FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Real-time Video Fusion for Surveillance Applications

André da Cunha Trancoso Ranito

MSc in Electrical and Computer Engineering

Supervisor: Prof.ª Maria Teresa Andrade

Second Supervisor: Eng. Luís Lucas

July 25, 2019

Resumo

A segurança e a protecção tornaram-se uma grande preocupação em quase todos os domínios do mundo actual. Aspectos como a prevenção e detecção de crimes ou vandalismo ou o controlo do fluxo de tráfego em áreas urbanas são especialmente abordados, mas é também utilizado na vigilância de locais remotos e em aplicações militares, como, por exemplo, na identificação e localização de forças inimigas.

A videovigilância tem vindo a ser utilizada há bastante tempo no sector militar para observar as actividades inimigas. Muitas vezes as imagens são capturadas em condições adversas que impedem a obtenção de imagens com qualidade suficientemente boa para compreender completamente a cena e identificar claramente os objectos nela contidos. Muitas vezes, para tentar superar tais problemas, a videovigilância é feita usando mais do que uma câmara de vídeo, equipada com diferentes tipos de sensores, cada um especialmente adequado a diferentes condições ambientais ou de visibilidade. Em aplicações de videovigilância militares, é cada vez mais comum o uso de uma combinação de sensores entre o infravermelho e o espectro total. No entanto, visto que esse tipo de sensores irão, provavelmente, capturar detalhes diferentes dependendo das condições de aquisição, não é possível decidir antecipadamente qual o *feed* que deve ser avaliado para garantir a vigilância ideal. Portanto, as imagens capturadas podem ser submetidas a um processamento adicional para extrair as informações que possam ser efetivamente relevantes para o objetivo desejado.

O tipo de processamento de imagens que normalmente é aplicado em aplicações de videovigilância pode ser amplamente classificado como processamento de alto nível e análise de cenas. O processamento de alto nível é aplicado aos sinais de vídeo capturados para melhorar a qualidade geral, nomeadamente em termos de detalhe oferecido. Entre outros, compreende técnicas para fundir eficientemente duas ou mais fontes de vídeo. A análise de cena inclui operações que extraem recursos das imagens das cenas de vídeo capturadas e as interpretam automaticamente, ou seja, em termos de objetos e atividades presentes na cena.

Esta dissertação explora as técnicas disponíveis para fundir imagens e propõe uma solução adequada para o ambiente militar, baseada na utilização de câmaras de vídeo de espectro visível e infravermelho. Os desafios abordados incluem a flexibilidade da solução para lidar com diferentes condições de aquisição, nomeadamente a presença de fumo ou nevoeiro nas cenas, e com a possibilidade de capturar imagens a distâncias variáveis. Outro aspeto importante que foi tido em consideração foi a possibilidade de se conseguir um funcionamento em tempo real.

Abstract

Safety and security have become a major concern in almost all real-world domains. Aspects such as crime or vandalism prevention and detection or traffic flow control in urban areas are especially addressed, but a strong focus is also placed on the surveillance of remote locations and on military applications, as for example, to identify and locate enemy forces.

Video surveillance has indeed been used for a long time in the military sector for observing enemy activities. Many times images are captured under adverse and variable conditions that prevent obtaining sufficiently good quality images to fully understand the scene and clearly identify the objects contained therein. Often, to try to overcome such problems, video surveillance is performed using more than one video camera, equipped with different types of sensors, each one especially suited to different environmental or visibility conditions. In military video surveillance applications, it is becoming commonplace to use a combination of sensors amongst infrared and full spectrum. However, given that these types of sensors will probably capture different details depending on the acquisition conditions, it is not possible to decide beforehand which feed should be used to ensure optimal surveillance. Therefore, captured images may undergo further processing to extract the information that might be effectively relevant for the desired goal.

The type of image processing that is normally applied in video surveillance applications can be broadly classified into high-level processing and scene analysis. High-level processing is applied to the captured video signals to enhance the overall quality, namely in terms of offered detail. Among others, it comprehends techniques to fuse efficiently two or more video sources. Scene analysis includes operations that extract features from the images of the captured video scenes and automatically interpret them, namely, in terms of objects and activities present in the scene.

This dissertation explores available techniques to fuse images and proposes a solution suitable for the military environment, based on the use of infrared and visible spectrum video cameras. Addressed challenges include the flexibility of the solution to cope with different acquisition conditions, namely the presence of smoke or fog in the scenes, and with the possibility of capturing images at variable distances. Another important aspect that was taken into consideration was the possibility of achieving real-time operation.

Acknowledgements

Esta dissertação marca o fim do meu percurso académico e, por isso, gostaria de agradecer a todos os que dele fizeram parte.

Inicialmente, gostaria de agradecer à instituição onde eu mais desenvolvi o meu conhecimento, onde alcancei inúmeras conquistas e, principalmente, onde me pude tornar engenheiro Electrotécnico e de Computadores. Um muito obrigado à FEUP por me ter proporcionado um ambiente de estudo e de trabalho exemplar.

Agradeço à Critical Software e a toda a sua equipa não só pelo estágio curricular que me propuseram, mas também pelas condições de trabalho excelentes que me concederam durante todo este processo.

Um especial agradecimento aos meus orientadores. A Professora Maria Teresa Andrade e o Engenheiro Luís Lucas revelaram-se fundamentais no acompanhamento e apoio dado durante o desenvolvimento deste estudo.

Um obrigado a todos os meus amigos e família que estiveram presentes nesta etapa importante da minha vida e que estiveram comigo durante os bons e os maus momentos.

Por fim, o maior agradecimento vai para os meus pais e para a minha irmã, a quem eu dedico todo este trabalho e todo o meu percurso académico, pois é graças a eles que eu me tornei na pessoa que sou e no engenheiro que me estou prestes a tornar. Muito obrigado.

André Ranito

"A stronger defense is an investment in peace."

Ronald Reagan

Contents

List of Figures

List of Tables

Abbreviations

CPU	Central Processing Unit
DWT	Discrete Wavelet Transform
GPU	Graphics Processing Unit
IR	Infrared
MR	Multi-Resolution
PSNR	Peak Signal to Noise Ratio
RMSE	Root Mean Squared Error
SSIM	Structure Similarity Index Measure
VIS	Visible

Chapter 1

Introduction

1.1 Context

This document covers the work done throughout my Master thesis dissertation. It focuses on the study and development of a real-time video fusion algorithm to be applied in the field of surveillance. The work conducted during the dissertation was proposed by the software development company Critical Software.

Critical Software is a Portuguese software company delivering solutions for mission and business critical systems. This company works on areas such as Aeronautics, Space, Defense, Railway, Telecommunications, Energy, Financial and Health. The scope of this dissertation falls within their Defense systems department mainly in the field of safety and security. In this sector this company has done projects in safety-critical systems, earth observation, embedded cyber security, verification & validation and systems integration for some well known organizations such as the Portuguese and Irish armed forces, for instance.

Safety and security have become a major concern in almost all real-world domains. In a civil perspective problems such as crime, vandalism and terrorism are some of the main issues for societies to deal with. To help addressing this, surveillance can be used in order to have more control of what is happening in the vicinities, which can help to detect and predict such events and, in the unfortunate case the occurrence already happened it can help to arrest the culprits. In a military domain, surveillance can have a strong impact as well. It can be used in remote locations to identify and locate enemy forces, as well as to help distinguish civilians from the targets.

Video surveillance consists in monitoring the behaviour and the activities of people or locations, usually with the purpose of assuring safety and security. Given this, video surveillance plays an important role in the lives of citizens as well as soldiers. Many times, the images are captured under adverse and variable conditions that may affect the quality of the collected information, making it impossible to clearly identify all the objects contained in the scene and fully understand what is happening. One solution to this problem is to use more than one video camera, each containing different types of sensors, each one collecting information in different wavelength bands of the electromagnetic spectrum. For instance, using an infrared sensor alongside a visible light

sensor is becoming quite common, allowing to collect more spectral information of the surroundings, thus getting a better perspective. However, given that these different types of sensors will most likely collect different details depending on the environment conditions, the different types of images being collected must be analyzed independently in order to fully perceive the situation.

This dissertation explores the idea of processing those different images collected by the various sensors and merging the details into a single image containing all the relevant information included in them, performing image fusion. Different techniques to perform image fusion were analyzed and a solution for the military environment, based on the use of infrared and visible spectrum video cameras was proposed. Some of the objectives consisted in the flexibility of the solution to cope with different acquisition conditions, for instance the presence of smoke or fog, the possibility of capturing images at variable distances and the possibility of the system to work in real-time.

The work developed in the scope of this dissertation can be of great use to Critical Software, since they can implement the system in many of their Defense projects. Moreover, this technology can also be adapted to other areas such as remote sensing, medical science and space, for instance, which can also be of great interest to the company.

1.2 Motivation and Objectives

The use of image fusion in Surveillance delivers a better and more complete description of the sensors' surroundings. For instance, it can be used to detect hidden weapons or in the identification, detection and tracking of targets. A good example of such application is the detection of people or vehicles in low visibility conditions, such as smoke or fog. This can be done by combining the information from visible light and infrared light sensors, for instance.

Another great advantage of this technology is the ability to reduce the amount of data necessary to describe the situation. This can be useful, for example, in a case where the sensors are attached to a vehicle, such as an Unmanned Air Vehicle - UAV. In this example the bandwidth of the ground connection may not be large enough to stream all the data being collected by the vehicle's sensors in real-time. By combining all the important information into a single image it is then possible to stream all the crucial details being collected by the sensors in a real-time manner.

The main objective proposed for this dissertation was to develop a video fusion solution suitable for use in military environments, or more generically, that would be applicable to situations where infrared and full spectrum video cameras would be simultaneously used to capture the same scenes. The methodology adopted to achieve such goal, included an initial study on the currently available techniques to fuse still images and the selection of those that would be more adequate for the scope of the work. The following step addressed the adaptation of the selected technique to video fusion. Finally, to address real-time requirements, a C++ implementation of the developed algorithm was accomplished.

1.3 Document Structure

Chapter 2 contains a description of the initial study done on the state of the art. This study led to a better understanding of the image fusion subject and, based on this information, it was possible to define a starting point for the work later done.

Chapter 3 starts by describing the selection over the different types of algorithms identified in chapter 2, which allowed to define which ones would proceed to the later stages of the study. Afterwards it describes the methodology of the developed algorithm which consists in assigning weights to both infrared and visible images. These weights then allowed to perform a weighted average between the two images.

The results and comparisons between the selected approaches is done in chapter 4. For this purpose, some evaluation methods described in chapter 2 were implemented in order to achieve such comparisons.

Chapter 5 is divided into two main parts. The first shows the work done to implement the developed algorithm and to adapt it to frame by frame video fusion. The latter highlights the work done within the scope of optimizing the system, as well as a study on its real time execution.

At last, chapter 6 contains the conclusion of this document. It shows a recap of all the work done and an analysis over the initial objectives of the internship proposed by the company. Finally, it shows an evaluation of the future work to be done in the scope of this project.

Chapter 2

Bibliographic Review

This chapter presents a study done on the state of the art of image fusion and it is divided in 4 main sections.

Starting by section 2.1 which provides a brief introduction on the theme and its applications.

Section 2.2 describes and compares the different levels in which various fusion techniques can be applied.

Section 2.3 characterizes some techniques used in image fusion. It focus in two types of methods, Pixel-level (2.3.1) and Region-based (2.3.2). It also explains in which case each technique type is more adequate.

Lastly, section 2.4 describes appropriate performance metrics that are commonly used to assess the success of image fusion techniques.

2.1 Image Fusion

With increased affordability and improvements in computing power, image fusion has started to become a feasible solution to help in many image analysis challenges. This technique merges different images into a single one containing enhanced information that is more suitable for human or machine perception when compared to the analysis of all the individual input images. Image fusion can also increase the signal-to-noise ratio, the robustness and the reliability of the system [9]. This process should also [7]:

- Preserve all relevant information in the fused image.
- Suppress irrelevant parts of the image and noise.
- Minimize any artifacts or inconsistencies in the fused image.

J. J. Lewis et al. [7] state that numerous imaging sensors have been developed, however, there are many scenarios where no sensor will give the complete picture. For example, while images taken in the visible spectrum tend to have good contextual information they do not easily show up targets if they are camouflaged or in low light conditions. Thus, redundant information from a

set of two or more different sensors can be fused together to produce a single fused image which preserves all relevant information from the original data.

Mostly used in a (a) Multimodal/Multisensor environment, image fusion can also be used for (b) Multifocus, (c) Multitemporal and (d) Multiview purposes as stated in [4]:

(a) In multisensor fusion, as the name implies, more than one type of sensor is used to collect information about the scene. Usually different sensor modalities (operating at different wavebands, ex. IR and visible light sensors) are used in order to obtain higher spectral data. Such data is then merged to output a single image containing information from all the different wavebands collected by the sensors, thus providing a more complete and informative view of the scene (Fig. 2.1).

(b) A Multifocus approach, as seen on Fig. 2.2, can be used in a situation where it is impossible to contain all the important objects in focus. A possible solution using image fusion can be collecting multiple images, each containing one object in focus, and then merging them in a way resulting in a final image with all objects focused.

(c) Multitemporal fusion uses images taken at the same scene but at difference instances to detect changes in the environment.

(d) Multiview combines images obtained at different view points, allowing for higher resolution representations and also the possibility for a 3D representation of the scene.

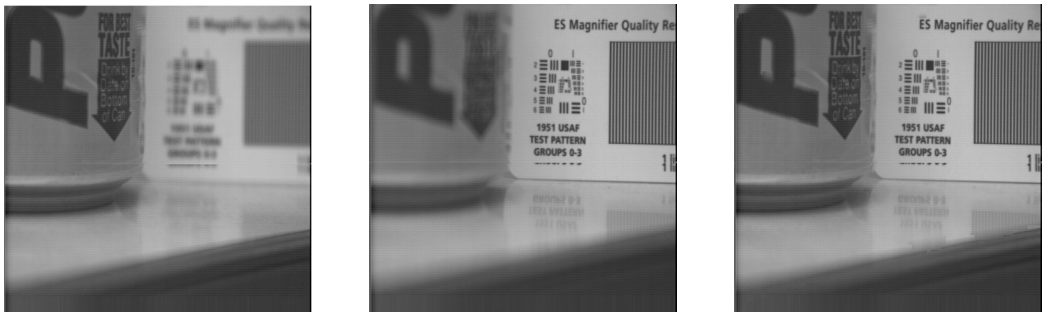
(a) Visible light image (b) IR image (c) Fused image

Figure 2.1: Multifocus image example (VIS+IR) [1].

2.1.1 Applications

Image fusion can have many applications. In a military environment it can be used as navigation guidance, object and target detection and recognition, surveillance, concealed weapon detection, tactical situation assessment and awareness [9][10][11].

It can be used in remote sensing which consists in capturing images of the earth's surface either from a plane or a satellite [12]. A high resolution, yet low spectral sensor (usually gray scale images) known as a panchromatic sensor is usually fused with a high spectral sensor providing

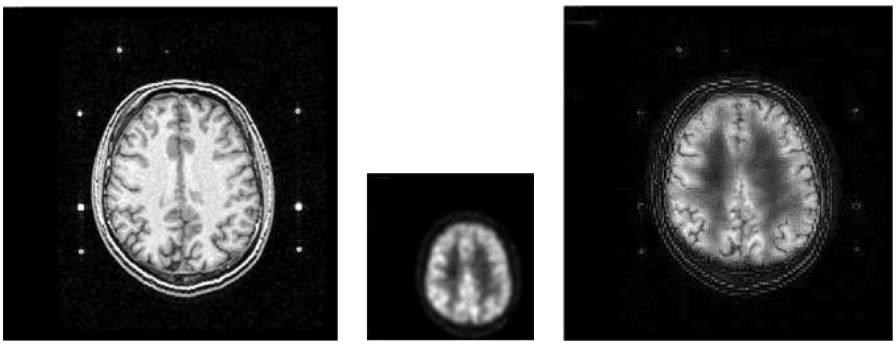


(a) Left focused image (b) Right focused image (c) Fused image

Figure 2.2: Multifocus image example [2].

color details in low resolution. The outcome is an image with high spectral information and high resolution.

Medical science applications are also of high interest since image fusion allows to merge the information from different exams each one containing separate and useful information [13]. For instance, in Fig. 2.3, a fusion between a MRI (Magnetic Resonance Imaging) and a PET (Positron Emission Tomography) was done to take advantage of the high spatial resolution of the MRI image and the low resolution, yet high functional information provided by the PET.



(a) (b) (c)

Figure 2.3: (a) MRI image, (b) PET image, (c) Fusion between both exams. [3]

Techniques in computer vision, industrial applications and robotics are also employed to allow for better understanding of the situation surrounding the sensors.

Despite being mostly applied and developed in a military environment, image fusion is starting to get used in civilian applications. Some of which get adapted from military developed systems and others are developed entirely for civilian purposes. This comes to show that this is a fast growing subject with a vast amount of uses and applications.

2.2 Fusion Levels

Image fusion can take place at many levels of abstraction. Starting from the lowest there is (a) pixel-level, (b) feature-level and (c) decision-level fusion [12] (Fig. 2.4). Abidi and Gonzalez in [14] also consider an even lower level of abstraction, the signal-level fusion, where the sensors' outputs are mixed in the signal domain. However such abstraction level is usually not used therefore it can be left aside.

