FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Artificial Intelligence for Vineyard Trap Segmentation and Perspective Correction

João Costa

FOR JURY EVALUATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Internal Supervisor: António Pedro Aguiar

External Supervisor: Pedro Faria, MSc

July 30, 2021

© João Costa, 2021

Resumo

O elevado valor económico associado à produção de uvas e os efeitos das alterações climáticas na viticultura tornam ainda maior a necessidade de um controlo rigoroso de pragas nas vinhas. Este controlo é realizado através da análise de armadilhas para insectos previamente implantadas nos campos de produção.

Uma vez que os sistemas de monitorização baseados em visão de computador são tendência em diversas áreas, existem algumas soluções disponíveis no mercado para este propósito. Estas consistem em nós de armadilhas inteligentes que identificam e contam automaticamente os insectos. Totalmente equipados, apresentam módulos para aquisição de imagens, poder de processamento, comunicação e fornecimento de energia. Contudo, a necessidade de hardware específico para cada nó torna-os inacessíveis para alguns produtores.

Tendo isso em conta, o projecto EyesOnTraps visa resolver este problema através da criação de uma solução baseada em smartphones que seja mais acessível e facilmente escalável. A utilização de smartphones para o processo de aquisição de imagem vem com a tarefa adicional de corrigir a perspectiva da armadilha. Ao contrário da estratégia anterior, o ângulo de captura de imagem não é fixo. Uma vez que este procedimento passa a ser feito por operadores humanos, as variações esperadas na aquisição de imagem podem ter impacto na detecção automática de insectos. De modo a tornar o resultado deste processo uniforme, é necessária uma transformação da imagem para obter uma vista frontal da armadilha.

Esta tese de mestrado foca-se na melhoria de uma metodologia previamente desenvolvida, implementada na aplicação móvel do projeto para segmentar e corrigir a vista da armadilha.

Ao contrário da abordagem anterior que recorria a métodos de visão de computador convencional, a dissertação explora uma metodologia que aplica técnicas de detecção de pontos de interesse para segmentar a armadilha numa imagem adquirida num smartphone seguido de um método de correcção de perspectiva diferente do original para obter a vista frontal pretendida.

A utilização desta abordagem leva a uma melhor segmentação, com um índice de Jaccard de 97,5% (comparado com um anterior 96,7%), e um tempo de inferência 59 vezes mais rápido quando comparado com a abordagem anteriormente utilizada.

Abstract

The high economic value associated with grape production and the effects of climate change in viticulture, make the need for close pest monitoring in the vineyard an even more concerning issue. This monitoring is performed by analyzing insect traps previously deployed in the production fields.

As image-based monitoring systems are trending in many areas, there are some insect trap solutions available in the market. These consist of intelligent trap nodes that automatically identify and count insects. Fully equipped, they feature modules for image acquisition, processing power, communication and power supply. However, the need for all this specific hardware for each trap node makes them unaffordable to some producers.

Tacking that into account, the EyesOnTraps project aims to tackle this problem by creating a smartphone-based solution that is more affordable and easily scalable. By using smartphones for the image acquisition process comes with the additional task of correcting the view of the trap. Unlike the previous intelligent node strategy, the image capture angle is not fixed. Since this procedure is now done by human operators, the expected variations in image acquisition could impact automatic insect detection. To make the result of this process uniform, an image transformation to achieve a frontal view of the trap is required.

This master thesis focus is to improve a previously developed methodology implemented as a mobile application to segment and correct the view of the trap.

Unlike the previous approach that resorts to conventional computer vision, the dissertation explores a methodology that applies landmark detection techniques in order to segment the trap in a smartphone acquired image and then applies a different perspective correction method to generate the frontal view of the trap.

Using this approach leads to a better segmentation, with a Jaccard index value of 97.5% (compared to a previous 96.7%), and a computing time 59 times faster when compared to the previously explored approach.

Agradecimentos

Agradeço aos envolvidos neste trabalho:

À Fraunhofer Portugal AICOS pela oportunidade. Especialmente ao Pedro Faria, pelo apoio constante no decorrer desta experiência.

Ao professor António Pedro Aguiar, pela supervisão durante este período.

A todos os que me acompanharam durantes estes 5 anos:

A todos os amigos que fiz na FEUP.

Ao meu grupo de amigos mais próximos, os mesmos de sempre.

À Marta, por ser quem é.

À minha familia, por todo o apoio que me deram e dão.

Pessoalmente e academicamente agradecido,

João Costa

"It never gets easier, you just get better."

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 3
- 1.3 Objectives . . . 4

2 Literature Review and Problem Statement **5**
- 2.1 Deep Learning and Computer Vision . . . 5
  - 2.1.1 Convolutional Neural Networks . . . 7
  - 2.1.2 Training Phase . . . 7
- 2.2 Image Segmentation . . . 8
- 2.3 Landmark Detection . . . 10
- 2.4 Planar Homography . . . 11
  - 2.4.1 Homography Estimation . . . 12
- 2.5 Similar Works . . . 13
  - 2.5.1 Document Perspective Correction . . . 13
  - 2.5.2 Document Unwarping . . . 14
- 2.6 Problem Statement . . . 15

3 Methodology **19**
- 3.1 Approach Definition . . . 19
- 3.2 Data Preparation . . . 20
  - 3.2.1 Available Data . . . 20
  - 3.2.2 Dataset Building . . . 20
- 3.3 Trap Segmentation . . . 22
  - 3.3.1 CNN Architectures . . . 22
  - 3.3.2 Parameters for Results Assessment . . . 23
- 3.4 Perspective Correction . . . 24
  - 3.4.1 Algorithms . . . 24
  - 3.4.2 Rejection of Demarked Transformations . . . 25
  - 3.4.3 Parameters for Results Assessment . . . 25
- 3.5 Implementation . . . 25

4 Results **27**
- 4.1 Trap Segmentation . . . 27
  - 4.1.1 Network Training and Selection . . . 27
  - 4.1.2 Dataset Critical Analysis . . . 28
- 4.2 Perspective Correction . . . 29
  - 4.2.1 Methodology Selection . . . 29

4.2.2 Demarked Corrections Rejection 30
4.3 Current Approach Comparison 30
4.3.1 Trap Segmentation 30
4.3.2 Perspective Correction 31
4.3.3 Inference Time 32
4.3.4 Visual Results 32

