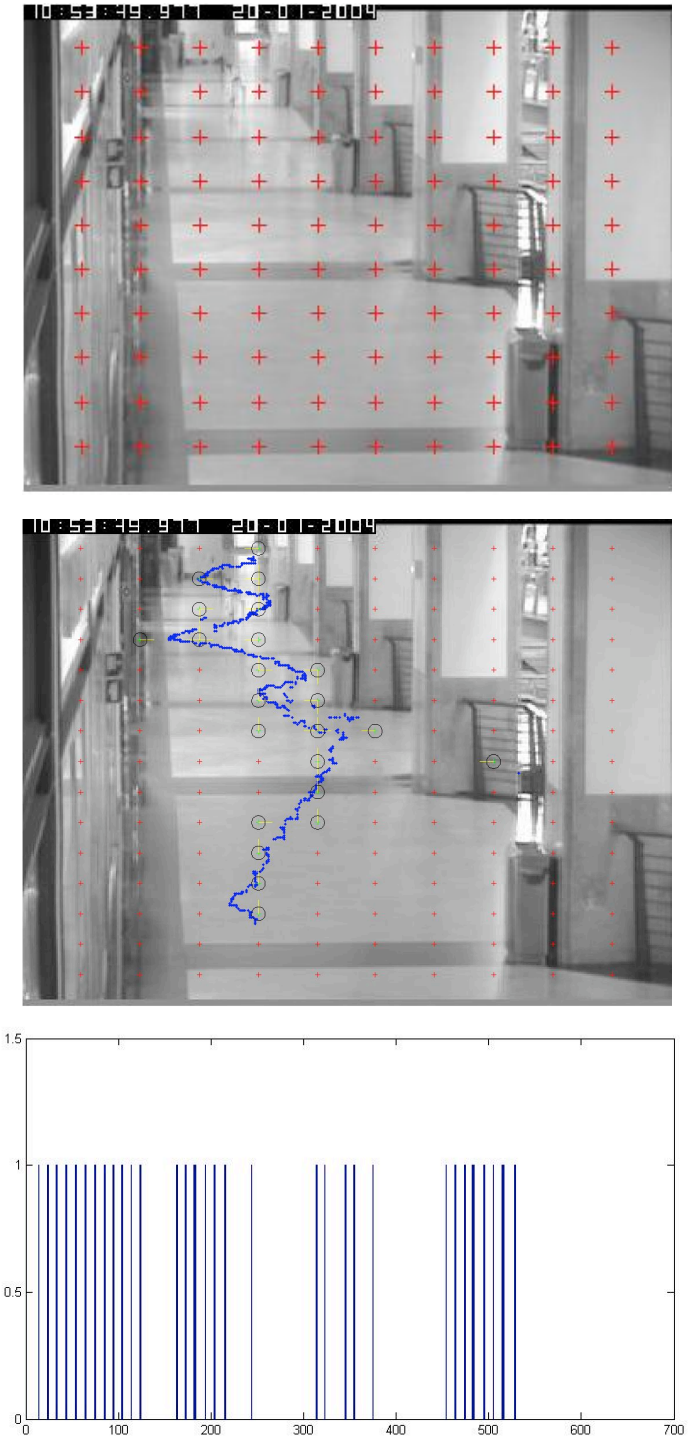


Figure 5.13: *Example of how the trajectory representation is created. The sampling grid of 10x10 points (left), the trajectory quantisation (center), and the resulting histogram representation (right) are shown.*

rence of a motion element e_j to be independent from the trajectory it belongs to given the latent variable z_k, that corresponds to the joint probability expressed by:

$$P(e_j, z_k, t_i) = P(t_i)P(z_k \mid t_i)P(e_j \mid z_k). \tag{5.7}$$

The joint probability of the observed variables is the marginalisation over the N_A latent aspects z_k as expressed by:

$$P(e_j, t_i) = P(t_i) \sum_{k=1}^{N_A} P(z_k \mid t_i)P(e_j \mid z_k). \tag{5.8}$$

The multinomial distributions $P(z \mid t_i)$ and $P(e \mid z_k)$ are estimated with an EM algorithm on a set of training documents. Since each motion element e_j represents a specific location and direction (see previous section) we can represent $P(e \mid z_k)$ for a specific z_k.

The aspect indices have no intrinsic relevance to a specific class, given the unsupervised nature of the PLSA model learning. We can however inspect each aspect to observe the meaning that they may have in terms of our target characteristic trajectories. Aspects can be conveniently illustrated by their most probable trajectories in a dataset. Given an aspect z, trajectories can be ranked according to:

$$P(t|z) = \frac{P(z|t)P(t)}{P(z)} \propto P(z \mid t), \tag{5.9}$$

where $P(t)$ is considered as uniform.

Figure 5.14 displays the 8 best-ranked trajectories for 6 aspects to illustrate its potential "semantic meaning". Among these 6 aspects it is possible to clearly identify the 5 defined trajectories. If we compare with Figure 5.11, the aspects from left to right and from top to bottom can be generally associated with trajectories 5, 3, 1, 2, 1, and 4, respectively.

Finally, we performed the same test described in the previous section, but in this case using PLSA. The results are summarised in Table 5.4.

method	error(%)
PLSA (10x10)	28.2
PLSA (10x15)	18.6
PLSA (15x15)	20.1

Table 5.4: *Results of trajectory classification using PLSA.*

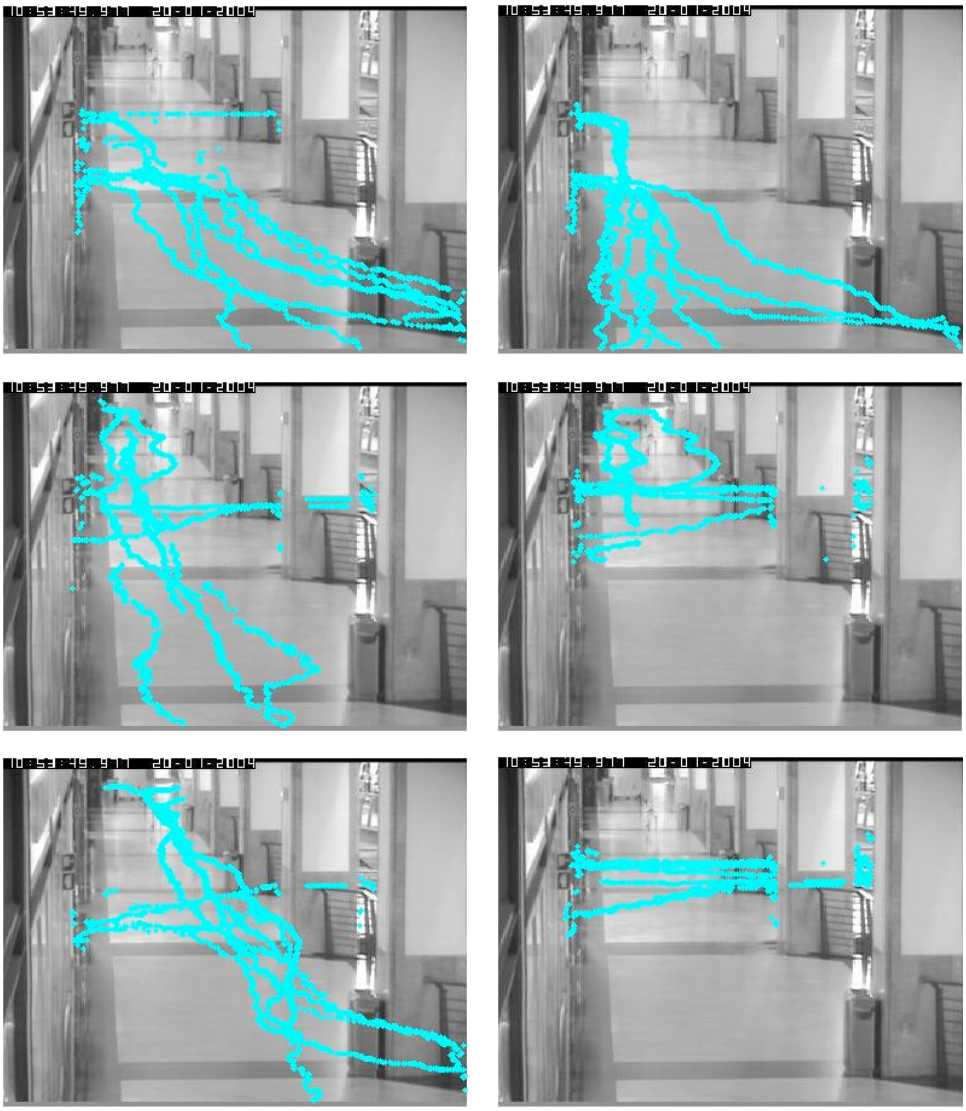


Figure 5.14: *Highest ranking 8 trajectories for 6 aspects from the PLSA model. If we compare to the trajectories defined in Figure 5.11 it is possible to verify that generally each aspect can be associated to one trajectory.*

5.7 Audio events detection

Most automatic event detection systems for real-world scenes, such as surveillance systems, are only based on visual analysis. However, the detection of many abnormal situations can benefit from the information conveyed by audio (Smeaton and McHugh, 2006). The hierarchical approach discussed in Section 3.4 can be used for this purpose. Specific models can be trained to detect abnormal events, like panic situations, gun shots, explosions, etc. Clavel *et al.* (2005) follow a similar approach to detect sounds produced by gun shots. The authors follow a statistical novelty detection approach (Markou and Singh, 2003), based on the comparison between the input audio and pre-trained models – an hierarchical model of gun shots of different nature and an environment sounds model. Both models rely on Gaussian Mixture Models (GMMs).

Many other event detection systems are based on binary GMM classifiers laid on an hierarchical classification scheme. However, different systems use different features to characterise the input audio signal. Mel Frequency Cepstrum Coefficients (MFCC) in combination with other features are often used, but essentially it depends on the scope of application and the events to be detected. With the goal of distinguishing four events/activities (talk, shout, knock and footsteps), Atrey *et al.* (2006) showed that Linear Predictor Coefficient (LPC) perform well with foreground/background segmentation and in distinguishing normal talk and shouting accurately, while Log Frequency Cepstral Coefficients (LFCCs) are more suited to segregate vocal and non-vocal events. Zhuang *et al.* (2008) proposed a method for feature selection to be used for audio event detection.

A major difficulty in audio detection systems is related to environmental noise. This noise is typically non-stationary and may have more energy than the audio event to detect. Cristani *et al.* (2004) introduced an online adaptive background modelling for audio (a very similar approach was also followed by Radhakrishnan *et al.*, 2005). They follow a similar rationale to Mixture of Gaussians (MoG), used for visual background modelling and subtraction as previously discussed in Section 3.2.1. The same authors also proposed combining audio and visual foreground/background segmentation to find multimodal events (Cristani *et al.*, 2006). The association between audio and visual information is done by analysing a AV Concurrence (AVC) matrix, which encodes the degree of simultaneity of audio and video. Valenzise *et al.* (2007) on the other hand, use solely audio sensors. They combined a GMM-based classifier system that identify screams and gun shots with a microphone array sound localisation system; the aim is to detect and localise the audio events in public spaces.

5.8 Summary

The automatic detection of events is an important task in order to effectively index and browse large libraries of multimedia content. Otherwise it would be difficult and time-

consuming to examine for example the sequences captured by a large-area surveillance system.

Events can be defined as real-world occurrences that unfold over space and time and can be classified according to their spatial and temporal uniformity. For example, temporal textures have indeterminate spatial and temporal extent, for example wind swinging trees or flowing water, activities are temporarily periodic but spatially restricted, for example a person walking or running, and motion events are isolated actions that do not repeat either in space or time, for example a door opening. Each of these types are associated with a characteristic approach for modelling and recognition. Alternatively, events can be distinguished according to their commonness of occurrence. Events can either be usual or unusual (alternatively, normal or abnormal).

In this chapter we focused on devising a complete system for detecting multiple appearances of persons, featuring a standard tracking algorithm, whose output was given to a multi-tracker object matching algorithm. The system outputs timelines representing the presence of persons in the different cameras along time. If a person is matched with the stored model of persons, the associated track is labelled accordingly. The obtained results demonstrate that the system is able to deal with scenarios composed of multiple cameras covering the same field of view or non-overlapping ones. Moreover, it is able to cope with differences in illumination and scale between the multiple cameras without using camera calibration parameters or a priori information. With this system, detecting and tracking a person is possible, even when observations of objects are not available for relatively large time periods. We evaluated the system using a dataset of sequences with diverse scenes. The more crowded scenes presented more difficulties and worse results were obtained.

In addition to the evolution in time approach of which the timeline representation of the appearances of persons is an example, event retrieval can also be based on the analysis of trajectories. We have evaluated a compact and simple representation for trajectories in video which can easily be adapted to many applications. We based trajectory classification in latent aspect modelling in order to create a space where the trajectory representation is invariant to occlusions, loss of tracking and noise.

Visual analysis can be found in nearly all recent intelligent analysis systems of real-world scenes, such as surveillance systems. However, the detection of many abnormal situations can benefit from the information conveyed by audio. We briefly discussed how recent work in audio-based event analysis is bringing a valuable contribution to these intelligent analysis systems.

In the next chapter we introduce a framework that can be used to integrate the multiple methods mentioned until now. The ultimate goal is to simplify the development, including testing and prototyping, of algorithms that rely on various modalities and with intricate relations between them.

6 Multimodal analysis framework

The design and implementation of multimedia signal processing systems is challenging especially when efficiency and real-time performance is desired. Researchers frequently have to rely on a combination of different software tools for each modality to assemble proof-of-concept systems which are inefficient, brittle and hard to maintain. MarsyasX, a new open-source cross-modal analysis framework follows a dataflow architecture where complex networks of processing objects can be assembled to form systems that can handle multiple and different types of multimedia flows with expressiveness and efficiency.

6.1 Overview

Over the last decade we have witnessed a proliferation of multimedia content that is easily and widely accessible. One of the main challenges facing multimedia research is how to analyse and search such huge amounts of information. The multimedia signal processing community has been actively working on these problems. Often this research is pursued by multiple groups in different and very specific areas. There are currently several examples of frameworks developed under such specific scenarios.

In the audio analysis and processing community, many examples of well known open source software frameworks exist, such as CLAM (Amatriain, 2007) [22], PD (Puckette, 1997) [23], Chuck (Wang and Cook, 2004) [24], FAUST (Orlarey *et al.*, 2006) [25], STK (Scavone and Cook, 2005) [26], HTK [27] (Young *et al.*, 2002), among others. Commercial tools also exist in this area, as is the example of MAX/MSP [29].

In what regards visual processing tools, there are also many open libraries that deal exclusively with image or video processing, namely OpenCV [17], LTI-Lib [18], VXL [19], and The Recognition and Vision Library (RAVL) [20], just to mention few. All these projects focus exclusively on computer vision and can be seen as utility libraries since they do not provide mechanisms to easily assemble algorithms based in building blocks.

Although such area-centric frameworks have been successfully and increasingly used by users within their respective areas, the lack of intercommunication with other fields of research leads to excessive overheads, re-implementation of similar techniques or even constant reinventions. It is very likely that the current small number of cross-fertilisation examples between different areas in multimedia analysis and processing

may be a result of the lack of common tools and frameworks. Such software tools are key factors in bridging the differences and distances between those communities.

MarsyasX is an open-source C++ framework that builds upon the original Marsyas project, and aims at just that – provide a common ground so that different methodologies and techniques with roots in different areas can coexist and cooperate towards a more integrated solution. In fact, MarsyasX represents a new line of development for Marsyas [21], an open-source software framework which finds its roots in the Music Information Retrieval community. Its name stands for Music Analysis, Retrieval and SYnthesis for Audio Signals, and the current version 0.2 (Tzanetakis, 2008) evolved from Marsyas 0.1 (Tzanetakis and Cook, 2000) which focused mostly on audio analysis, being especially suited for the development, testing and prototyping of analysis, processing and machine learning algorithms for audio signals. Marsyas was recently used for the submission of several algorithms to the MIREX2007 evaluation exchange [47], showing comparable results to other state-of-the-art algorithms (e.g. it was ranked first in the *Audio Mood Classification* task and second in the *Audio Artist Identification* task). Interesting to note are the computational runtimes achieved by the Marsyas algorithms when compared to those of other contestants which were systematically lower in several orders of magnitude (e.g. in the *Audio Mood Classification* task, the Marsyas based algorithm, ranked first, took 122 seconds per fold against the 521 seconds per fold taken by the second fastest, though ranked last, algorithm). Similar results were more recently achieved in the 2008 edition of MIREX [47].

Other frameworks and tools already exist that allow the integrated processing of audio and video streams. Although not originally created as multimodal platforms, Pd can be extended with visual processing modules from GEM [30], and Jitter [31] adds video and image processing abilities to the MAX/MSP environment. EyesWeb [32], on the other hand, has been originally conceived for supporting research on multimodal expressive interfaces and multimedia interactive systems (Camurri *et al.*, 2004), but although being freely available, is not an open-source initiative. These frameworks are mostly geared towards real-time applications. In addition to supporting real-time applications, MarsyasX has been designed with multimedia mining and retrieval applications in mind and has support for batch processing, machine learning tools, and interoperability with mining tools such as the Weka machine learning framework (Witten and Frank, 2005) [34].

In this chapter we will present the main architectural concepts of Marsyas, as well as the innovations brought by MarsyasX to incorporate multimodal analysis capabilities.

6.2 Marsyas architecture[§]

Systems in Marsyas are expressed as interconnected dataflow networks of processing modules. Each processing module performs a specific task that always consists of a matrix transformation. Processing networks are assembled using implicit patching.

6.2.1 Dataflow programming

Dataflow programming is based on the idea of expressing computation as a network of processing nodes/components connected by a number of arcs/communication channels. Dataflow programming has a long history. The original (and still valid) motivation for research into dataflow was to take advantage of parallelism. Motivated by criticisms of the classical von Neumann hardware architecture (Ackerman, 1982), dataflow architectures for hardware were proposed as an alternative in the 1970s and 1980s. During the same period a number of textual dataflow languages such as Lucid (Ashcroft and Wadge, 1977) were proposed. Despite expectations that dataflow architectures and languages would take over from von Neumann concepts this didn't happen. However during the 1990s there was a new direction of growth in the field of dataflow visual programming languages that were domain specific. In such visual languages programming is done by connecting processing objects with "wire" to create "patches". Successful examples include LabView, Simulink and in the field of Computer Music MAX/MSP (Zicarelli, 2002) and Pure Data (Puckette, 1997). In fact, expressing audio processing systems as dataflow networks has several advantages (Tzanetakis, 2008). The programmer can provide a declarative specification of what needs to be computed without having to worry about the low level implementation details of how it is computed. The resulting code can be very efficient and have a small memory footprint as data just "flows" through the network without having complicated dependencies. In addition, dataflow approaches are particularly suited for visual programming. One of the initial motivations for dataflow ideas was the exploitation of parallel hardware and therefore dataflow systems are particularly good for parallel and distributed computation. A comprehensive overview of audio processing frameworks from a Software Engineering perspective can be found in (Amatriain, 2007). Another recent trend has been to view dataflow computation as a software engineering methodology for building systems using existing programming languages (Manolescu, 1997).

As a result, the idea of dataflow programming has been determinant in the design of Marsyas 0.2, where complex networks of processing objects can be assembled to form systems that can handle audio and data flows with expressiveness and efficiency.

[§]This section is in part based on the article *Interoperability and the Marsyas 0.2 runtime* published in the International Computer Music Conference (Tzanetakis *et al.*, 2008).

6.2.2 Implicit patching

To assemble multimedia processing systems, modules are implicitly connected using hierarchical composition. Special "Composite" modules such as Series, Fanout, Parallel are used for this purpose. For example, modules added to a Series composite will be connected in series, following the order they were added - the first module's output is shared with the second module's input and so on. Moreover, the *tick*() method is called sequentially following the same order. Figure 6.1 shows an example of how composite and non-composite modules can be used. This paradigm differs from typical processing tools based on explicit patching such as CLAM (Amatriain, 2007), MAX/MSP (Puckette, 1991) or PD (Puckette, 1997). In explicit patching the user would first create the modules and then connect them by explicit patching statements. The interested reader may find a comprehensive discussion about the differences between implicit and explicit patching in (Bray and Tzanetakis, 2005)

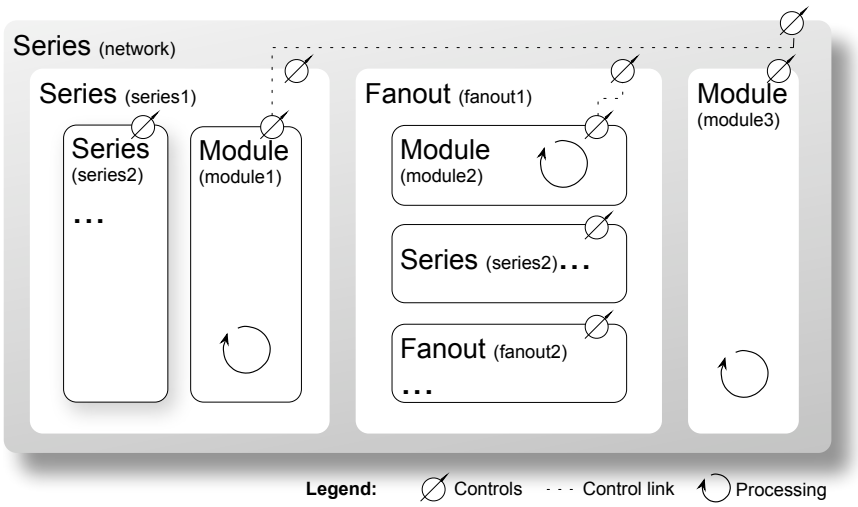


Figure 6.1: *Building blocks in Marsyas 0.2.*

6.2.3 Basic processing units and networks

As previously described, systems in Marsyas are expressed as interconnected dataflow networks of processing modules. Each processing module performs a specific task that always consists of a matrix transformation.

Audio and other types of data are represented by matrices with some semantics associated with them: rows represent observations (over time) and columns represent samples in time. For instance, a stereo audio signal might be processed in chunks of 2 observations (i.e., two channels) and 512 samples in time. This clean data structure,

although quite specific, suits well audio processing applications. All modules process one input matrix (known as a slice) and store the result on another matrix so it can be shared with the next processing module. Hence, each module accepts only one input and produces one output. Processing is performed on defined chunks of data and is executed whenever the *tick*() function of the module is called.

All processing blocks in Marsyas are called MarSystems and provide the basic components out of which more complicated networks can be constructed. Essentially any audio processing algorithm can be expressed as a large composite MarSystem (discussed below) which is assembled by appropriately connected basic MarSystems. Some representative examples of MarSystems include sound file reading and writing (e.g. wav, au, mp3 and ogg audio file formats), real-time audio input and output (i.e., from and to a soundcard), signal processing algorithms (e.g. filters, STFT, DWT), feature extraction (e.g. MFCC, centroid, rolloff, flux) and machine learning modules (e.g. k-NN, GMM, SVM, PCA, SOM).

To assemble multimedia processing systems, modules are implicitly connected using hierarchical composition. Special "Composite" modules such as Series, Fanout, Parallel are used for this purpose. The basic idea behind implicit patching (Bray and Tzanetakis, 2005) is to use object composition rather than explicitly specifying connections between input and output ports in order to construct the dataflow network. For instance, modules added to a Series composite will be connected in series, following the order they were added – the first module's output is shared with the second module's input and so on. Moreover, the *tick*() method is called sequentially following the same order. As an example, consider a small network consisting of a Series module which contains three other modules: a source, a processing module and a sink. When *tick*() is called in the Series, implicitly the *tick*() method is called in all three modules as shown in Figure 6.2.

Dataflow in Marsyas is synchronous which means that at every "tick" a specific slice of data is propagated across the entire dataflow network. This eliminates the need for queues between processing nodes and enables the use of shared buffers which improves performance. This is similar to the way UNIX pipes are implemented but with audio specific semantics.