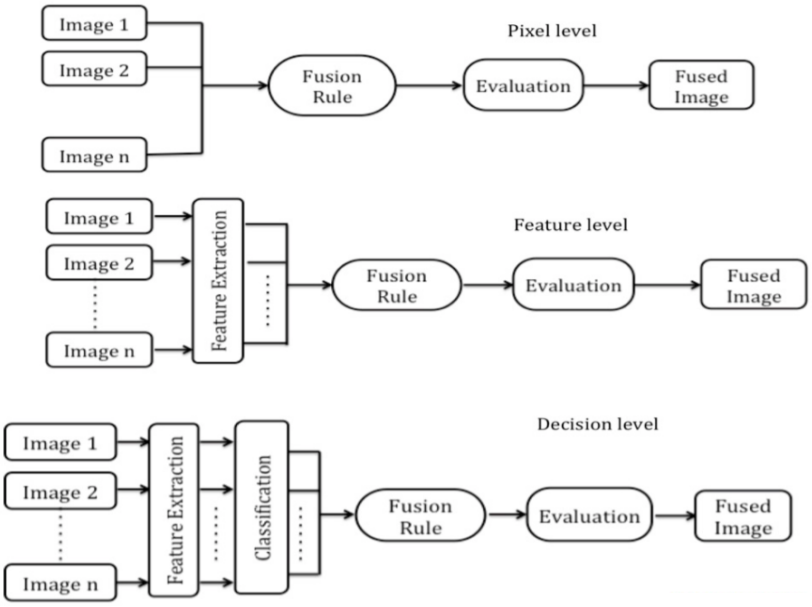


Figure 2.4: Fusion Levels [4]

(a) In a Pixel-level fusion technique the algorithm works directly or indirectly with the values of the pixels coming from the sensors. This level of abstraction merges the totality of the input images and allows to use either spatial or frequency domain algorithms [9]. A detailed explanation of such techniques is done at subsection 2.3.

(b) Feature-level fusion requires the extraction of characteristics from the input images using a segmentation algorithm. These features are later assigned to the respective features in the other input images and then merged to form the final fused image. Statistical approaches are usually employed to perform the fusion [12].

(c) In a Decision-level approach the individual input images are processed by doing a feature-extraction followed by a classification of such features. The information is fused using decision rules that have as input the object classifications calculated before.

The main advantages of using pixel-level techniques is its high temporal efficiency and easy implementation. On the other hand region-based (feature-level and decision-level) algorithms can be much more intelligent due to more complex fusion rules that are based on actual features of the

image rather than single or arbitrary groups of pixels [7]. However these methods will have higher performance costs since they will require either more time to execute or greater computing power. A trade-off between quality of the output and the computing power necessary is found here and it must be considered when choosing the algorithm to implement the fusion.

2.3 Fusion Techniques

This section covers some of the current image fusion techniques as well as the different types of available methods. Such techniques are divided into spatial and transform domain for pixel-based fusion as well as region-based techniques for feature-level and decision-level techniques. Examples of images are given for each technique as well as a comparison between all the methods in the different domains.

2.3.1 Pixel-level techniques

As mentioned in section 2.2, pixel-level techniques work on the totality of the input images and allow to fuse the values directly or indirectly. Such techniques can be defined as Spatial domain and Transform domain, respectively, and are further detailed in subsections 2.3.1.1 and 2.3.1.2.

2.3.1.1 Spatial Domain

Spatial domain algorithms consist in combining the pixel values in a linear or non-linear matter. The simplest algorithm in image fusion is in this domain and is known as **Averaging**. This method consists on computing the average (weighted or non-weighted) of all the input images as shown in Eq. (2.1).

$$F(x,y) = \alpha_1(I_1(x,y)) + \alpha_2(I_2(x,y)) + ... + \alpha_N(I_N(x,y)) \tag{2.1}$$

The α values are the respective weights for each input image where $\alpha_1 + \alpha_2 + ... + \alpha_N = 1$ and for non-weighted averaging $\alpha_1 = \alpha_2 = ... = \alpha_N = \frac{1}{N}$.

Select maximum is another fusion method that belongs in the spatial domain. It consists in choosing the highest pixel value from the input images using Eq. (2.2)

$$F(x,y) = max(I_1(x,y), I_2(x,y), ..., I_N(x,y)) \tag{2.2}$$

Another method known as **Color fusion** allows the merging of two or three different grey scale images. This process takes advantage of the RGB representation of an image by assigning each grayscale input image to a specific color plane in the RGB matrixes of the final image. An example of color fusion is described by J. J. Lewis et al. in [7] where a visible image can be assigned to the green part of the final image and the IR image to the red part. In the fused image objects form the visible spectrum (such as foliage) will appear green while hot objects from the IR image will appear red.

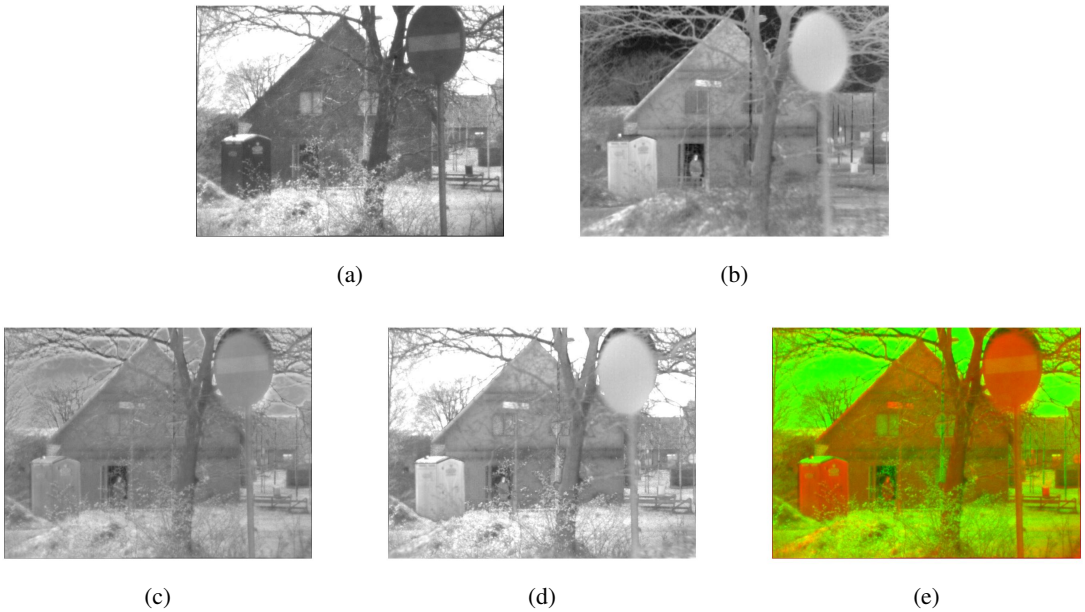


Figure 2.5: Example of spatial domain fusion techniques. (a) Visible image, (b) IR image, (c) Fusion using Average algorithm (non-weighted), (d) Fusion using Select Maximum algorithm, (e) Fusion using the Color fusion algorithm.

Due to the simplicity of its algorithms, spatial domain techniques have the advantage of requiring the least amount of computing power in between all the different fusion methods. Such algorithms can be useful for systems containing low computing power and/or requiring high temporal constraints. However the output of spatial domain algorithms can be quite poor in terms of quality. For instance, by analyzing Fig. 2.5, the Averaging method outputs an image with low contrast as seen in (c) where the sky is much darker than it actually was. The Select Maximum algorithm can hide some of the images' features, for example the "No Entry" sign is completely obscured in the output image (d). The color space fusion (e) is not ideal as well since it can introduce some misconceptions given the fact that the colors in the final image are not real.

For systems requiring higher quality fusions, other types of approaches must be considered. Transform domain or Region-based algorithms, covered in 2.3.1.2 and 2.3.2 respectively, can provide an excellent solution, however at higher costs (computing power and temporal).

2.3.1.2 Transform Domain

Another form of performing image fusion includes working on a different domain. In general, this is achieved by applying a Multi-resolution transformation (Ψ) to the input images decomposing each of the images in different coefficients, as seen in Fig. 2.6. These coefficients are then combined using a fusion rule (ϕ) in order to obtain the coefficients of the final fused image. To complete the process, the inverse transform operation (Ψ^{-1}) is performed to obtain the final image back in the spatial domain. This process is shown in Eq. (2.3).

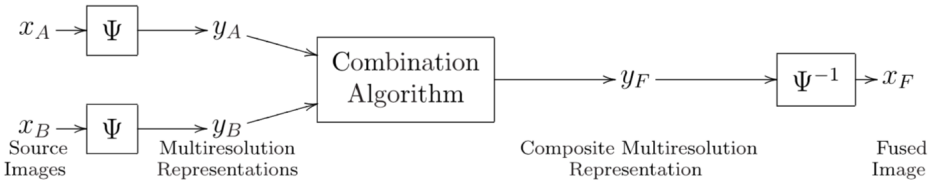


Figure 2.6: Multi-resolution image fusion scheme [5].

In image fusion the source images are usually transformed into another domain using a Multi-resolution decomposition (MR). This decomposes the signal being analyzed into several components, each of which captures information present at any given scale [5]. Pyramid and Wavelet approaches are examples of such MR transformations and are described below.

$$F = \Psi^{-1}(\phi(\Psi(I_1), \Psi(I_2), ..., \Psi(I_N))) \tag{2.3}$$

Pyramid-based Techniques

Some of the first multi-resolution techniques developed are pyramid-based. These algorithms try to extract the features from an image at several decomposition levels allowing for a more complete analysis of said image.

The simplest pyramid algorithm is known as the *Gaussian Pyramid*. This technique starts by applying a convolution between the source image and a kernel followed by a sub-sampling in the horizontal and vertical domain [6]. This is known as a *Reduce* operation and can be represented by Eq. 2.4.

(a) (b) (c) (d)

Figure 2.7: Gaussian pyramid example (3 reductions) [6].

$$G_{k+1}(x,y) = \sum_{u=-p}^{p} \sum_{v=-p}^{p} K(u,v) * G_k(2x+u, 2y+v) \tag{2.4}$$

where G_k and G_{k+1} are the source and the new down-sampled image respectively and $K(u,v)$ is the kernel used (usually 3x3 or 5x5). This process can be repeated on the new image to obtain another down-sampled image, and so on until the desired level of decomposition is reached and a pyramid of images (G_1, G_2, ..., G_N) have been generated (Fig. 2.7).

The inverse operation, known as *Expand* (E_k) is used to obtain a higher level image from a previously down-sampled one. This operation works by duplicating each row and column in G_{k+1} and by performing a convolution between the result and the Gaussian Kernel K, and can be represented in Eq. 2.5.

$$E_k(x,y) = \sum_{u=-p}^{p} \sum_{v=-p}^{p} K(u,v) * G_{k+1}\left(floor\left(\frac{x+u}{2}\right), floor\left(\frac{y+v}{2}\right)\right) \tag{2.5}$$

By performing the difference between images G_k and an expanded version of the same image from a lower level, E_k, a new image is created showing the high frequency details in the k^{th} level of the Gaussian pyramid. This process can be described as: $L_k = G_k - E_k$ and it allows the calculation of the Laplacian pyramid. This is a pyramid of N levels $(L_1, L_2, ..., L_N)$ containing the band-pass information of the image at different resolutions (Fig. 2.8).

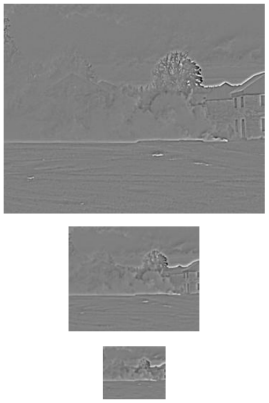


Figure 2.8: Laplacian Pyramid showing band-pass details of image in 2.7 [6].

The image fusion process starts by calculating the Gaussian and subsequently the Laplacian pyramids for the input images. The Laplacian images (L_k) from each source can be merged by comparing each respective pixel and selecting the one with the highest absolute value. The lowest resolution images from the Gaussian pyramids are also merged using a fusion method. This is known as the Base Fusion and can be done with any fusion algorithm, such as averaging, for instance. The fused image is then obtained performing some Expansions and Additions to the merged Laplacian and Gaussian images. The procedure can be exemplified in Fig. 2.9.

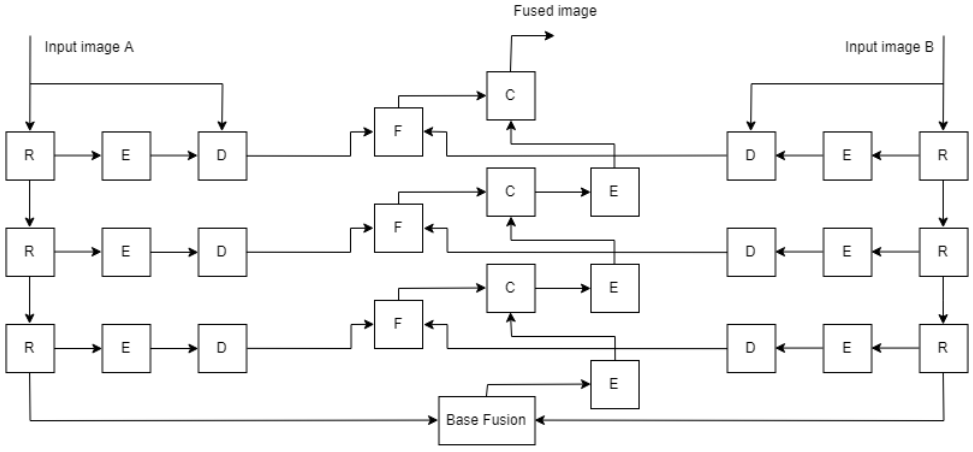


Figure 2.9: Pyramid fusion scheme [6]. R→ Reduce, E→ Expand, D→ Difference, C→ Combine.

Wavelet Transform

Introduced in the 1980s, Wavelet Transforms became a useful tool in the image fusion domain. Besides such application it is also used in image coding, pattern matching and fractal analysis [6].

The Discrete Wavelet Transform (DWT) is a well-known and a well developed variation of the Wavelet Transform and is greatly used in image processing.

The DWT works in a similar way as the Fourier transform. It provides an alternative way of representing data by changing the domain in which it is being represented. Unlike the Fourier Transform, which uses sinusoidal functions to represent a signal, the DWT uses variations of a function know as the *mother wavelet* (Fig. 2.10). A number of wavelet functions are then dilated and translated in order to fit the original signal. This representation is ideal to reproduce signals containing non-uniformities since it is well localised in terms of spatial and spectral frequencies.

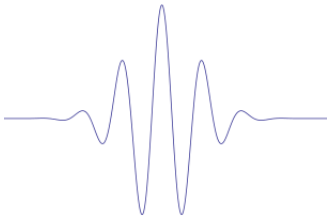


Figure 2.10: Mother wavelet example.

The following equation (2.6) describes how the original signal ($f(x)$) is represented using variations of the *mother wavelet* function [6].

$$f(x) = \sum_{p,q} c_{p,q} \psi_{p,q}(x) \tag{2.6}$$

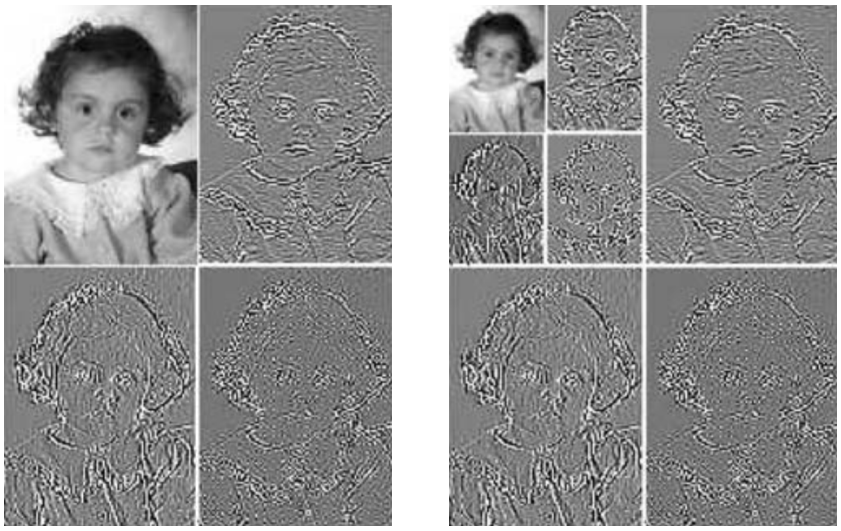
where $\psi_{p,q}(x)$, as stated before, are variations (dilations and translations) of the original *mother wavelet* function $\psi(x)$. This variation is done by following the given equation:

$$\psi_{p,q}(x) = 2^{-p/2} \psi(2^{-p}x - q) \tag{2.7}$$

where integers p and q are the scale index and the location index, respectively. These variables define the width and the position of each wavelet function that represent the original signal [6]. A more detailed description of such technique containing a full mathematical analysis can be found at [3].

The previous paragraph described the process of the DWT in one dimension. To apply such technique to an image, a 2D transform must be used. To convert it to a 2D transform, successive applications of the 1D transform are made, in both the horizontal and the vertical directions. After applying the DWT to an image four different bands of coefficients are generated, the Low-Low (LL), Low-High (LH), High-Low (HL) and High-High (HH). The LH, HL and HH represent the horizontal, vertical and diagonal details, respectively, while the LL represents a smooth subimage corresponding to the low-frequency details of the original image [3].

A multi-level DWT can also be applied to obtain a better multiscale result, since different levels will reveal details of different sizes. In Fig. 2.11a a representation of a one-level DWT is shown. The top left-hand corner refers to the LL band, the top right-hand corner to the LH band revealing the horizontal details, the bottom left-hand corner to the HL revealing the vertical details and last, the bottom right-hand corner to the HH band revealing the diagonal details. This process can be continued by applying another DWT to the LL band. This is show in Fig. 2.11b where the LL band from the first DWT was decomposed revealing another set of LH, HL and HH bands. This procedure can be applied over and over to obtain the level of detail necessary.



(a) (b)

Figure 2.11: Representation of a (a) one-level and a (b) two-level DWT image decomposition [3].

After applying the DWT to an image a set of coefficients is obtained. Image fusion can be done using the DWT by merging these coefficients with the ones from other images by using a fusion rule. Fig. 2.12 shows a generic scheme of how this process can be done, where W is the transform operation (DWT), ω the fusion rule and W' the inverse transform of the DWT.