5 Conclusions and Future Work **35**
5.1 Objectives Satisfaction 35
5.2 Future Work 36

A **37**
A.1 Architecture of our Network 37

References **39**

List of Figures

1.1 Alto Douro wine region . . . 1
1.2 Trapview monitoring node . . . 2
1.3 Trap image acquisition using smartphones . . . 3

2.1 ML and DL workflows . . . 6
2.2 Perceptron . . . 6
2.3 DL and data availability relation . . . 7
2.4 CNN typical architecture . . . 8
2.5 Segmentation of the trap and corner position regression used in the original approach 9
2.6 Brain tumor segmentation using a FCN . . . 9
2.7 Example of facial landmarks . . . 10
2.8 2D geometric transformations . . . 11
2.9 Image perspective transformation . . . 12
2.10 RANSAC algorithm . . . 13
2.11 Document perspective correction . . . 14
2.12 Document unwarping . . . 15
2.13 Loss of information when using only the corners for trap segmentation . . . 16
2.14 Bad results in the original approach . . . 17

3.1 Images of traps in different image acquisition environments . . . 20
3.2 Trap images with annotated landmarks . . . 21
3.3 Grayscale image visual impact . . . 21
3.4 Resized trap images . . . 22
3.5 Data augmentation generator . . . 23
3.6 Four point mapping for perspective correction of a trap . . . 24
3.7 Perspective correction destination grids . . . 24
3.8 Our algorithm block diagram . . . 26
3.9 Application flow . . . 26

4.1 Landmark detection JI distribution . . . 29
4.2 Landmark detection weaker results . . . 29
4.3 Decision tree to reject transformations . . . 31
4.4 Our algorithm visual results . . . 33
4.5 Comparison of our visual results with original approach . . . 34

List of Tables

2.1 Hierarchy of 2D transformations 11

3.1 Dataset distribution 20

4.1 Model training results for white traps 28
4.2 Model training results for yellow traps 28
4.3 Trap segmentation results for increased data augmentation for the MobileNetV2 model trained with MAE loss 28
4.4 Perspective correction methodologies results 30
4.5 Old versus new approach trap segmentation results 31
4.6 Difference in our algorithm using 16 and 4 points for the segmentation of the trap and 16 for the ground-truth mask 31
4.7 Original algorithm inference time 32
4.8 Our approach inference time 32

Abbreviations and Symbols

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
FCN	Fully Convolutional Network
FhP-AICOS	Fraunhofer Portugal AICOS
MAE	Mean Absolute Error
MSE	Mean Squared Error
JI	Jaccard Index
IoU	Intersection over Union
Std	Standard Deviation
DoF	Degrees of Freedom
DTC	Decision Tree Classifier
CE	Controlled Environment
NCE	Non Controlled Environment
NN	Neural Network
OCR	Optical Character Recognition
R-CNN	Regional Convolutional Neural Network
RANSAC	Random Sample Consensus
SOTA	State-Of-The-Art

Chapter 1

Introduction

1.1 Context

Grape production has a big socioeconomic importance in many countries, as it is estimated that the world area occupied by vineyards is approximately 7.4 million hectares [1]. In Portugal, the Alto Douro region economy is highly dependent on this, not only due to wine sales but also to its whole production chain and related businesses, such as tourism.



Figure 1.1: Alto Douro wine region [2]

Being a constant concern to the producers, pests pose a threat to grapevines health. Insects and mites can directly or indirectly damage the vines, by eating parts of them or beings vectors of diseases that interfere with key processes like photosynthesis or respiration [3].

The Food and Agriculture Organization of the United Nations states that climate change has increased crop damage [4], as warmer winters may not be so harsh on pests and diseases from warmer regions. This phenom changes the spectrum of existing threats in the different areas [3], which makes its monitoring even more relevant. In order to assess these plagues evolution, which serves as a support for corrective action decisions, human operators periodically examine sticky traps that are placed in the production fields, to identify and count the captured insects. The two

more common trap types are the delta (white) and the chromotropic (yellow) ones, which attract insects by pheromones or colour, respectively.

This trap monitoring is mainly done on-site by winegrowers. However, acquiring images of the traps allows a remote analysis by taxonomy specialists. This motivated the study of image-based techniques in areas like farming diseases [5, 6] and crops [7, 8] monitoring. With an intent similar to the one we purpose, there are some solutions being commercialized, like ForesightIPM [9], Metos [10] and Trapview [11], which make use of intelligent trap systems, such as the one in figure 1.2. They are composed of nodes featuring inbuilt cameras, set at a fixed distance and within optimal focus. Some of them even ensure that the obtained images are free from shadows and specular reflections, performing automatic insect classification and enumeration.



Figure 1.2: Trapview monitoring node

The high degree of automation on these solutions require a big investment in the infrastructure. Every monitoring module demands specific hardware for image acquisition, processing power, communication and power supply. Thus, the need of expanding the surveilled area involves additional investment, adding to the cost of the routine maintenance of the equipment, which can be prohibitive to some producers. Also, these monitoring nodes do not create a system independent of the rest of the vineyard work, as they still require trained employees to maintain the trap part of the nodes (cleaning, replacing the traps or changing the pest attractant), also needed in the normal placement of the traps in the production fields.

Interestingly, current smartphones have all the specifications required to build a monitoring node. The global smartphone penetration allied with the increasing computational power and high-definition cameras on these devices makes using them an interesting approach that can drastically cut costs [12].

From the mentioned existing solutions, Trapview [11], in addition to the monitoring nodes, also allows the user to acquire images of traps using a smartphone application, but the fact of it being a fully manual procedure, without any software layer for quality and adequacy control, makes it prone to human error.

Shifting from what is available in the market, the EyesOnTraps project [13], a partnership from Fraunhofer Portugal AICOS with ADVID [14] and GeoDouro [15] aims to develop an easily

scalable mobile solution for plague prevention in vineyards, supporting automated insect detection in traps and treatment recommendations for the recognized threats.

1.2 Motivation

Creating a smartphone-based trap monitoring solution requires extra work when compared to the previously existing ones. The handheld image acquisition under non-controllable settings is very difficult, due to variations in light exposure and the need of being sure that the image is of acceptable quality for further analysis. Also, unlike in the monitoring modules, where the image capture angle is fixed, the images can present various perspectives of the trap. Therefore, this new approach creates the need to perform an image transformation to get a uniform frontal view of the trap, since the insect detection part can be influenced by these variations. As an example, wings dimensions have a big importance in insect identification and can look different depending on the image capture angle. The normal trap image acquisition process can be seen in figure 1.3.