6.2.4 Dynamic access to modules and controls

In Marsyas, each module in a processing network can be accessed by querying the system with a path-like string. Taking the example shown in Figure 6.1, if we wanted to reach the processing module named `module1`, the query path would be:

```
/Series/network/Series/series1/Module/module1
```

The first "/" indicates the outermost module and the rest of the path is always composed by the concatenation of `Type/Name` strings. This naming scheme was inspired

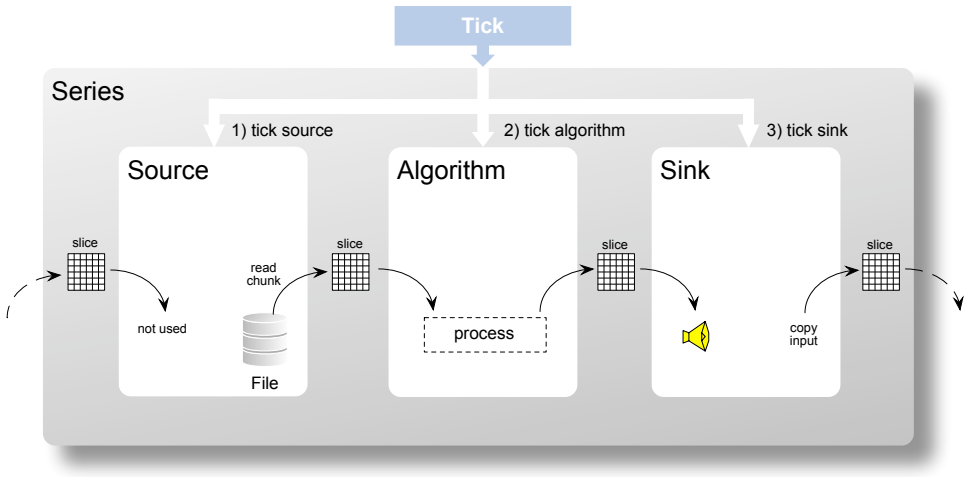


Figure 6.2: *Processing chunks of data in Marsyas.*

from the way messages are exchanged in Open Sound Control (OSC) (Wright *et al.*, 2003). It is possible to have access to some of the internal parameters of the modules using *controls*. Each module exports a list of controls which may be of different types (e.g. integers, floats, strings, vectors, or arbitrary user-defined types). They can be accessed for reading or writing by specifying the path to their parent module plus the `Type/Name` corresponding to the control.

One of the most useful characteristics of MarSystems is that they can be instantiated at run-time. Because they are hierarchically composed that means that any complicated audio computation expressed as a dataflow network can be instantiated at run-time. For example multiple instances of any complicated network can be created as easily as the basic primitive MarSystems. This is accomplished by using a combination of the Prototype and Composite design patterns (Gamma *et al.*, 1994).

6.2.5 Dynamic linking of controls

Controls can be linked as shown in Figure 6.1, so that changes to the value of one control are automatically propagated to all the others. There are plenty of interesting uses for this feature: parameter values that must be passed to more than one module in a system; feedback loops where results from modules ahead in the processing network are sent back to the first modules in the chain; shortcuts for other links, etc. Links can be defined (both at compile-time and at run-time) for controls with the same value type, either belonging to a same module, or to any other module in the network. Links can also be used to create *proxy controls* – in order to create a shortcut to a control from

a module deep inside other composite modules, it is possible to link it to a *proxy control* in the outmost module, created on demand for this task. This way multiple and easy to understand views for the control of the same algorithm can be created.

6.2.6 Interoperability

Considerable effort has been directed to developing interoperability layers with other software packages or projects. The objective is two-fold: on one hand to make use of features already provided by these packages or projects, and on the other hand to make Marsyas an important library other applications can interface with and build upon.

□ MATLAB

MATLAB is a powerful and widely used tool in several areas of research and development, with a large community of users and available routines for math and multimedia analysis and processing algorithms. Additionally, MATLAB provides easy to use and advanced plotting facilities, a major asset for researchers developing algorithms for audio, image and video processing. Until recently, developers always had to make a hard choice regarding their development language: either opt for the flexibility and ease of use of MATLAB or decide in favour of efficiency and performance as provided by an Object-Oriented Programming (OOP) language like C++. In its latest versions, MATLAB includes the ability to exchange data in run-time with applications developed in Fortran, C or C++, through and API named MATLAB Engine. Marsyas implements a singleton wrapper class for the MATLAB Engine API, enabling Marsyas developers to easily and conveniently send and receive data (i.e., integers, doubles, vectors and matrices) to/from MATLAB in run-time. It is also possible to execute commands in MATLAB from calls in the C++ code as if they have been called in the MATLAB command line. This enables the execution of MATLAB scripts and the access to all MATLAB functions and toolboxes from within Marsyas C++ code. The MATLAB wrapper class in Marsyas provides three basic methods:

- `PUTVAR(Marsyas_var, MATLAB_var_name);`
- `GETVAR(Marsyas_var, MATLAB_var_name);`
- `EVALUATE(MATLAB_cmd);`

By means of function overloading, these three methods allow exchanging different types of variables from Marsyas/C++. They can be called from anywhere in the Marsyas C++ code without any need of changes in the Marsyas interfaces, making it simple to send data structures to MATLAB for convenient inspection and analysis, calculations and plotting, and then get them back in Marsyas for additional processing. These features are only available when MATLAB is installed in the system and Marsyas is built with MATLAB Engine support. Any MATLAB Engine calls in Marsyas code are automatically ignored otherwise. A C++ snippet code is presented next, illustrating how a

vector of real values can be processed and exchanged with MATLAB, using the Marsyas MATLAB Engine wrapper class:

```
// create a std::vector of real numbers
std::vector<double> vector_real(4);
vector_real[0] = 1.123456789;
vector_real[1] = 2.123456789;
vector_real[2] = 3.123456789;
vector_real[3] = 4.123456789;

// send a std::vector<double> to MATLAB
MATLAB_PUT(vector_real, "vector_real");

// do some dummy math in MATLAB
MATLAB_EVAL("mu = mean(vector_real);");
MATLAB_EVAL("sigma = std(vector_real);");
MATLAB_EVAL("vector_real = vector_real/max(vector_real);");

// get values from MATLAB
double m, s;
MATLAB_GET(m, "mu");
MATLAB_GET(s, "sigma");
MATLAB_GET(vector_real, "vector_real");
```

□ Qt4

Trolltech's Qt is a comprehensive development toolkit that includes features, capabilities and tools that enable the development of cross-platform C++ applications. Such features include multi-platform Application Programming Interfaces (APIs) and classes for the development of Graphical User Interfacess (GUIs), signalling, and multi-threaded execution. In its 4th version, Qt is available as a dual-license software toolkit for all the supported platforms (i.e., Linux, Mac OS X and Windows). For open source applications such as Marsyas one of the licences is open-source General Public Licence (GPL).

Marsyas, although not bound specifically to Qt, uses this toolkit as its preferred solution for the development of GUIs. Its use is however totally optional, allowing the developer to choose any other library, or even including no GUI support at all (missing in this case all of the GUI extra features already implemented in some Marsyas classes and applications). There are currently two approaches available for GUI development using Qt4 in Marsyas.

The first way is using a *delegation design pattern*, where the core C++ Marsyas classes in charge of the actual processing are wrapped by an entity that takes care of all the message passing between the GUIs and the processing network. Additionally, using Qt's multithread features, this wrapper makes sure that GUIs and the processing code are executed in independent threads. This allows the implementation of responsive GUIs and the best use of the last generation multi-core processors. This approach is most

suited for the development of customized GUI front-ends for Marsyas based applications. It allows interacting with the processing network (by means of reads/writes to its module's controls) in an intuitive and real-time manner.

The second way is conditionally making all Marsyas modules inherit from Qt's base class *QObject*. This automatically embeds Qt's most advanced features (such as *signals* and *slots*) into most Marsyas core classes. This avoids the use of middle-layers for message exchange between GUIs and the processing modules. Additionally to improving efficiency, this approach facilitates the implementation of more advanced functionalities for GUI and multi-threaded processing. As a drawback, this implies a tighter compile-time bind between Marsyas and Qt, which can make the desired independence between the two frameworks more difficult to assure and maintain. Given the additional features inherited from Qt, this approach allows the common implementation of GUIs for all Marsyas modules, such as widgets for viewing/modifying the list of controls from any Marsyas module, or even the creation of specialised GUIs for data plotting or parameter modification.

□ Open Sound Control

Open Sound Control (OSC) (Wright *et al.*, 2003) is a protocol for communication among computers, sound synthesisers, and other multimedia devices that is optimised for modern networking technology. There are many implementations of OSC and most computer music environments (such as Max/MSP, PD, Chuck, CSound) have the ability to send and receive open sound control messages.

The control mechanism in Marsyas was inspired from OSC so the mapping of controls to OSC messages is very straightforward. The path notation is used to specify the full name of the control and the value of the message is directly mapped to the value of the control.

The mapping of OSC messages to Marsyas controls is part of the Qt4/Marsyas integration code. *OscMapper* is the interface between OSC, Marsyas and Qt4. It acts as both an OSC server and client and allows particular OSC hosts and clients to be associated with particular MarSystems. The communication is abstracted as signals and slots following the way Qt4 structures communicate between interface components. The user interface programmer only needs to specify the information about where the OSC messages will be coming from and all the rest is taken care directly by the mapping layer. For example, this way it is straightforward to use PD to send OSC messages to modify the parameters of a phasevocoder running in Marsyas. The data flowing through a Marsyas network is also accessible through controls so audio information can also be exchanged

□ Marsyas runtime as a Max/MSP external

Max/MSP7 allows the creation of so called "external" processing units which can be written in C/C++ following a specific API. These externals can then be used as building blocks in the visual programming environment.

Efforts have been put into the implementation of a general external that can be used to load any audio processing system expressed in Marsyas. The main challenge was to completely decouple the audio buffer rate of Max/MSP from the audio buffer size used by the Marsyas runtime. This is achieved through a dynamic rate adjusting sound source and sound sink for the input and output to the Marsyas part of the patch.

For example if the audio buffer size of the Max/MSP patch is 64 samples and Marsyas requires buffers of 256 samples then four buffers of 64 samples will be accumulated before sent to Marsyas for processing. Similarly at the Marsyas output the 256 samples will be broken into 64 sample buffers to be sent back to Max/MSP. Arbitrary sizes are supported and there is no requirement that one buffer should be smaller than the other. This is achieved by using circular buffers with dynamically adjustable length. Controls can be read and written through control outlets and inlets of the external.

6.3 MarsyasX – a step toward cross-modal analysis[§]

MarsyasX stands for Marsyas *"cross-modal"* and borrows from Marsyas 0.2 most of the concepts, namely the *hierarchical composition* paradigm and the implicit patching of modules. Similarly, data is processed in defined chunks by calling a *tick()* function and each module also has a set of controls that are used to access their internal parameters. The main conceptual difference is in the way data is exchanged between processing modules. Instead of using shared matrices of real values, MarsyasX exchanges data through a *payload mechanism*. Whenever data is produced in a given module at each tick, a payload is created. This payload, "carrying" the data, is then sent to the output channel as depicted in 6.3. A channel is a connection between adjacent modules where payloads are stacked while waiting to be processed. It is important to note that channels are established implicitly, according to the type of composite being used.

This data exchange mechanism is highly generic and flexible, supporting any type of data (e.g. images, audio frames, MIDI, XML, lists of points, etc.). However, it does not live without its own specific issues such as timing and synchronisation that are addressed later in this text. In addition, payloads can be hierarchically grouped into flows to enable typing and naming of time series of payloads. Alongside with this fundamental difference when compared to the previous version, MarsyasX includes additional improvements, like an integrated implementation of events associated with controls that can (1) trigger predefined actions, (2) connect controls with expressions (similarly

[§]This section is based on the article *MarsyasX: multimedia dataflow processing with implicit patching* published in the ACM Multimedia conference (Teixeira *et al.*, 2008).

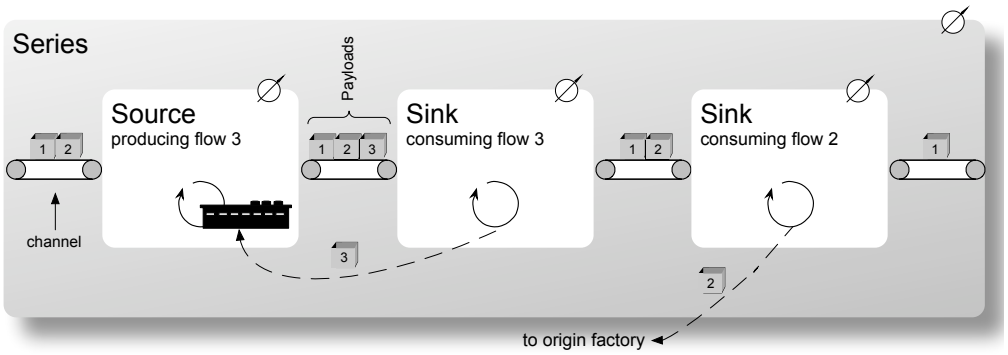


Figure 6.3: *Payload architecture in MarsyasX.*

to Marsyas 0.2), (3) synchronise with GUIs and (4) support distributed networks. These concepts are explained in more detail in the following subsections.

6.3.1 Payloads

Payloads are the basic transport entities of data across the network. They are sent and received from the channels connecting the modules, forming the processing network. A payload stores generic data and a set of metadata fields that characterise the payload, as summarised in Table 6.1.

Metadata field	Description
Type	Flow type identifier
Name	Flow name identifier
TOC	Time of Capture
TTS	Time to Schedule
Factory	Origin factory identifier
TemplateID	Template identifier
Header	Array of flow controls that changed

Table 6.1: *List of metadata fields contained by a payload.*

A more thorough description of the metadata fields is given in the following subsections.

6.3.2 Data flows

Different payloads can be grouped together in abstract entities called *flows*. A flow consists of all payloads that have the same type and are tagged with the same name. Hence it is always identified by the tuple $(Type, Name)$. The first two metadata fields shown in Table 6.1 form such a tuple.

The *flow type* field is used to distinguish different types of flows. Each payload of a given type must contain the same type of data. For example, a Visual flow payload should always contain an image, an Audio flow payload, an audio frame, and so on. Note that despite being of the same type, payloads of the same flow type may contain data with different characteristics – for example: images of different sizes or represented in different colour spaces.

On the other hand, the *flow name* field identifies different flows of the same type. Inherently, the same rules are applied for flows of the same type. Distinguishing flows of the same type can be useful for handling, for example, multiple video feeds – we might want to handle differently each feed. Moreover, it is often mandatory to distinguish different sources of data since many modules have stateful processing. If multiple sources are propagated as the same flow unexpected behaviour on these modules will occur.

Currently 5 types of flows are supported, namely Audio, Visual, XML, Multidata and Legacy. Each flow type is closely related to a data structure: matrix vector for multichannel audio frames, image supporting multiple colour spaces, XML tree structure, vector of independent matrices for generic data, and a matrix compatible with Marsyas 0.2 for legacy flows (detailed in Section 6.3.7).

6.3.3 Timing and synchronisation

It is important to preserve timing relations and constraints between data. For that purpose, payloads have two time metadata fields. The first, *Time of Capture TOC*, stores at what time the data held by the payload was created or captured. This time information is not supposed to change across the network. The *Time to Schedule TTS* field stores the time when the data should be processed by the modules that handle the payload. Unlike Time Of Capture (TOC), Time To Schedule (TTS) can be changed by the processing modules – for example, its value may be increased by a delay module. The difference between both TOC and TTS at the time when the payload is first created appears subtle although significant. Whereas TOC is used to synchronise data of different flows (with the same or different types and names), TTS is used to schedule payloads for processing. The latter is especially important if, for optimisation reasons, we want to parallelise the work load while at the same time maintaining coherent time relations. Both fields are integer values and are always referenced to a system-wide time base T_B. This time base is defined once at the beginning of the processing and

must remain fixed (e.g. 1ms). Whenever a new tick is triggered the current time T_{cur} is advanced by one unit of T_B. If there are any payloads of the respective flow on the input channel, such that $T_{cur} \geq TTS$, these are processed immediately and the result is output in a new payload. In case new data is created or if it is an in-place transformation, the same payload is forwarded. Any other payloads not satisfying this condition will remain in the input channel. This discrete behaviour implies a maximum time resolution error of $\epsilon_{max} = \frac{T_B}{2}$. Therefore, choosing T_B requires some knowledge of the time resolution required by the data being processed.

Let's consider the typical case of video and audio frames. In Figure 6.4 the values of TOC and TTS set at creation time for different frames are depicted.

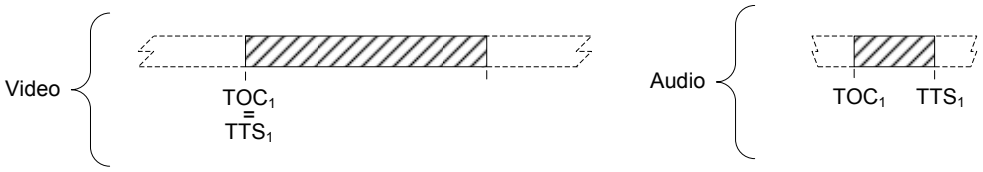


Figure 6.4: *Timing information associated to payloads.*

For the video case, each frame is "carried" by a payload with the same TOC and TTS. However, for the audio frames, the TOC will not be the same as the TTS since a frame of audio corresponds to an accumulation of samples along a period of time. The TOC will reference the time associated with the first sample while the TTS will be associated with the last sample.

6.3.4 Memory management

When a payload reaches a module that consumes the data without forwarding it or when it reaches the end of the network it becomes no longer useful. From an efficiency point of view, creating and destroying payloads continuously would be computationally expensive. To avoid this overhead, a simple recycling mechanism is used. If a module is a source of payloads, it will have an associated *payload factory*. A data *template* is attached to each factory – for instance, an image with a given dimension, planes, etc. If the properties of the data to be outputted do not change, this template should only be attached once and remain the same till further notice. When a payload is required, it will be requested to the factory, which will either reuse an existing one or create a new payload (i.e., allocate memory for it). The fields *Factory* and *TemplateID* are also set accordingly at this stage. If a new payload is required, the template will be used to create a copy of it, generating a payload with the exact same properties. When the same payload is no longer needed anywhere in the processing network, it will be returned to the corresponding factory where it will be stored in a recycle bin for future use. However, if the *TemplateID* no longer corresponds to the one currently assigned

to the factory (i.e., the properties have changed) it will be destroyed. This means that payloads of the same flow may contain at a given instant data with different properties. A simple example would be a network containing a Source, a Delay and a Sink module. If the source changes its properties after some time, no immediate reconfiguration is done across the network. The payloads in standby at the Delay module will still be able to be processed as before. When the payloads with the new properties reach the following modules any required reconfiguration will then be performed. The mechanism that flags and propagates these changes is based on the *Header* metadata field. This is a form of highly efficient type-specific garbage collection (or more accurately recycling) that is enabled by the strict semantics of time and dataflow processing used in MarsyasX.

In order to evaluate the impact in the performance of a typical network we tested the payload architecture alongside the slice architecture that is currently adopted by Marsyas 0.2. The setup consisted of a source and N test MarSystems. The source creates a 500x500 image at each tick and test MarSystems do nothing to the incoming image. Whereas the payload architecture avoids copying the whole image at each MarSystem tick, the slice architecture does not (cf. Figure 6.1). The impact of this can be verified in Figure 6.5. While the time taken to process 10 ticks rapidly increases with increased N for the slice architecture, the elapsed time remains almost stable for the range of N depicted in that figure. With such overheads the slice architecture can become unfeasible for typical visual processing networks.

Nevertheless, the overhead using the payload architecture increases, albeit slowly. In Figure 6.6 we show that a significant impact to the performance happens only for a very large number of MarSystems (> 10000).

6.3.5 Initial configuration and propagation of flow control updates

Many processing modules require some kind of initialisation like buffer allocation or device configuration. Often these depend on the type of data being processed and its properties. Again, for the video case, it is expected that the source module reading, for example, a local file, "knows" the width and height of each frame and therefore somehow propagates this information to the other modules. Moreover, if the file suddenly changes and so do the frame dimensions, it is once more expected that the other modules are informed of the changes. In MarsyasX this information is passed across the network the same way data is: using payloads. Each payload has a *Header* metadata field that contains a list of controls that have changed and whose change needs to be notified. Not all controls need to be propagated, only *flow controls*. When a module and the corresponding controls are created, the latter can be tagged as flow or local controls. Moreover, if the module has (or might have) different input and output properties, two controls are added: an *in-flow control* and an *out-flow control*. Whenever one flow control changes its value the next payload sent by the module will have the control's name and value included in the header. All other modules receiving the same

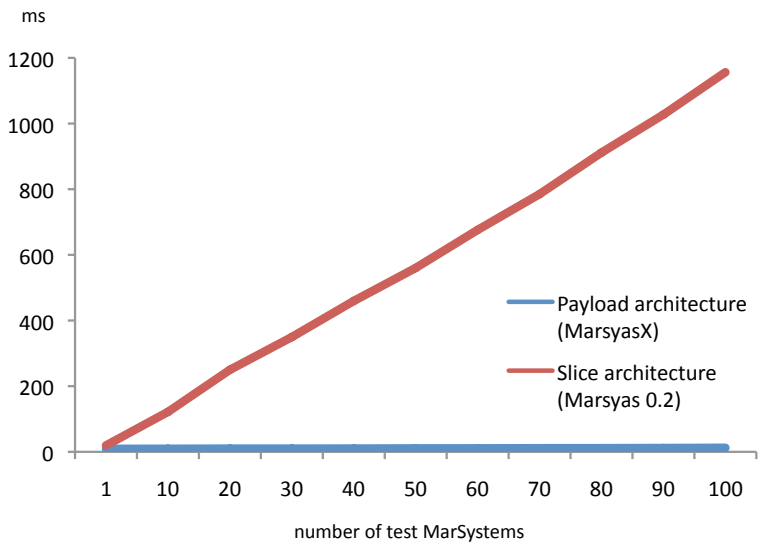


Figure 6.5: *Comparison of performance between the slice and payload architectures. The experimental setup consisted of 10 ticks in a network comprising one 500x500 image source and N test MarSystems. While with the slice architecture the time taken to process the ticks increases rapidly with N, using the payload architecture results in a much less noticeable ascent. Values were taken with a Pentium 4 3.4GHz with 1GB of RAM.*

flow will automatically read the header, update their own controls accordingly and forward the changes the same way. For the in- and out-flow controls, only the latter is sent and, naturally, only the corresponding in-flow control is set automatically. Ideally, all modules processing the same type of data should share the same set of flow controls. However, if no control exists in the current module with a name included in the header it is ignored but nevertheless forwarded. Figure 6.7 shows an example of how this process unfolds.

Before the first tick is triggered, it is also possible to explicitly request that all changes are propagated at once, avoiding unexpected delays due to initialization overheads.