Smith and Moira in [6] state that the DWT generally produces good results in image fusion and is also computationally efficient. However, it is not shift-invariant because of the sub-sampling process used in its calculation. To address this problem, Rockinger, in 1997, discarded the down-sampling process creating a shift invariant discrete wavelet transform (SIDWT) [15]. Despite resulting in a visibly better fused imagery, the SIDWT was a lot more computationally expensive than the DWT. This approach was further developed by Bull et al. in 2002 with the dual-tree complex wavelet transform (DT-CWT) [16].

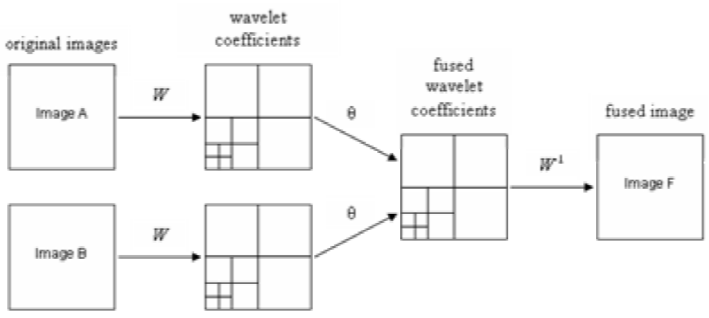


Figure 2.12: Discrete Wavelet Fusion scheme [6].

2.3.2 Region-Based Algorithms

Systems that apply fusion techniques are usually interested in some of the features from the input images instead of the totality of the information. However, pixel-based algorithms merge all the data coming in from the sensors. This can lead to the creation of artifacts or inconsistencies that may mislead the human observer or any posterior processing unit.

Another approach, know as Region-Based techniques can be the solution to this problem. This feature/decision-level methods start by detecting all the salient features of the input images. The fusion is then performed on such features, generating a final image containing only the relevant information from each of the inputs. Besides being more robust, this approach also avoids the well-known problems in pixel-level fusion such as blurring effects and high sensitivity to noise and misregistration [5].

Accoring to J. J. Lewis et al. in [7], region-based approaches allow for:

- **Intelligent fusion rules**: Fusion rules are based on combining groups of pixels which form the regions of an image. Thus, more useful tests for choosing the regions of a fused image (...) can be implemented.

- **Highlight features**: Regions with certain properties can be either accentuated or attenuated in the fused image depending on a variety of the region's characteristics.

- **Reduced sensitivity to noise**: Processing semantic regions rather than individual pixels or arbitrary regions can help overcome some of the problems with pixel-fusion methods such as sensitivity to noise, blurring effects and misregistration [17].

- **Registration and video fusion**: The feature information extracted from the input images could be used to aid the registration of the images. Region-video fusion schemes could use motion estimation to track the fused features, allowing the majority of frames to be quickly predicted from some fully fused frames.

2.3.2.1 Segmentation process

The first step of a region-based algorithm is the segmentation or feature detection. This consists in the detection of salient information that can later be related and fused with features in other images.

Any feature detection algorithm can be used in this part. However, it can greatly affect the next stages of the image fusion process. An over-segmentation algorithm will lead to an excessive amount of information in the final image. On the other hand an under-segmentation method can cause lack of information in the final image. So a balanced feature detection algorithm must be selected to provide a good quality fusion. In [7] a MR dual-tree complex wavelet transform (DT-CWT) based segmentation algorithm was applied (Fig. 2.13), while in [5] a linked pyramid MR decomposition was used to detect the images' features.

In Fig. 2.13 the authors firstly transform the N registered images $I_1, I_2, ..., I_N$ using ω (DT-CWT). The transform outputs a set of coefficients $D_{n(\theta,l)}$ for each image, I_n. Any given set of coefficients consists in six sub-bands, θ, for each level of decomposition, l. The combination of the images' intensity values, I_n, and the coefficient values, D_n using a segmentation algorithm, $\sigma(I_n, D_n)$ will output a list of all the T_n regions for each image: $R_1, R_2, ..., R_N$, where $R_n = \{r_{n,1}, r_{n,2}, ..., r_{n,T_n}\}, n \in N$. This was a solution presented in [7] to segment the input images allowing for the next stage, the fusion process.

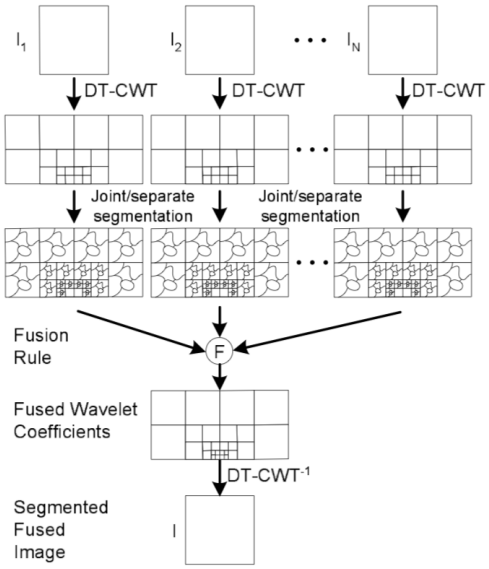


Figure 2.13: Region-based image fusion scheme using the DT-CWT [7].

2.3.2.2 Priority calculation and fusion rules

After the segmentation process the regions detected must then be merged. This operation has to follow some ruling in order to obtain the best result. Priorities are usually calculated for the regions detected in each image and then the merging is done following these said priority values. J. J. Lewis et al. in [7] used three different methods to calculate the regions' priorities. The first one is based on the average energy of the wavelet coefficients in said region. This is usually a good measure of the importance of a region and it is done by using Eq. (2.8), where $|r_n|$ is the size of the region calculated with the detail coefficients $d_{n(\theta,l)}(x,y)$.

$$P(r_{t_n}) = \frac{1}{|r_{t_n}|} \sum_{\forall\theta,\forall I,(x,y)\in r_{t_n}} \left|d_{n(\theta,l)}\right| \tag{2.8}$$

The variance of the wavelet coefficients over a region could also be used as a priority:

$$P(r_{t_n}) = \frac{1}{|r_{t_n}|} \sum_{\forall\theta,\forall I,(x,y)\in r_{t_n}} (d_{n(\theta,l)}(x,y) - \overline{d}_{n(\theta,l)})^2 \tag{2.9}$$

where $\overline{d}_{n(\theta,l)}$ is the average wavelet coefficient of the region r_{t_n}.

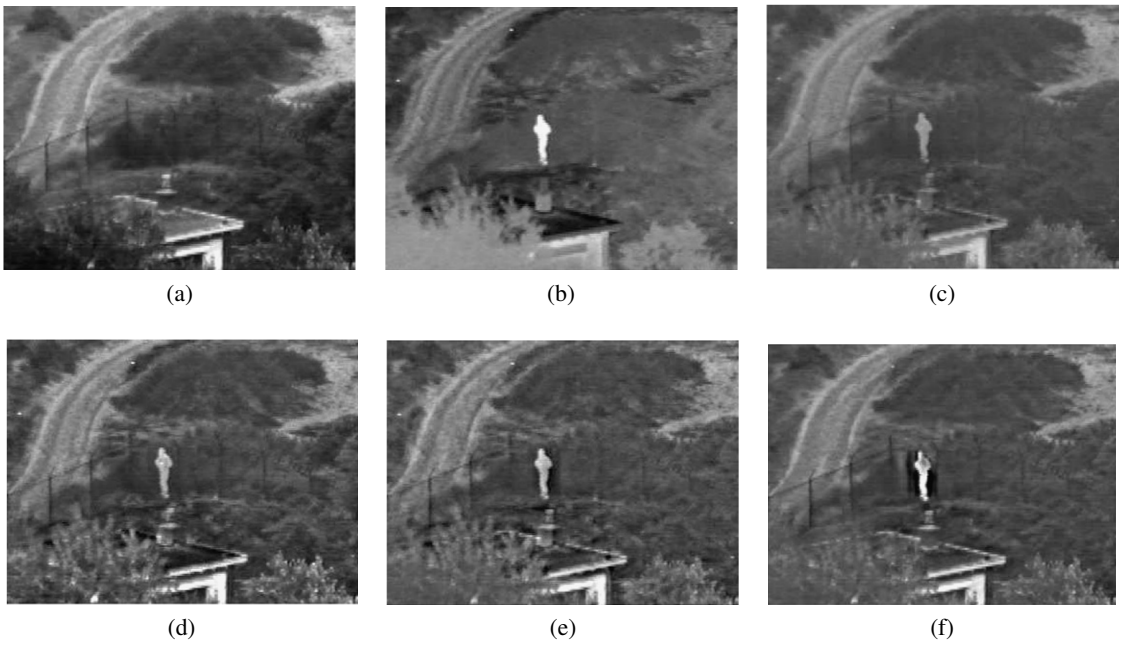
Another form of priority calculation could be based on the entropy of the wavelet coefficients. The normalized Shannon entropy of a region can be calculated as such:

$$P(r_{t_n}) = \frac{1}{|r_{t_n}|} \sum_{\forall\theta,\forall I,(x,y)\in r_{t_n}} d^2_{n(\theta,l)}(x,y) \log d^2_{n(\theta,l)}(x,y) \tag{2.10}$$

Other methods could be used to calculate the priority map, for instance an approach based on the geometry of the regions, such as size, shape and position. In this case smaller regions should be given higher priority since bigger regions are more likely to contain background information.

After calculating the priority maps, the fusion can simply be achieved by merging the pixel information following the respective regions' priorities.

As mentioned in section 2.2, region-based approaches can output a higher quality image at the cost of a higher computing power necessity. A prior study of the problem must be done in order to understand which approach is better for the system being developed. If computing power and/or time constraints are flexible enough a better quality fusion using a region-based approach can be done. Otherwise a pixel-level based approach should be the correct approach.



(a) (b) (c)

(d) (e) (f)

Figure 2.14: Comparison of image fusion techniques [7]. (a) Visible image, (b) IR image, (c) Fusion using an Averaging technique, (d) Fusion using a MR Pyramid technique, (e) Fusion using a DWT technique, (f) Fusion using a Region-based technique.

Fig. 2.14 shows an example of the difference between most of the methods covered in this section (2.3). As stated before the averaging method does not show the best result as a lot of contrast seems to be lost after the process. Both multi-resolution methods (Pyramid and Discrete Wavelet Transform) seem to have a good result even though the DWT looks a bit sharper. The Region-based approach seems to have the best result however, as stated before, at the cost of higher complexity.

2.4 Performance evaluation

As previously stated there are many different techniques to perform image fusion. The quality of each one may differ depending on the application. Therefore some evaluation metrics should be used to determine which is the best approach to follow.

In [4] Kalaivani and Phamila did an analysis of different image fusion techniques and to compare them they selected 6 quality assessment metrics. The authors selected the following methods:

1. **Root Mean Squared Error** [15][18] → This technique is used to find dissimilarities between the reference and the fused image. Low values show that the test image is similar to the reference image.

$$RMSE = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(A(m,n)-B(m,n))^2} \tag{2.11}$$

2. **Peak Signal to Noise Ratio** → Measurement of the quality of the image. The value will be high if the fused and reference image are identical.

$$PSNR = 10\log_{10}\frac{r^2}{MSE} \tag{2.12}$$

where r is the peak value of the reference image and MSE the mean squared error.

3. **Petrovic** ($Q^{AB/F}$) [19][20] → Giving the output image (F) and input images (A and B), the Petrovic metric measures the information preservation of A and B in F.

$$Q^{AB/F} = \left(\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}\left[Q^{AF}(m,n)W^{A}(m,n)+Q^{BF}(m,n)W^{B}(m,n)\right]}{\sum_{m=1}^{M}\sum_{n=1}^{N}\left[W^{A}(m,n)+W^{B}(m,n)\right]}\right) \tag{2.13}$$

where Q^{AF} and Q^{BF} are calculated over the edge values and W^A and W^B are weight factors. The output value can be between 0 and 1. An output of 0 will reveal a complete loss of information and 1 and ideal fusion.

4. **Structure Similarity Index Measure** [21] → This technique measures the structural resemblance between two images.

$$SSIM(A,B) = \frac{(2\mu_A\mu_B+C_1)(2\sigma_{AB}+C_2)}{(\mu_A^2+\mu_B^2+C_1)(\sigma_A^2+\sigma_B^2+C_2)} \tag{2.14}$$

where μ_A and μ_B are the mean, σ_{AB} is the cross co-variance and c_1 and c_2 are constants.

5. **Spatial Frequency** → This method is used to find the clarity of the resultant fused image using edge information. The higher the spatial frequency the higher clarity of the image.

$$RF = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(B(m,n)-B(m,n-1))^2} \tag{2.15}$$

$$CF = \sqrt{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(B(m,n)-B(m-1,n))^2} \tag{2.16}$$

$$SF = \sqrt{RF^2 + CF^2} \tag{2.17}$$

6. **Piella** (Q^{w_t}) [22] → This metric finds the quantity of information from the input images captured in the fused image.

$$Q^{w_t} = \sum_{w_t \in W} c(w_t)(\lambda(w_t)Q_0(A, F(w_t) + (1 - \lambda(w_t)Q_0(B, F(w_t))) \tag{2.18}$$

where Q_0 is the Wang-Bovik image quality index [] and $\lambda(w_t)$ is a weight referring the relative importance of input image A compared to B.

2.5 Conclusions

In this chapter the different image fusion techniques were shown. Each algorithm can be more or less efficient depending on the type of images to be fused. Furthermore there are a high number of variations available in each different algorithm that could also be fitted, again, depending on the types of images to be fused. For these purposes performance measures were selected to define which algorithms and respective variations adapt better to the images in question.

It is also important to consider that the final objective of the project is to perform video fusion, meaning a frame-by-frame fusion of two sensor streams (visible and IR). For this reason, the execution time of the algorithm should also be considered.

As mentioned in this chapter (sec 2.2) there are fusion techniques in different levels, such as pixel and feature level. Included in pixel level there are spatial and transform domain techniques. As an initial stage of this project an algorithm from each different technique will be selected. Meaning two pixel level techniques will be selected (a spatial domain and a transform domain) as well as a feature-level. The one that performs the best will then be selected for the next stages of the project.

Chapter 3

Methodology

This chapter starts by describing an initial analysis done in order to select the type of solution to be used. This selection was done based on the specifications and the objectives of the dissertation, thus the final objective of performing video fusion in real-time was considered throughout this study.

After selecting the solutions to be used, the overview of the algorithm developed in the ambit of this dissertation is described. This algorithm is further explained in the remainder of this chapter and it consists in developing an algorithm to calculate and assign weights to the images.

The results and comparisons are shown in chapter 4 where the final algorithm to be implemented will be selected.

3.1 Algorithm Selection

An analysis of the algorithms studied in chapter 2 is done in this section. In order to perform such analysis, an algorithm of image fusion would be chosen and then adapted to video. This adaption would consist in fusing the video frame by frame by executing the image fusion algorithm repetitively.

Therefore, the chosen image fusion approach must have a low execution time in order to perform the fusion in the available interval between two consecutive frames. For instance, in the case of a video at 25 FPS, the time between to consecutive frames is 40 ms, which is approximately the time available to perform such fusion.

Some work was initially done in order to understand which would be the best approach to this problem. There was some experimentation work done with different algorithms, as well as attempts to make improvements in some of them. The next sections will explain the selections made to reach the chosen algorithm.

3.1.1 Pixel-level vs. Region-based Techniques

As mentioned in chapter 2, there are two main types of techniques, pixel-based and region-based. Region-based techniques generally provide a fusion of better quality at the cost of higher

computing power, leading to higher execution times. On the other hand, pixel-based techniques generally provide a simpler solution to the problem. The output quality is not as good as the region-based ones however, these algorithms can be extremely fast to execute.

Considering that the system being developed must run in real-time, the execution time of the solution should be the main characteristic to be considered. Therefore a Pixel-level approach was used in the extent of this dissertation.

However, as previously mentioned, the output quality of a pixel-level approach can be of inferior value when compared to a feature-level one. So, for this reason there was an attempt to maximize the quality while using a pixel-level approach.

3.1.2 Pixel-level Techniques - Comparison

In this section a comparison of a generic version of the pixel-level techniques mentioned in section 2.3.1 (averaging, pyramid-based and wavelet transform) is done in order to understand which technique is best in the context of this problem. Again, the key characteristic of the chosen algorithm will be the execution time.

So, to test the speed of the algorithms a grayscale video provided by [23] was used as a dataset. The chosen video contains 2000 grayscale frames of 720x480 pixels and a fusion was done on each frame resulting in 2000 executions of each algorithm. The results can be seen in table 3.1.

Table 3.1: Execution times of the 3 pixel-level fusion algorithms (2000 frames fusion).

Algorithm	**Time (s)**
Average	15.26
Pyramid	14.00
DWT	38.29

These results were obtained by measuring the runtime of python scripts on a 1.80GHz i7 processor. Such times could be different if running on another processor, however, the ratio between those times would be similar. Another factor that might have influenced these times is how busy the processor is when running the scripts, therefore it was made sure all of them were running in the same conditions to provide a more accurate comparison between them.

As shown in table 3.1 the DWT approach is quite slower than the other two and, for this reason, it was opted out. Leaving the two remaining approaches to be considered for the next stages of this study.

3.2 Algorithm Overview

As mentioned in the previous section, the two selected solutions to be tested are the Average based and the Pyramid based ones. As such, the developed algorithm will be applied directly to both images in the case of the Average based approach, in this case performing a weighted average. In the Pyramid based approach, as mentioned in 2.3.1.1 there is a base fusion to start

the procedure and afterwards the image starts to expand to its original size. So, in this case, the developed algorithm will be applied as the base function of the Pyramid fusion approach. Meaning that the smallest versions of the images will be fused using the weighted average as a base function.

The final objective of the system consists in calculating the weighted average between two images. The overview of such system can be seen in Fig. 3.1. It is divided in three main parts, the processing of the IR image, the processing of the visible spectrum image and, at the end, the fusion of both images. On the right of the figure a detailed version of the processing of the IR image is shown.