Figure 1.3: Trap image acquisition using smartphones

The method implemented in the application to correct the view of the trap works by segmenting the trap from the background and performing its perspective correction. The methodologies used are built around conventional computer vision and geometric approaches, which have proved to sometimes fail under in-the-wild conditions, such as complex scenarios and deformed traps, causing severe image distortions. This creates a need to develop another approach using a different technique. Handheld image capture allied with Artificial Intelligence (AI) has shown promising results in the health area [16, 17, 18]. Also, Deep Learning (DL) methods were able to achieve great results in many computer vision tasks, performing better when compared to traditional methods [19]. However, it has never been explored in the viticulture sector.

Therefore, this master thesis, in the scope of the EyesOnTraps project, plans to rebuild the mentioned modules of the application by developing an algorithm that presents a better performance under those adverse conditions.

1.3 Objectives

Given the known drawbacks in the current explored methodologies, we will explore the usage of state-of-the-art (SOTA) AI-based approaches to address the segmentation and the perspective correction of traps in more complex and deformed scenarios. The main goals of this master thesis were separated into the following topics:

- **Image segmentation using DL techniques, such as automatic landmark detection, in order to deal with folds, curved and/or partially broken corners of the trap in a more efficient way.**
- **Comparative study of multiple methods for perspective correction**.

Chapter 2

Literature Review and Problem Statement

To understand the concepts referred in the bibliography, we did some preliminary work. Section 2.1 covers the DL and computer vision relationship. We then further explain what are Convolutional Neural Networks (CNN), followed by how these can be trained.

We then present a broad study of image segmentation techniques in section 2.2 and landmark detection in section 2.3. Section 2.4 covers a general view of homography and how can it be estimated.

In section 2.5, methodologies that can be used as inspiration to solve our problem are shown, such as recovering the homography of documents and their unwarping.

Lastly, we clear up the problems of the current methodology being used in the project and what we aim to solve in section 2.6.

2.1 Deep Learning and Computer Vision

Computer vision, one of the fields of AI, focuses on making computers process visual data. It has been able to surpass human capabilities in some tasks, like defect detection, due to a much quicker inference time and capacity of identifying small imperfections, imperceptible to the human eye. Some of the algorithms used have the ability to improve their performance with experience. They belong to the Machine Learning (ML) domain, which plays a major role in self-driving cars, medical condition diagnosis, and face recognition. Although these models can get progressively better at their function, they still need to be manually adjusted when an inaccurate prediction is returned. A comparison of the ML and DL workflows can be seen in figure 2.1.

The DL field is related to Artificial Neural Networks (ANN), a model inspired by the human nervous system. The fundamental block of these networks are artificial neurons that receive an input and produce outputs. The links between neurons have weights and biases that are adjusted during training. The weights control the influence of a certain input in the output of the neuron.

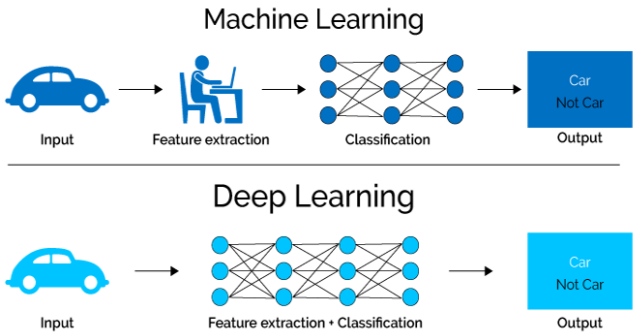


Figure 2.1: Machine Learning and Deep Learning workflows [20]

Representing a single neuron, the most basic model is the Perceptron, seen in figure 2.2, used to classify binary data in a linear way.

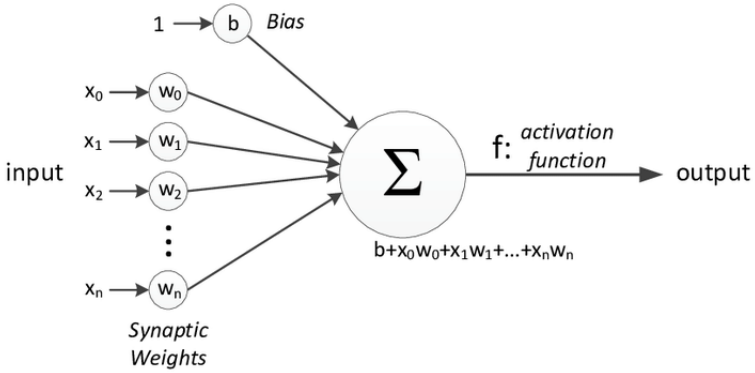


Figure 2.2: Perceptron model [21]

However, when data classification requires a non-linear function, a multiple layer Perceptron is needed, which already classifies as a Neural Network (NN). Usually, the more complex the problem, the more layers this model needs, which leads to the adjective "deep".

Although DL growth has been immense in the last years and showed great results in many application domains, there are still many areas left where to explore its capabilities. The key reason for this success is the current availability of large annotated datasets for certain applications that allow a better training of a network that has a very large learning capacity (figure 2.3). Experimental results show SOTA performance using DL compared to ML approaches in image processing and computer vision [19].

Regarding the way these models learn, it can be divided into three types: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

Supervised learning, the most common one, uses example input-output pairs to teach a model to predict the output for an unlabeled input. It needs enough labeled data to train the algorithm. The labels are used to correct and modify the model in case of wrong predictions. Problems that use this type of learning can be classified as classification (predicts a class label) or regression problems (predicts a numerical value).

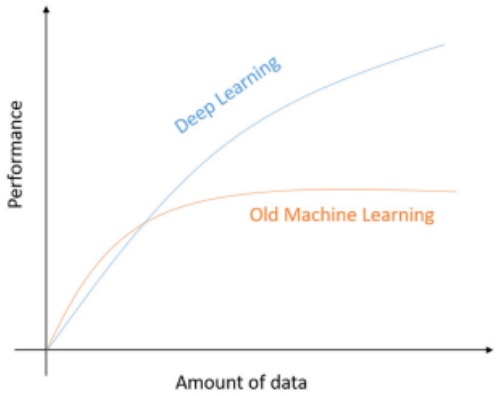


Figure 2.3: DL and data availability relation [22]

2.1.1 Convolutional Neural Networks

This Deep Neural Networks (DNN) family is more suitable for image processing when compared to others. The pattern of the links between neurons resembles the animal visual cortex organization. The typical architecture of a CNN can be seen in figure 2.4. It has three main parts:

- Convolutional Layer: Extracts a feature map containing important features, such as corners and edges, by performing a convolution between the input and kernels that represent the weights. The output of this convolution then goes through an activation function that is used to add non-linearity to the network, like ReLu or sigmoid, depending on the task purpose.
- Pooling Layer: Normally placed after a convolutional layer, the pooling layer reduces the spatial size of the input feature maps to decrease computational costs. This operation can be of various types, being Max-pooling the most used, which selects the highest value from the input maps, discarding the others.
- Classification Layer: Outputs the result of the problem. The image obtained from the previous operations is flattened and given to this layer. It is fully connected, meaning that each neuron is connected to every other in the following layer.