6.3.6 Events

When a control changes its value an *Event Manager* is notified as shown in Figure 6.7. The role of this manager is to handle the propagation of the change across the system. Note that by system we are referring not only to all the modules composing the network but also other remote networks and modules, GUI widgets connected to the control and other registered "observers" or "listeners". The event `CHANGED` is always sent when the control's value changes but other events can be defined. Probes such as Threshold, HysteresisThreshold and Peak can be attached to controls and send custom

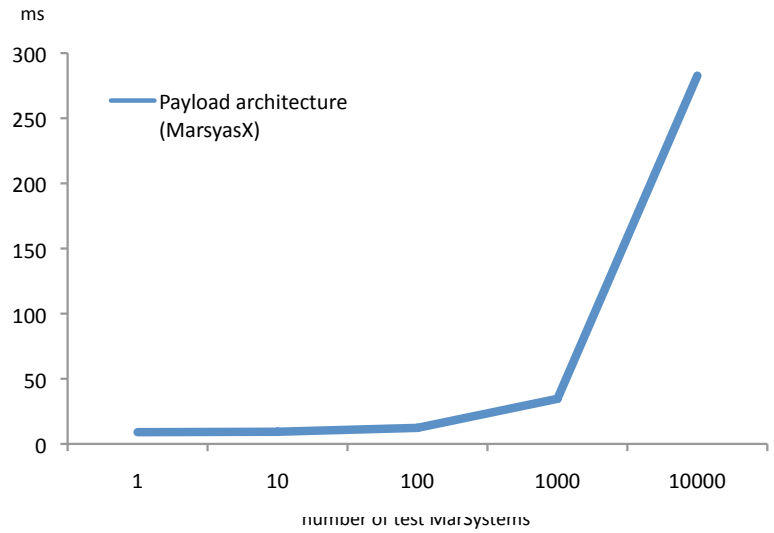


Figure 6.6: *Evolution of the performance using the payload architecture. We used a log10 scale and with this scale it is noticeable that the overhead increases with the number of MarSystems, but in much smaller proportion than with the slice architecture. The same experimental setup described in Figure 6.5 was used.*

events like `MAXREACHED`, `ZEROREACHED`, etc. Controls can be connected directly through a *link* (see Section 6.2.4) or through an *expression*. For example, suppose we want to watch the Peak Signal-to-Noise Ratio (PSNR) calculated by a specific module and change the parameter of an algorithm if it rises above a certain maximum value and falls under a minimum value. To achieve this, a HysteresisThreshold probe could be attached to the PSNR control and two connections, possibly with an expression, would be made between PSNR and the algorithm's parameter control: one for the case `MAXREACHED` event occurs and another for `MINREACHED`.

An implementation of events already exist in Marsyas 0.2 but have a different objective (Burroughs and Tzanetakis, 2006). The next step, under development, is to integrate the Marsyas 0.2 events systems under the same event framework described here.

6.3.7 Legacy interface with Marsyas 0.2

An important feature of MarsyasX is the legacy interface with Marsyas 0.2. Undoubtedly it is very important to still be able to use the large collection of modules available now and or in the future in Marsyas 0.2 releases. Since the framework base follows closely the previous, it is rather straightforward to support legacy modules. Figure 6.8 shows how this is done. A MarsyasX module called MarSystemLegacy wraps a legacy module and synchronises the controls of both. The most important difference is the

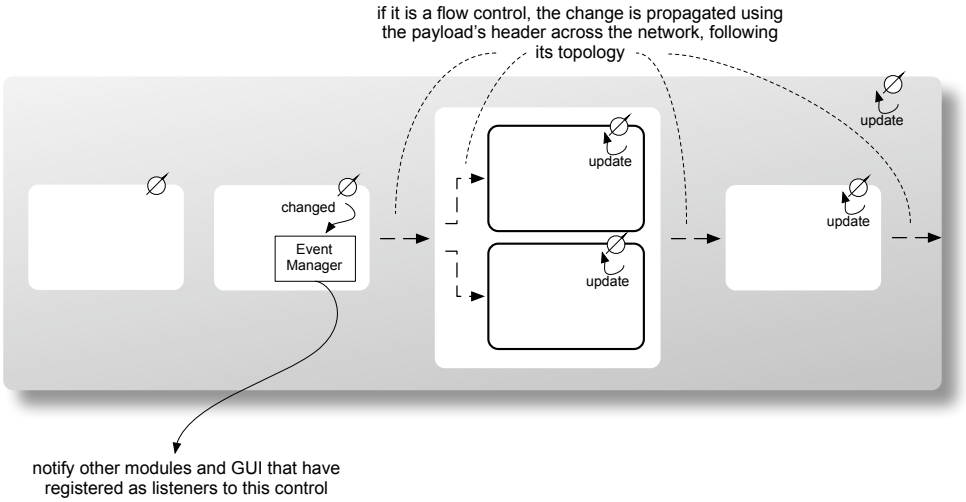


Figure 6.7: *Propagation of flow controls updates.*

way data is exchanged between modules. It is also the most costly operation, since it implies copying the data stored in the payload to the input slice of the legacy module and, after processing, from the output slice to the payload that will be sent to the output channel.

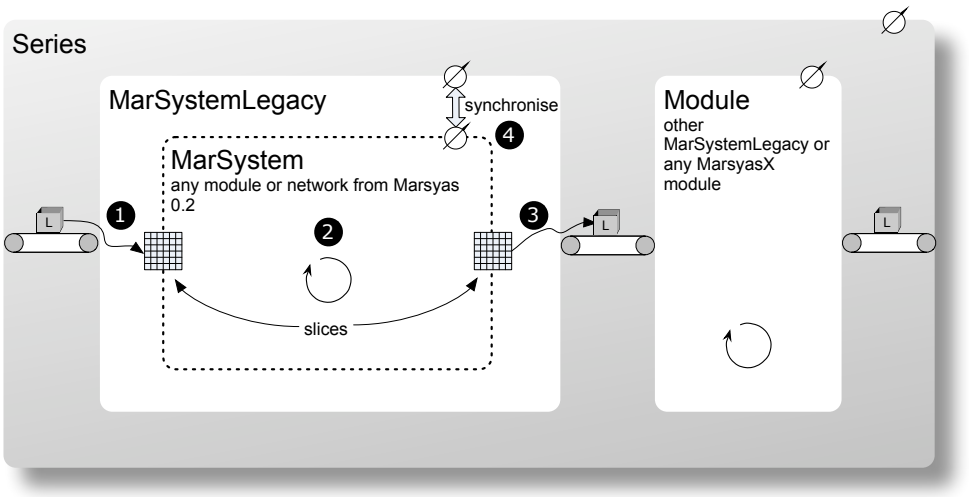


Figure 6.8: *Legacy interface with Marsyas 0.2.*

6.3.8 Modules

The similar architecture of MarsyasX and Marsyas 0.2, simplifies the porting of modules. Moreover, with the legacy interface it is possible to mix modules created with both versions. The plethora of Marsyas 0.2 modules for reading audio files, feature extraction, and audio analysis and synthesis can easily be made available for MarsyasX users. In addition to that, visual processing modules are being created, including modules for video and image IO, filtering, optical flow estimation, segmentation and feature extraction. Support for XML handling is also available. This is an ongoing process and more modules are expected to be developed.

Despite the complex architecture underlying the MarsyasX modules, the process of creating modules has been simplified. Typically, a user only needs to define the input and output flows and create a process function accepting the data according to the defined flows. This does not require any knowledge about the payload mechanism. Additionally, modules are managed using a simple plugin system. Plugins are dynamically loaded whenever necessary. Each plugin includes modules that are somewhat related. This has multiple advantages, namely: avoids excessive startup times due to module initialisation, allows the deployment of smaller packages containing only the strictly necessary, and opens the possibility for third-party modules (possibly with different licenses).

□ Creating new modules

Although the underlying some concepts of Marsyas 0.2 and MarsyasX can be somewhat complex to fully grasp, the process of creating new modules is fairly straightforward. In fact, the user can be oblivious of core details like payloads and factories. Starting with the simpler case, if we want to create a module that accepts an image, processes it, and outputs it, the C++ class would be something like:

```
class SimpleVisualModule : public MarSystemVisual
{
public:
  SimpleVisualModule(std::string name) :
    MarSystem(''SimpleVisualModule'', name),
    MarSystemVisual()
  {
  }

  MarSystem* clone() const
  {
    return new SimpleVisualModule(*this);
  }

  void build()
  {
```

```
    configureFlows(
      new Flow(FlowHandlerVisual::type(), ''*'', Flow::FLOW_INOUT),
      &SimpleVisualModule::process
    );

    MarSystemVisual::build();
  }

  bool process(mrs_image &img)
  {
    //... do something with img ...
    return true;
  }
};
```

The typical methods that need to be defined in a new module are: the *constructor* with the name of the module as an argument, a *clone* method that is called by a module manager called MarSystemManager, a *build* method where the handled flows are configured, and a *process* method where the actual processing occurs. The key to avoid handling directly with payloads and factories is the *configureFlows* call in the build method. Flows are associated with different types of handling; in this example the handling is `in-out`, but `in` and `out` are also possible. With the `in-out` handling mode the data carried by a flow is given as input to the module and outputted after the respective processing. With `in` handling flow data is inputted but never outputted. On the other hand, with `out` handling a new flow is outputted with no inputs. Arbitrarily complex combinations of flow type and flow handling are possible. For instance, if we need to create a module that handles both audio and video, the C++ class would be similar to:

```
class SimpleAVModule : public MarSystemVisual, public MarSystemAudio
{
public:
  SimpleAVModule(std::string name) :
    MarSystem(''SimpleAVModule'', name),
    MarSystemVisual(),
    MarSystemAudio()
  {
  }

  MarSystem* clone() const
  {
    return new SimpleAVModule(*this);
  }

  void build()
  {
    configureFlows(
      new Flow(FlowHandlerAudio::type(), ''*'', Flow::FLOW_INOUT),
      &SimpleAVModule::processAudio
    );
```

```
    configureFlows(
      new Flow(FlowHandlerVisual::type(), ‘‘*’’, Flow::FLOW_INOUT),
      &SimpleAVModule::processVisual
    );

    MarSystemAudio::build();
    MarSystemVisual::build();
  }

  bool processAudio(mrs_audioframe &frame)
  {
    //... do something with frame ...
    return true;
  }

  bool processVisual(mrs_image &img)
  {
    //... do something with img ...
    return true;
  }
};
```

If for instance we need to update the module to generate a visual representation of the audio frame, only two changes are needed. First, add the new visual flow by changing the first `configureFlows` call:

```
configureFlows(
  new Flow(FlowHandlerAudio::type(), ‘‘*’’, Flow::FLOW_INOUT),
  new Flow(FlowHandlerVisual::type(), ‘‘*’’, Flow::FLOW_OUT),
  &SimpleAVModule::processAudio
);
```

and update the processAudio method accordingly:

```
bool processAudio(mrs_audioframe &frame, mrs_image &out_img)
{
  //... do something with frame and out_img ...
  return true;
}
```

For more complex flow handling a lower-level access to the underlying payloads and factories is also possible. However this is rarely necessary for typical data handling.

6.3.9 Scripting with Python

A very important motivation of building and using Marsyas is to have a way to rapidly assemble a prototype and batch test it with different parameters, test sets, network configurations, etc. One way of doing it is to have an interface of Marsyas to a scripting language, like Python. Other scripting languages such as Ruby, Perl or Lua can also be adopted, based on the same rationale. Marsyas has been designed to expose a lot of functionalities at run-time including module creation, network assembly, changing of controls, and starting/stopping processing. Essentially the only part that requires compilation is the addition of new processing modules or changing the processing code of existing modules. This was a conscious decision as it is essential for fast performance. All the run-time functionalities of Marsyas can be interfaced to a scripting language.

Python is an interpreted, interactive, object-oriented programming language that has a increasingly large supporting community. Its clean syntax, small number of powerful high-level data types, easy integration with C/C++ and large set of supporting tools are decisive features. Using the Python C API it is possible to create extensions to Python. The extensions are implemented in dynamic-load libraries whose functionalities are then accessed by Python. Having created an extension containing Marsyas, Python can be used to create batch tests or simple applications faster than with C/C++. The first obvious reason is that if some part of the code needs to be changed to do, for example a different test, a compilation that usually is time consuming, is not needed. And, since it consists only of a thin wrapper over Marsyas, the overhead is minimal and the execution time should be very close to an equivalent C/C++ implementation.

Let us consider a simple example. To create an application that estimates the motion of a video, calculates its centroid and displays both the video and the estimated motion, we would first import the marsyas module, create a manager – which is responsible for the creation of new modules – and load the needed extensions which in this case would only be visual-ext and av-ext – containing visual processing algorithms, such as the MotionDetector and audiovisual utility modules like FFMPEGFileSource, respectively.

```
import marsyas
manager=marsyas.MarSystemManager()
manager.loadExtension('marsyas-visual-ext')
manager.loadExtension('marsyas-av-ext')
```

Next, the network needs to be created. This is accomplished using the manager by passing as arguments a type and a name for the new module. The network will only consist of a file source, a motion detector that estimates the motion and a visual sink to display both the original video and the motion vectors and centroid. Some of the details of how data flows between modules will become clear along Section 6.3.

```
series=manager.create('Series', 'network')
series.addMarSystem(manager.create('VisualSource', 'src'))
series.addMarSystem(manager.create('MotionDetector', 'motion'))
```

```
series.addMarSystem(manager.create('VisualSink', 'snk'))
```

Initialisation of the modules is performed through the controls, using the `updctrl` method. To simplify the use of paths we can beforehand create proxy controls by linking them to internal controls. Finally the bootstrap is called to completely initialise the network before starting the processing.

```
series.linkctrl('mrs_string/filename',
  'VisualSource/src/mrs_string/filename')
series.linkctrl('mrs_bool/notEmpty',
  'VisualSource/src/mrs_bool/notEmpty')
series.updctrl('mrs_string/filename', 'test.avi')
series.updctrl(
  'MotionDetector/motion/mrs_natural/blockSize', 4)
series.updctrl(
  'MotionDetector/motion/mrs_bool/drawVectors', True)
series.bootstrap()
```

Finally, we use the tick method to process each chunk of data, which in this case corresponds to the video frames – in each tick called a frame will be processed and nothing else.

```
notEmpty=series.getctrl('mrs_bool/notEmpty')
while notEmpty.isTrue():
  series.tick()
```

Note that most of the code is rather generic, i.e., the C/C++ would be very similar to this, differing only in the syntax, however with the added flexibility of scripting and easy prototyping without recompilation. The example applications described in Section 6.4 were implemented using the Python interfaces.

6.4 Example applications

In this section we present some examples of applications that can be implemented in MarsyasX. The focus is on how one can do it rather than on the algorithmic details. The first can actually be considered simplistic but are meant to be so in order to simplify comprehension. Some additional information as well as these and other examples are presented online [4] with code available for download. Also, the MarsyasX library is available in a SourceForge SVN repository [5].

6.4.1 Simple motion detection

The first example is a simple network that can be used for motion detection (as represented in Figure 6.9). In Section 6.3.9 we discussed the Python code to create this network. The network comprises simply three modules – one source, one processing

module, and one sink. The processing module estimates motion with an optical flow estimation algorithm, for instance the Lucas-Kanade method (Lucas and Kanade, 1981). Optical flow vectors are outputted as a Multidata data type and, since the *drawVectors* control is set to true, the vectors are also visually represented and overlaid on the input image. Finally, the resulting image with the vectors is shown by the VisualSink.

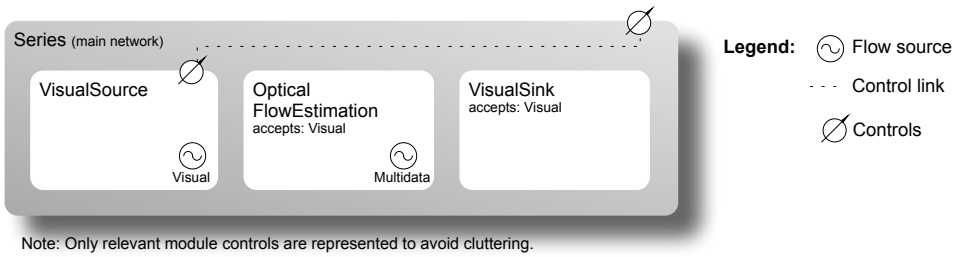


Figure 6.9: *Network used for motion detection.*

6.4.2 Segmentation of audio streams

This application example takes an audio stream and automatically segments it in two classes: music and speech (as in Section 3.4). The network configuration employed for this purpose is shown in Figure 6.10.

We start by extracting some discriminative features from the audio signal, as depicted in Figure 6.10. A *Fanout* composite module is used for concatenating features such as *Zero Crossings* and *MFCC* coefficients into a single feature vector. The figure also depicts the use of pre-defined prototype for the calculation of spectral features (in this case, *Spectral Centroid*, *Spectral Rolloff* and *Spectral Flux*), without the need to redefine its use each time they need to be calculated. The resulting feature vector is then fed into a *Classifer* module, which implements a simple multi-dimensional Gaussian model. In a training setup, the feature vectors come from labelled audio files (i.e., either music or speech signals) and are used to train the Gaussian model for the corresponding class. Afterwards, it is possible to configure the same network (using a control available at the *Classifer* module for setting training/classification mode) to a classification setup. In this mode, feature vectors extracted from an unknown audio stream are sent to the classifier, which outputs the most likely class according to its internal pre-trained models.

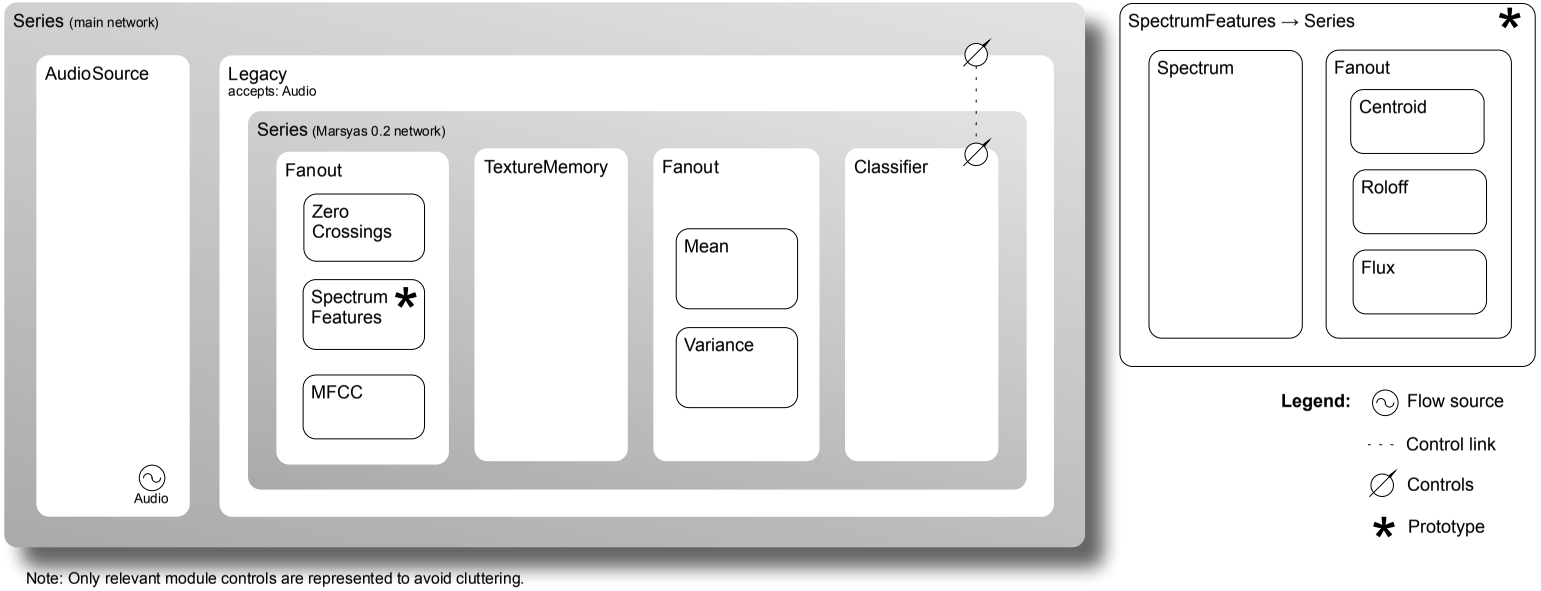


Figure 6.10: *Network used for the music/speech segmentation scenario.*

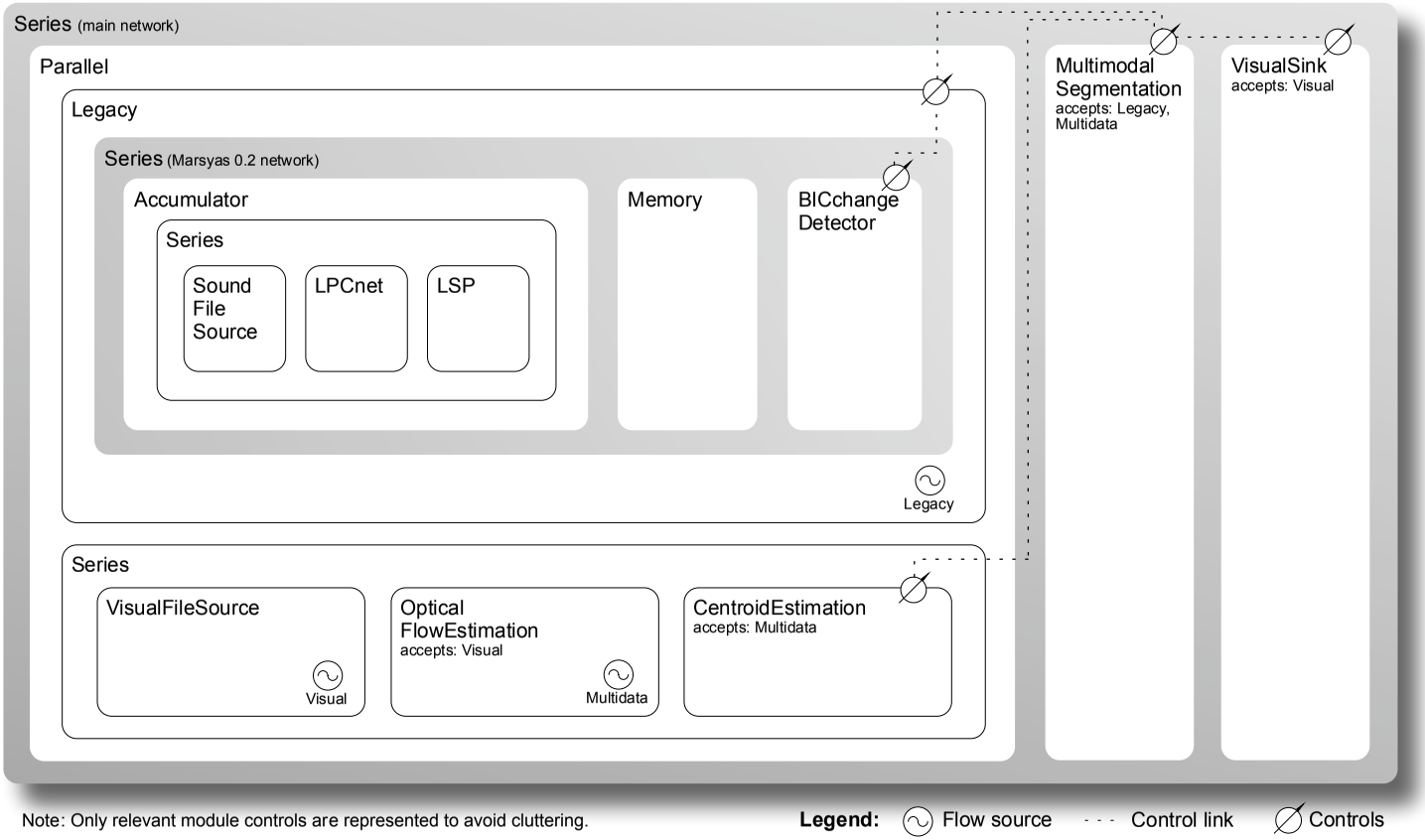


Figure 6.11: *Network used for the speaker segmentation scenario.*

6.4.3 Speaker segmentation using audio and video

The algorithm used in this example for the speaker segmentation is based on the the one presented in Section 2.3.4. The algorithm and the corresponding network can be broadly separated in three parts: the audio speaker segmentation algorithm, the visual motion estimation and centroid calculation, and finally a multimodal speaker segmentation module that combines the visual and audio results. The processing network is depicted in Figure 6.11.

The audio algorithm used for the speaker segmentation assumes no prior knowledge about the number of speakers or their identities and presumes that the audio input contains only speech. The method follows a metric-based approach for coarse speaker segmentation using Line Spectral Pairs (LSP), which is subsequently validated by means of the Bayesian Information Criterion (BIC).