The processing of both IR and visible spectrum images consists in assigning weight values to both images, respectively. These weights will be assigned following the developed algorithm presented in the remainder of this chapter.

The fusion part will consist in merging the information of both images considering the weights previously calculated by the developed algorithm. In this case, as mentioned before, the two fusion algorithms to be tested are the weighted average and the Pyramid based approach with the weighted average as the base function.

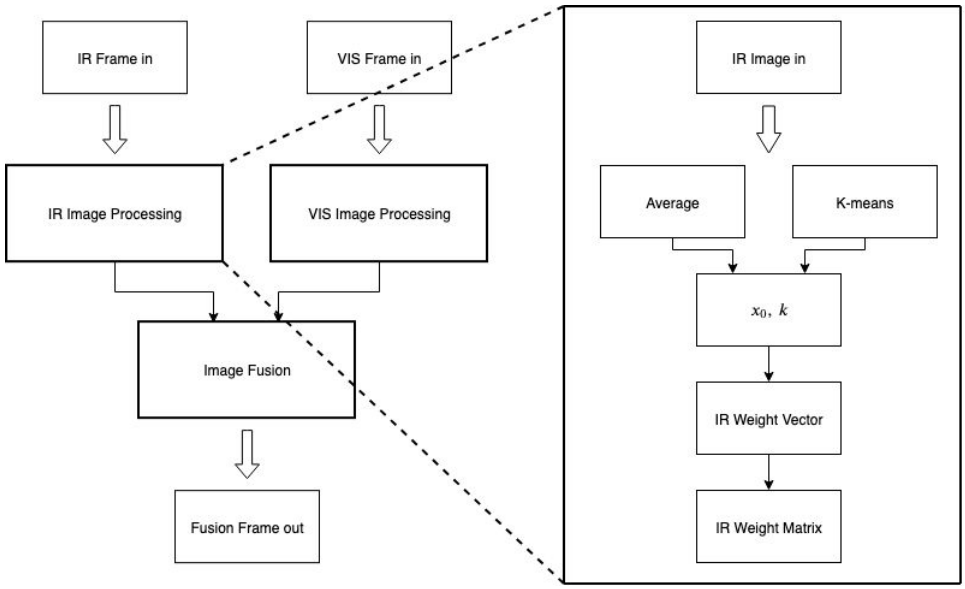


Figure 3.1: Overview of the developed fusion system.

3.3 Pixel Weighting

Averaging two images can be done following the generic formula shown in 2.3.1.1. A simplified version of such formula, used only for two images, is given by Eq. 3.1 and represented in Fig. 3.2 (it is assumed that both images are exactly the same size, as they should be).

$$F(x,y) = \alpha_1.I_1(x,y) + \alpha_2.I_2(x,y) \tag{3.1}$$

where $\alpha_1, \alpha_2 \in [0,1]$ and $\alpha_1 + \alpha_2 = 1$. If $\alpha_1, \alpha_2 = 0.5$ an unweighted average is performed, meaning that every pixel in both images will have the same impact in the output image.

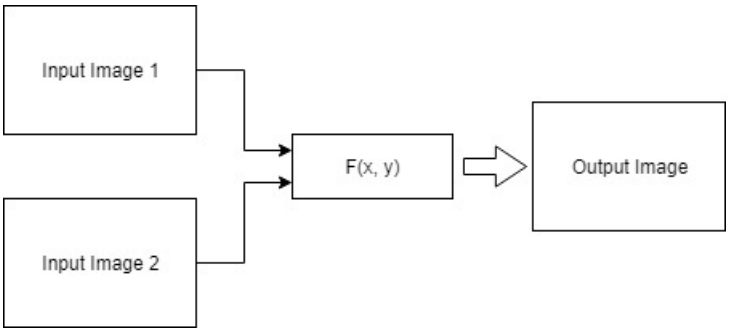


Figure 3.2: Diagram of a two image averaging algorithm. ($F(x,y)$ is detailed in 3.1)

On the other hand, if $\alpha_1 \neq \alpha_2$ the input images will have different impacts on the output image. This means, for instance, if $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$ then the values of I_1 will be 3 times more present in the output than the ones from I_2.

Fig. 3.3 shows an example of how varying the α values can impact the resulting image. Image *(a)* is the original image and images *(b)* and *(c)* are variations of the original image that will be used as inputs of the average.

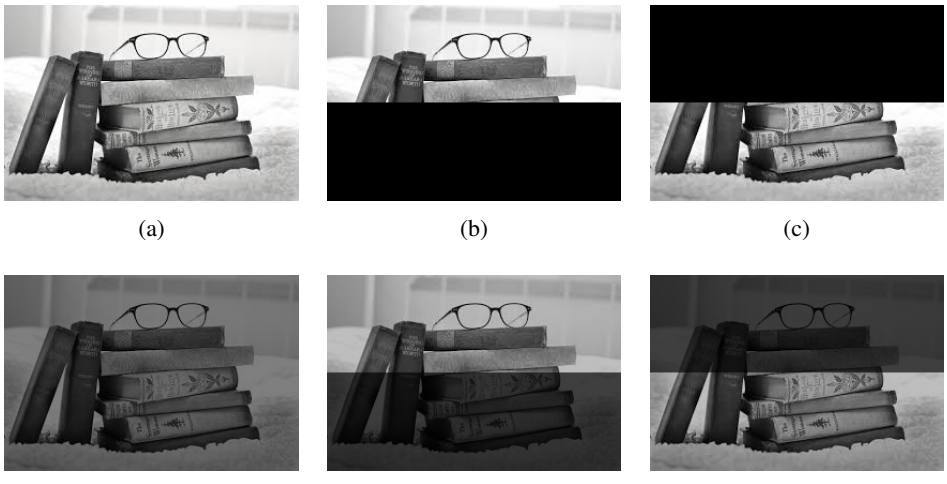


Figure 3.3: Examples of averaging with different α values.

(d), *(e)* and *(f)* are the resulting outputs from different α values. Image *(d)* is the output for $\alpha_1 = \alpha_2 = 0.5$, meaning it is an unweighted average. This resulted in a slightly darker image than the original one due to the darker pixels introduced to images *(b)* and *(c)*. Image *(d)* shows the output of the average when $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$. This can be seen in the output image as the dark strip in image *(b)* overlays the bottom details in image *(c)* while the upper part of the image is perfectly visible. The opposite happens in image *(f)* where $\alpha_1 = 0.25$ and $\alpha_2 = 0.75$. In this image the visible part of image *(b)* is darkened by the dark strip in image *(c)*.

So far all the α values have been constant for every pixel in the images. However, and as the name of this section suggests, different pixels in the same image could have different weights. This could be done by using a matrix of α values for each input image, where each matrix is the same size of its respective image and each value represents the weight for the respective pixel in the image. Fig. 3.4 shows a diagram of this algorithm.

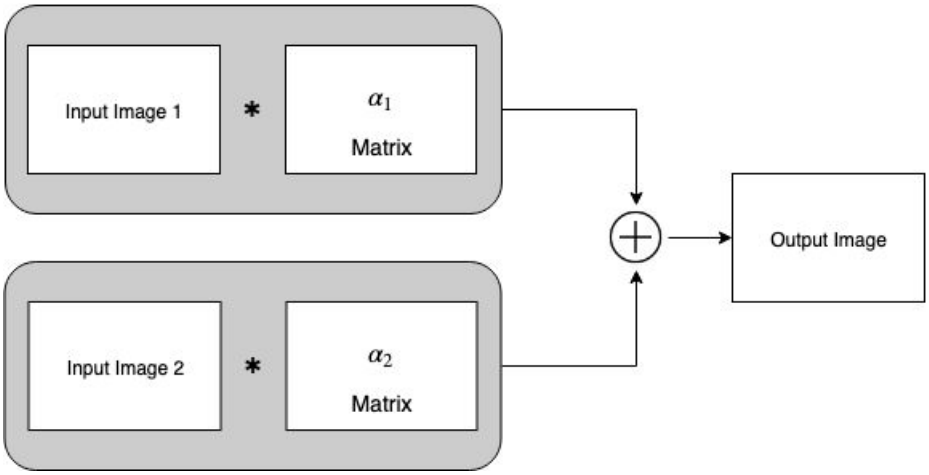


Figure 3.4: Weighted pixels average diagram.

After multiplying each pixel by their respective weight, this value is then added with the corresponding value from the other image and assigned to the output image. After repeating this process for every pixel in the input images the average output image is then complete.

The biggest advantage of using different weight values for the pixels in the same image is the ability to select the most important parts of the image and ignore the ones that are not as important. This is done by assigning high weight values to the pixels in the important areas and otherwise in the remaining areas.

This, however, bears one problem which is how to assign weights to the pixels in an image. The next sections will show the method developed to assign those weights taking into account the image fusion problem at hand.

Please note that in the following sections instead of using the α notation an ω notation is used. This was done to facilitate the weighting algorithm. Instead of having $\alpha_1, \alpha_2 \in [0,1]$ and $\alpha_1 + \alpha_2 = 1$, an ω value is used where $\omega \in \mathbb{R}$. This allows to set the weights without having the constraints present in the α notation.

After assigning ω values as weights, a transformation can be done using the following equations:

$$\alpha_1 = \frac{\omega_1}{\omega_1 + \omega_2} \tag{3.2}$$

$$\alpha_2 = \frac{\omega_2}{\omega_1 + \omega_2} \tag{3.3}$$

After using equations 3.2 and 3.3 the previous detailed averaging algorithms can be used as described.

3.4 K-means Clustering

The first part of the pixel weighting algorithm developed in the context of this dissertation starts by running a K-means clustering algorithm over the image.

Clustering consists in the association of different data points into groups by analyzing a set of features given by those points. The main objective is to organize those data points by similarity, meaning that identical ones will be put in the same group and consequently different ones will most likely be assigned to different groups. Such methods can have applications in Machine Learning and Data science, Image segmentation and compression, etc...

The K-means algorithm proposed by MacQueen [24] is one of the most used clustering algorithms. This method uses multiple iterations to obtain the best clustering given the data points. Considering each cluster has a central point called centroid the algorithm works as follows:

1. Select k points as the initial centroid values.

2. Assign every data point to the cluster with closest centroid.

3. Recalculate the new centroids by averaging all the data points assigned to a each cluster.

4. Repeat steps 2 and 3 until the iteration termination criteria is met. This termination criteria could be defined by reaching a maximum number of iterations or by reaching a specified accuracy value.

This process can be followed in Fig. 3.5 where in the first iteration random centroid values are given to the clusters. In the second iteration the data points are assigned to those centroids, which then get their new positions recalculated. After a certain amount of iterations, the algorithm converged and the final position of the centroids is determined.

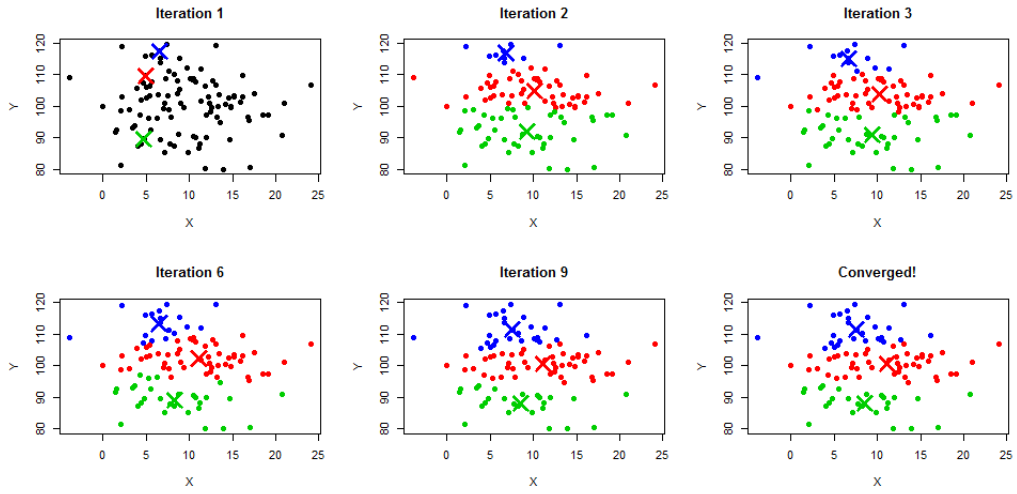


Figure 3.5: K-means iteration process.

3.4.1 K-means Clustering in Image Processing

This algorithm can be applied to the color intensities on an image (or just the intensity on a grayscale image). This can be used for image compression by reducing the amount of different intensity values in the image and also for image segmentation as a part of the thresholding algorithm.

Thresholding techniques are used to determine which part of the image belongs to the background and which parts belong to the foreground. In bi-level thresholding, the pixel whose gray level is less than the threshold will be assigned to the background, else to the foreground. However, it is not always good to find a single threshold point for an image in the cases where there are more complex objects in them, either in the foreground or background [25]. For this reason a multilevel threshold method could be used in more complex images. By using more than two clusters, a multilevel partition of the image is achieved. Fig. 3.6 shows an example of both bi-level and multilevel thresholding using the k-means algorithm.

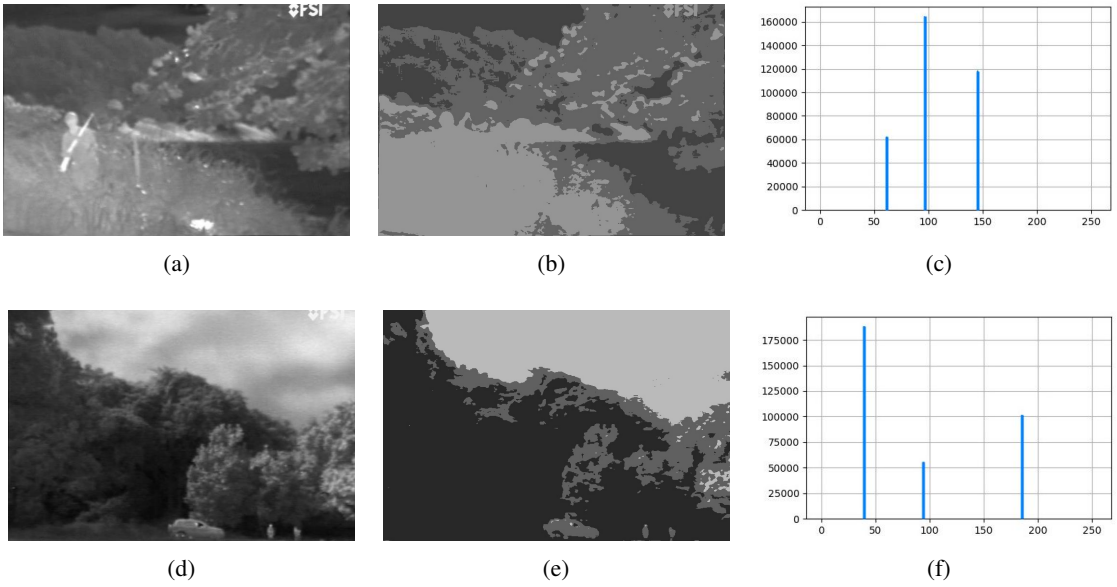
(a) (b) (c)

Figure 3.6: Example of K-means applied to a colored image. *(a)* original image, *(b)* k-means image with 2 clusters (bi-level thresholding), *(c)* k-means image with 4 clusters (multilevel thresholding).

3.4.2 K-means on Pixel Weighting

The K-means algorithm can be useful for pixel weighting because it can divide an entire image into groups. This will provide some useful information about the image such as how the image intensities are divided, which intensity value has the larger amount of pixels, if it is a dark or a bright image, etc... This information can then be used to determine which pixels are important and which ones are not.

Fig. 3.7 shows the result of two IR images after being clustered using the K-means algorithm. In this example, 3 clusters were used ($K = 3$) and their values, as well as the amount of pixels per cluster, can be seen in their respective histograms.



(a) (b) (c)

(d) (e) (f)

Figure 3.7: Example of K-means on two grayscale images with close and faraway objects. *(a)* and *(d)* are the original images, *(b)* and *(e)* are the clustered versions and *(c)* and *(f)* are the histograms of the clustered images, respectively.

Using $K = 3$ allows to break down the image in three parts regarding their intensity values. Those parts can be divided as:

- dark pixels;
- average intensity pixels;
- bright pixels;

This information can then be used to weight the pixels by determining which parts of the image are more important and which ones are not.

3.5 Infrared Image Weighting

In this section a brief introduction on thermography is done. The main parts of an IR image are then identified in order to assign the weights accordingly. Afterwards, the weighting function for the IR image is explained an exemplified.

3.5.1 Thermography

Thermography takes advantage of the fact that all the objects with the temperature above absolute zero (-273.15^{o}C or 0^{o}K) radiate energy with a range of wavelengths between 0 to ∞. This includes infrared, which wavelengths typically fall between 1-15 μm [26]. Infrared sensors are used to capture this information and then an image is produced based on the intensity of radiation being emitted by those objects. An object with higher temperature will emit higher intensity radiation and vice-versa. This will allow to view and map an environment independently of the amount of visible radiation available. This technique can be very useful for surveillance in low visibility situations since warm objects, such as humans, will stand out from the cooler background.

To provide some examples, the infrared images (a) and (d) from Fig. 3.7 will be used in this section.

3.5.2 Image Analysis

As previously said, in the IR images the interesting parts will have higher intensity values than the background. For this reason, high weights should be assigned to pixels with higher values and low weights to the ones with lower values. In order to assign such weights a sigmoid type function will be used. It can be expressed as:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \tag{3.4}$$

where

- $x_0 \rightarrow$ the midpoint in the x-axis,
- $L \rightarrow$ the maximum value,
- $k \rightarrow$ the steepness of the function.