There are other types of layers that are not mandatory for a CNN model but can improve it, such as Batch Normalization layers, which standardize the inputs for each batch, making the network training faster and more stable, and Dropout layers, that help reduce the overfitting chances by randomly dropping some neurons contribution during the training phase.

2.1.2 Training Phase

To assess how good a NN is for a certain task, a loss function (or cost) is used. The training phase's goal is to reduce these by achieving good predictions. This procedure is the last computation step in the output calculation process (Forward Pass) and the first one in the refreshing of the weights based on the loss and the optimizer being used (Backward Pass). These optimizers have a learning

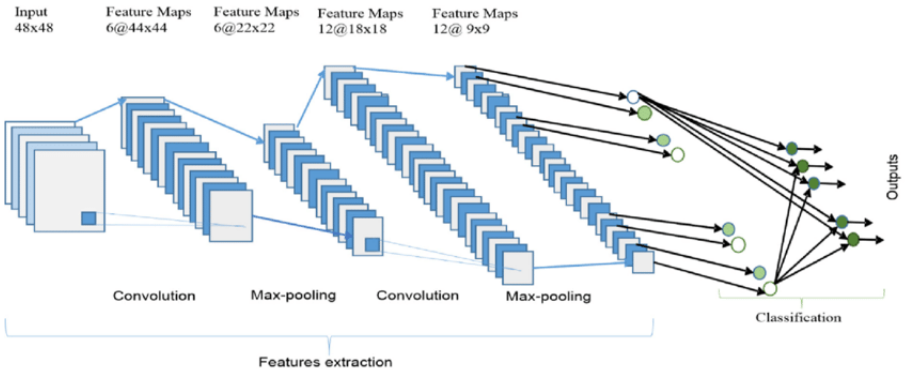


Figure 2.4: CNN typical architecture [19]

rate that can be adjusted in order to control how much the model changes when reacting to the loss. Both of these passes make an iteration, where a subset of data, also known as batch, is passed through the model. It can be of variable size. After passing the whole dataset in batches, the model went through what is called an epoch. We can train the models for how many epochs we want. The more we train the model, loss decreases and its performance gets better, until it reaches a plateau.

In some applications, it is possible to reduce the training time of the network and the amount of data it needs to achieve good performance by training a model that was used in a previous related problem. This technique is named transfer learning.

However, training can lead to overfit if the amount of data and its complexity is too small. This happens when the model learns the details of the training data so well that it weakens the performance when handling new cases. Data augmentation can be used to help avoid this problem. The available data is artificially expanded by performing geometric transformations and cropping. It should mimic the conditions our model can encounter.

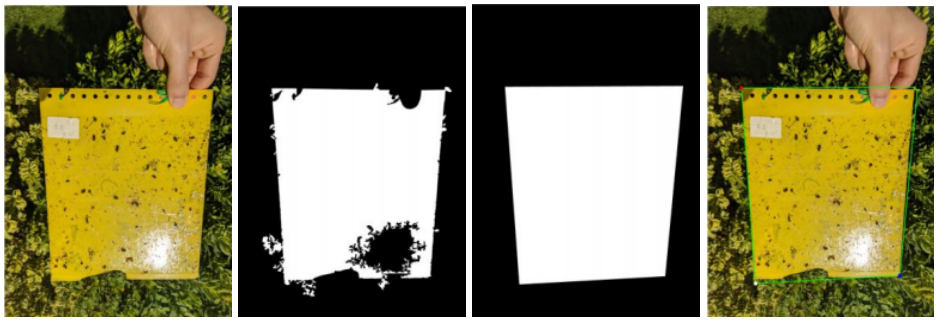
To assess the predictive performance of the trained models and assess their generalization ability, a k-fold cross validation can be used. This method is very useful in cases of limited data. By splitting it in k equal parts, k-1 are used for training and 1 is used for testing purposes, repeating this process k times until all sets have used as test set.

2.2 Image Segmentation

Image segmentation is the act of partitioning images into multiple segments or objects [23]. It is used to extract the desired object from its complex background. Over the last few years, DL models lead to a new generation of methods that achieved the highest accuracy rates in several popular benchmarks [24].

This procedure can be done by giving labels to each pixel (semantic segmentation) or by, extending this first, identifying to what instance it belongs (instance segmentation).

Work-wise, the project's original algorithm we want to improve works by creating a segmentation mask for the trap using Otsu's method [25], followed by the Douglas-Pecker algorithm [26] to find the corners of the trap, as seen in figure 2.5.



(a) original (b) segmentation mask (c) polygon approximation (d) located corners

Figure 2.5: Segmentation of the trap and corner position regression used in the original approach [1]

During the literature review of solutions for image-based plague prevention, only Xuesong et al. [27] segmented insect traps from the background and corrected its perspective. They used the GrabCut method [28] and the Hough Transform [29] to obtain the four vertices of the distorted trap in the image.

However, we performed a review of SOTA segmentation methods, not going further in traditional computer vision ones.

Starting with Fully Conventional Networks (FCN), they have been applied to problems such as segmentation of brain tumors [30](example in figure 2.6), skin lesions [31], and iris [32].

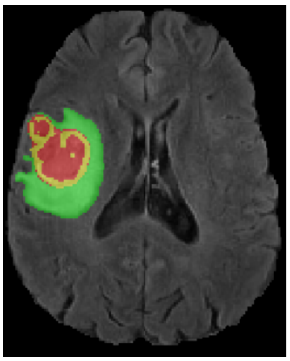


Figure 2.6: Brain tumor segmentation using a FCN [30]

Convolutional encoder-decoder architectures are widely used too. Noh et al. [33] work and Badrinarayanan et al. [34] SegNet proved to be good approaches for general segmentation problems [24]. One limitation of these models is the loss of image fine-grained information through the encoding process [24]. HRNet [35] showed great progress regarding this issue, serving as the backbone to newer research on semantic segmentation [24].