The visual algorithm part of the network considers scenarios with only two speakers facing the camera, such as interviews or lectures. It is assumed that the speaker will be located in the region containing the most amount of motion. The separation of these regions is defined by a boundary that for simplicity is kept as a vertical straight line splitting the image in two halves. A centroid of the motion is calculated and is used to detect the potential speaker.

The multimodal speaker segmentation algorithm takes into account two constraints: people tend to move their bodies, arms and lips before producing any sounds and the first sounds produced are usually non speech vocalisations such as breath, etc, hence the visual change detector is then more likely to be fired before the audio one; also, this last audio detector is more likely to detect the correct boundary but with a higher false alarm rate due to the presence of non speech sounds and background noise. When a speaker segmentation is detected by the multimodal module it is signalled in a control. This control is linked to the VisualSink that will display which speaker is speaking.

This work implements a late fusion scheme where one classifier is attached to each modality and the decisions of the classifiers are finally combined. Future work will concentrate on early fusion, namely the use of only one classifier that considers all the modalities at once, which is usually considered more reliable but harder to implement. MarsyasX can be powerful in such scenarios, since audio and video data can be conveniently aligned and combined within the same network of data.

6.4.4 Simple surveillance with multiple inputs

This application includes inputs from multiple cameras and microphones capturing different areas. The goal is to select the input with the most relevant activity. For this simple example we will consider two features for the activity detection: one audio feature, the short-time energy, and a video feature, the global motion. A simple weighted average of the normalised feature values are used as a measure of activity.

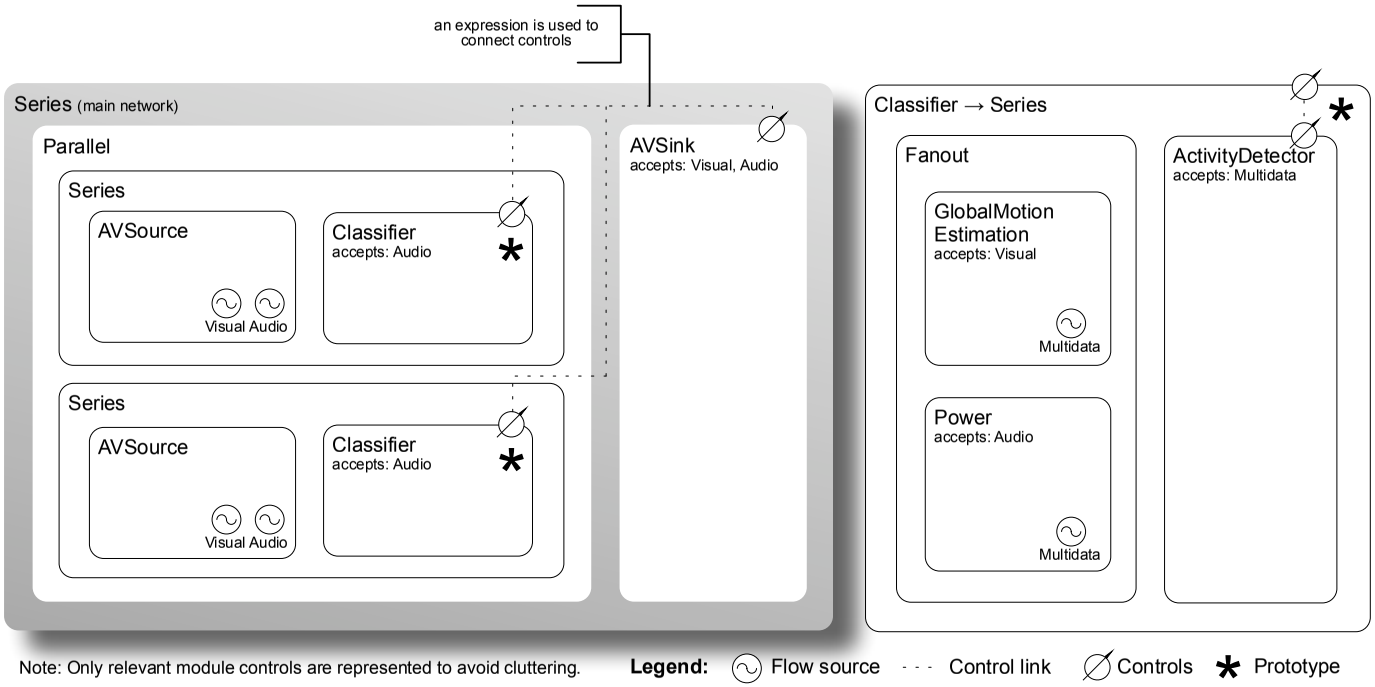


Figure 6.12: *Network used for the surveillance scenario.*

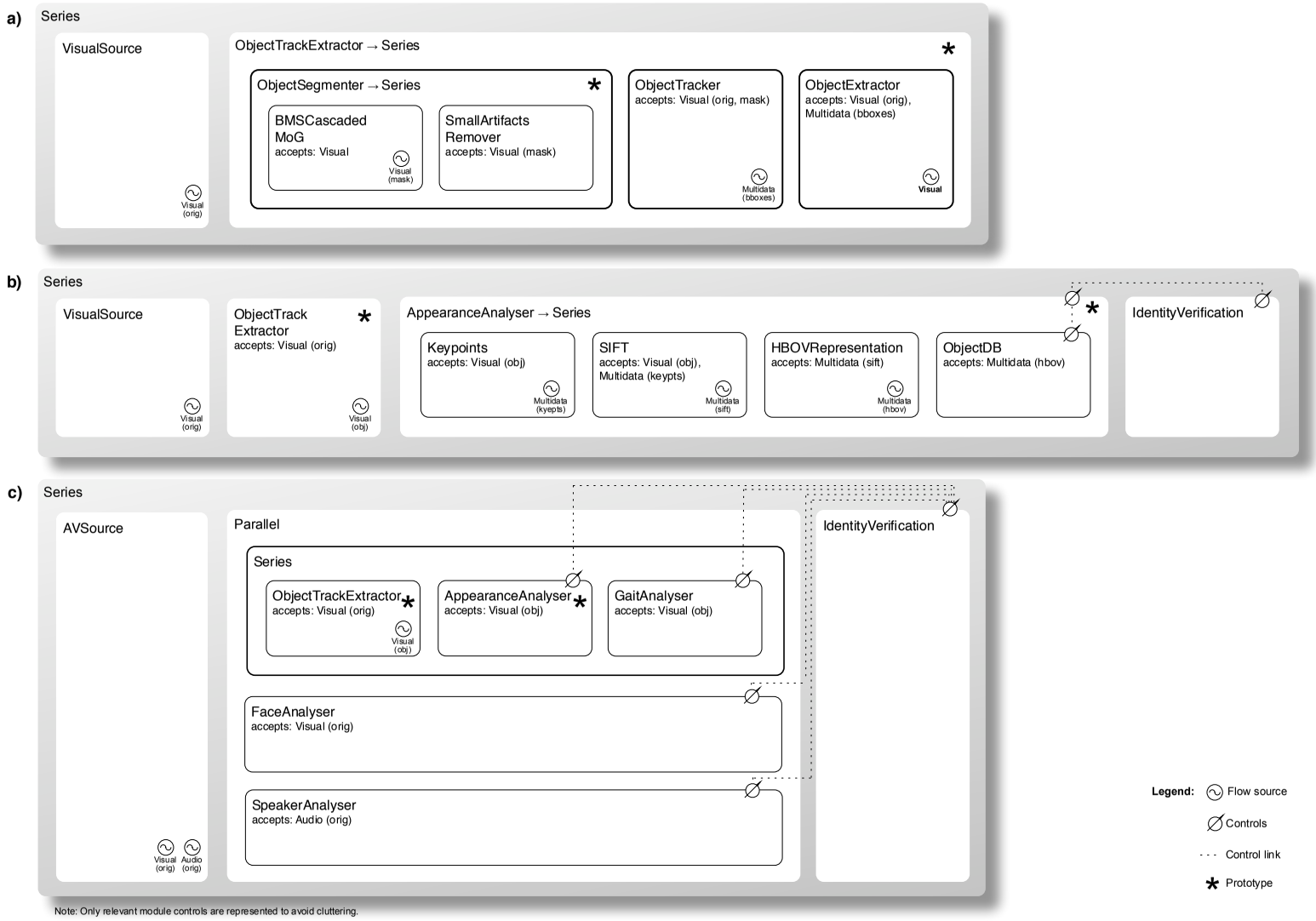


Figure 6.13: *Network used for the person detection system.*

Figure 6.12 shows the network that is used for this scenario. To read the input videos we use the AVSource module that produces two flows: Audio and Visual. This means that, when a video frame is captured a new payload containing an image is sent to the output channel and, when an audio frame is captured another payload is outputted containing now a vector of audio samples. Note that despite the two depicted inputs, more are possible. For the two sources case we will have a total of four flows: $(Visual, Area1)$, $(Audio, Area1)$, $(Visual, Area2)$ and $(Audio, Area2)$. The next module is a Classifier which is represented by a prototype. A prototype is another module, possibly composite, that is registered in the module manager with a given name. This classifier is in fact a Series containing a Fanout and the Activity Detector. The Fanout is a composite that sends the input to each module inside, like a Parallel but, unlike the latter, aggregates the flows at the output. In this case, both modules inside the Fanout will produce a new flow: Features. The output of this Fanout will then be a feature matrix containing the two features. In the flow aggregation, the payload timing information is used to correctly align the features in time, since the visual and audio sources have different output rates. Finally the Activity Detector will take both features to calculate the activity measure and store it in a control which is linked to the Classifier module. A control is used to signal the AVSink which source (i.e., audio and visual flows) to display. Another link is used between each Classifier and the AVSink using an expression. The expression, in Python, could be:

```
if in1 >= in2: out = "Area1"
else: out="Area2"
```

with `in1` and `in2` representing the activity measures of each source, and out representing the selected source. Whenever the control that selects the input is changed the AVSink starts displaying it.

6.4.5 Person detection system

The last example of application implements a person detection system based on the object matching algorithm described in more detail in Section 5.5. Only visual processing modules are used. The algorithm consists of five steps: (1) segment and track each relevant visual object, (2) extract a representation for each tracked object, (3) compare this representation with a database of objects, (4) if a given object is known, label it accordingly, and (5) update the database with the new information, if it is found relevant. In MarsyasX each of these steps corresponds to one or more modules performing a specific task.

The main modules for this application include: background modelling and subtraction for object segmentation, object tracking, extraction of local descriptors and vocabulary-based representation, and object classification. The first two modules are incorporated in the network shown in Figure 6.13a and the other two modules are part of the network shown in Figure 6.13b. The former network outputs the multiple object tracks

detected by the combined `ObjectSegmenter` (described in Section 3.3) and `ObjectTracker` (described in Section 5.5.3). This network is combined with the `AppearanceAnalyser` module (described in Section 4.3) to identify persons. The object identity of each frame extracted is outputted by a control which in turn is linked with an `IdentityVerification` module. The system can be further improved by integrating other modalities, like face, gait, and speech information (all discussed in Section 4.4). The modularity of MarsyasX simplifies this process and only straightforward changes are needed (Figure 6.13c).

6.5 Summary

Creating applications that rely on audio or visual processing can often be a cumbersome task. There is a wide offer of specific tools and libraries both commercial and free (open source or not) but if one wants to combine some of these problems arise. The first problem is that data structure and semantics are almost always different and the user ends up creating custom wrappers or sometimes re-implementing functionalities. This is even more evident with a combination of audio and visual processing libraries. In fact, cross-modal processing is an important and growing field of research among the scientific community. Having the ability to, under the same framework, use or develop new tools and algorithms is undoubtedly important. By abstracting both data structures as well as its flow, and by using uniform procedures to define and set parameters a considerable effort of integration can be removed from the user. We have described in this chapter MarsyasX, a broad framework that attempts to solve these problems.

MarsyasX is based in Marsyas 0.2, an established framework for music and audio signals analysis. With Marsyas, processing networks are assembled using configurable pre-implemented modules by means of implicit patching, differing from most commonly used frameworks. MarsyasX extends the functionalities of Marsyas 2 to visual support alongside audio. It is however not limited to audio and visual processing but can in fact be seamlessly used for generic data processing. Data is exchanged between modules using timed payloads, which in turn are implicitly grouped in flows. A flow "carries" data of the same type along the network always following its topology. The topology is defined implicitly by the use of composite modules that aggregate other modules (simple or composite) in an arbitrarily complex configuration. Typical composite modules include Series, Parallel, Fanin, Fanout, etc. The payload timing information consists of two timestamps – time of creation (TOC) and time-to-schedule (TTS) – which are used to synchronise different flows. This flow of data across the network can be classified as synchronous as opposed to the asynchronous behaviour of control data flow. Controls interfaces to the modules and are used to read or set internal values of the modules and can be accessed at all times. However, any change that will imply the modification of the flow will be propagated synchronously by the data flow. These and any other changes are at the same time asynchronously notified by events.

Event recipients can be defined or registered by other modules or by higher layers, like a GUI.

Some example applications showed how simple or more complex algorithms can easily be implemented in MarsyasX. As the framework evolves more modules are expected to be included allowing the creation of much more complex networks. It is also envisioned that multiple processing networks will be able to work on top of distributed systems, transparently exchanging payloads between them.

7 Conclusions

Building a complete solution to automatically describe real-world scenes is a complex undertaking. Such solution comprises different methods performing various associated tasks. Each of these tasks represent per se a research challenge. Also, often real-world scenes encompass multiple modalities that convey information in different forms. Typically modalities are processed independently and the resulting output is then aggregated. However, in many cases the information conveyed by modalities have interactions that can be explored to better extract their intrinsic information. The analysis of each modality as well as the way individual modalities should be combined, represents a multidimensional challenge.

7.1 Discussion

Systems that can automatically detect and identify events in complex scenes and track objects across multiple cameras in real-time still present technological challenges that need to be tackled. Currently, systems that are employed for these tasks (e.g. surveillance systems) are almost all operated manually. Tedious and error-prone inspection of long hours of data are often the only possible method to know when an event occurred. Typically these systems rely only on data acquired by cameras capturing low quality images at a small frame rate, which makes the posterior analysis even more arduous. The main reason behind this option is the difficulty of storing the huge amounts of data being captured, if no selection of data is performed. Some recent systems only store segments that have some activity, based on the motion estimated by the system. Although this significantly reduces the amount of information, it hardly provides the necessary means to efficiently search for particular events.

Most of the information in scenarios like surveillance is conveyed by the sequence captured images but, more often than not, there is important information that can be obtained from analysing other types of data, or modalities. In fact, the concept of multimodal scene analysis relies on that premiss, i.e., if adequate methods are applied, additional information can be extracted from a given scene when using a combination of different modalities. Some challenges need to be overcome in this process, namely, to know which modalities are really relevant and how can the multiple modalities be combined.

The description of real-world scenes can be approached considering two semantic entities: *objects* and *events*. Different methods are employed to analyse both – methods for object analysis and methods for event analysis. Object analysis can further be subdivided in object detection and object tracking. From object detection in a complex

scene results a separation in its composing objects. Each object can then be separately analysed, identified, or classified. However, detecting or segmenting objects is a very complex problem since, to attain results comparable to human-performed segmentations, semantic or high-level a priori knowledge is very often required. Object recognition on the other hand involves comparing an unknown sample with a model. A related task is object tracking that relies on both detection and recognition to follow an object along time. Finally, with event analysis, activities and events that are considered relevant are identified. Similarly to object analysis, detection and recognition tasks are also performed but, unlike in object analysis, these tasks usually are intrinsically related in event analysis. Also, while object analysis is essentially related to the content, event analysis has a strong context-dependent component. This means that, to understand what is a relevant event, we must know the context of the captured scene.

With this thesis we presented some original contributions that are related to object detection, object recognition, and event analysis. Regarding the former area, we presented a visual object segmentation algorithm. In scene description object segmentation is an important first step, which means that it should represent a very small fraction of processing time while performing as effectively as possible. The proposed algorithm uses a cascade of change detection tests, including noise-induced changes, illumination variation and structural changes. For the detection of structural changes the algorithm is based on other commonly used per-pixel modelling algorithms. When compared to these, our algorithm achieves faster processing and results show that even without additional post-processing segmentation quality is better.

Object recognition usually follows object detection. In this scope we introduced an object matching system for multiple cameras. The goal is to match tracked objects based on their appearance across independent views. The objects are first tracked by commonly used single-view tracking algorithms and these individual results are the inputs of the matching method. This methodology is suitable for implementing an "object handover system" between cameras with non overlapping views. Encouraging results were obtained using static models for a set of known objects. However typical scenes may present changes to the objects' appearance; also, new object models may also need to be included. The matching method should therefore be scalable in two distinct dimensions: an undetermined number of new objects can be added at any given instant and, existing object representations can be updated to reflect changes in time. For this purpose we proposed and tested an incremental model which also presented a good recognition performance.

This object matching system was later extended to an event navigation system based on timelines representing person appearances. From an application point of view, the complete system can be seen as a summarisation application that finds multiple object occurrences. These occurrences can be captured by the same camera or by any other cameras, located in completely independent positions. The system takes as inputs raw sequences, segments them into relevant objects, tracks the detected objects

and processes the tracks to find multiple person appearances. This in essence acts as a linking mechanism between sequences. The result is a timeline that allows more efficient browsing through the scene.

Developing these types of algorithms and related systems is a problem in itself. Typical visual and audio analysis algorithms are created and tested using generic mathematical tools like Matlab. However, faster implementations need to be carried out in lower-level languages and frameworks (typically C/C++). It is possible to find numerous libraries that help with this task but if the scope of the task involves integrating different types of data, this process can easily become burdensome and inefficient. Another contribution of the work developed in this thesis is a multimodal framework called MarsyasX that could process many types of data following a common scheme. The framework follows a dataflow architecture where complex network of processing objects can be assembled to form systems that can handle multiple and different types of multimedia flows with expressiveness and efficiency.

7.2 Current work

The ultimate goal when outlining this work was to have an efficient way of browsing the history of a wide-area multi-camera system capturing real world scenes – with surveillance systems representing the prominent instance of such systems. We envisioned a multimodal system that could separate and represent relevant objects captured along time, such as persons. Moreover, the objects' representation should allow a search by content. In this case, the search query would be the object itself and the search target would be the collection of other appearances of the same object. Recalling Figure 1.3 in Chapter 1, we identified then the different areas related to multimodal scene analysis. We also defined a coherent line of work encompassing some of the identified areas and essentially centred on visual data analysis. That path was pursued during the development of the work that is reported in this thesis and in a number of peer-reviewed publications as well (refer to Section 1.4.5).

The current tangible result is a system that integrates all the methods devised until now and implemented in the multimodal framework and generates browsable timelines summarising captured scenes. Multiple appearances of persons captured by the same or by different cameras are identified in the process, allowing also to know the path followed by a given person.

7.3 Future work

Multiple future directions can be envisioned for this work. We can broadly divide them in global and specific lines of work. The former are related to the system perspective of

this work, while the latter include specific improvements to the algorithms developed in the scope of the thesis.

Regarding the global lines of work, the ultimate goal is the development of a complete system that gathers data from different types of sources – mainly visual and audio sources, but other inputs from other sensors should also be incorporated. As we mentioned in the previous section, most of the work until now has been on the development of visual object analysis algorithms. A near-term goal is to incorporate audio processing algorithms, especially for two purposes: improve person identification and improve event detection. Audio sensors are often not used in typical surveillance scenarios but, when available, the information extracted from the additional audio sources may be valuable to improve detection and recognition performance.

In Figure 1.4 we defined three layers in which development would occur – framework layer, analysis layer, and application layer. Until now the two lower layers (framework and analysis) received the most attention. A mid-term goal is to develop an application that manages the system sensors, the analysis algorithms and the output these algorithms generate. These outputs should typically be in the form of information about the analysed content, usually defined as metadata. The way metadata is visualised depends greatly on the type of application. In this case we identify two forms of metadata visualisation – temporal-based and spatial-based. The temporal-based metadata consists of identified events that can be presented as a timeline (or timelines) such as the ones created for the specific case of person (described in Section 5.5). The timeline representation provides a good way of showing when objects are detected and synchronisation relationships between events. Layout is specified per object per time unit, so an overview of all objects at a given time is possible. More information about metadata representation for visual object editing can be found in (Teixeira, 2004). Regarding spatial-based metadata, each captured image is enriched by the processed metadata, such as identification about persons and other objects, localisation of audio sources, etc. If additional information about the covered area is available, a map representation of the area with overlaid information can also be displayed.

Regarding specific lines of work, it is possible to identify various possible paths. Specifically, further improvements should be done on the object segmentation algorithm to effectively handle challenging sequences with difficult illumination changes. Using hybrid solutions is a possible way, i.e., combining region segmentation and global measurements with per-pixel modelling. Also, the use of additional information like depth can contribute to better segmentation results. Finally, we expect to include layered modelling that can "memorise" current and past objects. The layered modelling would allow to significantly improve performance in scenarios where objects with a moving-static-moving behaviour are present.

The work on the object track matching method will be further developed. In particular, the information gathered by object matching can be used as feedback to the tracking algorithm in order to improve tracking performance. This was already discussed in

Section 5.5.7 but a more thorough analysis still needs to be conducted. Also, better incremental models need to be devised in order to achieve higher scalability.

Finally, the proposed multimodal framework MarsyasX is an ongoing work that is expected to have a significant impact when it becomes the engine of the Marsyas project. Until now it has been an experimental branch of Marsyas but there is a consensus among developers that MarsyasX will play an important role in Marsyas' future. Among the long list of goals, the most important considering a near-term time window include: developing a more stable and tested core; devising a complete event management system; adding support for XML-based flows (e.g. MPEG-7, MPEG-4 XMT, X3D, etc), MIDI-based flows, and OSC-based flows; and, more ambitiously, support distributed processing networks.

7.4 Final thoughts

An endeavour as challenging as this and covering as many topics could never be concluded here. The initial collaborations described in Sections 3.4 and 5.6 are examples of that. Additional theses are also expected to be spawned from this, such as the work that currently explores the possibility of improving person tracking taking as input the results of object segmentation and using as feedback the results of object matching (Carvalho, 2009). We believe that this work lays the foundation for other efforts.

Appendices

A Datasets

A.1 Segmentation dataset $\mathcal{D}_s$

The $\mathcal{D}_s$ dataset used to evaluate the proposed method performance consists of 12 sequences, shown in Figure A.1, which were manually segmented in selected frames, shown in Figure A.2. The test set is available at the BGMS page [3]. The first sequence, called shopping (SH) shows a view of a shopping corridor and is one of the test case scenarios made publicly available by the EC Funded CAVIAR project/IST 2001 37540 (Fisher *et al.*, 2002). The scene consists of people walking, browsing the stores' displays or waiting for others. It has stable illumination conditions, except for a small portion in the right side of the field of view. However, hard shadows and reflections in the floor and in the display's glass are present. The second sequence, labelled outdoor (OD) shows an outdoor scene with several people passing along the camera field of view and is available in the MPEG-7 test set (results are presented for stream A). The sequence has some noise and although the illumination conditions are fairly stable, the background presents significant vegetation swing. The speedway (SW) sequence was captured from a bridge over a speedway and is also available in the MPEG-7 test set. It shows different sorts of vehicles moving in both directions. Overall, it is the most stable stream regarding background changes but some relevant shadows are present. The following frames were manually segmented by visual inspection: 350, 355,... and 400 in the SH sequence; 880, 885,... and 930 in the OD sequence; and 2510, 2515,... and 2560 in the SW sequence. Besides these three sequences, the test set also includes nine openly available segmented sequences, namely: meeting room with moving curtain (MR), campus with wavering tree branches (CAM), lobby in an office building with switching on/off lights (LB), shopping centre (SC), hall of an airport (AP), restaurant (BR), subway station (SS), water surface (WS), and fountain (FT). All sequences from this data set present challenging scenes with considerable background activity – mainly in MR,

CAM, WS and FT – and sudden changes of illumination conditions are also present – with special difficulty in LB. More details about these sequences can be found in (Li *et al.*, 2004a).