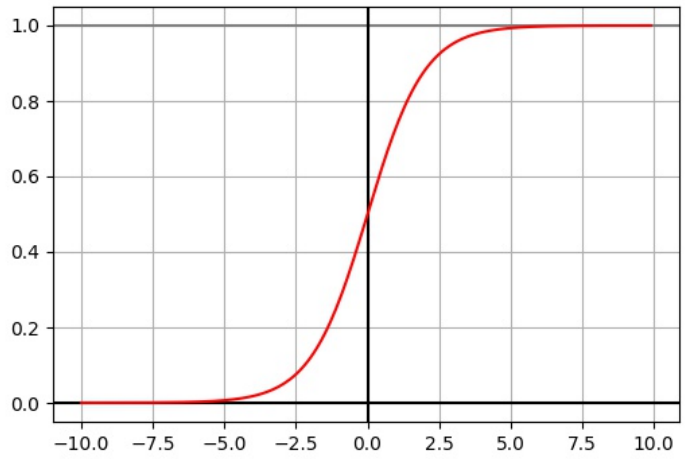


Figure 3.8: Sigmoid function with $x_0 = 0$, $L = 1$, $k = 1$.

This function was chosen because of its shape. As seen in Fig. 3.8, this function starts with a low value (asymptote at $y = 0$) and then smoothly rises until a higher value, L (in this case, asymptote at $y = 1$).

Using a variation of the sigmoid function as a weight function will allow low intensity valued pixels to be assigned low weights and high intensity valued pixels to be assigned high weights. It will also provide a smooth transition between these two levels of weights for the pixels with intermediate values.

Having decided the weight function, the next stages are to define and calculate the values for the L, x_0 and k variables.

3.5.2.1 Defining L

This value defines the maximum point in the sigmoid function. Meaning that, in the context of our problem, it will define the maximum value of the weights to assign to the pixels in the IR image. As of now, this maximum weight value will be denoted as `MAX_IR_WEIGHT` and, consequently, $L =$ `MAX_IR_WEIGHT`. Fig. 3.9 shows the sigmoid function with the new L value and, therefore, $\omega_{ir} \in]0,$ `MAX_IR_WEIGHT`$[$.

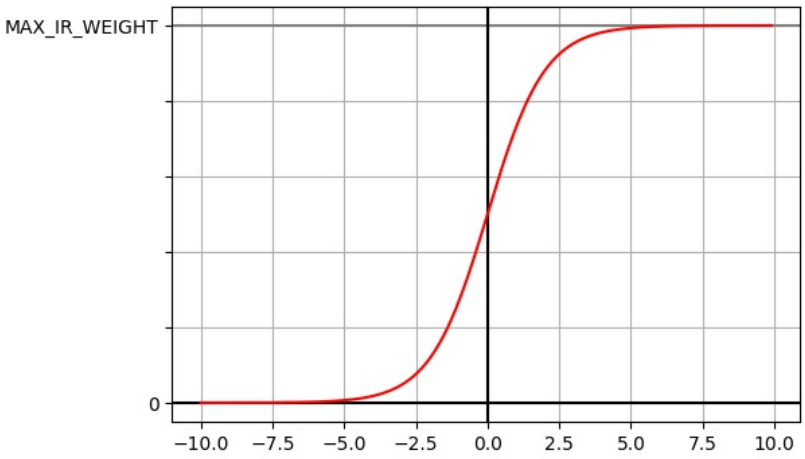


Figure 3.9: Sigmoid function with $L = \texttt{MAX\_IR\_WEIGHT}$

3.5.2.2 Calculating x_0 and k

These values will adjust the function in the x-axis. The value of x_0 will define the position of the function horizontally, while the k value will compress/stretch the function, changing the steepness in the x-axis. The first step is to define the domain of the weighting function. Considering we will be using grayscale images as inputs to our system, the pixel values will range from 0 to 255, so the domain of our weighting function should also be $[0, 255]$.

The next step should be to get the equations for both the variables. By analyzing the equation regarding the sigmoid function, an expression for each variable is obtained. Such expressions are the following:

$$y = \frac{L}{1 + e^{-k(x - x_0)}} \quad \Leftrightarrow \quad x_0 = \frac{ln\left(\frac{L}{y} - 1\right) + k.x}{k} \tag{3.5}$$

$$y = \frac{L}{1 + e^{-k(x - x_0)}} \quad \Leftrightarrow \quad k = -\frac{ln\left(\frac{L}{y} - 1\right)}{x - x_0} \tag{3.6}$$

Since there are two equations, two points must be defined in order to be able to calculate both x_0 and k. The first one (P_1) will be the point concerning the image's average intensity value. This value can be calculated by performing the average of the IR image using the following expression:

$$\mu = \frac{1}{mn} \sum_{i=0}^{m} \sum_{j=0}^{n} I(i, j) \tag{3.7}$$

So the x coordinate for P_1 will be μ_{ir}. The y coordinate for P_1 is the weight for the pixels which value is the same as the image's average. This value was defined to be $1/3$ of the maximum value, already known as `MAX_IR_WEIGHT`. It was defined as 1/3 because the values around the

average of an image will, most likely, be the most common values in said image. Therefore, and as a way to highlight the pixel values that are clearly above the average, this weight was set to 1/3 the maximum weight value. Given this, $P_1 = (\mu_{ir}, \texttt{MAX\_IR\_WEIGHT}/3)$, which can be seen in Fig. 3.10.

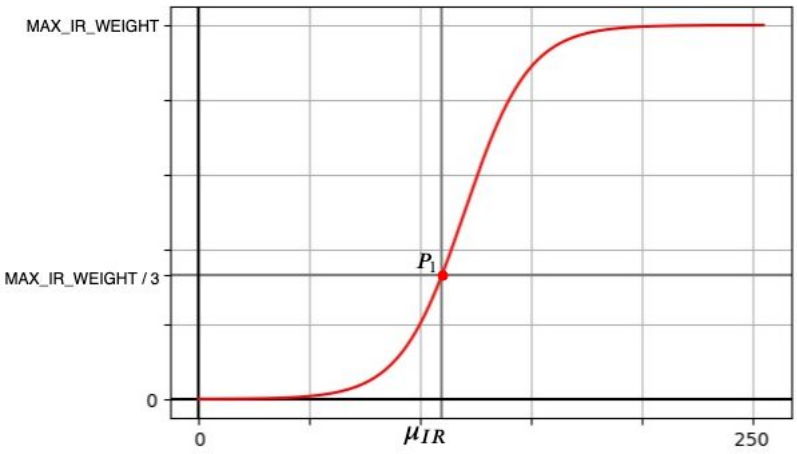


Figure 3.10: Example of $P_1 = (\mu_{ir}, \texttt{MAX\_IR\_WEIGHT}/3)$

The second point (P_2) will be the point where the sigmoid function starts to rise noticeably from its asymptote in $y = 0$. This point was denoted as *Rising Point* (RP) and is shown in Fig. 3.11.

To determine the value of RP the K-means clustering algorithm described in 3.4.2 will be used. Being the point where the sigmoid starts to rise, the RP value will separate the pixels that will get a close to null weight from the ones that will not. The RP value will be defined as the value of the cluster containing the darker pixels, meaning that every pixel with a value lower than the darkest cluster will have a null weight and the pixels with higher values will be weighted according to remaining part of the the sigmoid function.

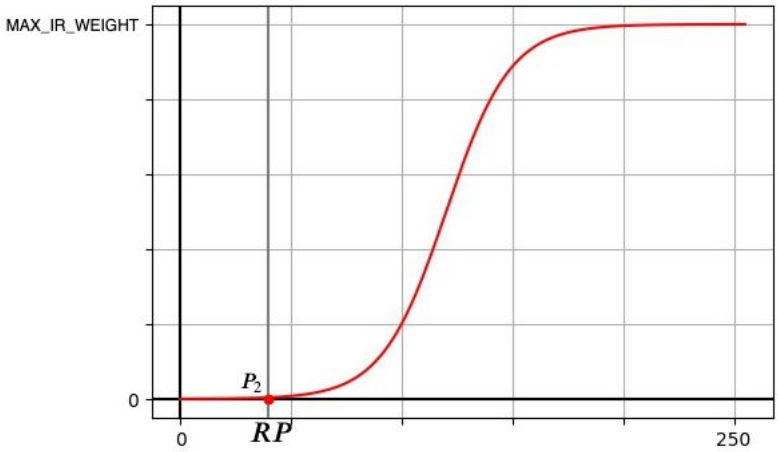


Figure 3.11: Example of $P_2 = (RP,\ 0.005)$

Fig. 3.12 shows an example of the output of the K-means algorithm from image (d) in Fig. 3.7. It has a cluster in intensity levels 40, 96 and 186. As previously mentioned, the `RP` will get the value of the cluster with the lowest intensity value, meaning that, in this case, its value would be 40.

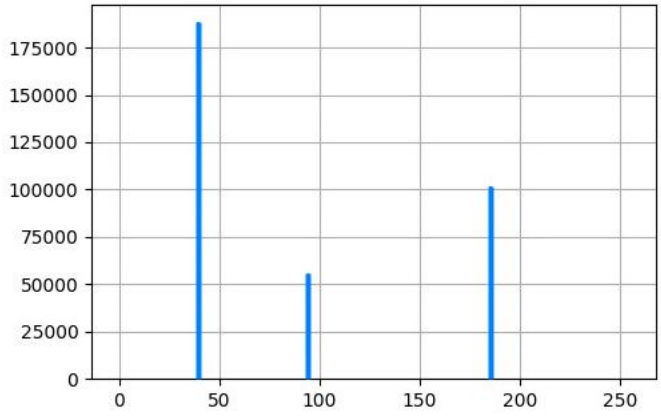


Figure 3.12: Output of the K-means algorithm with 3 clusters with values 40, 96, 186, respectively.

Knowing the value of `RP`, the x coordinate for P_2 is determined. The y coordinate should be a fixed value that allows for the `RP` to be positioned at the point the sigmoid function starts to rise significantly. $y = 0.005$ provided a good result as it is the coordinate value in the y-axis after which the sigmoid steepness starts to increase significantly. Therefore, $P_2 = (\texttt{RP},\ 0.005)$.

Having defined both P_1 and P_2, the next step is to solve the system of equations shown below:

$$\begin{cases} x_0 = \dfrac{ln\left(\frac{L}{y_{P_1}}-1\right)+k.x_{P_1}}{k} \\ \\ k = -\dfrac{ln\left(\frac{L}{y_{P_2}}-1\right)}{x_{P_2}-x_0} \end{cases} \tag{3.8}$$

Solving the system for x_0 and k, the following final expressions are obtained:

$$\begin{cases} x_0 = \dfrac{x_{P_2}.ln\left(\frac{L}{y_{P_1}}-1\right)-x_{P_1}.ln\left(\frac{L}{y_{P_2}}-1\right)}{ln\left(\frac{L}{y_{P_1}}-1\right)-ln\left(\frac{L}{y_{P_2}}-1\right)} \\ \\ k = \dfrac{ln\left(\frac{L}{y_{P_2}}-1\right)-ln\left(\frac{L}{y_{P_1}}-1\right)}{x_{P_1}-x_{P_2}} \end{cases} \tag{3.9}$$

Having defined L and calculated x_0 and k, the weighting function is obtained and is shown in Fig. 3.13

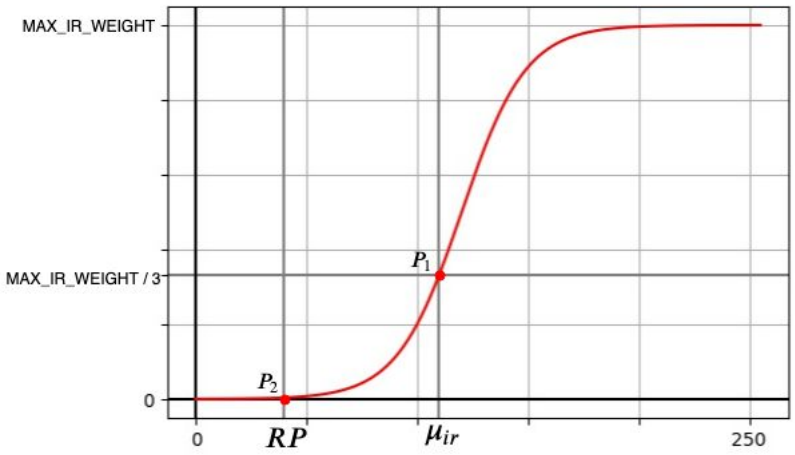


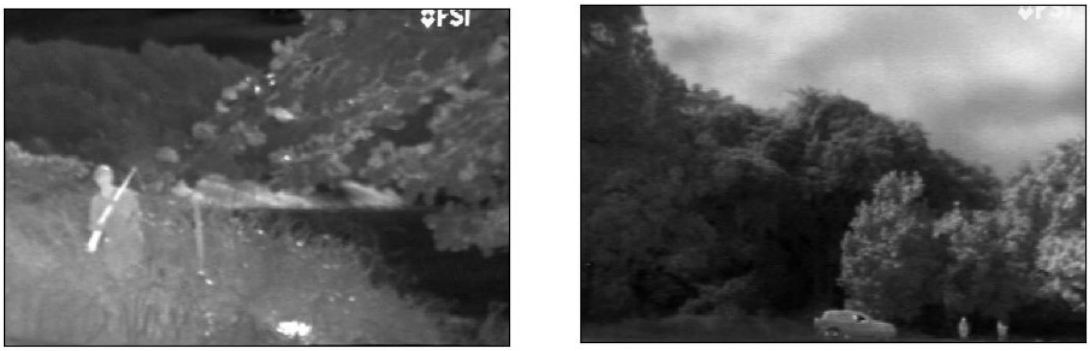
Figure 3.13: An example of a weighting function with $x_0 \approx 120$ and $k \approx 0.0689$.

In this example, in an IR image with an average value of 110 and with the lowest valued cluster of the K-means algorithm located at 40, and also considering the `MAX_IR_WEIGHT` to be 1, the two auxiliary points will be $P_1 = (110,\ 1/3)$ and $P_2 = (40,\ 0.005)$. Leading to the values of x_0 and k to be:

$$\begin{cases} x_0 = 120.057 \\ k = 0.069 \end{cases} \tag{3.10}$$

3.5.3 Examples

The next two images (Fig. 3.14) will be used to exemplify the whole weighting process for IR images. These images were chosen because one of them has an object in the vicinity of the sensors and the other has the objects farther away. This will show how the algorithm works for both this cases.



(a) (b)

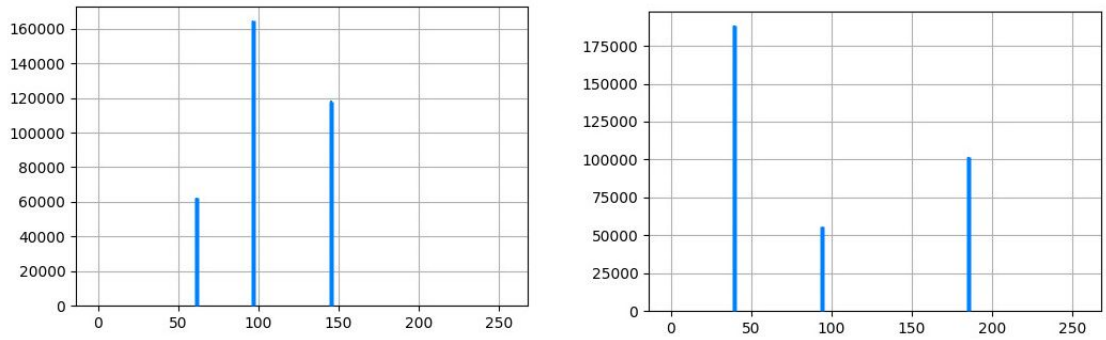
Figure 3.14: Example of two IR images. (a) shows an object close to the sensors while (b) contains objects farther away.

In this example `MAX_IR_WEIGHT` will be defined as 1 ($L = 1$) in both images. So the next step is to calculate the values for x_0 and k. To do so the values for auxiliary points P_1 and P_2 must be defined/calculated.

To define P_1, the average values of the IR image must be calculated. In these examples, the average value for the image in (a) is approximately 108 while for the image in (b) is approximately 92. Given this, the P_1 values for each IR image are:

$$P_1(a) = (108,\ 1/3) \qquad P_1(b) = (92,\ 1/3)$$

The next step is to calculate P_2 using the K-means algorithm. The output of this algorithm for each of the example images is shown in Fig. 3.15.



(a) (b)

Figure 3.15: Outputs of the K-means algorithm respectively to IR images from Fig. 3.14.

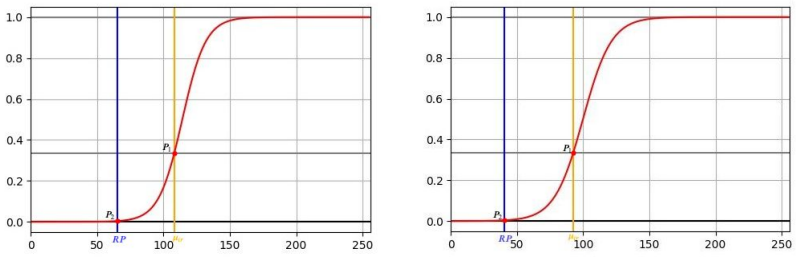
As previously mentioned, the x coordinate for P_2 will be defined as the value of the lowest valued cluster from the output of the K-means algorithm. In these cases, for (a) the lowest valued cluster is located at $x \approx 65$ while for (b) it is located at $x \approx 40$. The values of P_2 can now be defined as:

$$P_2(a) = (65,\ 0.005) \qquad\qquad P_2(b) = (40,\ 0.005)$$

After having defined both P_1 and P_2, the next step is to calculate x_0 and k using Eq. 3.9. The following final values are obtained for both the example images:

$$(a) \Rightarrow \begin{cases} x_0 \approx 108.04 \\ k \approx 0.129 \end{cases} \qquad\qquad (b) \Rightarrow \begin{cases} x_0 \approx 92.4 \\ k \approx 0.106 \end{cases}$$

Therefore, the final weight functions for the two IR images shown above are the following, respectively:



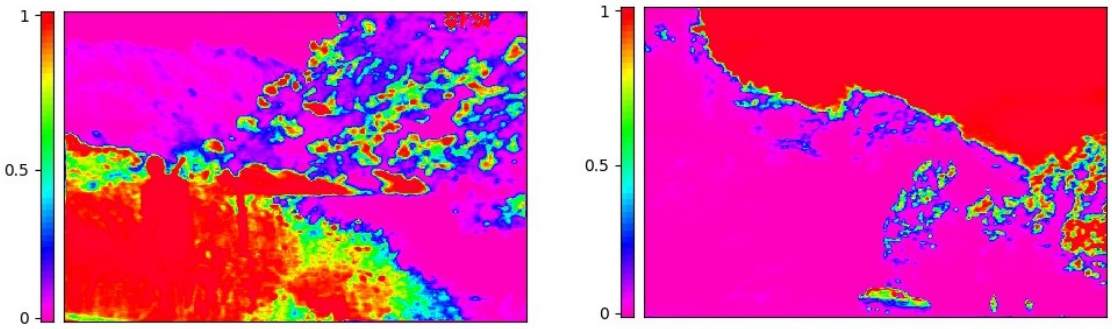
(a) (b)

Figure 3.16: Weight functions for the two example images in Fig. 3.14, respectively.