Other models inspired by FCN and encoder-decoders initially developed for medical/biomedical image segmentation tasks, such as U-Net [36] and V-Net [37] also proved to produce good results outside this domain [24].

The Regional Convolutional Neural Network (R-CNN) [38] and its extensions (Fast R-CNN [39], Faster R-CNN [40], Masked-RCNN [41]) have proven to be successful in object detection applications. Some of the extensions of R-CNN have been used to address the task of simultaneously performing object detection and semantic segmentation. Mask R-CNN, presented by He et al. [41] beat all previous benchmarks on many COCO [42] challenges for object instance segmentation [24].

2.3 Landmark Detection

Landmarks are important points in an image. Its detection is a fundamental component in many face analysis tasks, such as attribute inference [43], verification [44] and recognition [45]. The used algorithms are capable of locating points around facial contour, capturing the rigid and non-rigid deformation due to head movements and expressions, as seen in figure 2.7 [46]. Other fields of study, like human pose estimation [47], medical imaging [48] and clothing analysis [49] also make use of this technique.

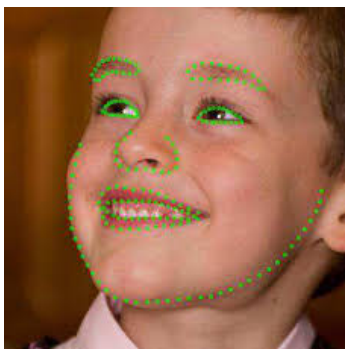


Figure 2.7: Example of facial landmarks [50]

In a survey, Wu and Ji [46] state that DL-based regression methods achieve SOTA performances on several databases for facial landmark detection, being more robust than any other methods when handling images captured under extreme conditions like illumination, occlusions and extreme head poses.

Work-wise, Sun et al. [51] predicted five facial keypoints on an input image using a CNN with four convolutional layers. By applying several changes to this model, Ranjan et al. [52] performed face detection, landmark localization, pose estimation, and gender recognition all at once. Zhou et al. [53] also changed [51] model to detect 68 landmarks instead of 5.

2.4 Planar Homography

We can transform an object in a 2D plane by applying a geometric transformation. A small illustration of the different types that exist can be seen in figure 2.8, where their Degrees of Freedom (DoF) and characteristics they preserve are in table 2.1.

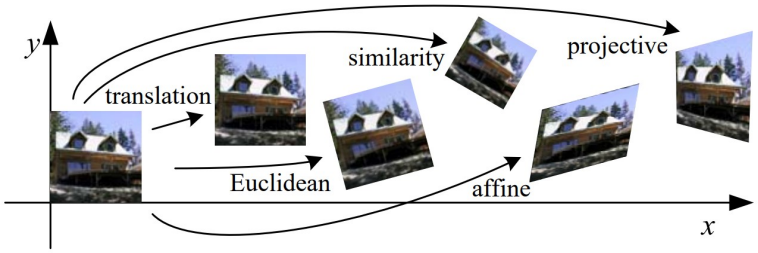


Figure 2.8: 2D geometric transformations [23]

Transformation	DoF	Preserves
Translation	2	Orientation
Rigid (Euclidean)	3	Lenghts
Similarity	4	Angles
Affine	6	Parallelism
Projective	8	Straight Lines

Table 2.1: Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below, being the column Preserves incremental from the bottom to the top [23]

Being the most interesting to our work from the ones mentioned above, planar homography, also known as perspective transform or projective transformation, relates any two images of the same planar surface in space, assuming a pinhole camera model. This means that points of a surface in one view are associated to points in another, using a homogeneous representation (equation 2.1). The projective classification comes from the fact of this transformation only depending on the intersection of planes with lines. This transformation is usually represented by a matrix H, up to a scale s. As mentioned in table 2.1, the transformation is determined by 8 parameters, being generally normalized with $h_{33} = 1$. An example of this transformation can be seen in figure 2.9.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$



(a) original (b) desired view (c) result

Figure 2.9: Image perspective transformation in order to get a desired view of a chessboard [54]

2.4.1 Homography Estimation

The estimation of the homography between two views is very important in applications involving multiple view geometry [55], such as calibration [56] and metric rectification [57]. Applications that use perspective geometry, like image registration and image mosaicing [58, 59] also rely on this, as well as tracking applications that use multiple cameras to build projector-camera systems [60].

The general pipeline for homography estimation is composed of two steps:

- Correspondences Detection: Done by identifying unique image features across views. Famous algorithms for that purpose are the Scale Invariant Feature Transform (SIFT) [61], the Speeded Up Robust Features (SURF) [62] and the Binary Robust Independent Elementary Features (BRIEF) [63].
- Homography Estimation: The correspondences are used to estimate the homography using a least-squares estimation that uses the best fit for all the points.

However, using incorrect matches to perform this action can lead to poor results. They are usually filtered using the Random Sample Consensus (RANSAC) [64], a robust technique that can be applied to a set of points that contains a lot of outliers. It works by picking a random subset of n point pairs to perform a least-squares estimation, where n is minimum number of points needed to perform it. If the remaining ones are good fits, it classifies them as hypothetical inliers. After a fixed amount of iterations, the estimation with the less amount of hypothetical outliers is given as the output of the algorithm, as seen in figure 2.10.

Regarding evaluation metrics, the Euclidean image distance is used. It represents the sum of distances between the transformed points x' and their corresponding correct annotation in the second image, as seen in equation 2.2. The good or bad fits previously mentioned are classified using this distance and a fixed threshold.

$$E = \sum_i d(x_i^{'}, Hx_i)^2 \tag{2.2}$$

These methods' performance gets worse when handling severely contaminated data. Applying DL in this field also proved to be a good choice. DeTone et al. [65] and Zheng et al. [66] works

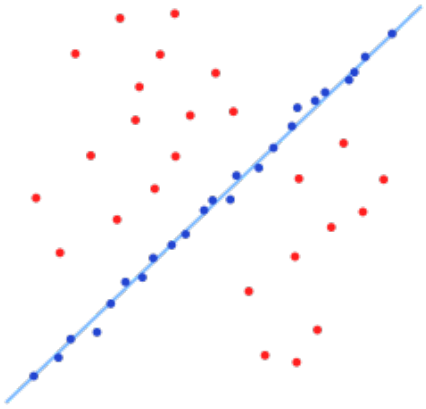


Figure 2.10: Visual explanation of the RANSAC algorithm. Blue - Inliers, Red - Outliers

showed better performances under variations in noise, illumination, and very different perspectives when compared to several traditional methodologies.