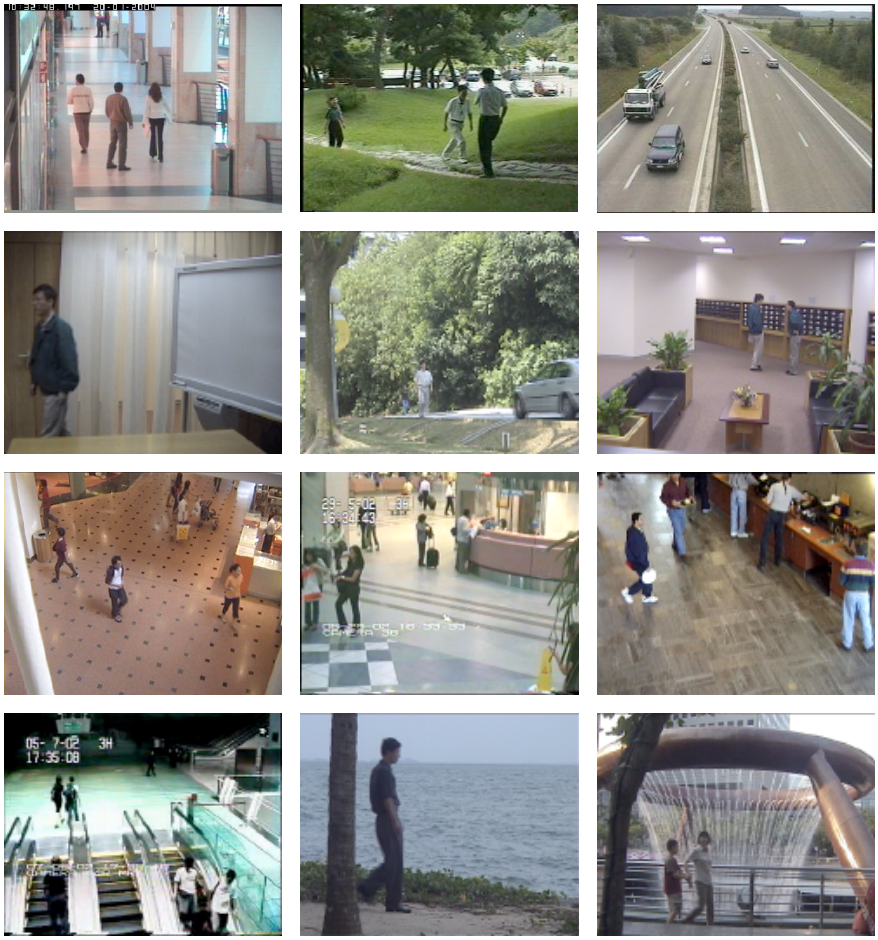


Figure A.1: *Segmentation dataset (original frames). Segmentation dataset (original frames). From left to right, and from top to bottom, shopping (SH), outdoor (OD), speedway (SW), moving curtain (MR), campus with wavering tree branches (CAM), lobby in an office building with switching on/off lights (LB), shopping centre (SC), hall of an airport (AP), restaurant (BR), subway station (SS), water surface (WS), and fountain (FT).*

A.2 4-class audio dataset $\mathcal{D}_4^a$

The audio dataset $\mathcal{D}_4^a$ is used to test the segmentation of audio streams in 4 possible classes: silence, noise, speech, and music. The files comprising the dataset were col-

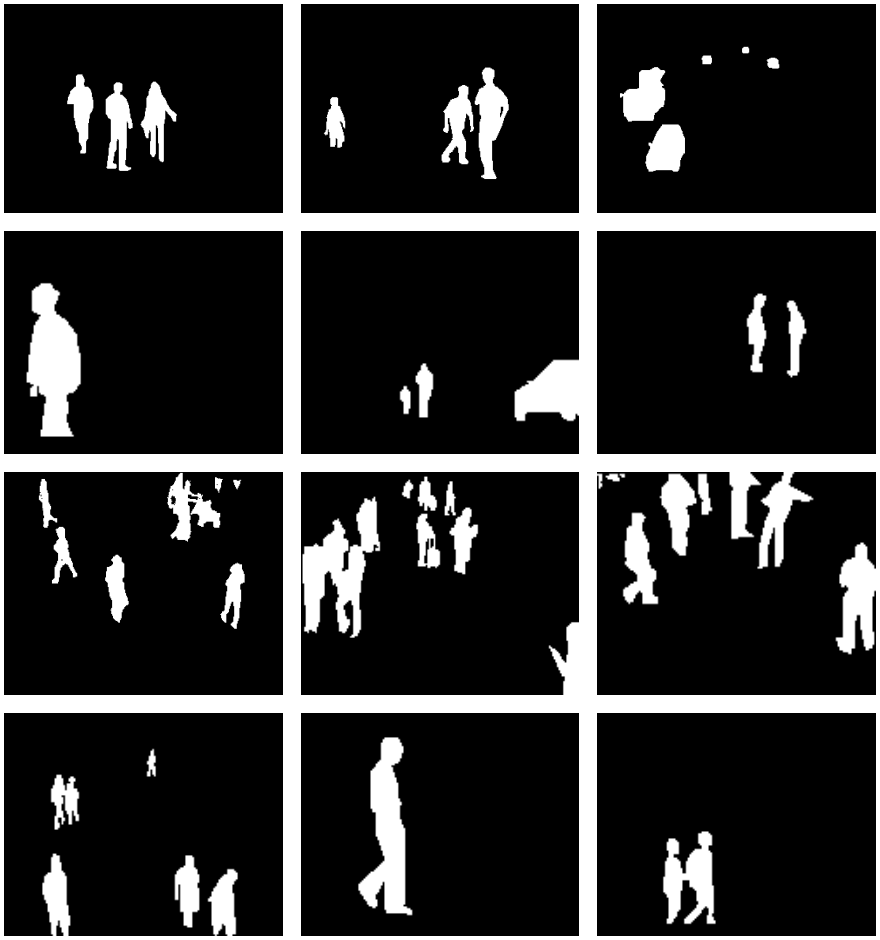


Figure A.2: *Segmentation dataset (masks). Segmentation dataset (masks). From left to right, and from top to bottom, shopping (SH), outdoor (OD), speedway (SW), moving curtain (MR), campus with wavering tree branches (CAM), lobby in an office building with switching on/off lights (LB), shopping centre (SC), hall of an airport (AP), restaurant (BR), subway station (SS), water surface (WS), and fountain (FT).*

lected from audio CDs, television videos and web radio streams. It represents a wide spectrum of real-world sequences, subject in some cases to adverse capturing conditions. In particular, background noise is present, as are linear and non-linear distortions. Also, some recordings originally had different sampling rates, but for this dataset all were resampled to 44100Hz with one channel and 16-bit resolution.

In total, approximately one hour of audio for each of the 4 classes is stored in 342 10-second files.

A.3 Object matching dataset $\mathcal{D}_{30}$

The dataset $\mathcal{D}_{30}$ consists of individual sequences containing 30 visual objects extracted from the Shopping Center dataset of the CAVIAR project [44]. We extracted each visual object from the 26 sequences that comprise the original set, using the provided CVML-based ground-truth information. The ground-truth consists of a bounding box defined in each frame, for each object.

Figure A.3: *Images of all 30 visual objects used in the dataset. First row shows* Person[01-10], *second row shows* Person[11-20] *and third row shows* Person[21-30]. *The dataset consists of a total of 14506 images, with an average of 468.5 images per person.*

The total number of images used to train and test the complete system is 14506. This number includes 30 individual tracks of 30 different persons. Fig. A.3 shows all visual objects composing the dataset. Individual tracks were extracted from the original video according to the bounding box defined in the ground-truth. These tracks in-

clude images with partial occlusions as well as severe occlusions by other persons (see Fig. 4.6 for an example). Additionally, we tested different tracks of the same person, captured at different instants and areas (see Fig. A.4 and Fig. A.4 for examples); they were however not used for training.

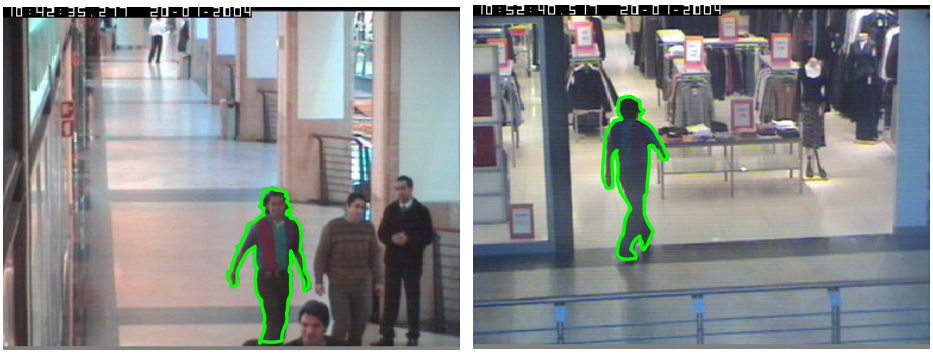
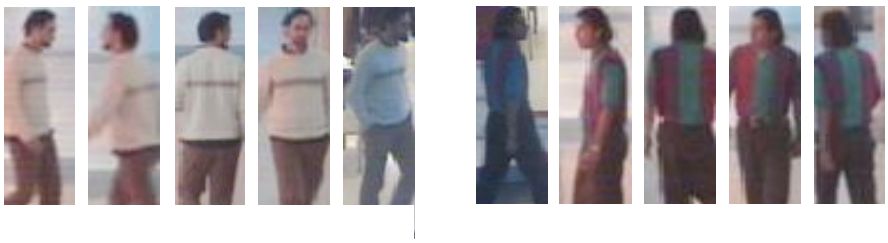


Figure A.4: *Same visual object captured at different instants by different cameras.*



(a) Person01 (b) Person12

Figure A.5: *Different perspectives of the same person are present in the dataset. Two examples of this diversity in appearance is shown in the figure.*

A generic vocabulary was created with the PASCAL Visual Object Classes (VOC) database [45]. All objects tagged as PASperson and all its variants (Sitting, Standing and Walking) were extracted from the VOC 2006 contest (Everingham *et al.*, 2006) dataset; these images formed the generic dataset $\mathcal{D}_g$. The total number of images for this vocabulary training set is 2688.

The dataset $\mathcal{D}_{30}$ was further partitioned into subsets $\mathcal{D}^i, i \in \{1, 2, \ldots, 5\}$ to test the learning adaptability of the system. Each subset represents a time interval of 5 seconds (150 frames at 30 fps) of accumulated data. In Fig. A.6 a representation of how the subsets are created is depicted. Note that no frames are repeated between sets and that all frames from $\mathcal{D}_{30}$ are included in the subsets, such that $\bigcup_{i=1}^{10} \mathcal{D}^i = \mathcal{D}_{30}$ and

$\bigcap_{i=1}^{10} \mathcal{D}^i = \emptyset$. This incremental dataset simulates a typical visual surveillance scenario where objects are detected at different time instants. Initially only some objects are represented but, as new subsets are added new objects are added. Likewise, objects previously detected can no longer be represented in future subsets.

Figure A.6: *Subsets $\mathcal{D}^i, i \in \{1, 2, \ldots, 5\}$ created to test incremental learning of objects. Each object starts to be captured in a given instant and leaves the field of view at another instant – the time interval when objects are is represented by the bars. Each dataset corresponds to 5 seconds of activity.*

A.4 Independent views dataset $\mathcal{D}_5$

To complement the results with the CAVIAR dataset we captured 3 additionally sequences at two independent locations (Figure A.7). These sequences were named Hall1, Lobby1, and Lobby2 and present challenging situations with cluttered scenes, high rates of occlusion, different illumination conditions as well as different scales of the persons being captured. Similarly to $\mathcal{D}_30$, a set of track images from 5 persons was extracted from the sequences, forming in this case the $\mathcal{D}_5$.

A.5 Event detection dataset $\mathcal{D}_e$

The event dataset uses also CAVIAR dataset [44]. The scene consists of the surveillance sequences capturing the area in front of a store. During the videos people enter and leave the store from several directions, others just pass by the store. Figure A.8 is an image of the area under the camera field of view.

The full dataset contains 26 sequences with a total of 235 ground-truth tracks. From the set of existing tracks we selected those which are not interrupted in the sequence. These are the tracks which begin and end on the limits of the video frame. The final dataset consists of 124 trajectories, each annotated which one of the 5 characteristic

(a) `Hall` scene (b) `Lobby` scene

Figure A.7: *Two additional independent locations. Distinct conditions of capture are visible, namely different scale and illumination.*

Figure A.8: *Image of the area in front of a store where the sequences were recorded.*

trajectories defined in Section 5.6. In Figure A.9 we show the full 124 trajectories, divided into the specific characteristic trajectories.

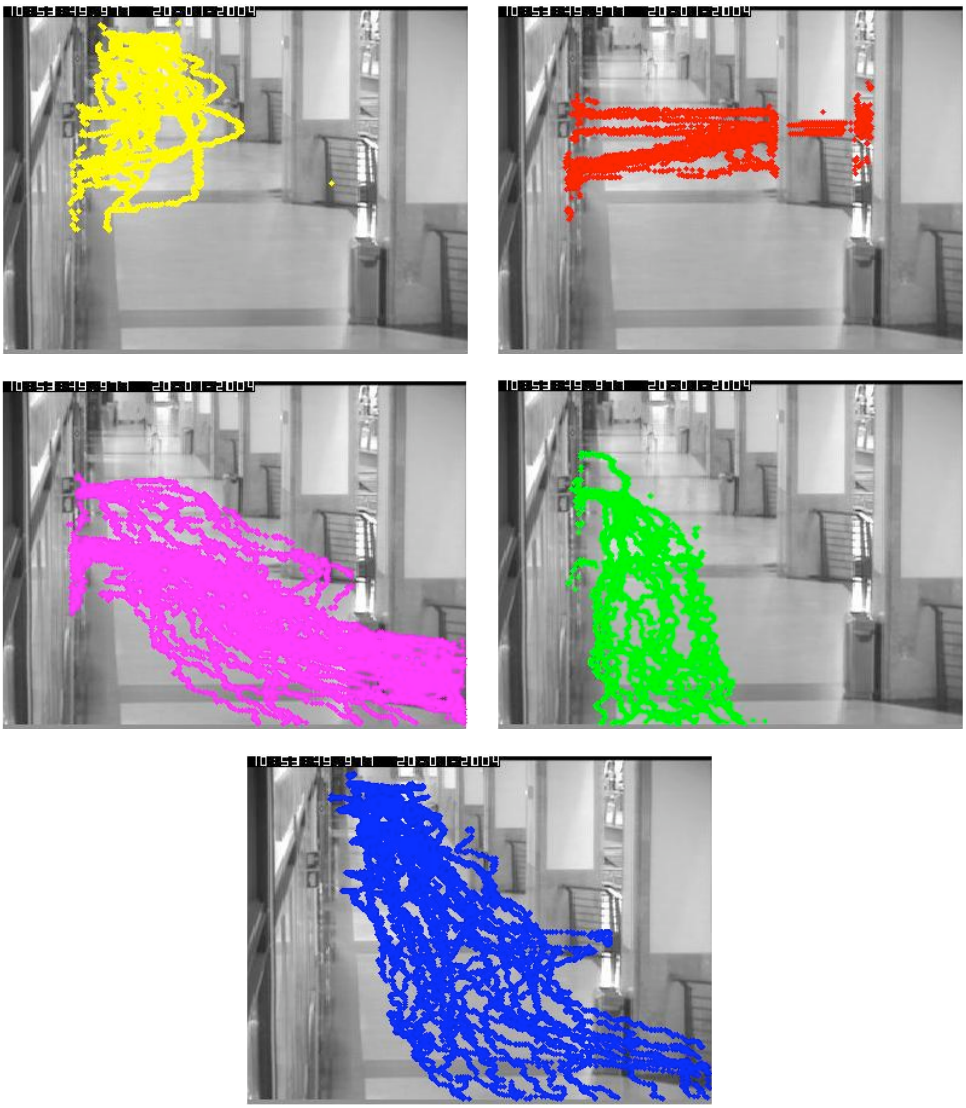


Figure A.9: *Aglomerated plot of all tracks in the database. The tracks are divided into 5 diferent directions (124 tracks in total).*

B Evaluation metrics

B.1 Performance metrics

B.1.1 Precision and recall

$$precision = \frac{N_{tp}}{N_{tp} + N_{fp}} \tag{B.1}$$

$$recall = \frac{N_{tp}}{N_{tp} + N_{fn}} \tag{B.2}$$

where N_{tp} is the number of true positives, N_{fp} is the number of false positives and N_{fn} is the number of false negatives.

B.1.2 False Acceptance Rate (FAR) and Miss Detection Rate (MDR)

$$FAR = \frac{N_{fp}}{N_{tp} + N_{fn} + N_{fp}} \tag{B.3}$$

$$MDR = \frac{N_{fn}}{N_{tp} + N_{fn}} \tag{B.4}$$

The relation between the pairs (FAR, MDR) and ($precision$, $recall$) is given by:

$$FAR = \frac{recall \cdot N_{fp}}{precision \cdot N_{tp} + recall \cdot N_{fp}} \tag{B.5}$$

$$MDR = 1 - recall \tag{B.6}$$

B.1.3 F-measure

A measure that combines both precision and recall is called F-measure. The most commonly used, F, is defined by:

$$F = 2\frac{precision \cdot recall}{precision + recall} \tag{B.7}$$

This is a special case for a more generic F_β measure with $\beta = 1$. F_β weighs either the precision if $\beta < 1$ or the recall if $\beta > 1$.

$$F_\beta = (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{B.8}$$

B.2 A metric to compare foreground segmentations[§]

When working with object-based spatial segmentation of video, the major objective is to design an algorithm that produces appropriate segmentation results for the particular goals of the application being addressed. The demonstration of the usefulness of a particular algorithm entails necessarily a comparative analysis of the algorithm against similar algorithms. And the fair assessment requires suitable metrics providing a meaningful and objective measure of performance.

We proposed recently a unifying model for the comparison of image segmentations (Cardoso and Corte-Real, 2005, 2006). Here, we recover the general framework and instantiate a new metric, particularly adapted for the application at hand.

[§]This section is in part based on the article *Partition-distance methods for assessing spatial segmentations of images and videos* accepted for publication in the Computer Vision and Image Understanding international journal (Cardoso *et al.*, 2009).

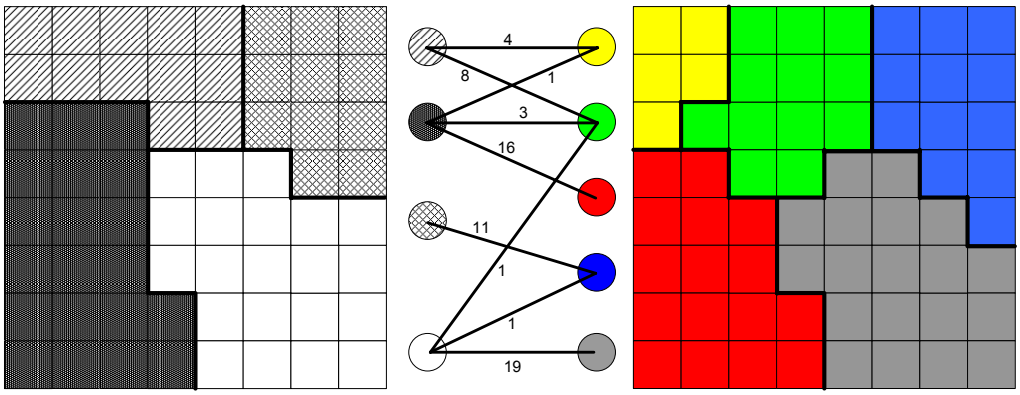


Figure B.1: *Intersection-graph for two segmentations. The weights shown correspond to the number of pixels in the intersection.*

At the core of the framework is the intersection-graph between two segmentations, defined as the bipartite graph with one node for each region of the segmentations. Two nodes are connected by an undirected, weighted edge if and only if those two regions intersect each other. Figure B.1 exemplifies such setting. The intersection-graph associated with two image segmentations can now be used as a factory of indices of similarity between partitions. The partition-distance (Cardoso and Corte-Real, 2005) has been defined as the problem of finding a maximum weighted matching in the intersection-graph. The sum of the weights of the unmatched edges on this matching process provides the distance between both segmentations. And the pixels corresponding to these unmatched edges constitute a informative error mask.

The weight assigned to an edge should express the importance of the corresponding intersection. The simplest way is to assign the area of the intersection to the weight of each edge. But this formulation is not necessarily the one better capturing the perceived quality of a segmentation. In particular, all pixels in the same intersection of two regions are being equally weighted. However, to accommodate human perception, the different error contributions should be weighted according to their visual relevance. Therefore, it is, arguably, more sensible that the cost of erring a pixel increases with pixel distance to the object border.

The foregoing argument motivates the introduction of the cost-based partition-distance, d^c_{sym}, as a generalisation of the partition-distance. Start by computing the distance of each pixel to the object border in both segmentations, d_1 and d_2. Define a monotonically increasing cost function on these two distances, $C(d_1, d_2)$. Different laws can be considered, such as linear, exponential or logarithmic. A suitable strategy is to set $C(d_1, d_2) = \max(d_1, d_2)$ or $C(d_1, d_2) = 2^{\max(d_1, d_2)}$. Finally, the weight of an edge will be the mere sum of the individual costs of the pixels in the intersection. Note that setting $C(d_1, d_2) = 1$ results in edges weighted by the area of the intersection. The cost-

based partition-distance will be the sum of the weights of the unmatched edges on the matching process that follows. The cost-based partition-distance will penalise thick discrepancies between two segmentations, favouring thin, along the borders, differences. This metric can also be seen as generalisation of the perceptual spatial measure proposed by Cavallaro *et al.* (2002).

It is possible to confirm that the cost-based partition-distance still enjoys of most of the useful properties of the original partition-distance (Cardoso and Corte-Real, 2005). Most notably, the cost-based partition-distance is still non-negative (being zero iff the two partitions coincide), symmetric, and transitive. It is, therefore, a metric. The transitive property is especially significant in the context of comparing more than two algorithms. It conveys the desirable behaviour that if segmentation A is similar to segmentation B and segmentation B is similar to segmentation C, then segmentation A is similar to segmentation C.

C Local descriptors

C.1 Overview

Local interest point detectors and descriptors are designed to extract specific points from images and produce features that allow for a robust matching between similar points across images. Point matching is an essential task in the wide-baseline matching process. Wide-baseline matching is the task of finding corresponding points between images of the same scene or object, in the case where the images are taken from widely separated viewing angles.

Local interest point detectors are designed to localise points that contain distinctive information in their local surrounding area and whose extraction is stable with respect to geometric transformations and noise. These points are *characteristic* points in the image, where the signal changes bi-dimensionally. In addition, local interest point detectors need to automatically specify an area around the characteristic point that will have a certain amount of invariance to image transformations. We will refer to this area as *local interest area*. Invariance to transformations means that given two images of a certain object taken from different viewing angle, the detector will be able to extract local points and areas in both images that correspond to the same point on the surface of the object.

Local descriptors are compact and distinct features, extracted from local interest areas. These descriptors are designed to be as specific as possible, while providing some invariance to imaging conditions and to compensate for possible errors in the local interest area definition. Local interest points must be as specific as possible because each local interest point is compared with a large amount of other local interest points to assess the similarity between each possible pair of points. This is especially important in the case of wide-baseline matching, where the point-to-point correspondence between images is the final objective.

Local point detectors and descriptors were originally proposed to enable efficient point-to-point matching in wide-baseline matching problems (Lowe, 2004; Mikolajczyk and Schmid, 2004). In more recent work these techniques have been exploited in other areas like object recognition, scene recognition, image annotation, image segmentation and video browsing.