At last, after determining the weight function, a matrix of weights for the IR image will be

created. This matrix will be the exact size of the image and each position of the matrix will contain the weight value for their respective pixel on the IR image.

A representation of such matrices for the examples given above is shown in Fig. 3.17. This weight maps show the weights assigned to each pixel, where the purple pixels have a value of 0 and the red pixels have the maximum value, in this case 1. All the other pixels have intermediate weight values which increase with the color gradient assigned to them.



(a) (b)

Figure 3.17: Weight maps for the examples given above, respectively.

By looking at the images above it is possible to analyze that the pixels in the IR images that were brighter got assigned higher valued weights while, on the other hand, the pixels with low intensity values got assigned low weights.

The next section explains the process of weighting the visible spectrum image.

3.6 Visible Spectrum Image Weighting

Since we want to keep all the visible spectrum information and then average it with the weighted IR image, the visible image should have constant weight for all its pixels. The main idea is that the IR values with high weights will stand out on the final fused image and the ones with low weight will be suppressed by the ones in the visible spectrum image.

If every pixel in the visible image has a weight value half the maximum assigned to the IR image, then any pixel in the IR image with a weight above this value would stand out and any pixel with a value below would be suppressed. This can be seen in Fig. 3.18, where both visible spectrum and IR weight functions are shown.

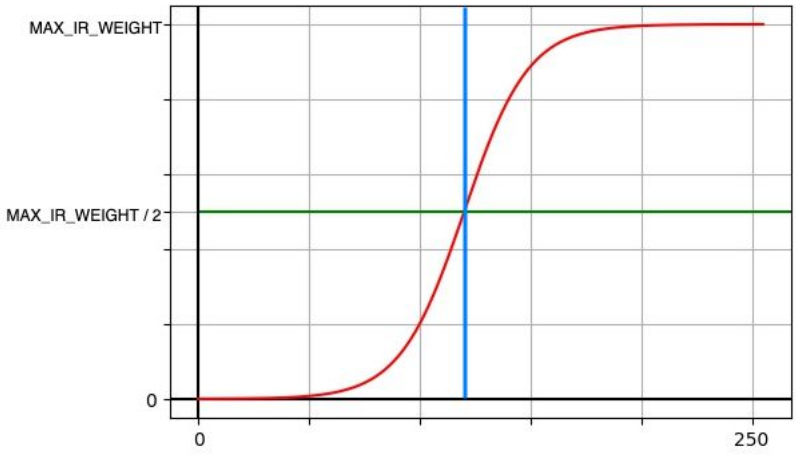


Figure 3.18: Representation of an example of both weighting functions. Visible Spectrum function in green and IR weight function in red.

The red line represents the weight function for the IR image, while the green one represents the one for the visible spectrum image. The blue line separates the intensity values in which the IR pixels will be suppressed (left of the line) and the ones which will stand out (right of the line).

Given this, the visible spectrum image will have a constant weight for all its pixels. This value half the value of `MAX_IR_WEIGHT`.

3.6.1 Examples

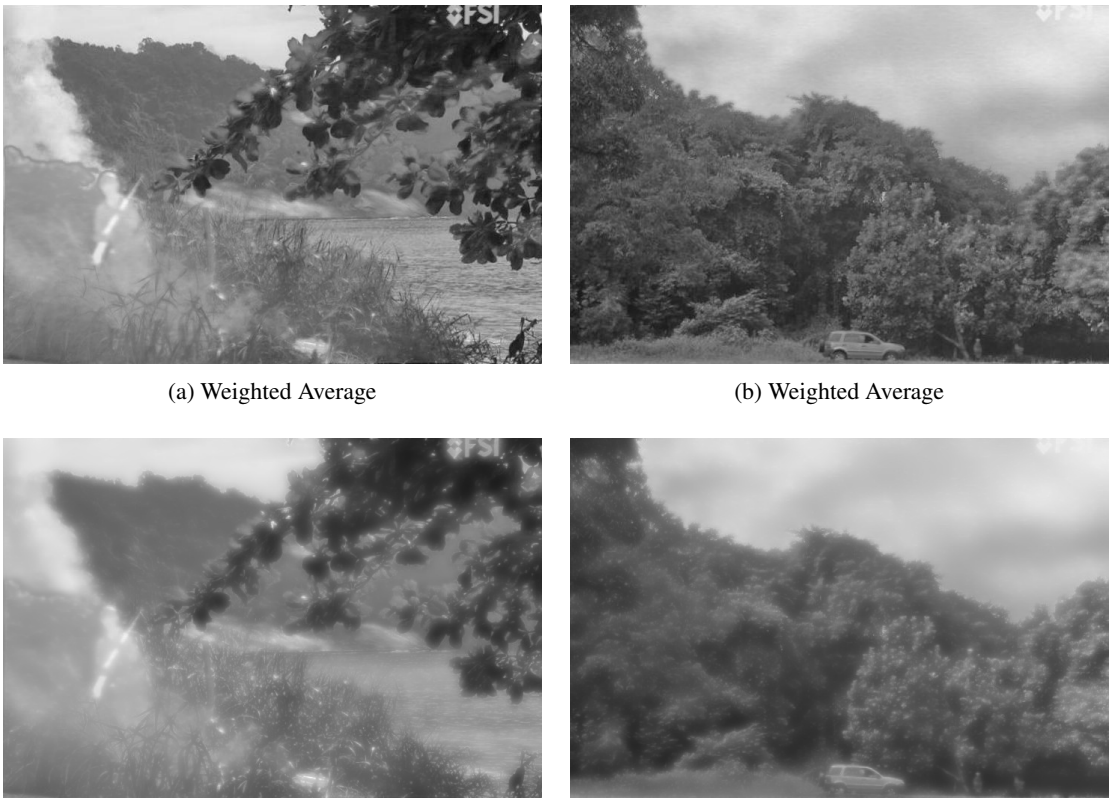
Using the same example images from subsection 3.5.3, the following pairs of images (Fig. 3.19) will be used to further demonstrate this algorithm.

(a)

(b)

Figure 3.19: Example images in the visible spectrum and IR.

Using the IR weights calculated in subsection 3.5.3 and the weight of the pixels in the visible spectrum to be 0.5 (half the maximum of the IR weights), the fusion images shown in Fig. 3.20 were generated. These images maintain the information contained in the visible spectrum image, while smoothly adding only the brighter parts from the IR image.



(a) Weighted Average

(b) Weighted Average

(c) Pyramid (Weighted Average based)

(d) Pyramid Weighted Average based

Figure 3.20: Result of the fusion of the example images using the developed method.

The images were calculated using both approaches selected so far, the weighted average and the Pyramid decomposition. For the weighted average approach, the developed algorithm was used to calculate the weights of both images and then the average was calculated based on those. The results can be seen in images (a) and (b).

For the Pyramid decomposition approach, the images were reduced and then a weighted average using the developed algorithm to calculate the weights was done on those reduced images as the base fusion for the pyramid approach. Afterwards the fusion image was expanded multiple times until it reached the initial size.

By looking at the pictures it is possible to see that the brighter parts of the IR image are highlighted in the final images, yet the visible details on the remaining parts are still maintained.

The next chapter compares these solutions as well as some reference approaches based on the normal average to compare the results and to select the final solution to be implemented.

Chapter 4

Results and Comparisons

In this chapter, a comparison of the results obtained from different image fusion algorithms is presented as a way to evaluate the weighting algorithm developed in the context of this dissertation. In the remainder of this document, the weighted average using the algorithm developed in this dissertation as a weighting function will be denominated as **Sigmoid**.

The approaches to be compared are the unweighted Average, Pyramid decomposition with unweighted average as base fusion, weighted Average with the Sigmoid algorithm to determine the weights and, lastly, the Pyramid decomposition with Sigmoid weighted average as the base function.

To make this comparison sufficiently representative, five images with different characteristics were selected in order to compare the different methods in different scenarios. Also, three quantitative performance evaluation tests from the ones described in section 2.4 were implemented.

As these performance evaluation tests require a reference image, a fusion image for each example was designed. This was done using an image editor program and the objective was to create what would be a perfect fusion image from the two input images. These test images and their respective fusion reference can be seen in Figs. 4.1, 4.3, 4.5, 4.7 and 4.9. The original images were provided by [23].

The chosen performance evaluation methods were Peak Signal to Noise Ratio (PSNR), Root Mean Squared Error (RMSE) and Structure Similarity Index Measure (SSIM). In image processing, the PSNR method is usually used to measure the difference of quality between two images. The higher the PSNR value is, the closer the image being evaluated is to the reference. So, in this case, we will be looking for the algorithm with the highest PSNR value.

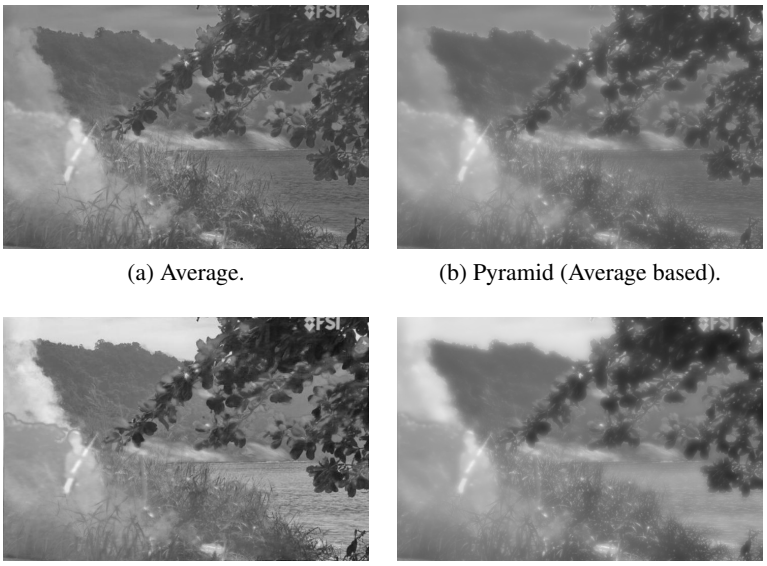
The RMSE method, in image processing, can be used to find dissimilarities between two images. In this case it was used to find the dissimilarities between the reference image and the output from the algorithms being compared. The lower the RMSE value, the closer the output is to the reference image.

The SSIM is used to measure the similarity between two images. Its output value is located in $[0,\ 1]$, where 0 indicates no correlation with the reference image and 1 means the images are exactly the same [27]. In the context of this problem the closer the value is to 1 the better.

The next pages show the results of these comparative tests in tables 4.1, 4.2, 4.3, 4.4 and 4.5 where the best results are highlighted in bold.

(a) Visible image. (b) IR image. (c) Fusion reference image.

Figure 4.1: Test image *A*.



(a) Average. (b) Pyramid (Average based).

(c) Sigmoid. (d) Pyramid (Sigmoid based).

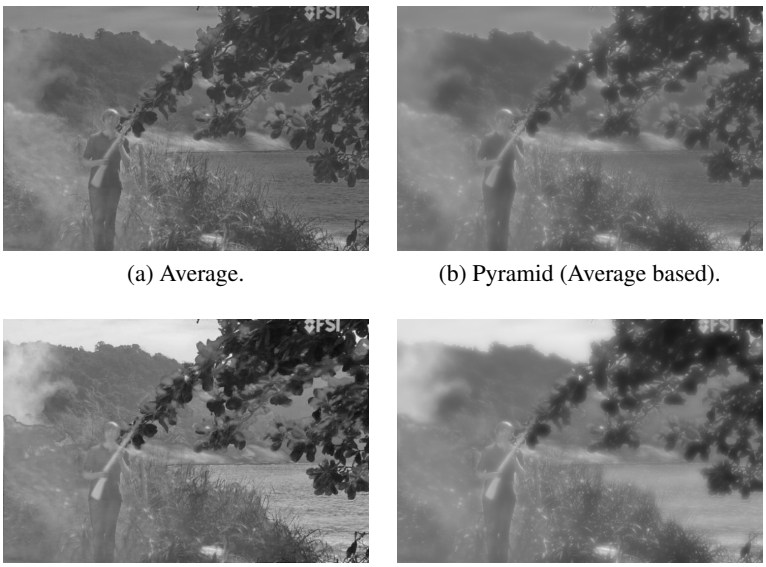
Figure 4.2: Output of the different fusion algorithms for test image *A*.

Table 4.1: Comparison of the fusion results from test image *A*.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	16.52871	38.02811	0.79952
Pyramid (Average based)	16.00817	40.37677	0.66599
Sigmoid	**19.69167**	**26.42145**	**0.81796**
Pyramid (Sigmoid based)	18.17455	31.46391	0.68558

(a) Visible image. (b) IR image. (c) Fusion reference image.

Figure 4.3: Test image *B*.



(a) Average. (b) Pyramid (Average based).

(c) Sigmoid. (d) Pyramid (Sigmoid based).

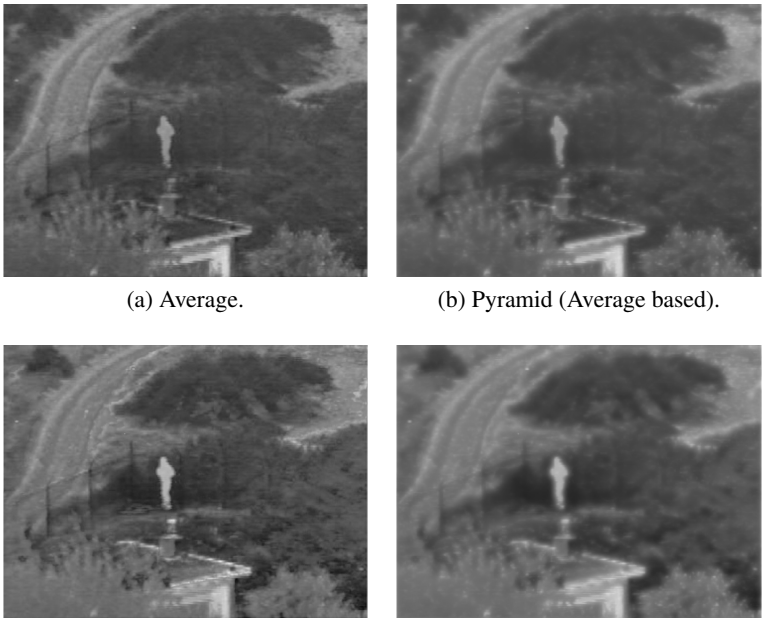
Figure 4.4: Output of the different fusion algorithms for test image *B*.

Table 4.2: Comparison of the fusion results from test image *B*.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	16.80988	36.81683	0.82078
Pyramid (Average based)	16.53495	38.00081	0.69240
Sigmoid	**22.49180**	**19.14035**	**0.85459**
Pyramid (Sigmoid based)	20.52013	24.01781	0.72095

(a) Visible image. (b) IR image. (c) Fusion reference image.

Figure 4.5: Test image *C*.



(a) Average. (b) Pyramid (Average based).

(c) Sigmoid. (d) Pyramid (Sigmoid based).

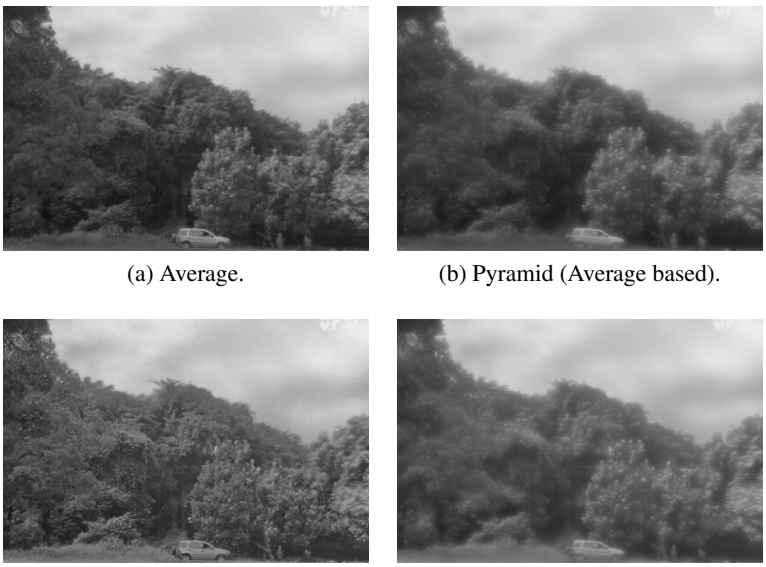
Figure 4.6: Output of the different fusion algorithms for test image *C*.