2.5 Similar Works

We covered the following study fields since their procedure can be replicated in our work. Instead of insect detection in traps, these following works study transformation methods to provide a document image that is suitable for automatic information extraction and content analysis.

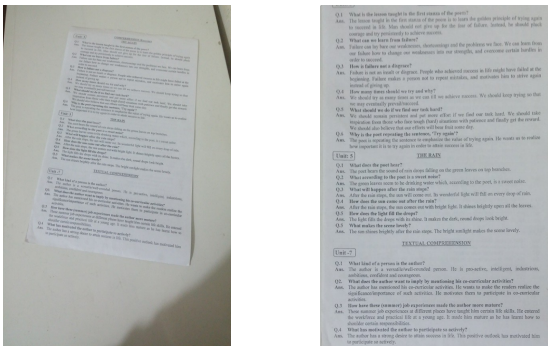
2.5.1 Document Perspective Correction

Applications for document image segmentation and perspective correction have gained popularity with the increased use of smartphones. These images are significantly different from the ones captured with dedicated hardware. The sheets can be laying in complex backgrounds, under various illumination conditions, making the segmentation process difficult. Also, the product image can be distorted due to a weird capture angle. The following works come in this scope, using a DL-based approach to correct the document view.

Abbas et al. [67] used a CNN model to locate the document's corner's position to perform its perspective correction. Their method does not depend on document content and does not require manual input. An example can be seen in figure 2.12.

The homography matrix is calculated using the corners and their desired location in the rectified image. This methodology proved to be more efficient than predicting the 3x3 homography matrix as the output of the network, which they also tested.

[67] mentions that the common practice in deep learning of converting the input to grayscale removes pivotal information that could help in the corner regression, as colour can be important to distinguish the document from its background.



(a) original image (b) result

Figure 2.11: [67] approach

Regarding loss functions for the CNN's training, [67] achieved the best results using Mean Absolute Error (MAE) and Mean Squared Error (MSE).

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_p^i - y^i| \tag{2.3}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_p^i - y^i)^2 \tag{2.4}$$

Using MAE as an evaluation metric directly shows in pixel units how the keypoint prediction is performing, as it represents the distance between ground truth points and the predicted ones. Applying the same function for the training loss leads to better results. [67] explain this using the hypothesis that for the problem at hand, MAE handles extreme scenarios, such as occluded corners or laying out the image frame, better than MSE. This happens since this last one gives a high penalty in case of misdetections, forcing the network to overfit.

In a similar approach to this problem, Dobai and Teletin [68] used a model with a MobileNet [69] backbone to regress the document corners' positions. MobileNet is a lightweight deep neural network designed to run on mobile devices. The best results were achieved using as loss function a variation of MSE that includes an angular error term.

Using more complex architectures, Javed and Shafait [70] used a custom CNN to detect the document corners followed by another network that works as a region extractor and refiner. It refines the vertices' location recursively through numerous iterations. It uses a smaller cropped patch of the previous image containing this corner, repeating this process until it gets to its precise location. Zhu et al. [71] use a similar methodology, making use of a VGG16 [72] architecture.

2.5.2 Document Unwarping

Further in the scope of document image correction, the following works additionally consider folds, curves, and crumples in the sheet, focusing on straightening them out.

Ma et al. [73] presented the first end-to-end Neural Network (NN) to flatten and rectify distorted document images, by predicting the mapping that can remove the distortion. Results can be seen in figure 2.12. Due to the lack of a dataset for this technique, the authors created an artificial one using ground truth documents images obtained using scanners and 2D meshes with distortions.

Das et al. [74] worked on a DL approach for document image unwarping, building a method that takes predicting a 3D-coordinate map of the document using a shape prediction network. Subsequently, this map goes into a texture mapping module that outputs the backward mapping used to remove the distortion. Finally, it has a refinement network that works as a post-processing component to adjust illumination effects. The dataset used is an own creation that features real 3D document shapes obtained using document data and rendering software.



(a) original image (b) result

Figure 2.12: [74] approach

2.6 Problem Statement

The proposed contributions of this master thesis look to overcome the difficulties of properly segmenting and correct traps in complex environments.

The current methodology only considers the four outer points of the trap to segment it and thereafter compute the homography matrix, which leads to losing information in some cases, as shown in figure 2.13.

Background complexity and capturing conditions strongly influence the results of the trap segmentation process (figure 2.14). Therefore, the presented methodology in the next chapter proposes a different approach to solve these problems using deep learning capabilities.

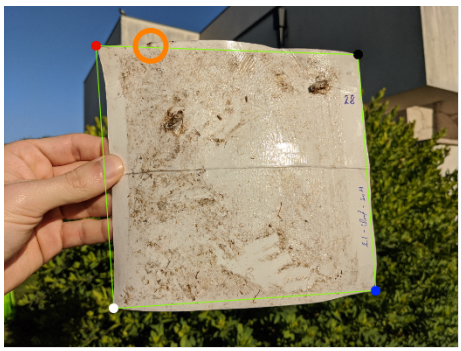


Figure 2.13: Loss of potentially important information (orange circle) when using only the corners for trap segmentation