These new applications explore the use of local descriptors in quantised form, where quantised local descriptors are usually entitled *visterms*. This quantisation allows, by counting the number of visterms' instances in an image, to produce a global image representation, the Bag-Of-Visterms (BOV). Due to the different nature of the application of local descriptors for image categorisation, instead of point-to-point matching, it has been argued that some properties that are important when performing point-to-point matching may not be necessary when applying visterms to image categorisation, and may even by detrimental to the performance.

One of the most important local descriptors scheme is Scale-Invariant Feature Transform (SIFT). This was the local descriptor used in this thesis (refer to Section 4.3).

C.2 SIFT

The SIFT feature describes the local interest area using a concatenation of local histograms of edge orientation computed over the a grid sub-division of the local interest area's gradient map (Lowe, 1999, 2004).

SIFT features have become widely used for both wide-baseline matching and quantised local descriptor approaches and have been found to perform best for many tasks by several authors (Lowe, 2004; Mikolajczyk and Schmid, 2004; Quelhas, 2006).

The SIFT extraction process is based on the extraction of gradient samples from the image at the scale of the local interest point to describe. In its original formulation (Lowe, 1999), SIFT feature extraction was coupled with the scale-space representation of the Difference of Gaussians (DoG) interest point detector. To increase the speed of the feature extraction process, the author used the pre-computed scale-space smoothed images to compute the SIFT descriptor feature. If we apply the SIFT descriptor together with some other local interest point detector, which does not have a scale-space representation, we must Gaussian smooth (or re-sample) the image to the scale of the detected point. We consider the case where we have access the scale-space representation which was used to extract the local interest point.

The SIFT feature extraction process, as illustrated in Figure C.1, is summarised in Figure C.2. In short, the image's gradient is sampled and its orientation is quantised. Using a grid division of the local interest area, local gradient orientation histograms are created where the gradient magnitude is accumulated. The final feature is the concatenation of all the local gradient orientation histograms. A Gaussian weighting is introduced in the SIFT feature extraction process (Figure C.2) to give more importance to

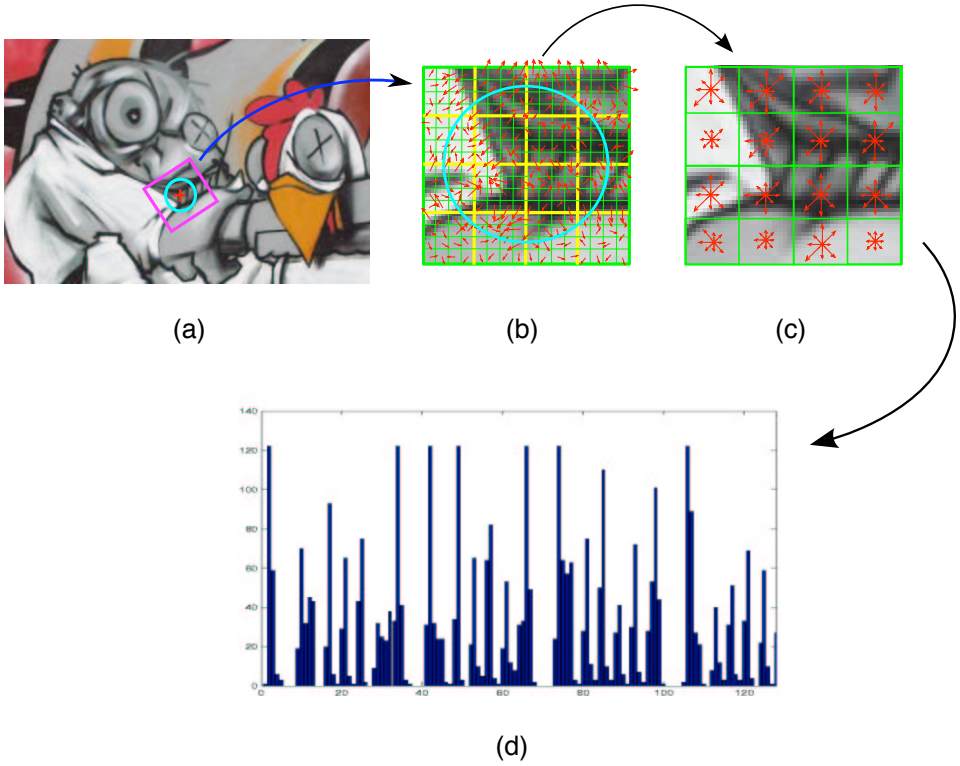


Figure C.1: *Illustration of the SIFT feature extraction process. (a) original image with the local interest point to describe, showing the detected location, scale and area used for sampling. (b) local interest area with gradient samples at each grid point, blue circle illustrates the Gaussian weighting window. (c) local individual orientation histograms which result of accumulating each sample into the corresponding bin of its local histogram. (d) final 128 dimensional SIFT feature (before normalisation). (image from Quelhas* et al.*, 2007)*

samples closer to the centre of the local interest area. This contributes to a greater invariance of the SIFT descriptor, since samples closer to the centre of the local interest areas are more robust to errors in the local interest area estimation.

Lowe (2004) found that the best compromise between performance an speed was obtained by using a 16×16 gradient sampling grid and a 4×4 sub-histogram grouping (cf. Figure C.1). The final descriptor proposed in this formulations is 128 (4x4x8) dimensional.

As mentioned in the beginning of this subsection, the SIFT descriptors is one of most prominent local interest point descriptor. One of the main reasons for its success is its low complexity, which makes this detector fast and easy to implement. Another reason

SIFT feature extraction procedure.

1. select the Gaussian smoothed image corresponding to the local interest point's characteristic scale (σ_D),
2. sample the image gradient based on the scale and orientation of the local interest point, using a regular grid around the local interest point location $\mathbf{x}_i$ (Figure C.1(b)),
3. normalise the sampled gradient's orientation with relation to the local interest point's orientation,
4. apply a Gaussian weighting to the gradient's magnitude, with $\sigma = \sigma_D/2$. (light blue circle in Figure C.1(b)),
5. quantise the gradients orientation into n orientations ($n = 8$ in Figure C.1(c)),
6. create grid division orientation histograms in which to accumulate the magnitude of the previously quantised local gradient (yellow grid in Figure C.1(b)),
7. form a vector by concatenating the grid histograms (Figure C.1(c)) into one histogram (Figure C.1(d)),
8. normalise the feature vector to further increase illumination invariance.

Figure C.2: *SIFT feature extraction process. SIFT feature extraction process (more details in Lowe, 2004).*

for the success of SIFT is the intrinsic invariance to small errors in the calculation of the position and area, resulting from representing the local image information with a histogram.

Bibliography

Til Aach, André Kaup, and Rudolf Mester. Statistical model-based change detection in moving video. *Signal Processing*, 31(2):165–180, March 1993.

Andrea F. Abatea, Michele Nappi, Daniel Riccioa, and Gabriele Sabatino. 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906, October 2007.

William B. Ackerman. Data flow languages. *Computer*, 15(2):15–25, February 1982.

W. H. Adams, Giridharan Iyengar, Ching-Yung Lin, Milind Ramesh Naphade, Chalapathy Neti, Harriet J. Nock, and John R. Smith. Semantic indexing of multimedia content using visual, audio, and text cues. *EURASIP Journal on Applied Signal Processing*, 2003(2):170–185, 2003.

Zaher Al Aghbari, Kunihiko Kaneko, and Akifumi Makinouchi. Content-trajectory approach for searching video databases. *IEEE Transactions on Multimedia*, 5(4):516–531, December 2003.

Pedro M.Q. Aguiar and José M.F. Moura. Figure-ground segmentation from occlusion. *IEEE Transactions on Image Processing*, 14(8):1109–1124, August 2005.

Xavier Amatriain. CLAM, a framework for audio and music application development. *IEEE Software*, 24(1):82–85, January/February 2007.

M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–189, February 2002.

Edward A. Ashcroft and William W. Wadge. Lucid, a nonprocedural language with iteration. *Communications of the ACM*, 20(7):519–526, July 1977.

Pradeep K. Atrey, Namunu C. Maddage, and Mohan S. Kankanhalli. Audio based event detection for multimedia surveillance. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 5, pages 813–816, Toulouse, France, May 2006.

Shai Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, August 2004.

Yaakov Bar-Shalom. *Tracking and data association*. Academic Press, 1988.

Marian Stewart Bartlett, Javier R. Movellan, and Terrence J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, November 2002.

Peter N. Belhumeur, Jo ao P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, July 1997.

Gordon Bell and Jim Gemmell. A digital life. *Scientific American*, March 2007.

Chiraz BenAbdelkader, Ross Cutler, and Larry Davis. Motion-based recognition of people in eigengait space. In *Proceedings of IEEE International Conference on Face and Gesture Recognition*, pages 267–272, Washington, DC, May 2002.

Keni Bernardin, Rainer Stiefelhagen, and Alex Waibel. Probabilistic integration of sparse audio-visual cues for identity tracking. In *Proceedings of ACM International Conference on Multimedia*, pages 151–158, Vancouver, BC, October 2008.

Marcelo Bertalmio, Guillermo Sapiro, and Gregory Randall. Morphing active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):733–737, July 2000.

Bir Bhanu and Xiaotao Zou. Moving humans detection based on multi-modal sensor fusion. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 136–136, Washington, DC, June/July 2004.

Michael J. Black and Allan D. Jepson. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

James Black, Tim Ellis, and Paul Rosin. Multi view image surveillance and tracking. In *Proceedings of IEEE Workshop on Motion and Video Computing*, pages 169–174, December 2002.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

Jean-Francois Bonastre, Perrine Delacourt, Corinne Fredouille, and Christian Wellekens Teva Merlin. A speaker tracking system based on speaker turn detection for nist evaluations. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 2, pages 1177–1180, Istanbul, Turkey, June 2000.

Alan Conrad Bovik, Marianna Clark, and Wilson S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, January 1990.

Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999, San Juan, Puerto Rico, June 1997.

Stuart Bray and George Tzanetakis. Implicit patching for dataflow-based audio analysis and synthesis. In *In Proceedings of International Music Conference (ICMC)*, 2005.

Albert S. Bregman. *Auditory scene analysis: The Perceptual Organization of Sound*. MIT Press, Cambridge, MA, 1990.

Thomas Brox, Andr'es Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on theory for warping. In *Proceedings of European Conference on Computer Vision*, pages 25–36, Prague, Czech Republic, 2004.

Aurélie Bugeau and Patrick Pérez. Detection and segmentation of moving objects in highly dynamic scenes. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, June 2007.

Wray L. Buntine. Variational extensions to EM and multinomial PCA. In *Proceedings of European Conference on Machine Learning*, pages 23–34, Helsinki, August 2002.

Cristopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Neil Burroughs and George Tzanetakis. Flexible event scheduling for data-flow audio processing. In *OOPSLA Companion*, pages 724–725, 2006.

Torsten Butz and Jean-Philippe Thiran. From error probability to information theoretic (multi-modal) signal processing. *Signal Processing*, 85(5):875–902, 2005.

William A. S. Buxton. *Human skills in interface design*, pages 1–12. Wiley, 1994.

Dan Buzan, Stan Sclaroff, and George Kollios. Extraction and clustering of motion trajectories in video. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 521–524, Cambridge, UK, August 2004.

Qin Cai and J. K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241–1247, November 1999.

Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 599–610, Paris, France, June 2004.

W. Campbell, D. E. Sturim, and D. A. Reynolds. Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311, May 2006.

W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20(2–3):210–229, April 2006.

Joseph P. Campbell, Jr. Speaker recognition: a tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, September 1997.

Antonio Camurri, Paolo Coletta, Alberto Massari, Barbara Mazzarino, Massimiliano Peri, Matteo Ricchetti, Andrea Ricci, and Gualtiero Volpe. Toward real-time multimodal processing: EyesWeb 4.0. In *AISB Convention: Motion, Emotion and Cognition*, Leeds, UK, March 2004.

Jaime S. Cardoso and Luís Corte-Real. Toward a generic evaluation of image segmentation. *IEEE Transactions on Image Processing*, 14(11):1773–1782, November 2005.

Jaime S. Cardoso and Luís Corte-Real. A measure for mutual refinements of image segmentations. *IEEE Transactions on Image Processing*, 15(8):2358–2363, August 2006.

Jaime S. Cardoso, Pedro Carvalho, Luis F. Teixeira, and Luis Corte-Real. Partition-distance methods for assessing spatial segmentations of images and videos. *Computer Vision and Image Understanding*, 113(7):811–823, July 2009.

Jaime S. Cardoso. *Metadata Assisted Image Segmentation*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, May 2006.

Pedro Carvalho. Data combination for people tracking in video sequences. Thesis Research Plan, Faculdade de Engenharia da Universidade do Porto, January, 2009.

Yaron Caspi and Michael Irani. A step towards sequence-to-sequence alignment. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 682–689, Hilton Head Island, SC, June 2000.

Andrea Cavallaro and Touradj Ebrahimi. Video object extraction based on adaptive background and statistical change detection. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 465–475, 2001.

Andrea Cavallaro, Elisa Drelie Gelasca, and Touradj Ebrahimi. Objective evaluation of segmentation quality using spatio-temporal context. In *Proceedings of IEEE International Conference on Image Processing*, September 2002.

Amit Chakraborty and James S. Duncan. Game-theoretic integration for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):12–30, January 1999.

Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):602–615, September 1998.

T. Chateau, V. Gay-Belille, F. Chausse, and J.T. LaprestÈ. Real-time tracking with classifiers. In *Workshop on Dynamical Vision at ECCV 2006*, May 2005.

Scott Shaobing Chen and P. S. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proceedings of DARPA Speech Recognition Workshop*, pages 127–132, 1998.

Yunqiang Chen, Yong Rui, and Thomas S. Huang. JPDAF based HMM for real-time contour tracking. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 543–550, Kauai, HI, December 2001.

Datong Chen, Robert Malkin, and Jie Yang. Multimodal detection of human interaction events in a nursing home environment. In *Proceedings of International Conference on Multimodal Interfaces*, pages 82–89, State College, PA, October 2004.

Shu-Ching Chen, Mei-Ling Shyu, Chengcui Zhang, and Min Chen. A multimodal data mining framework for soccer goal detection based on decision tree logic. *International Journal of Computer Applications in Technology*, 27(4):321–323, 2006.

Olivier Chomat and James L. Crowley. Probabilistic recognition of activity using local appearance. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 101–109, Ft. Collins, CO, June 1999.

Brian Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 3037–3040, Phoenix, AZ, March 1999.

C. Clavel, T. Ehrette, and G. Richard. Events detection for an audio-based surveillance system. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 1306–1309, Amsterdam, The Netherlands, July 2005.

Robert T. Collins, Ralph Gross, and Jianbo Shi. Silhouette-based human identification from body shape and gait. In *Proceedings of IEEE International Conference on Face and Gesture Recognition*, pages 351–356, Washington, DC, May 2002.

Dorin Comaniciu and Peter Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, May 2003.

Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.

Ingemar J. Cox, Joumana Ghosn, and Peter N. Yianilos. Feature-based face recognition using mixture-distance. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 209–216, San Francisco, CA, June 1996.

Marco Cristani, Manuele Bicego, and Vittorio Murino. On-line adaptive background modelling for audio surveillance. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 399–402, Cambridge, UK, August 2004.

Marco Cristani, Manuele Bicego, and Vittorio Murino. Audio-visual foreground extraction for event characterization. In *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop*, June 2006.

Ross Cutler and Larry Davis. Look who's talking: Speaker detection using video and audio correlation. In *Proceedings of IEEE International Conference on Multimedia & Expo*, volume 3, pages 1589–1592, New York, NY, July/August 2000.

Ross Cutler and Larry Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, August 2000.

Belur V. Dasarathy. Sensor fusion potential exploitation - innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, January 1997.

Belur V. Dasarathy. Industrial applications of multi-sensor multi-source information fusion. In *Proceedings of IEEE International Conference on Industrial Technology*, volume 1, pages 5–11, 2000.

James W. Davis and Aaron F. Bobick. The representation and recognition of action using temporal templates. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 928–934, San Juan, Puerto Rico, June 1997.

Karen K. De Valois, editor. *Seeing*. Academic Press, 2nd edition, 2000.

Alberto del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, 1999.

Yining Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, August 2001.

Simon Denman, Vinod Chandran, and Sridha Sridharan. Tracking people in 3D using position, size and shape. In *IEEE Proceedings of International Symposium on Signal Processing and Its Applications*, volume 2, pages 611–614, August 2005.

Ajay Divakaran and Huifang Sun. A region based descriptor for spatial distribution of motionactivity for compressed video. In *Proceedings of International Conference on Image Processing*, volume 2, pages 287–290, Vancouver, BC, September 2000.

Marie-Pierre Dubuisson and Anil K. Jain. Contour extraction of moving objects in complex outdoor scenes. *International Journal of Computer Vision*, 14(1):83–105, January 1995.

Richard O. Duda and Peter E. Hart. *Pattern classification and scene analysis*. John Wiley & Sons, 1973.

Stéphane Dupont and Jurgen Luttin. Audio-visual speech modeling for continuous speech recognition. *IEEE Transactions on Multimedia*, 2(3):141–151, September 2000.

Daniel Duraes, Luis F. Teixeira, and Luis Corte-Real. Building modular surveillance systems based on multiple sources of information – architecture and requirements. In *Proceedings of International Conference on Signal Processing and Multimedia Applications*, pages 314–319, Porto, Portugal, July 2008.

Shahram Ebadollahi, Lexing Xie, Shih-Fu Chang, and John R. Smith. Visual event detection using multi-dimensional concept dynamics. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 881–884, Toronto, ON, July 2006.

Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *Proceedings of International Conference on Computer Vision*, volume 2, pages 726–733, Nice, France, October 2003.

Ahmet Ekin, A. Murat Tekalp, and Rajiv Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, July 2003.

Ahmed Elgammal, David Hardwood, and Larry Davis. Non-parametric model for background subtraction. In *Proceedings of European Conference on Computer Vision*, volume 2, pages 751–767, 2000.

How-Lung Eng, Kar-Ann Toh, Alvin H. Kam, Junxian Wang, and Wei-Yun Yau. An automatic drowning detection surveillance system for challenging outdoor pool environments. In *Proceedings of IEEE International Conference on Computer Vision*, pages 532–539, Nice, France, October 2003.

Mark Everingham, Andrew Zisserman, Chris K. I. Williams, and Luc Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. `http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf`, 2006.

John M. Findlay and Iain Gilchrist. *Active Vision: The Psychology of Looking and Seeing*. Oxford University Press, 2003.

Robert Fisher, Jose Santos-Victor, and James Crowley. CAVIAR: Context Aware Vision using Image-based Active Recognition. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR`, 2002.

John W. Fisher III and Trevor Darrell. Speaker association with signal-level audiovisual fusion. *IEEE Transactions on Multimedia*, 6(3):406–413, June 2004.

Ronald A. Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8:376–386, 1938.

Gian Luca Foresti, Christian Micheloni, Lauro Snidaro, Paolo Remagnino, and Tim Ellis. Active video-based surveillance system: the low-level image and video processing techniques needed for implementation. *IEEE Signal Processing Magazine*, 22(2):25–37, March 2005.

Thomas E. Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data sets. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, July 1983.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

Dennis Gabor. Theory of communication. *Journal of IEE*, 93(26):429–457, November 1946.

Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

Yongsheng Gao and Maylor K.H. Leung. Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):764–779, June 2002.

Yongsheng Gao and Yutao Qi. Robust visual similarity retrieval in single model face databases. *Pattern Recognition*, 38(7):1009–1020, July 2005.

Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. The LIMSI broadcast news transcription system. *Speech Communication*, 37:2002, 2002.

A. Jay Goldstein, Leon D. Harmon, and Ann B. Lesk. Identification of human faces. *Proceedings of the IEEE*, 59(5):748–760, May 1971.

Gaile G. Gordon. Face recognition based on depth maps and surface curvature. *Proceedings of SPIE*, 1570:234–247, September 1991.

Venu Govindaraju, David B. Sher, Rohini K. Srihari, and Sargur N. Srihari. Locating human faces in newspaper photographs. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 549–554, San Diego, CA, June 1989.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2, pages 1458–1465, Beijing, China, October 2005.

Richard L. Gregory. *Eye and Brain: The Psychology of Seeing*. Princeton University Press, 1978.

Abhinav Gupta and Larry S. Davis. Objects in action: An approach for combining action understanding and object perception. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, June 2007.

Amarnath Gupta and Ramesh Jain. Visual information retrieval. *Communications of the ACM*, 40(5):70–79, May 1997.

Mihai Gurban and Jean-Philippe Thiran. Multimodal speaker localization in a probabilistic framework. In *Proceedings of European Signal Processing Conference*, Florence, Italy, September 2006.

Thien Ha-Minh, Alessandro Tortelli, Luis F. Teixeira, Lutz Goldmann, Mustafa Karaman, Stewart Worrall, Tim Masterton, and Charles Attwood. First set of contributions and evaluation of tools for video segmentation and tracking. Deliverable D2.2.2, Visnet II consortium, November 2007.

Thien Ha-Minh, Luis F. Teixeira, Pedro Carvalho, Lutz Goldmann, Mustafa Karaman, Stewart Worrall, Tim Masterton, Charles Attwood, and Krystian Ignasiak. Update set of developments and evaluation of tools for video segmentation and tracking. Deliverable D2.2.7, Visnet II consortium, August 2008.

James Hafner, Harpreet S. Sawhney, Will Equitz, Myron Flickner, and Wayne Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):729–736, July 1995.

Raffay Hamid, Amos Johnson, Samir Batta, Aaron Bobick, Charles Isbell, and Graham Coleman. Detection and explanation of anomalous activities: Representing activities as bags of event n-grams. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1031–1038, San Diego, CA, June 2005.

Arum Hampapur, Lisa Brown, Jonathan Connell, Sharat Pankanti, Andrew Senior, and Yingli Tian. Smart surveillance: applications, technologies and implications. In *Proceedings of IEEE Pacific-Rim Conference on Multimedia*, volume 2, pages 1133–1138, Singapore, December 2003.

Ju Han and Bir Bhanu. Statistical feature fusion for gait-based human recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 842–847, Washington, DC, June 2004.

Ju Han and Bir Bhanu. Individual recognition using gait energy image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):316–322, February 2006.

Bohyung Han, Dorin Comaniciu, and Larry Davis. Sequential kernel density approximation through mode propagation: Applications to background modeling. In *Proceedings of Asian Conference on Computer Vision*, 2004.

Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, 3(6):610–621, November 1973.

Ismail Haritaoglu, David Harwood, and Larry S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

John Hershey and Javier R. Movellan. Audio vision: Using audio-visual synchrony to locate sounds. *Advances in Neural Information Processing Systems*, (12):813–819, 1999.

Erik Hjelmas and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, September 2001.

Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1–2):177–196, January/February 2001.

Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981.

Ian P. Howard and William B. Tempelton. *Human spatial orientation*. Wiley, London, 1966.

Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):696–706, May 2002.

Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, August 2004.

Weiming Hu, Dan Xie, Tieniu Tan, , and Steve Maybank. Learning activity patterns using fuzzy self-organizing neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(3):1618–1626, June 2004.

Jing Huang, S Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 762–768, San Juan, Puerto Rico, June 1997.

Jennifer Huang, Bernd Heisele, and Volker Blanz. Component-based face recognition with 3d morphable models. In *Proceedings of International Conference on Audio- and Video-Based Biometric Person Authentication Surrey*, pages 27–34, Guildford, UK, July 2003.

Daniel P. Huttenlocher, Jae J. Noh, and William J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proceedings of IEEE International Conference on Computer Vision*, pages 93–101, Berlin, Germany, 1993.

Michael Isard and Andrew Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

Yuri A. Ivanov and Aaron F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, August 2000.

Anil K. Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, August 1996.

Omar Javed, Zeeshan Rasheed, Khurram Shafique, and Mubarak Shah. Tracking across multiple cameras with disjoint views. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2, pages 952–957, October 2003.

Omar Javed, Khurram Shafique, and Mubarak Shah. Appearance modeling for tracking in multiple non-overlapping cameras. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 26–33, San Diego, CA, June 2005.