Table 4.3: Comparison of the fusion results from test image *C*.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	24.24836	15.63588	0.91388
Pyramid (Average based)	24.42881	15.31439	0.88205
Sigmoid	**27.31682**	**10.98245**	**0.91728**
Pyramid (Sigmoid based)	26.29216	12.35754	0.89303

(a) Visible image. (b) IR image. (c) Fusion reference image.

Figure 4.7: Test image *D*.



(a) Average. (b) Pyramid (Average based).

(c) Sigmoid. (d) Pyramid (Sigmoid based).

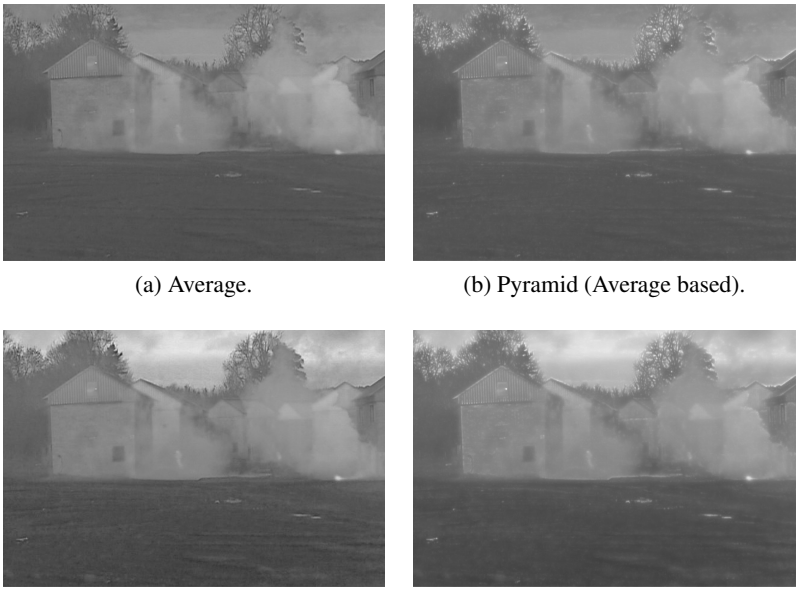
Figure 4.8: Output of the different fusion algorithms for test image *D*.

Table 4.4: Comparison of the fusion results from test image *D*.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	23.19384	17.65420	0.89463
Pyramid (Average based)	23.58599	16.87488	0.78796
Sigmoid	**32.15116**	**6.29480**	**0.96854**
Pyramid (Sigmoid based)	27.65052	10.56853	0.80450

(a) Visible image. (b) IR image. (c) Fusion reference image.

Figure 4.9: Test image E.



(a) Average. (b) Pyramid (Average based).

(c) Sigmoid. (d) Pyramid (Sigmoid based).

Figure 4.10: Output of the different fusion algorithms for test image E.

Table 4.5: Comparison of the fusion results from test image E.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	21.50770	21.43655	0.95680
Pyramid (Average based)	21.59982	21.21042	0.93995
Sigmoid	**28.51262**	**9.56994**	**0.96170**
Pyramid (Sigmoid based)	28.12199	10.01015	0.94945

4.1 Conclusions

As seen in the previous tables, the sigmoid algorithm provided the best quantitative results and also in a consistent way. The next table shows the average values of all the results obtained.

Table 4.6: Average results.

Algorithm	**PSNR (high value)**	**RMSE (low value)**	**SSIM (closer to 1)**
Average	20.45769793	25.91431412	0.877121635
Pyramid (Average based)	20.43154798	26.35545382	0.793669876
Sigmoid	**26.03281322**	**14.48179938**	**0.904015832**
Pyramid (Sigmoid based)	24.15186914	17.68358795	0.810702407

By analyzing the images above, it is clear that the downside of the unweighted average approach (figure (a) on the results above) mentioned in 2.3.1.1 is visible. This approach most often results in a low contrast image which deteriorates the visible part of the image. This can be seen in the sky on examples *A*, *B* and *E*, the lake in examples *A* and *B* and the clay pathway in example *C*.

The same effect can be seen on the Pyramid approach with the unweighted average as the base function. In addition to that the fusion image also gets some blur, this is due to the reductions and expansions done. As previously explained in 2.3.1.2, every time there is a reduction, every other line and column are deleted. So, when expanding back, every other row and column are duplicated, resulting in an image a bit blurry.

The Sigmoid fusion (weighted average) performs the fusion in the parts of the IR image which are relevant, while usually maintaining the original visible parts in the non relevant parts of the IR image. This can be seen and compared with the examples given above for the unweighted average. The sky is a lot brighter and realistic in example images *A*, *B* and *E* in the Sigmoid fusion. This can also be seen in example image *D* where the foliage of the trees is clearer in the case of the Sigmoid fusion.

The Pyramid approach using the sigmoid weighted average as the base fusion has a good performance however, the same problem as the unweighted average based Pyramid examples, the output can get a bit blurry, due to the reasons mentioned in the other similar case.

Considering this and the quantitative results, the Sigmoid approach shown the best results amongst all four approaches, therefore it was chosen to be the implemented solution. The next chapter contains details on such implementation and posterior optimization of the system.

Chapter 5

Video Fusion

This chapter starts by describing the implementation of the selected approach. It shows an overview of the classes implemented followed by a flowchart on the adaption for consecutive sets of frames (video stream).

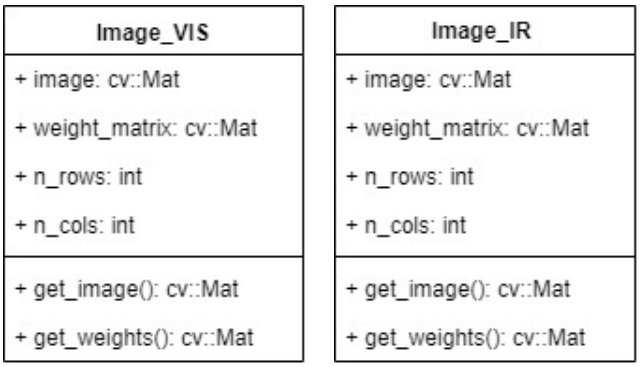
Afterwards it shows the study done on the real-time viability of the system as well as the optimizations done to minimize the execution time. In order to achieve this, the critical path had to be defined, path in which these optimizations would be applied.

5.1 Implementation

Initially the algorithm described in chapter 3 was implemented in C++. In order to do so, 3 classes were developed, one regarding the IR image, another regarding the VIS image and, finally, a Frame class to hold both images (VIS and IR) and take care of the fusion between the two images.

The classes regarding the IR image and the VIS image both receive their respective images, store them and, lastly, create the weight matrix for their image. The weight matrix generation for the IR image is, however, as previously explained, much more complex than the matrix generation for the VIS image. This is because the weights regarding the VIS image are constant for the entire image, making it much simpler to create it. Fig. 5.1 shows a representation of both these classes, including their variables and methods.

The Frame class contains one instance of the VIS image class and one instance of the IR image class. It then takes care of generating the fusion image. Fig. 5.2 shows the details of the Frame class.



(a) (b)

Figure 5.1: IR and VIS image classes.

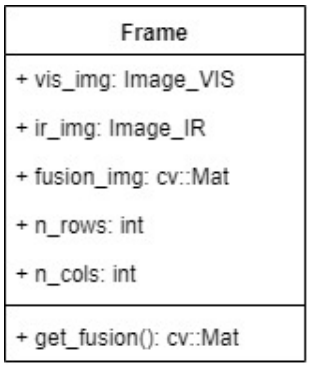


Figure 5.2: Frame class.

So far, the implemented algorithm only works for a set of images, however, to be used in a video environment it has to be able to work for various set of images (frames), making it possible to create a fused video. Therefore, the system developed until now must be executed in a loop, one time for each set of frames.

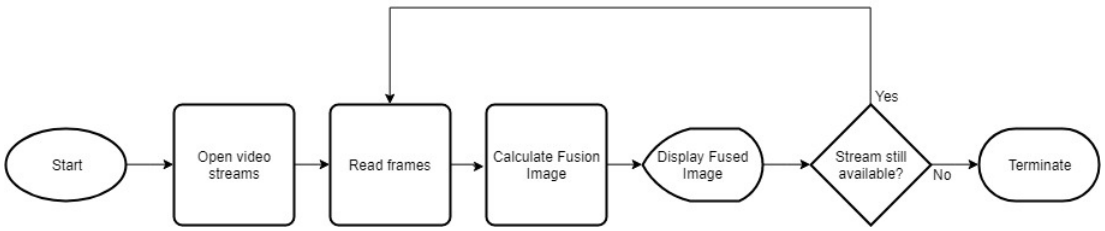


Figure 5.3: Flowchart of the Video class developed.

Fig. 5.3 shows the flowchart of the last developed class (Video class), which takes care of the entire process to make it possible to generate video fusion. This includes opening the streams of both types of image (VIS and IR) and then, in a loop, read the frames from both streams and instantiate an object from the Frame class which will take care of generating a fusion of from the frames in the video streams.

5.2 Real-time Analysis and Optimization

5.2.1 Critical Path

The critical path defines which parts of the algorithm have the most influence in the overall execution time. It is quite important to define this path since this allows to optimize this critical part and lower the execution time.

By analyzing the algorithm overview shown previously in Fig. 3.1 it is possible to verify that, initially, the algorithm is divided in two parts (IR Image Processing and VIS Image Processing). Both these parts are required to be completed in order to initiate the next and final part, the Image Fusion.

Considering the much higher complexity of the IR processing part over the VIS processing one, the critical path must be defined as the red stripped line in Fig. 5.4. This path goes through the IR processoing and the Image Fusion parts.

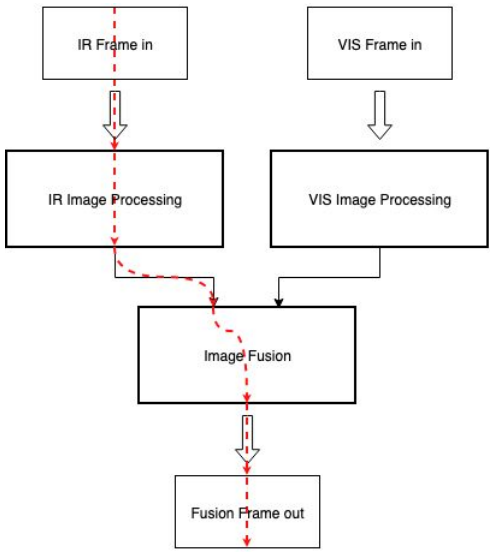
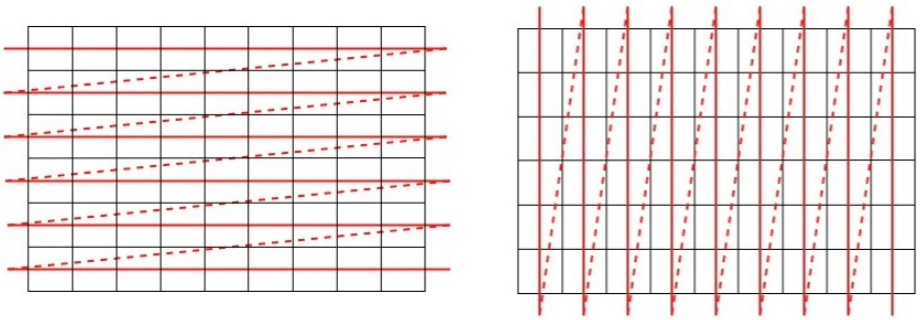


Figure 5.4: Critical path of the developed image fusion algorithm.

In order to obtain the maximum performance and the lowest execution time, an optimization of the IR Image Processing system detailed in Fig. 3.1 must be done. This optimization will be further explained in the next subsection.

5.2.2 Image Traversing

Image traversing consists in the iteration over an entire image, accessing every pixel. There are two main ways of doing it, row major and column major. Row major traversing consists in a horizontal scanning of the image, while column major traversing consists in a vertical one. Fig. 5.5 shows both types of image traversing.



(a) Row major traversing (b) Column major traversing

Figure 5.5: Diagram of Matrix/Image traversing.

Both (row major and column major) ways of traversing the image access exactly the same amount of memory, however, considering the image was stored in memory row by row, column major traversing can be a lot slower. In the case of this implementation, the images are stored using an object from the OpenCV library of type *cv::Mat* which, by default, stores an image by row. Fig. 5.6 shows the traversal time vs. matrix size graph from the study provided by [8] using both row and column major in a matrix which values were stored row by row. The yellow dots correspond to a row major traversing using a gcc compiler, while the blue dots also correspond to a row major traversing, this time using a Microsoft compiler. The cyan and purple dots correspond to column major traversals using gcc and Microsoft compilers, respectively. By looking at the graph it is quite noticeable that column major traversing (purple and cyan dots) is quite slower than row major (yellow and blue dots). Without getting in too much detail, this difference in execution time happens due to cache memories and the way the cache lines are copied from main memory into the cache [8].

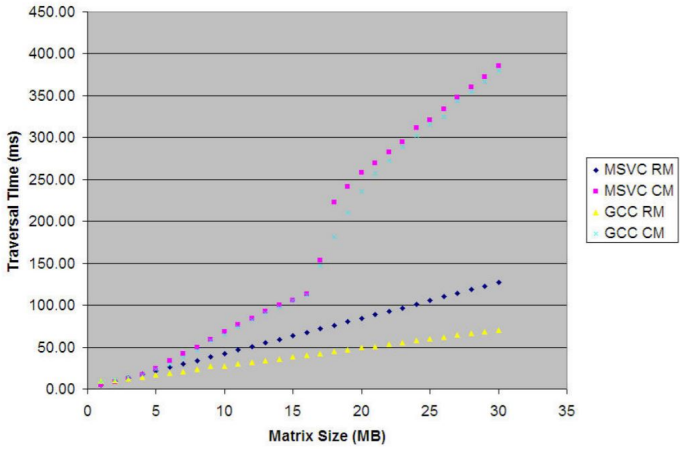


Figure 5.6: Matrix traversing speed comparison, row major vs. column major [8].

This information was considered every time any image had to be traversed when implementing the system, meaning that a row major image traversing was always used whenever it was possible.

Even though this allows for faster access to the image's pixel values, traversing an image (in any given way) is always a slow process with a lot of main memory accesses and, for this reason, there will be attempts to improve this speed using different techniques described in the next sections.

5.2.3 IR Image Processing Optimization

The first part of this section of the algorithm consists in calculating the image's average value and its centroids using the K-means algorithm. As mentioned before, both of these parts require at least one image traversing (even though the K-means algorithm will require more than one most of the times).

As a way to speed up this task and considering that the size of the image clearly influences the time it takes to traverse all the image's pixels, a study was done to measure the influence of an image size reduction in the final results of calculating the image's average and its centroids.

In order to do so, an image was resized multiple times and the average and centroids were calculated for each resized copy of the image. Then, the ratio between the values obtained in the resized images and the ones from the original image was calculated. The results of this study are shown in Figs. 5.7 and 5.8.

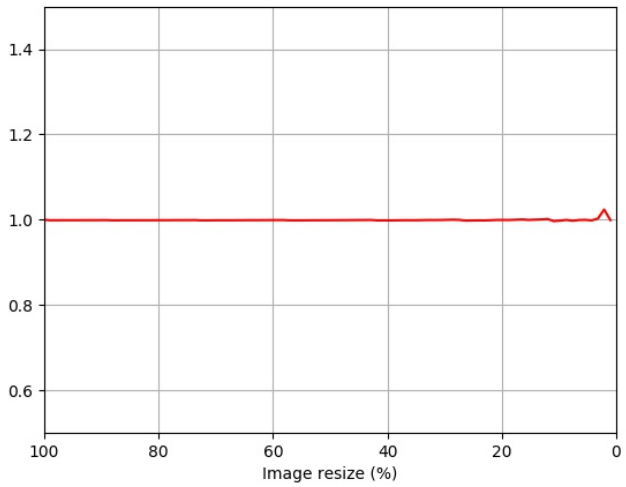


Figure 5.7: Ratio between the average value of the resized image and the original one.

Fig. 5.7 shows the ratio of the average values as a function of the resize percentage. This graph shows that by resizing the image, the value of its average does not change significantly compared to the original sized image value. This will allow to calculate the average on a reduced copy of the current frame, leading to a much faster calculation.

Fig. 5.8 shows the ratio of the cluster values as a function of the resize percentage. By analyzing this graph we are able to see that, by resizing the image, the cluster values will not change very significantly until the 5% resize value is reached. For this reason, a reduced copy

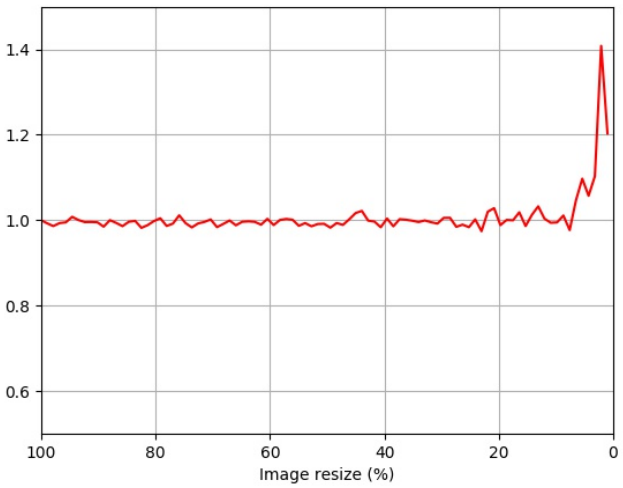


Figure 5.8: Ratio between the centroid values of the resized image and the original one.

of the current frame can also be used to calculate the K-means cluster values, speeding up the process.

This study concluded that the use of a copy 20% the size of the original image can be used to calculate both the average and the K-means centroids, allowing for a speed up of 1.5.

Another optimization that was done to the K-means part was to use the centroids from the previous frame as the starting centroid values to the current frame being processed. This will allow for a faster calculation of the centroid values since less iterations of the K-means algorithm will be needed. This is due the fact that the centroids of the previous frame will be, unless there is a scene cut, very similar to the ones of the current frame.