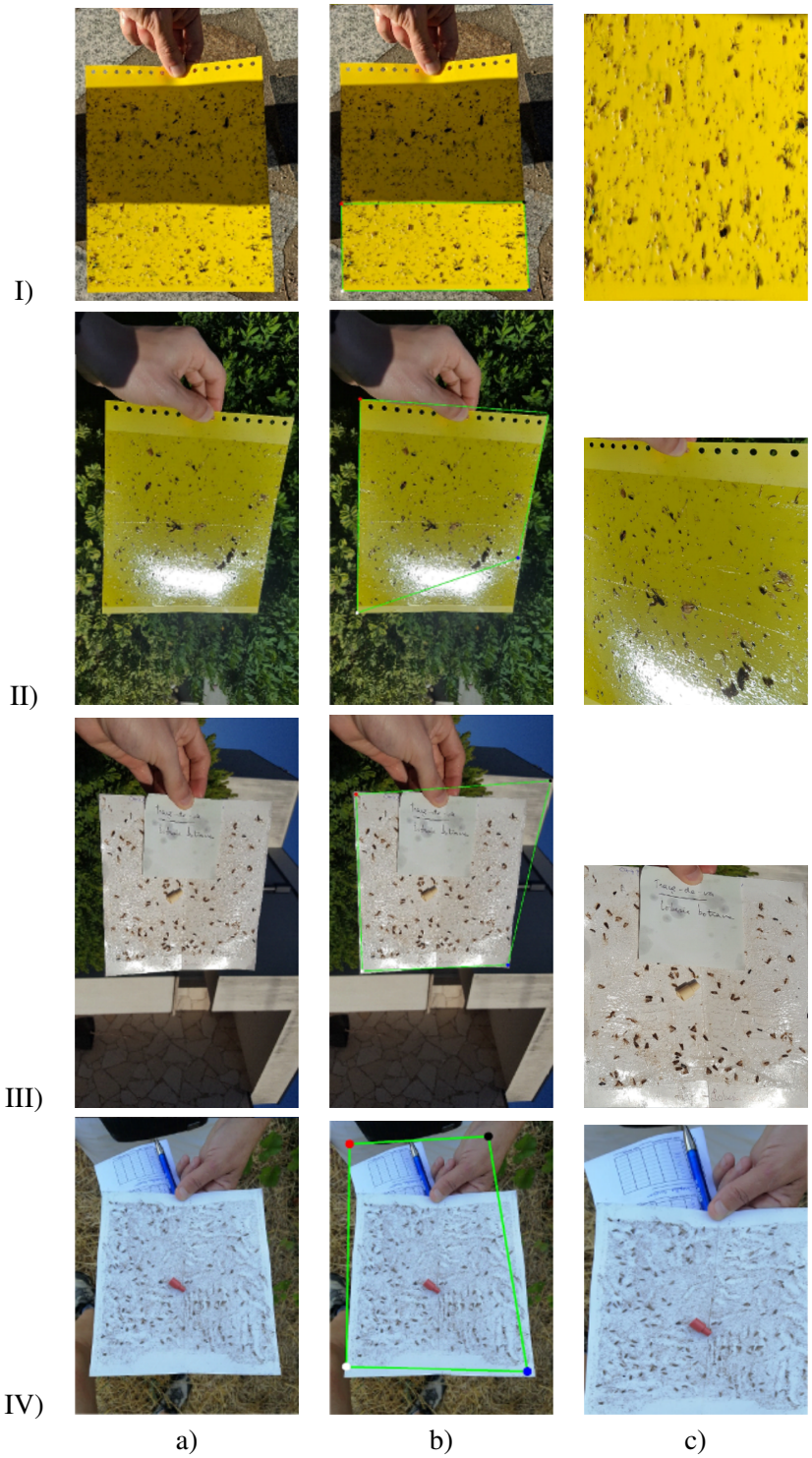


Figure 2.14: Examples of bad results in the original approach due to I) shadows II) reflections III),IV) complex backgrounds in the capturing environment. a) original image b) corner detection c) image after perspective correction, as seen on [1].

Chapter 3

Methodology

In this chapter, we briefly describe our approach to solve the problem in section 3.1. Afterwards, we present how we prepared the data in section 3.2. Section 3.3 shows how we chose a model for the trap landmark detection and in section 3.4 we describe what algorithms we tested to correct the perspective of the trap.

3.1 Approach Definition

We chose to follow a strategy inspired by [67, 68] works. However, we used 16 points instead of 4 for segmentation purposes and gave more emphasis to the perspective correction part. Our approach uses a CNN model to predict the landmarks on a trap image and performs a perspective correction in order to get a frontal view of the trap.

The methods used by [73, 74], which we chose not to follow, require an unwarped ground truth for the trap, that we did not have. Additionally, they were far different from the current methodology, that we want to compare ours against.

Therefore, the task division for this approach consisted of two phases:

- **Trap Segmentation:** A CNN model was picked and trained to detect the 16 landmarks that make the contour of the trap. We used the TensorFlow library [75].

- **Perspective Correction:** We used the OpenCV [54] library to build a more robust perspective correction technique than the one used in the original approach.

Previously to the segmentation part, a dataset was built to perform an efficient training of the chosen CNN algorithms. Since our data collection is small for deep learning purposes, data augmentation techniques were used to reduce the possible overfitting of the model.

3.2 Data Preparation

3.2.1 Available Data

The dataset is composed of 340 trap images, white or yellow, taken with different smartphones in controlled (laboratory) and non-controlled environments (production field), as seen in figure 3.1. This last set was collected by the target end-users of the system in three different vineyards during their weekly monitoring of the traps. The distribution of these images can be seen in the table 3.1.



(a) CE (b) CE (c) NCE (d) NCE

Figure 3.1: Images of traps in different image acquisition environments. CE - Controlled Environment, NCE - Non Controlled Environment

Trap Type	CE	NCE	Total
W	60	102	162
Y	139	39	178
Total	199	141	340

Table 3.1: Dataset distribution

The use of different smartphones for the image capturing process leads to having a wide range of resolutions (1224 × 1632 px to 2988 × 5312 px).

All the photos have the borders and corners of the trap visible and there are no repeated images of them in similar conditions (perspective and illumination).

16 landmarks were manually annotated in each trap, being the corners, the middle points of these, and the middle ones of the previously mentioned, making the contour of the trap, as illustrated in the figure 3.2.

3.2.2 Dataset Building

The amount of training data and its quality are very important for the effectiveness of a deep learning application. Therefore, after matching the images and each respective landmark annotations, there was a need to establish some requirements and specifications for the dataset. From the literature, we cannot define the requirements for our data. The criteria usually relies on a more empirical

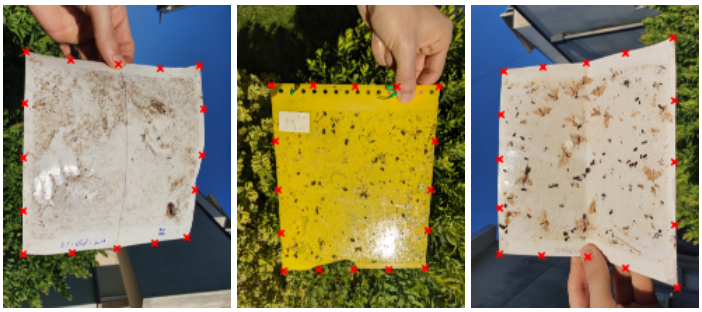


Figure 3.2: Trap images with annotated landmarks

approach that is used for deep learning computer vision tasks: If a feature in an image would be identified clearly by the human eye, it would then be applicable for a deep learning module. Since we have a clean dataset in terms of data representation and format (all JPG files), the following steps were to choose the color scheme and the resizing format.