Omar Javed, Khurram Shafique, and Mubarak Shah. Automated surveillance in realistic scenarios. *IEEE Multimedia*, 14(1):30–39, January-March 2007.

Sylvie Jeannin, Radu Jasinschi, Alfred She, Thumpudi Naveen, Benoit Mory, and Ali Tabatabai. Motion descriptors for content-based video representation. *Signal Processing: Image Communication*, 16(1):59–85, 2000.

Yonggang Jin and Farzin Mokhtarian. Efficient video retrieval by motion trajectory. In *Proceedings of British Machine Vision Conference*, pages 667–676, London, UK, September 2004.

Gunnar Johansson. Visual motion perception. *Scientific American*, (232):76–88, January 1975.

Neil Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615, 1996.

Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 199–206, Kauai, HI, December 2001.

Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, January 2002.

Young-Kee Jung, Kyu-Won Lee, and Yo-Sung Ho. Content-based event retrieval using semantic scene interpretation for automated traffic surveillance. *IEEE Transactions on Intelligent Transport Systems*, 2(3):151–163, September 2001.

Amit Kale, Aravind Sundaresan, A. N. Rajagopalan, Naresh P. Cuntoor, Amit K. Roy-Chowdhury, Volker Krüger, and Rama Chellappa. Identification of humans using gait. *IEEE Transactions on Image Processing*, 13(9):1163–1173, September 2004.

Rudolf Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Takeo Kanade. *Computer Recognition of Human Faces*, volume 47. Birkhauser Verlag, Stuttgart, 1977.

Mustafa Nazmi Kaynak, Qi Zhi, Adrian David Cheok, Kuntal Sengupta, Jian Zhang, and Chi Chung Ko. Analysis of lip geometric features for audio-visual speech recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 34(4):564–570, July 2004.

Christoph Kayser, Christopher I. Petkov, Michael Lippert, and Nikos K. Logothetis. Mechanisms for allocating auditory attention: An auditory saliency map. *Current Biology*, 15:1943–1947, November 2005.

Thomas Kemp, Michael Schmidt, Martin Westphal, and Alex Waibel. Strategies for automatic segmentation of audio data. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 3, pages 1423–1426, Istanbul, Turkey, June 2000.

Patrick Kenny, Gilles Boulianne, and Pierre Dumouchel. Eigenvoice modeling with sparse training data. *IEEE Transactions on Speech and Audio Processing*, 13(3):345–354, May 2005.

Vera Kettnaker and Ramin Zabih. Bayesian multi-camera surveillance. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 253–259, Ft. Collins, CO, June 1999.

Sohaib Khan and Mubarak Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, October 2003.

Zia Khan, Tucker Balch, and Frank Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1819, November 2005.

Einat Kidron, Yoav Y. Schechner, and Michael Elad. Cross-modal localization via sparsity. *IEEE Transactions on Signal Processing*, 55(4):1390–1404, April 2007.

Kyungnam Kim. *Algorithms and evaluation for object detection and tracking in Computer Vision*. PhD thesis, University of Maryland, Baltimore, MD, 2005.

Mike Kirby and Lawrence Sirovich. Application of the Karhunen-Loeve procedure for thecharacterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, January 1990.

Genshiro Kitagawa. Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1041, December 1987.

Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In *Proceedings of European Conference on Computer Vision*, pages 189–196, 1994.

Stefanos Kollias and Kostas Karpouzis. Multimodal emotion recognition and expressivity analysis. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 779–783, Amsterdam, The Netherlands, July 2005.

Margarita Kotti, Luis G. Martins, Emmanouil Benetos, Jaime S. Cardoso, and Constantine Kotropoulos. Automatic speaker segmentation using multiple features and distance measures: A comparison of three approaches. In *Proceedings of IEEE International Conference on Multimedia & Expo*, pages 1101–1104, Toronto, Canada, July 2006.

Young Ho Kwon and Niels da Vitoria Lobo. Face detection using templates. In *Proceedings of International Conference on Pattern Recognition*, pages 764–767, 1994.

Martin Lades, Student Member, Jan C. Vorbruggen, Joachim Buhmann, Jorg Lange, Christoph V. D. Malsburg, Rolf P. Wurtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, March 1993.

Mathieu Lagrange, Luis G. Martins, Luis F. Teixeira, and George Tzanetakis. Speaker segmentation of interviews using integrated video and audio change detections. In *Proceedings of International Workshop on Content-Based Multimedia Indexing*, pages 219–226, Bordeaux, France, June 2007.

Mathieu Lagrange, Luis G. Martins, Jennifer Murdoch, and George Tzanetakis. Normalized cuts for predominant melodic source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):278–290, February 2008.

Andreas Lanitis, Chris J. Taylor, and T. F. Cootes. Automaitc interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, July 1997.

Lily Lee, Raquel Romano, and Gideon Stein. Monitoring activities from multiple video streams: establishing a common coordinate frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):758–767, August 2000.

L. Lee, G. Dalley, and K. Tieu. Learning pedestrian models for silhouette refinement. In *Proceedings of IEEE Conference on Computer Vision*, pages 663–670, Nice, France, October 2003.

Dar-Shyang Lee. Effective Gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):827–832, May 2005.

Riccardo Leonardi, Pierangelo Migliorati, and Maria Prandini. Semantic indexing of soccer audio-visual sequences: a multimodal approach based on controlled markov chains. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):634–643, May 2004.

Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472, November 2004.

Ying Li, Shrikanth Narayanan, and S. Kuo. Content-based movie analysis and indexing based on audiovisual cues. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(8):1073–1085, August 2004.

Shang-Hung Lin, Sun-Yuan Kung, and Long-Ji Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1):114–132, January 1997.

Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, November 1998.

Zongyi Liu and Sudeep Sarkar. Improved gait recognition by gait dynamics normalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):863–876, June 2006.

Chengjun Liu and Harry Wechsler. Evolutionary pursuit and its application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):570–582, June 2000.

Zhu Liu, Yao Wang, and Tsuhan Chen. Audio feature extraction and analysis for scene segmentation and classification. *The Journal of VLSI Signal Processing*, 20(1):61–79, October 1998.

Yanxi Liu, Robert Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354 – 371, March 2004.

J. Lou, Q. Liu, T. Tan, and W. Hu. Semantic interpretation of object activities in a surveillance system. In *Proceedings of International Conference on Pattern Recognition*, volume 3, Quebec, Canada, August 2002.

David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, September 1999.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Lie Lu and Hong-Jiang Zhang. Real-time unsupervised speaker change detection. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 358–361, Quebec, Canada, August 2002.

Lie Lu, Hong-Jiang Zhang, and Hao Jiang. Content analysis for audio classification and segmentation. *IEEE Transactions on Speech and Audio Processing*, 10(7):504–516, October 2002.

Juwei Lu, K. N. Plataniotis, and K. N. Plataniotis. Regularized discriminant analysis for the small sample size problem in face recognition. *Pattern Recognition Letters*, 24(16):3079–3087, December 2003.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

Xiao-Guang Lv, Jie Zhou, and Chang-Shui Zhang. A novel algorithm for rotated human face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 760–765, Hilton Head Island, SC, June 2000.

John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, August 2000.

Christopher Madden, Eric Dahai Cheng, and Massimo Piccardi. Tracking people across disjoint camera views by an illumination-tolerant appearance representation. *Machine Vision and Applications*, 18(3):233–247, August 2007.

Dimitrios Makris. *Learning an Activity-Based Semantic Scene Model*. PhD thesis, City University, London, UK, July 2004.

Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modelling with front propagation: a level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, February 1995.

Dragos-Anton Manolescu. A data flow pattern language. In *Proceedings of Conference on Pattern Languages of Programs*, volume 1, Monticello, IL, September 1997.

J. Markel and S. Davis. Text-independent speaker recognition from a large linguistically unconstrained time-spaced data base. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27(1):74–82, February 1979.

Markos Markou and Sameer Singh. Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83(12):2003, December 2003.

David Marr. *Vision*. W.H.Freeman & Co, 1982.

Luis G. Martins. *A Computational Framework for Sound Segregation in Music Signals*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2009.

Stephen McAdams and Emmanuel Bigand, editors. *Thinking in sound: the cognitive psychology of human audition*. Oxford University Press, 1993.

Harry McGurk and John MacDonald. Hearing lips and seeing voices. *Nature*, (264):746–748, 1976.

Gérard Medioni, Isaac Cohen, cois Bérmond Fran Somboon Hongeng, and Ramakant Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):873–899, August 2001.

Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. *International Journal of Computer Vision*, 60(1):63–86, 2004.

Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(10):1615–1630, 2005.

Anurag Mittal and Larry S. Davis. M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51(3):189–203, February 2003.

Anurag Mittal and Nikos Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–302 – II–309, Washington, DC, June/July 2004.

Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.

Farzin Mokhtarian, Sadegh Abbasi, and Josef Kittler. Robust and efficient shape indexing through curvature scale space. In *Proceedings of British Machine Vision Conference*, pages 53–62. Edinburgh, UK, 1996.

Gianluca Monaci, Oscar Divorra Escoda, and Pierre Vandergheynst. Analysis of multimodal sequences using geometric video representations. *Signal Processing*, 86(12):3534–3548, December 2006.

Michael Muhlbaier, Apostolos Topalis, and Robi Polikar. Learn++.MT: A new approach to incremental learning. In *Workshop on Multiple Classifier Systems, Springer Lecture Notes in Computer Science (LNCS)*, volume 3077, pages 52–61, June 2004.

Andrew Naftel and Shehzad Khalid. Motion trajectory learning in the DFT-coefficient feature space. In *Proceedings of IEEE International Conference on Computer Vision Systems*, pages 47–54, New York, NY, January 2006.

Ara V. Nefian and Monson H. Hayes III. Hidden markov models for face recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 2721–2724, Seattle, WA, May 1998.

Ulric Neisser. *Cognitive psychology*. Appleton-Century-Crofts, New York, NY, 1967.

L. Nigay and J. Coutaz. A design space for multimodal systems - concurrent processing and data fusion. In *Proceedings of Conference on Human Factors in Computing Systems (INTERCHI 93)*, pages 172–178. Addison Wesley, 1993.

Peter Nillius, Josephine Sullivan, and Stefan Carlsson. Multi-target tracking – linking identities using bayesian network inference. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2187–2194, New York, NY, June 2006.

David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, New York, NY, June 2006.

M. S. Nixon, J. N. Carter, D. Cunado, P.S. Huang, and S. V. Stevenage. *Biometrics: Personal Identification in Networked Society*, chapter Automatic gait recognition, pages 231–249. Kluwer, Norwell, MA, 1999.

Harriet J. Nock, Giridharan Iyengar, and Chalapathy Neti. Assessing face and speech consistency for monologue detection in video. In *Proceedings of ACM International Conference on Multimedia*, Juan-les-Pins, France, 2002.

Harriet J. Nock, Giridharan Iyengar, and Chalapathy Neti. Speaker localisation using audio-visual synchrony: An empirical study. In *Proceedings of International Conference on Image and Video Retrieval*, pages 488–499, 2003.

Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of European Conference on Computer Vision*, May 2006.

Timo Ojala, Matti Pietikainen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, January 1996.

Yann Orlarey, Albert Gräf, and Stefan Kersten. DSP programming with Faust, Q and SuperCollider. In *Linux Audio Conference*, Berlin, Germany, 2006.

George Orwell. *Nineteen Eighty-Four*. Secker and Warburg, 1949.

Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, San Juan, Puerto Rico, June 1997.

Jonathan Owens and Andrew Hunter. Application of the self-organising map to trajectory classification. In *Proceedings of IEEE Workshop on Visual Surveillance*, pages 77–83, 2000.

Nikos Paragios and Visvanathan Ramesh. A MRF-based approach for real-time subway monitoring. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1034–1040, Kauai, HI, December 2001.

Rama Chellappa Pavan Turaga, Ashok Veeraraghavan. Mining videos for events using a cascade of dynamical systems: From videos to verbs. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, June 2007.

I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: from the laboratory to the commercial world. *Proceedings of the IEEE*, 89(10):1478–1497, 2001.

Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 84–91, Seattle, WA, June 1994.

Valery A. Petrushin. Mining rare and frequent events in multi-camera surveillance video using self-organizing maps. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 794–800, Chicago, IL, August 2005.

P. Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J. Rauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, April 1998.

Massimo Piccardi and T. Jan. Mean-shift background image modelling. In *Proceedings of International Conference on Image Processing*, pages 3399–3402, 2004.

Massimo Piccardi. Background subtraction techniques: a review. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, October 2004.

Ramprasad Polana and Randal Nelson. Detecting activities. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2–7, New York, NY, June 1993.

Robi Polikar, Lalita Udpa, Satish S. Udpa, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 31(4):497–508, 2001.

Gerasimos Potamianos, Chalapathy Neti, Guillaume Gravier, Ashutosh Garg, and Andrew W. Senior. Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 91(9):1306–1326, September 2003.

Miller Puckette. Combining event and signal processing in the MAX graphical programming environment. *Computer Music Journal*, 15(3):68–77, 1991.

Miller Puckette. Pure data. In *In Proceedings of International Music Conference (ICMC)*, pages 269–272, 1997.

Shrinivas Pundlik and Stan Birchfield. Motion segmentation at any speed. In *Proceedings of British Machine Vision Conference*, September 2006.

Wei Qu, Dan Schonfeld, and Magdi Mohamed. Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. *EURASIP Journal on Advances in Signal Processing*, 22(8):758–767, August 2000.

Pedro Quelhas, Florent Monay, Jean-Marc Odobez, Daniel Gatica-Perez, and Tinne Tuytelaars. A thousand words in a scene. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1575–1589, September 2007.

Pedro Quelhas. *Scene Image Classification and Segmentation with Quantized Local Descriptors and Latent Aspect Modeling*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, December 2006.

Regunathan Radhakrishnan, Ajay Divakaran, and Paris Smaragdis. Audio-visual foreground extraction for event characterization. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 158–161, New Paltz, NY, October 2005.

Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *International Journal of Computer Vision*, 50(2):203–226, November 2002.

K. Ravishankar, B. Prasad, S. Gupta, and K. Biswas. Dominant color region based indexing for CBIR. In *Proceedings of International Conference on Image Analysis and Processing*, pages 887–892, Venice, Italy, September 1999.

Carlo S. Regazzoni, Visvanathan Ramesh, and Gian Luca Foresti. Scanning the Issue/Technology - Special issue on video processing, understanding and communications in third generation surveillance systems. *Proceedings of the IEEE*, 89(10):1419–1440, October 2001.

James M. Rehg, Kevin. P. Murphy, and Paul W. Fieguth. Vision-based speaker detection using bayesian networks. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 110–116, Ft. Collins, CO, June 1999.

D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, December 1979.

Douglas A. Reynolds and Richard C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, January 1995.

Christof Ridder, Olaf Munktel, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on Recent Advances in Mechatronics*, pages 193–199, 1995.

David F. Rosenthal and Hiroshi G. Okuno. *Computational auditory scene analysis*. Lawrence Erlbaum, 1998.

David A. Ross, Jongwoo Lim, and Ruei-Sung Lin. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, August 2007. (online).

Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998.

David Sadlier and Noel E. O'Connor. Event detection in field sports video using audio-visual features and a support vector machine. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1225–1233, October 2005.

Payam Saisan, Gianfranco Doretto, Ying Nian Wu, and Stefano Soatto. Dynamic texture recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 58–63, Kauai, HI, December 2001.

V. Salari and Ishwar K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):87–91, January 1990.

Ashok Samal and Prasana A. Iyengar. Human face detection using silhouettes. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6):845–867, December 1995.

Conrad Sanderson. *Automatic person verification using speech and face information*. PhD thesis, Griffith University, 2002.

Sudeep Sarkar, P. JonathonPhillips, Zongyi Liu, Isidro Robledo Vega, Patrick Grother, and Kevin W. Bowyer. The humanID gait challenge problem: data sets, performance, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):162–177, February 2005.

Koichi Sato and J. K. Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vision Image Understand*, 96(2):100–128, November 2004.

John Saunders. Real time discrimination of broadcast speech/music. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 993–996, 1996.

Gary Scavone and Perry Cook. RtMidi, RtAudio, and Synthesis ToolKit (STK) update. In *Proceedings of International Computer Music Conference*, Barcelona, Spain, September 2005.

Eric Scheirer and Malcolm Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 1331–1334, Munich, April 1997.

Robert Sekuler, Allison B. Sekuler, and Renee Lau. Collisions between moving visual targets: what controls alternative ways of seeing an ambiguous display? *Nature*, (385):308, 1997.

Seong-Wook and Rama Chellappa. Attribute grammar-based event recognition and anomaly detection. In *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop*, page 107, June 2006.

Ishwar K. Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):56–73, January 1987.

Khurram Shafique and Mubarak Shah. A non-iterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, January 2005.

G. Shakhnarovich, L. Lee, and T. Darrell. Integrated face and gait recognition from multiple views. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 439–446, Kauai, HI, December 2001.

Linlin Shen and Li bai. A review on Gabor wavelets for face recognition. *Pattern Analysis Applications*, 9(2–3):273–292, October 2006.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, Hilton Head Island, SC, June 1994.

Yifan Shi, Yan Huang, Minnen, David Minnen, Aaron Bobick, and Irfan Essa. Propagation networks for recognition of partially ordered sequential action. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 862–869, Washington, DC, June/July 2004.

Nils T. Siebel and Steve J. Maybank. Fusion of multiple tracking algorithm for robust people tracking. In *Proceedings of European Conference on Computer Vision*, pages 373–387, Copenhagen, Denmark, June 2002.

Matthew A. Siegler, Uday Jain, Bhiksha Raj, and Richard M. Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proceedings of DARPA Speech Recognition Workshop*, pages 97–99, 1997.

Josef Sivic and Andrew Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477, Nice, France, October 2003.

Valene Skerpac. Speaker identification and verification (SIV) glossary. Technical report, VoiceXML Forum Speaker Biometrics Committee, May 2007.

Malcolm Slaney and Michele Covell. Facesync: A linear operator for measuring synchronization of video facial images and audio tracks. In *Proceedings of Neural Information Processing Systems (NIPS)*, volume 13, pages 814–820, 2000.

Alan F. Smeaton and Michael McHugh. Event detection in an audio-based sensor network. *Multimedia Systems*, 12(3):1432–1882, December 2006.

Alex Solomonoff, W. M. Campbell, and I. Boardman. Advances in channel compensation for SVM speaker recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal ProcessingI*, volume 1, pages 629–632, Philadelphia, PA, March 2005.

F. Soong, A. Rosenberg, L. Rabiner, and B. Juang. A vector quantization approach to speaker recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 387–390, Tampa, FL, 1985.

Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 19, pages 246–252, Ft. Collins, CO, June 1999.

Markus A. Stricker and Alexander Dimai. Color indexing with weak spatial constraints. In *Proceedings of Storage and Retrieval for Image and Video Databases (SPIE)*, pages 29–40, 1996.

William H. Sumby and Iriwin Pollack. Visual contribution to speech intelligibility in noise. *The Journal of the Acoustical Society of America*, 26(2):212–215, 1954.

Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

Martin Szummer and Rosalind W. Picard. Temporal texture modeling. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 823–826, Lausanne, Switzerland, September 1996.

Hai Tao, Harpreet S. Sawhney, and Rakesh Kumar. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, January 2002.

Luis F. Teixeira and Luis Corte-Real. Cascaded change detection for foreground segmentation. In *Proceedings of IEEE Winter Vision Meeting - Motion and Video Computing*, Austin, TX, February 2007.

Luis F. Teixeira and Luis Corte-Real. Video object matching across multiple independent views using local descriptors and adaptive learning. *Pattern Recognition Letters*, 30(2):157–167, January 2009.

Luis F. Teixeira, Jaime S. Cardoso, and Luis Corte-Real. Object segmentation using background modelling and cascaded change detection. *Journal of Multimedia*, 2(5):55–64, September 2007.

Luis F. Teixeira, Luis G. Martins, Mathieu Lagrange, and George Tzanetakis. MarsyasX: multimedia dataflow processing with implicit patching. In *Proceedings of ACM International Conference on Multimedia*, pages 873–876, Vancouver, BC, October 2008.

Luis F. Teixeira, Pedro Carvalho, Jaime S. Cardoso, and Luis Corte-Real. Automatic description of object appearances in a wide-area surveillance scenario. *IEEE Transactions on Circuits and Systems for Video Technology*, 2009. (submitted).

Luis F. Teixeira. Editing and description framework for video objects. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, July 2004.

Demetri Terzopoulos and Richard Szeliski. *Active vision*, chapter Tracking with Kalman snakes, pages 3–20. MIT Press, Cambridge, MA, 1992.

Mike J. Tomlinson, Martin J. Russell, and N.M. Brooke. Integrating audio and visual information to provide highly robustspeech recognition. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 2, pages 821–824, Atlanta, GA, May 1996.

Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedings of IEEE International Conference on Computer Vision*, volume 1, pages 255–261, 1999.

Nikolaus F. Troje, Cord Westhoff, and Mikhail Lavrov. Person identification from biological motion: effects of structural and kinematic cues. *Perception & Psychophysics*, 67(4):667–675, 2005.

Sofia Tsekeridou and Ioannis Pitas. Content-based video parsing and indexing based on audio–visual interaction. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(4):522–535, April 2001.

Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, Maui, HI, June 1991.

George Tzanetakis and Perry Cook. Multi-feature audio segmentation for browsing and annotation. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 103–106, New Paltz, NY, October 1999.

George Tzanetakis and Perry Cook. MARSYAS: a framework for audio analysis. *Organized Sound*, 3(4), 2000.

George Tzanetakis, Luis G. Martins, Luis F. Teixeira, Carlos Castillo, Randy Jones, and Mathieu Lagrange. Interoperability and the Marsyas 0.2 runtime. In *Proceedings of International Computer Music Conference*, Belfast, Ireland, August 2008.

George Tzanetakis. *Intelligent Music Information Systems: Tools and Methodologies*, chapter Marsyas: a case study in implementing Music Information Retrieval Systems, pages 31–49. Information Science Reference, 2008.

Akira Utsumi and Jun Ohya. Multiple-camera-based human tracking using non-synchronous observations. In *Proceedings of Asian Conference on Computer Vision*, pages 1034–1039, January 2000.

Giuseppe Valenzise, Luigi Gerosa, Marco Tagliasacchi, Fabio Antonacci, and Augusto Sarti. Scream and gunshot detection and localization for audio-surveillance systems. In *Proceedings of IEEE Conference on Advanced Video and Signal-based Surveillance*, pages 21–26, London, UK, September 2007.

M. Valera and S. A. Velastin. Intelligent distributed surveillance systems: A review. *IEE Proceedings: Vision, Image and Signal Processing*, 152(2):192–204, 2005.

Cor J. Veenman, Marcel J.T. Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, January 2001.

G.V. Veres, L. Gordon, J.N. Carter, and M.S. Nixon. What image information is important in silhouette-based gait recognition? In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 776–782, Washington, DC, June 2004.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Kauai, HI, December 2001.

Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2002.

Xia Wan and C.-C. Jay Kuo. A multiresolution color clustering approach to image indexing and retrieva. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 6, pages 3705–3708, Seattle, WA, May 1998.

J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, May 1994.

Ge Wang and Perry Cook. ChucK: A programming language for on-the-fly, real-time audio synthesis and multimedia. In *Proceedings of ACM International Conference on Multimedia*, pages 812–815, New York, NY, October 2004.

Yao Wang, Zhu Liu, and Jin-Cheng Huang. Multimedia content analysis using both audio and visual clues. *IEEE Signal Processing Magazine*, 17(6):12–36, November 2000.

Shiuh-Ku Weng, Chung-Ming Kuo, and Shu-Kang Tu. Video object tracking using adaptative Kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, December 2006.

Jutta Willamowski, Damian Arregui, Gabriella Csurka, Christopher R. Dance, and Lixin Fan. Categorizing nine visual classes using local appearance descriptors. In *Proceedings of Workshop on Learning for Adaptable Visual Systems in IEEE International Conference on Pattern Recognition*, August 2004.

Laurenz Wiskott, Jean-Marc Fellous, Norbert Kr®uger, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.

Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edition, 2005.

L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):774–780, August 2000.

Jeremy M. Wolfe. *Seeing*, chapter Visual Attention, pages 335–386. Academic Press, 2nd edition, 2000.

Cristopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

Matthew Wright, Adrian Freed, and Ali Momeni. Opensound control: State of the art 2003. In *International Conference on New Interfaces for Musical Expression (NIME'03)*, Montreal, Canada, 2003.

TingYao Wu, Lie Lu, Ke Chen, and Hong-Jiang Zhang. UBM-based real-time speaker segmentation for broadcasting news. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 2, pages 193–196, Hong Kong, April 2003.

Yi Wu, Edward Y. Chang, Kevin Chen-Chuan Chang, and John R. Smith. Optimal multimodal fusion for multimedia data analysis. In *Proceedings of ACM International Conference on Multimedia*, pages 572–579, New York, NY, October 2004.

Lexing Xie, Hari Sundaram, and Murray Campbell. Event mining in multimedia streams. *Proceedings of the IEEE*, 96(4):623–647, April 2008.

Gaungzheng Yang and Thomas S. Huang. Human face detection in complex background. *Pattern Recognition*, 27(1):53–63, 1994.

Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Face recognition: A literature survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, January 2002.

Alper Yilmaz and Mubarak Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, November 2004.

Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13.1–13.45, December 2006.

S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book (for HTK version 3.2)*. Cambridge University Engineering Department, 2002.

Kin Choong Yow and Roberto Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713–735, 1997.

Hua Yu and Jie Yang. A direct LDA algorithm for high-dimensional data - with application to face recognitiolda algorithm for high-dimensional data - with application to face recognitio. *Pattern Recognition*, 34(10):2067–2070, October 2001.

Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, August 1992.

Lihi Zelnik-Manor and Michal Irani. Event-based analysis of video. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 123–130, Kauai, HI, December 2001.

Lihi Zelnik-Manor and Michal Irani. Statistical analysis of dynamic actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1530–1535, September 2006.

DongQing Zhang and Shih-Fu Chang. Event detection in baseball video using superimposed caption recognition. In *Proceedings of ACM International Conference on Multimedia*, pages 315–318, Juan-les-Pins, France, December 2002.

Tao Zhao and Ram Nevatia. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1208–1221, September 2004.

Wenyi Zhao, Rama Chellappa, and Arvindh Krishnaswamy. Discriminant analysis of principal components for face recognition. In *Proceedings of IEEE International Conference on Face and Gesture Recognition*, pages 336–341, Nara, Japan, April 1998.

Wenyi Zhao, Rama Chellappa, P. Jonathon Phillips, and Azriel Rosenfeld. Face recognition: a literature survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.

Tao Zhao, Manoj Aggarwal, Rakesh Kumar, and Harpreet Sawhney. Real-time wide area multi-camera stereo tracking. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 976–983, San Diego, CA, June 2005.

Hua Zhong, Jianbo Shi, and Mirko Visontai. Detecting unusual activity in video. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 819–826, Washington, DC, June/July 2004.

Xiaoli Zhou and Bir Bhanu. Feature fusion of side face and gait for video-based human identification. *Pattern Recognition*, 41(3):778–795, March 2008.

Hanning Zhou and Don Kimber. Unusual event detection via multi-camera video mining. In *Proceedings of International Conference on Pattern Recognition*, volume 3, pages 1161–1166, Hong Kong, August 2006.

Wensheng Zhou, Asha Vellaikal, and C.-C. J. Kuo. Rule-based video video classification system for basketball video indexing. In *Proceedings of ACM International Conference on Multimedia*, pages 213–216, Los Angeles, CA, October 2000.

Xingquan Zhu, Xindong Wu, Ahmed K. Elmagarmid, Zhe Feng, and Lide Wu. Video data mining: semantic indexing and event detection from the association perspective. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):665–677, May 2005.

Xiaodan Zhuang, Xi Zhou, Thomas S. Huang, and Mark Hasegawa-Johnson. Feature analysis and selection for acoustic event detection. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 17–20, Las Vegas, NV, April 2008.

David Zicarelli. How i learned to love a program that does nothing. *Computer Music Journal*, 26(4):44–51, 2002.

Resources

Thesis

1. Author's page
   http://www.fe.up.pt/~lfpt
2. UVic internship
   http://www.fe.up.pt/~lfpt/UVic
3. Background Modelling and Subtraction
   http://www.fe.up.pt/~lfpt/BGMS
4. MarsyasX
   http://www.fe.up.pt/~lfpt/MarsyasX
5. MarsyasX SVN repository
   http://marsyas.svn.sourceforge.net/viewvc/marsyas/MarsyasX
6. VISNET II Network of Excellence
   http://www.visnet-noe.org/

Surveillance projects and applications

7. IBM Smart Surveillance System
   http://www.research.ibm.com/peoplevision/
8. iOmniscient
   http://www.iomniscient.com/

9. Keeneo
   http://www.keeneo.com/uk_index.php

10. DETEC
    http://www.detec.no/

11. 3VR
    http://www.3vr.com/

12. BRS Labs
    http://www.brslabs.com/

Multimodal-based projects

13. Semantic Multimodal Analysis of Digital Media – COST 292
    http://www.cost292.org/

14. M4 – MultiModal Meeting Manager
    http://www.dcs.shef.ac.uk/spandh/projects/m4/

15. AMI – Augmented Multi-party Interaction
    http://www.amiproject.org/

16. Oxygen Project
    http://oxygen.csail.mit.edu/

Generic libraries and frameworks

Visual

17. OpenCV
    http://www.intel.com/research/mrl/research/opencv

18. LTI-Lib
    http://ltilib.sourceforge.net

19. VXL
    http://vxl.sourceforge.net

20. Recognition And Vision Library (RAVL)
    http://ravl.sourceforge.net

Audio

21. Marsyas
    http://marsyas.sf.net/
22. CLAM
    http://www.iua.upf.es/mtg/clam
23. Pd – Pure Data
    http://puredata.info/
24. Chuck – Strongly-timed, Concurrent, and On-the-fly Audio Programming Language
    http://chuck.cs.princeton.edu/
25. Faust Signal Processing Language
    http://faust.grame.fr/
26. Synthesis ToolKit in C++ (STK)
    http://ccrma.stanford.edu/software/stk/
27. HTK Speach Recognition Toolkit
    http://htk.eng.cam.ac.uk/
28. SPro – Speech signal processing toolkit
    http://gforge.inria.fr/projects/spro
29. MAX/MSP
    http://www.cycling74.com/products/maxmsp

Multimodal

30. Graphics Environment for Multimedia (GEM) for Pd
    http://gem.iem.at/
31. Jitter for MAX/MSP
    http://www.cycling74.com/products/jitter
32. EyesWeb
    http://www.infomus.org/EywMain.html

Learning

33. LIBSVM – A Library for Support Vector Machines
    http://www.csie.ntu.edu.tw/~cjlin/libsvm
34. Weka Machine Learning Project
    http://www.cs.waikato.ac.nz/~ml

Other

☐ Local descriptors extraction

35. SIFT in C/MATLAB (David Lowe's reference)
    http://www.cs.ubc.ca/~lowe/keypoints/
36. SIFT in C (Krystian Mikolajczyk)
    http://www.robots.ox.ac.uk/~vgg/research/affine/
37. SIFT in C++ (Andrea Vedaldi)
    http://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html
38. Speeded Up Robust Features (SURF)
    http://www.vision.ee.ethz.ch/~surf

☐ Kalman and particle filtering

39. Kalman filter toolbox for Matlab
    http://www.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html
40. Recursive Bayesian Estimation Library (ReBEL)
    http://choosh.csee.ogi.edu/rebel
41. Sequential Monte Carlo methods software
    http://www-sigproc.eng.cam.ac.uk/smc/software.html

☐ Tracking

42. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker
    http://www.ces.clemson.edu/~stb/klt

☐ Biometrics authentication

43. Mistral – Open Source platform for biometrics authentification
    http://mistral.univ-avignon.fr/en/

Datasets and competitions

Visual

44. CAVIAR Test Case Scenarios
    http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

45. PASCAL Visual Object Classes (VOC) database
    http://www.pascal-network.org/challenges/VOC/

46. PETS: Performance Evaluation of Tracking and Surveillance
    http://www.cvg.rdg.ac.uk/slides/pets.html

Audio

47. Music Information Retrieval Evaluation eXchange (MIREX)
    http://www.music-ir.org/mirex/2007/index.php/Main_Page (MIREX 2007)
    http://www.music-ir.org/mirex/2008/index.php/Main_Page (MIREX 2008)

Multimodal

48. CUAVE Database – Audio-Visual Speech Processing
    http://www.ece.clemson.edu/speech/cuave.htm

Bio

Luis F. Teixeira was born in Porto, Portugal in April 20, 1979. During the first 17 years of his life, he lived and studied in a small city near Porto called S. Mamede de Infesta, where he completed high school at Escola Secundária Abel Salazar (Abel Salazar High School), named after a renowned Portuguese scientist and artist.

In 2001 he graduated in Electrotecnical and Computer Engineering (ECE) at Faculdade de Engenharia da Universidade do Porto (FEUP), the School of Engineering of the University of Porto. In 2004 he finished the M.Sc. in Networks and Communication Services, also at FEUP and soon after started his Ph.D.

Until late 2008 he was a researcher at INESC Porto, an R&D institute that acts as an interface between the academic and industrial worlds. His career at INESC Porto started in 2001 with a scientific investigation scholarship in the Telecommunications and Multimedia Unit (UTM). He joined the MOG group that was, at the time, actively working in the ORBIT project with BBC R&D. Since then he worked in other projects, namely ASSET, MetaVision and NUGGETS, all in the digital television and distributed systems areas. More recently he collaborated in the VISNET II Network of Excellence, where the scope of his work was on visual analysis applied to surveillance.

Currently he is a senior scientist at the Fraunhofer Portugal (FhP) Assistive Information and Communication Solutions (AICOS) research center. FhP AICOS began its activity in May 2008 and focuses on developing applied R&D solutions in the area of assistive Information and Communication Technologies (ICT) solutions that are able of lowering technical and financial barriers that usually hamper persons from using ICT and effectively participating in the Information Society.

He published several articles in international conferences and journals and was part of reviewing boards. He also actively contributes to Marsyas, an open-source project that provides a framework to develop audio (and more recently video) analysis algorithms and applications.

Curriculum Vitae

Personal information

Surname(s) / First name(s)	**Pinto de Almeida Teixeira, Luís Filipe**
Address(es)	Rua de Rebordãos, 193 – 4435-416 Rio Tinto, Portugal
Telephone(s)	+351 224860754 Mobile: +351 963953024 (preferable)
Email(s)	luis@hubz.net lfpt@fe.up.pt
Homepage(s)	http://luisteixeira.net
Nationality(-ies)	Portuguese
Date of birth	April 20, 1979 (Porto)
Gender	Male

Occupational field

Researcher in Computer Science

Work experience

Dates	Nov. 2008 – current
Occupation or position held	Senior Scientist
Main activities and responsibilities	Collaborated in industry, european and internal projects with management, and research and development responsibilities. Co-supervised projects by FEUP Master students.
Name and address of employer	Fraunhofer Portugal AICOS
Type of business or sector	R&D

Dates	Oct. 2006 – Dec. 2006
Occupation or position held	Visiting Researcher
Main activities and responsibilities	Collaborated in the Marsyas project with the orientation of Prof. George Tzanetakis.
Name and address of employer	University of Victoria, Victoria, BC, Canada
Type of business or sector	R&D

Dates	Sep. 2003 – Oct. 2008
Occupation or position held	Researcher with FCT scholarship
Main activities and responsibilities	Collaborated in both european projects and internal projects with research and development responsibilities. Co-supervised projects by FEUP Master students.
Name and address of employer	Fundação para a Ciência e Tecnologia (FCT)/INESC Porto
Type of business or sector	R&D

Dates	Sep. 2001 – Aug. 2003
Occupation or position held	Researcher
Main activities and responsibilities	Collaborated in both european projects and internal projects with research and development responsibilities.
Name and address of employer	INESC Porto
Type of business or sector	R&D

Dates	Mar. 2001 – Aug. 2001
Occupation or position held	Junior Researcher
Main activities and responsibilities	Collaborated in an internal project of the ARSIS group with development responsibilities.
Name and address of employer	INEB
Type of business or sector	R&D

Teaching experience

Dates	Oct. 2009 – current
Occupation or position held	Invited Assistant
Name and address of employer	Department of Informatics Engineering (DEI) of the Faculdade de Engenharia da Universidade of Porto (FEUP)
Type of business or sector	Education
Dates	Mar. 2004 – Jul. 2006
Occupation or position held	Teaching Assistant
Name and address of employer	Department of Electrical and Computer Engineering (DEEC) of the Faculdade de Engenharia da Universidade of Porto (FEUP)
Type of business or sector	Education

Education and training

Dates	Oct. 2004 – Nov. 2009
Title of qualification awarded	Ph.D. in Electrical and Computer Engineering.
Principal subjects/Occupational skills covered	Thesis Title: Contributions for the Automatic Description of Multimodal Scenes.
Name and type of organization providing education and training	Faculdade de Engenharia da Universidade do Porto (FEUP)
Dates	Mar. 2002 – Nov. 2004
Title of qualification awarded	M.Sc. in Computer Networks and Communication Services.
Principal subjects/Occupational skills covered	Title: Editing and Description Framework for Video Objects; Keywords: Multimedia authoring, content reuse, content repurpose, MPEG-4, multimedia content description, semantic content, MPEG-7
Name and type of organization providing education and training	Faculdade de Engenharia da Universidade do Porto (FEUP)
Level in national or international classification	*Muito Bom* (Very good, top grade), with a first-year classification of 18/20
Dates	Sep. 1996 – Jul. 2001
Title of qualification awarded	*Licenciatura* (5-year engineer degree) in Electrical and Computer Engineering.
Principal subjects/Occupational skills covered	Major in Telecommunications.
Name and type of organization providing education and training	Faculdade de Engenharia da Universidade do Porto (FEUP)
Level in national or international classification	17/20 (2^{nd} among 116 finalists)

Personal skills and competences

Mother tongue(s)	**Portuguese**
Other language(s)	English, French, Finnish

Self-assessment
European level(*)

	Understanding		Speaking		Writing
	Listening	Reading	Spoken interaction	Spoken production	
English	C2 Proficient user	C2 Proficient user	C2 Proficient user	C2 Proficient user	C2 Proficient user
French	B1 Independent user	B2 Independent user	A2 Basic user	A2 Basic user	A2 Basic user
Finnish	A2 Basic user	B1 Independent user	A2 Basic user	A2 Basic user	A2 Basic user

(*) *Common European Framework of Reference (CEF) level*

Technical skills and competences

Excellent knowledge of multimedia processing and coding technologies, especially in what refers to visual processing. Broad knowledge of the MPEG standards (incl. MPEG-4, MPEG-7 and MPEG-21). Good knowledge of IP technology, covering different layers: network and transport protocols (incl. IPv4/6, MPLS, SIP, TCP and UDP) and especially service protocols (incl. HTTP, RPC, RTCP, RTSP, XML-RPC, and SOAP). Good technical writing skills. Good simulation and modelling skills (incl.. UML modelling). Capacity to conduct autonomous self-motivated projects, as well as to manage a team of researchers and participate in large-scale projects.

Computer skills and competences

Experienced C/C++ programmer with 7+ years of development experience in R&D projects, as well as open-source projects. Also taught C++ programming classes associated to the Informatics Engineering course at FEUP. Developed a broad range of applications and modules, covering aspects from specification and implementation of multimedia framework architectures, implementation of distributed systems, algorithm-level implementation, GUI design to 3D programming. Practical knowledge of libraries and frameworks, including CORBA using ACE/TAO, XML using Xerces and other libraries, GStreamer, DirectShow, Qt, OpenGL, among others. Large experience (4+ years) using the Python programming language, developing smaller projects and bindings of C/C++ modules for higher-level integration. Experience also in web development using Zope, PHP and Javascript with associated technologies such as Ajax, jQuery and JSON. Knowledge of other programming languages, including Java, Perl, SQL and C#. Advanced user, with some administration experience, of the Windows, Linux, and Mac OSX operating systems. Knowledge of specific software tools, including MATLAB, Maple, SPSS, MySQL and LaTeX. Familiar with the software development cycle and with collaborative development tools.

Research

Publications

Luis F. Teixeira.
Contributions for the Automatic Description of Multimodal Scenes.
PhD thesis, Faculdade de Engenharia da Universidade do Porto, Portugal, 2009

Jaime S. Cardoso, Pedro Carvalho, Luis F. Teixeira, and Luis Corte-Real.
Partition-distance methods for assessing spatial segmentations of images and videos.
Computer Vision and Image Understanding, 113(7):811–823, July 2009

Luis F. Teixeira and Luis Corte-Real.
Video object matching across multiple independent views using local descriptors and adaptive learning.
Pattern Recognition Letters, 30(2):157–167, January 2009

Jaime S. Cardoso, Ricardo Sousa, Luis. F. Teixeira, and M. J. Cardoso.
Breast contour detection with stable paths.
Communications in Computer and Information Science, 25:439–452, 2008

Luis F. Teixeira, Luis G. Martins, Mathieu Lagrange, and George Tzanetakis.
"MarsyasX: multimedia dataflow processing with implicit patching.
In *Proceedings of ACM International Conference on Multimedia*, pages 873–876, Vancouver, BC, October 2008

George Tzanetakis, Luis G. Martins, Luis F. Teixeira, Carlos Castillo, Randy Jones, and Mathieu Lagrange.
Interoperability and the marsyas 0.2 runtime.
In *Proceedings of the International Computer Music Conference*, Belfast, Northern Ireland, August 2008

Daniel Duraes, Luis F. Teixeira, and Luis Corte-Real.
Building modular surveillance systems based on multiple sources of information – architecture and requirements.
In *International Conference on Signal Processing and Multimedia Applications*, pages 314–319, Porto, Portugal, July 2008

Filipe Coelho, Luis Baptista, Luis F. Teixeira, and Jaime S. Cardoso.
Automatic system for the recognition of amounts in handwritten cheques.
In *International Conference on Signal Processing and Multimedia Applications*, pages 320–324, Porto, Portugal, July 2008

Jaime S. Cardoso, Luis F. Teixeira, and Maria J. Cardoso.
Automatic breast contour detection in digital photographs.
In *Proceedings of International Conference on Health Informatics*, pages 91–98, Funchal, Portugal, January 2008

Luis F. Teixeira, Jaime S. Cardoso, and Luis Corte-Real.
Object segmentation using background modelling and cascaded change detection.
Journal of Multimedia, 2(5):55–64, September 2007

M. J. Cardoso et al.
Turning subjective into objective: The BCCT.core software for evaluation of cosmetic results in breast cancer conservative treatment.
The Breast, 16(5):456–461, October 2007

Mathieu Lagrange, Luis G. Martins, Luis F. Teixeira, and George Tzanetakis.
Speaker segmentation of interviews using integrated video and audio change detections.
In *Proceedings of International Workshop on Content-Based Multimedia Indexing*, pages 219–226, Bordeaux, France, June 2006

Luis F. Teixeira and Luis Corte-Real.
Cascaded change detection for foreground segmentation.
In *Proceedings of IEEE Winter Vision Meeting - Motion and Video Computing*, Austin, TX, February 2007

Luis F. Teixeira and Luis Corte-Real.
Integrated multimedia authoring and description framework.
In *Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services*, pages 265–268, Incheon, South Korea, April 2006

Luis F. Teixeira.
Editing and description framework for video objects.
Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, July 2004

Summary: 4 international journal articles, 9 international conference articles, 2 theses

Projects

IST NoE VISNET II (2006–2009) – Audiovisual systems and algorithms. Researcher/Ph.D. student. Research tasks: develop segmentation and tracking algorithm for visual surveillance systems.

IST NoE VISNET (2004–2005) – Audiovisual systems and algorithms. Sporadic contributions as Ph.D. student.

IST NUGGETS (2003–2004) – Distributed systems for digital television. Researcher and implementation responsibilities within a team of 2 elements. Development tasks: MXF SDK support.

IST ASSET (2003) – Distributed systems for digital television. Researcher and implementation responsibilities within a team of 3 elements. Development tasks: Metadata handling SDK in C++.

MXF SDK (2002) – Distribution formats for digital television. Implementation responsibilities within a team of 5 elements. Development tasks: MXF SDK implementation in C++.

IST METAVISION (2002–2003) – Distributed systems for digital television. Researcher and implementation responsibilities within a team of 3 elements. Development tasks: Distributed audiovisual file management system in C++.

ORBIT (2001–2002) – Video processing, distributed systems for digital television. Researcher and implementation responsibilities within a team of 15 elements. Development tasks: CORBA modules in C++ and video annotation GUI in JAVA.

Supervisory activities

– Luís António Alves Ferreira, M.Sc. FEUP, "Hyperbolic tree visualization on mobile devices", co-supervisor with Professor Ademar Aguiar, 2009.

– Daniel Filipe Martins Durães, M.Sc. FEUP, "Arquitectura de sistema de vigilância integrada" (Integrated surveillance system architecture), co-supervisor with Professor Luís Corte-Real, 2008.

– Guilherme Artur Conceição Capela, M.Sc. FEUP, "Reconhecimento de símbolos musicais manuscritos na framework Gamera" (Recognition of handwritten musical symbols in the framework Gamera), co-supervisor with Professor Jaime S. Cardoso, 2008.

– Filipe Emanuel Amaro Coelho, M.Sc. FEUP, "Sistema automático de reconhecimento do montante de um cheque" (Automatic system for the recognition of check amounts), co-supervisor with Professor Jaime S. Cardoso, 2008.

– Moisés Emanuel Adrega Medeiros, final year project FEUP, 'Descrição e manipulação de objectos para composição de multimédia" (Description and manipulation of objects for multimedia composition), co-supervisor with Professor Luís Corte-Real, 2005.

Invited presentations

– *Marsyas and MarsyasX*, University of Minho, March 2008

– *Object segmentation*, Laboratory for System Integration, Master in Electrical and Computers Engineering, University of Porto, December 2007

Reviewer of scientific articles

– International Conference on Image Analysis and Recognition (ICIAR)
– International Computer Music Conference (ICMC)
– Pattern Recognition Letters

Participation in conferences and other events

– ACM Multimedia 2008, Vancouver, BC, Canada, 27-31 October, 2008.
– IEEE Winter Vision Meeting - Motion and Video Computing (Motion 2007), Austin, TX, 23-24 February, 2007.
– Data and Models in Engineering, Science and Business, Parts I & II short summer course organized by the Massachusetts Institute of Technology Professional Institute (MIT-PI), Cambridge, MA, USA, 10-13 July, 2006.
– Summer School NN2005 on Neural Networks, Porto, Portugal, 4-8 July, 2005.
– International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2002), Padova, Italy, 19-21 June, 2002.

Additional information

References

Luís Corte-Real (Associate Professor at FEUP and Senior Researcher at INESC Porto) – supervised both the M.Sc. and Ph.D. theses
José Ruela (Associate Professor at FEUP and Coordinator of the Telecommunications and Multimedia Unit at INESC Porto) – coordinated research activities
George Tzanetakis (Assistant Professor at University of Victoria and Coordinator of the MISTIC research group) – supervised the internship at UVic

Honors and awards

Doctoral scholarship awarded by the FCT (portuguese foundation for science and technology), 2004-2008
Master's scholarship awarded by the FCT (portuguese foundation for science and technology), 2003-2004

Main contributions to open-source projects

Marsyas/MarsyasX [http://marsyas.sf.net] – Several core contributions to the Marsyas project, an audio processing library; main developer of MarsyasX, an evolution of Marsyas toward multimodal processing.

Last updated on November 15, 2009

Anonymous

...brick walls are there for a reason. The brick walls are not there to keep us out. The brick walls are there to give us a chance to show how badly we want something. Because the brick walls are there to stop the people who don't want it badly enough. They're there to stop the *other* people.

—Randy Pausch, 2007