Having sped up the image traversing time in the calculation of the average and the centroids values by resizing the image to 20% its original size and reducing the amount of K-means iterations by using the previous centroids as starting points, the next step is to find a way to speed up the process of obtaining the weight matrix. Since this process consists in individual calculations over every pixel in the IR image, this process can then be optimized by the use of multithreading in case the algorithm is being executed on a multi-core hardware. The next section will further explain this optimization process.

5.2.4 Multithreading

A thread is an execution point within a process. Therefore, a multithread process has multiple concurrent points of execution. Multithreading can be useful in a situation in which there is a need to wait for resources. For example, one thread can be waiting for the resources while another thread can be running in the meantime. For instance, in a task in which there is an I/O request or a database access, one thread could be waiting for those resources while another thread continues to work on the CPU.

Multithreading can also be useful if there is the ability to run individual threads concurrently in different cores in the CPU. This is shown to be a powerful approach for boosting a system by taking advantage of said parallelism [28]. In the ambit of our problem, this is the type of optimization we are interested in. Using multithreading to concurrently process an image using multiple cores will allow the optimization and the acceleration of some of the tasks of the algorithm (including image traversing).

Fig. 5.9 illustrates one way to traverse an image concurrently, in this case, using 9 threads. The next sections will explain how to optimize the developed algorithm using multithreading.

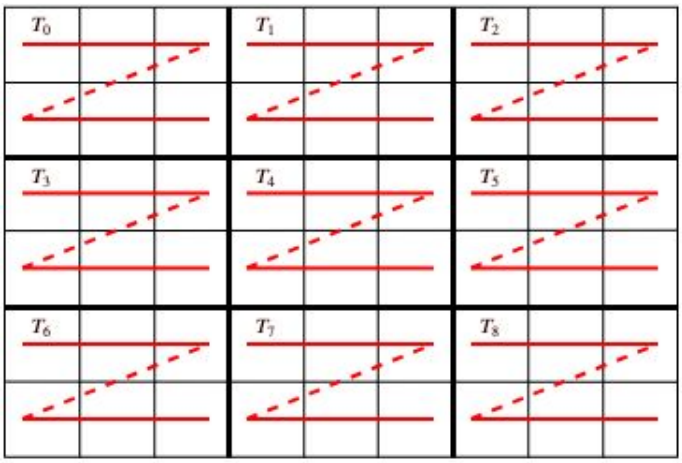


Figure 5.9: Example of image traversing using multithreading.

5.2.4.1 Weight Matrix calculation

After having calculated the sigmoid as mentioned in chapter 3, the next step is to create the weight matrix. This consists in a matrix the same size as the original image in which each element will contain the weight of the pixel in its respective position in the image. To obtain this matrix it is necessary to iterate over every pixel in the IR image and assign a weight to those pixels based on its value on the sigmoid function previously calculated.

As opposed to iterate over the entire image one pixel at a time, such image could be divided into different parts allowing the use of multithreading to, concurrently, calculate the weight matrix more rapidly. Each thread will then calculate the weights to the part of the image assigned to it and, in the end, those separate parts will be joined to form the weight matrix. Fig. 5.10 shows a diagram which exemplifies this process. In this example the image was divided in 9 total parts, and a thread was assigned to each one, producing a total of 9 threads.

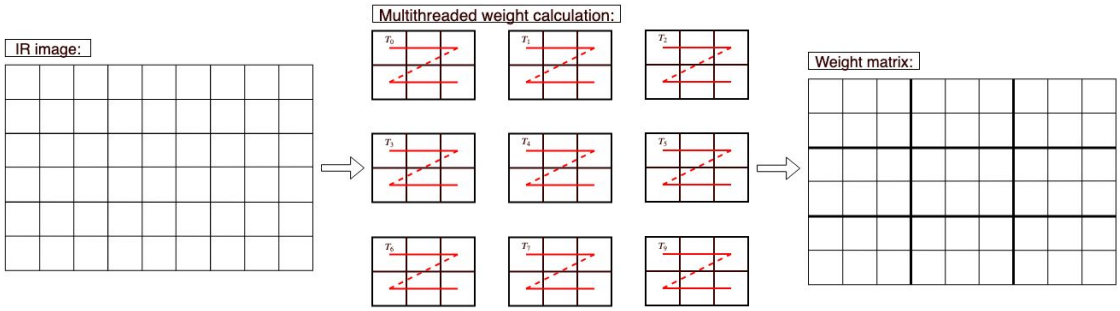


Figure 5.10: Multithreaded weight matrix calculation.

This optimization was implemented and tested in an intel i7 quad core processor running a Linux operating system. Considering that the processor is not hyperthreading, meaning it is only running one thread on each core, there are only four physical threads available. For this reason we tested the algorithm for up to 6 threads, after which the results starting getting slightly worse due to the overhead of context switching the threads. Given this, the results are displayed in table 5.1.

Table 5.1: Processing time of two example videos after optimized weight matrix generation. Both videos were 720x480 pixels and 25 FPS.

No. of Threads	**Video No. 1 (46 s)**	**Video No.2 (2 m 58 s)**
Single (reference)	17.558 s	1 m 5 s
2 Threads	15.210 s	56.116 s
3 Threads	14.976 s	55.253 s
4 Threads	**14.545 s**	**53.488 s**
5 Threads	14.747 s	54.380 s
6 Threads	14.576 s	53.713 s

After optimizing the weighting function when processing the IR image, the multithreading versions shown a speedup of over 1.2.

5.2.4.2 Image Fusion

Another optimization that could be done using multithreading is the image fusion part. It can be seen in the diagram in Fig. 3.1 as the last part of the algorithm. It takes both VIS and IR images as well as their respective weight matrices and creates the output of the algorithm which is the fused image.

This consists in averaging every pixel in the images with each other considering the weights assigned to each pixel. In other words, for every pixel in the fused image, the following formula

is applied:

$$F(x,y) = \frac{vis\_img(x,y).vis\_weight\_matrix(x,y) + ir\_img(x,y).ir\_weight\_matrix(x,y)}{vis\_weight\_matrix(x,y) + ir\_weight\_matrix(x,y)} \quad (5.1)$$

Again, this calculation is completely independent from pixel to pixel, so it is possible to use multithreading to speed up this task. After performing this optimization and using the same test videos and conditions as the previous example, the following results are obtained:

Table 5.2: Processing time of the same example videos in table 5.1 after optimization of weight matrix generation and image fusion.

No. of Threads	**Video No. 1 (46 s)**	**Video No.2 (2 m 58 s)**
Single (reference)	17.558 s	1 m 5 s
2 Threads	11.928 s	43.599 s
3 Threads	9.942 s	35.926 s
4 Threads	**9.355 s**	**33.726 s**
5 Threads	10.727 s	38.996 s
6 Threads	10.990 s	39.023 s

After completing these multithreading optimizations and by analyzing the results, it is possible to determine that this resulted in a speed up of around 1.9. This allowed to process the example videos used in about 20% of the actual video time.

5.2.5 Optimization Recap

In order to demonstrate the results obtained from the optimization processes in a clearer way, this section will sum up such optimizations as well as the speedup expected from those.

Fig. 5.11 shows the high-level diagram of the developed algorithm, highlighting the parts which were optimized, namely the calculation of the average, K-means clusters and weight matrix on the IR Image Processing part, as well as the fusion of the two images in the final part of the algorithm.

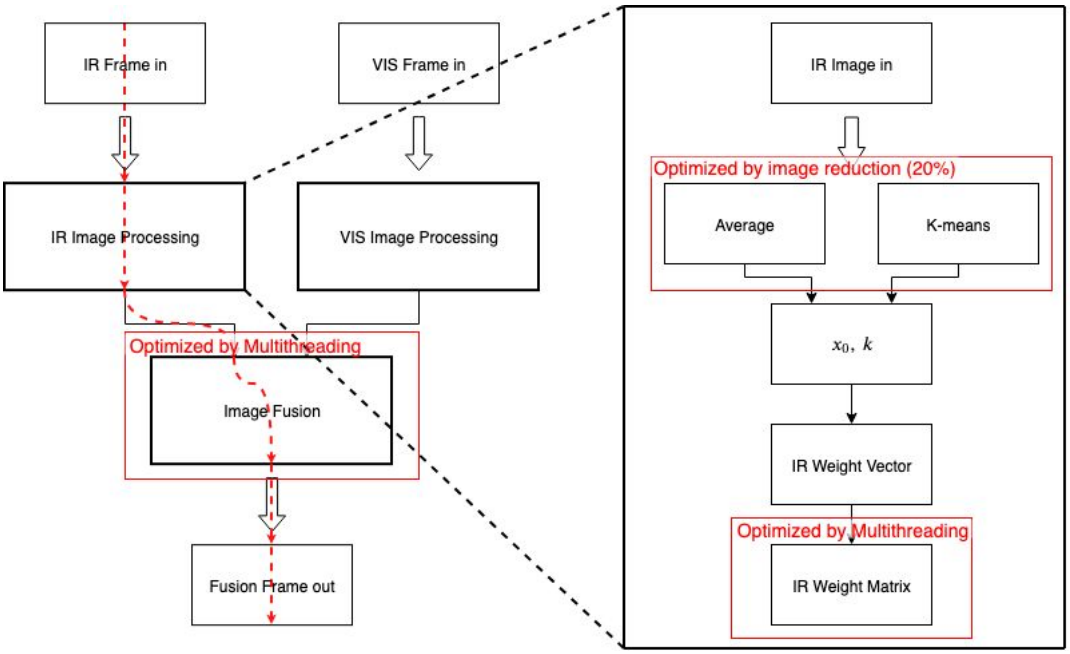


Figure 5.11: Optimization diagram.

The first optimization was to use reduced copies (20%) of the IR image to calculate the average and the K-means cluster values. This optimization led to a speedup of 1.5.

The second optimization was the use of multithreading to calculate the weight matrix of the IR image. This optimization led to a speed up if 1.2 in relation to the previous optimizations. So far the total speedup value is 1.8.

The third, and final, optimization was the use of multithreading to finally calculate the fusion between the VIS and IR images. This optimization led to a speed up of 1.9 in relation to the first optimization (the resize optimization). Leading to a total optimization speedup of 2.85.

As previously mentioned, and considering the video settings used for the tests (25 FPS, 720x480 pixels) and the conditions in which the program was running (laptop, i7 quadcore processor, 4 threads total), the processing time of performing the image fusion using the developed algorithm is around 20% the total video time. This proves that, in these conditions, the algorithm can comfortably run in real-time.

By rising the framerate or the video resolution, or by decreasing the processing power in which the program will be executed, this processing time percentage will rise. However, it is still quite far from reaching 100% which is the limit at which the program can no longer run in real time. Thus, it is possible to conclude that there is a big enough margin for this algorithm to fulfil the real time requisites of the system for the most common surveillance video settings.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Initially, a study of the literature was done in order to select the algorithms that best fitted the purpose. Considering the main objective of this dissertation, which was to achieve video fusion in real-time, the execution time would be essential for the selection of the algorithm. For this reason, the algorithms which had higher execution times were discarded, meaning that pixel-level algorithms, which were the fastest to execute, would be selected for further studies. Unfortunately however, the main disadvantage of these types of algorithms is the quality, which can be a little inferior than other types, such as feature based algorithms. This led to the development of a pixel-level solution, based on the average of two images, that improves the fusion quality while maintaining a low execution time.

A study on thermal imaging (infrared) was then made as a way to understand what are, generally, the most important parts on this type of images. Parts which will then be considered for the fusion while, on the other hand, the parts with no interest will be discarded. After this study it was concluded that the most interesting parts are the ones which intensity value is higher. With this in mind an algorithm that analyzed the image and assigned weights to every pixel was developed. This weights are given as a function of the intensity value of the respective pixel, and the function is obtained from characteristics of the image as a whole, such as the average for instance.

After assigning all the weights, the weighted average between the visible and the infrared image was done, thus generating the final image, the fusion.

After being developed, the final solution was adapted to video streams and implemented in C++. Lastly a study about the feasibility of developing a real-time system for video fusion was done. To achieve this, optimizations, such as task parallelization, had to be made. These optimizations led to a speedup of the execution time of the algorithm which, consequently, allowed the system to work in real-time for the most commonly used video settings in surveillance.

To sum up, this study allowed to achieve the objectives proposed by the company. The internship aimed at:

- Performing a study on the literature of image fusion;

- Select an algorithm that could be adapted for video fusion;
- Improve the algorithm (if necessary);
- Implement the system for real-time execution

6.2 Future Work

In the development of this work, one of the great difficulties that arose was the lack of data variety for performing quality tests on the algorithm. It was quite difficult to find synchronized videos of visible and infrared images with which it was possible to work. As such, an important work to be develop would be a system with which it would be possible to generate test data containing various different features. Such features could include day, night, foggy or smoky images, as well as images containing both faraway and close objects, etc. Some images taken from vehicles could also be collected, like aerial images, for instance. This would allow to make adjustments to the algorithm to try to adapt it to many different cases.

In relation to the work already developed, more specifically in the implementation part, the limits of the algorithm developed can be measured. As mentioned in the document, the algorithm was tested for images at 25 FPS, with a resolution of 720x480 pixels and processed on a laptop with four cores. These tests resulted in a large margin for real-time processing, however, it was also mentioned that this margin would decrease if the frame rate or resolution was increased, or if the processing power was reduced. Having said that, an analysis of the limits of the algorithm can be made for which it will still work in real time.

Lastly, also in the implementation part, the optimization of the algorithm using the GPU could be implemented and tested. This implementation might allow to further increase the parallelism level of the system and, most likely, reduce its execution time. This would require a study on the parallelization using the GPU, to try to understand if it is a viable option and, if so, the subsequent implementation and testing.

References

[1] Mayssaa Al Najjar, Milad Ghantous, and Magdy Bayoumi. *Video surveillance for sensor platforms algorithms and architectures*. Springer, 2014.

[2] Mohammad Bagher Akbari Haghighat, Ali Aghagolzadeh, and Hadi Seyedarabi. Multi-focus image fusion for visual sensor networks in dct domain. *Computers & Electrical Engineering*, 37(5):789–797, 2011.

[3] Gonzalo Pajares and Jesus Manuel De La Cruz. A wavelet-based image fusion tutorial. *Pattern recognition*, 37(9):1855–1872, 2004.

[4] K Kalaivani and Y Asnath Victy Phamila. Analysis of image fusion techniques based on quality assessment metrics. *Indian Journal of Science and Technology*, 9(31), 2016.

[5] Gemma Piella. A general framework for multiresolution image fusion: from pixels to regions. *Information fusion*, 4(4):259–280, 2003.

[6] Moira I Smith and Jamie Paul Heather. A review of image fusion technology in 2005. In *Thermosense XXVII*, volume 5782, pages 29–46. International Society for Optics and Photonics, 2005.

[7] John J Lewis, Robert J O'Callaghan, Stavri G Nikolov, David R Bull, and Nishan Canagarajah. Pixel-and region-based image fusion with complex wavelets. *Information fusion*, 8(2):119–130, 2007.

[8] Scott Meyers. Cpu caches and why you care, 2013.

[9] SS Bedi and Rati Khandelwal. Comprehensive and comparative study of image fusion techniques. *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, 3(I):2231–2307, 2013.

[10] Zhong Zhang and Rick S Blum. Region-based image fusion scheme for concealed weapon detection. In *Proceedings of the 31st Annual Conference on Information Sciences and Systems*, pages 168–173, 1997.

[11] Alexander Toet, Jan Kees IJspeert, Allen M Waxman, and Mario Aguilar. Fusion of visible and thermal imagery improves situational awareness. *Displays*, 18(2):85–95, 1997.

[12] Cle Pohl and John L Van Genderen. Review article multisensor image fusion in remote sensing: concepts, methods and applications. *International journal of remote sensing*, 19(5):823–854, 1998.

[13] Belur V Dasarathy. Information fusion in the realm of medical applications-a bibliographic glimpse at its growing appeal. *Information Fusion*, 13(1):1–9, 2012.

[14] Mongi A Abidi and Rafael C Gonzalez. *Data fusion in robotics and machine intelligence*. Academic Press Professional, Inc., 1992.

[15] Oliver Rockinger. Image sequence fusion using a shift-invariant wavelet transform. In *Image Processing, 1997. Proceedings., International Conference on*, volume 3, pages 288–291. IEEE, 1997.

[16] Paul R Hill, Cedric Nishan Canagarajah, and David R Bull. Image fusion using complex wavelets. In *BMVC*, pages 1–10. Citeseer, 2002.

[17] Gemma Piella. A region-based multiresolution image fusion algorithm. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 1557–1564. IEEE, 2002.

[18] Zhou Wang and Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002.

[19] Dejan Drajic and Nedeljko Cvejic. Adaptive fusion of multimodal surveillance image sequences in visual sensor networks. *IEEE Transactions on Consumer Electronics*, 53(4), 2007.

[20] CS Xydeas, , and Vladimir Petrovic. Objective image fusion performance measure. *Electronics letters*, 36(4):308–309, 2000.

[21] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[22] Gemma Piella and Henk Heijmans. A new quality metric for image fusion. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III–173. IEEE, 2003.

[23] Eduardo Antônio Barros da Silva and Andreas Ellmauthaler. Geração de um banco de imagens e vídeos infravermelhos e visíveis, 2013.

[24] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[25] Dongju Liu and Jian Yu. Otsu method and k-means. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 1, pages 344–349. IEEE, 2009.

[26] CA Balaras and AA Argiriou. Infrared thermography for building diagnostics. *Energy and buildings*, 34(2):171–183, 2002.

[27] Zhou Wang, Ligang Lu, and Alan C Bovik. Video quality assessment based on structural distortion measurement. *Signal processing: Image communication*, 19(2):121–132, 2004.

[28] M Shanthi and A Anthony Irudhayaraj. Multithreading-an efficient technique for enhancing application performance. *International Journal of Recent Trends in Engineering*, 2(4):165, 2009.