3.2.2.1 Colour Scheme

We had to choose between using RGB and a grayscale colour system. Grayscale, by using a third of the data RGB uses, saves storage and reduces the dataset loading and model training time in experiences. However, it should only be used if it does not omit important information for the task we are performing. In some of our images, it impacts the recognition of the borders, as can be seen in the figure 3.3. This assumption was later confirmed when evaluating the performance of our baseline model using both colour schemes. We picked RGB due to the variability of situations where the colour of the trap can positively influence its contour detection.

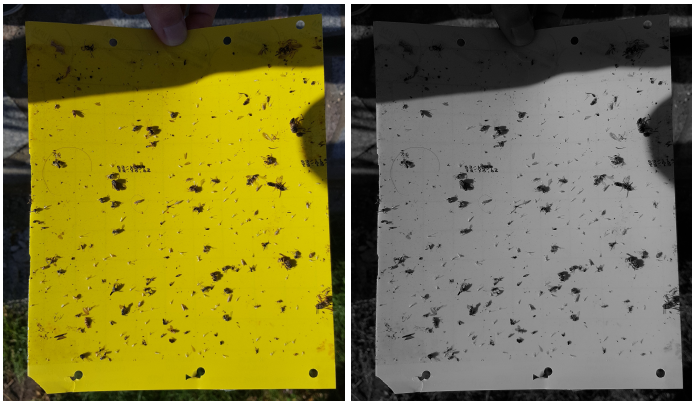


Figure 3.3: Grayscale image visual impact

3.2.2.2 Image Size

CNN models take a fixed input size, creating a need to define this parameter. Regarding the shape of our images, they are originally rectangles and almost all of them were annotated in a

vertical orientation. However, the traps presenting border deformations mostly have them in the side margins, not in the top or bottom ones. Since when being used our algorithm can see this deformations anywhere, we chose to follow a square shape to make the most of the existing data using data augmentation techniques. We resized all images to 256 x 256 px, as seen in figure 3.4.

Although it causes stretching or compressing of the trap shape due to all the various original image sizes, we followed the same rule with what we made the decision about the colour scheme.

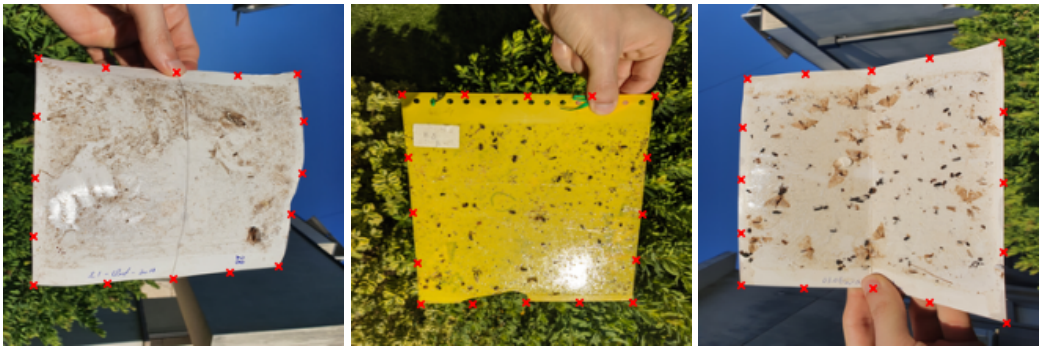


Figure 3.4: Resized trap images

3.2.2.3 Data Augmentation

Keras [76], the interface for the Tensorflow library provides a class named ImageDataGenerator that was designed to provide real-time data augmentation. It generates augmented images on the fly while training a model, requiring lower memory usage when compared to doing it previously and storing it.

However, this class was designed for classification problems. Since ours is a regression one, we implemented our own generator with custom data augmentation methods. We built modules to perform zooms, flips, translations and rotations over the pitch, roll, and yaw dimensions on every batch of images before the model training step. Figure 3.5 shows an example of how this generator works.

3.3 Trap Segmentation

3.3.1 CNN Architectures

We started by developing a model inspired by the top performer networks for the face landmark detection competition in Kaggle [77] and the ones presented in document corners location papers, such as Abbas and Hussain [67] and Javed and Shafait [70] works. Our work requires 32 output nodes, representing the x and y values of the 16 landmarks. After some training and fine-tuning, the final architecture is the one described in the appendix A. It features dropout and batch normalization layers.

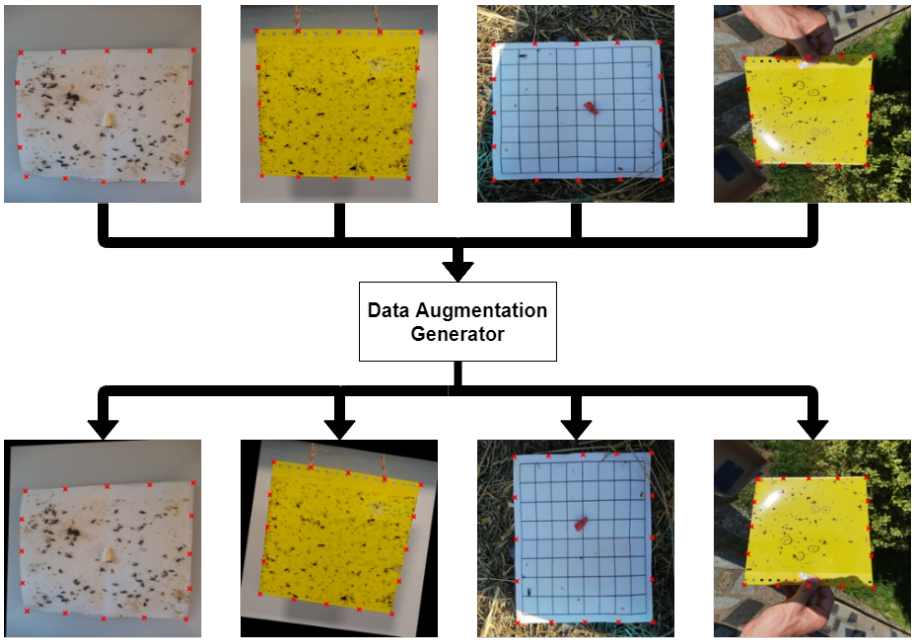


Figure 3.5: Example of the behaviour of data augmentation generator for a batch of 4 images

We used this model to support different important choices in our work. We usually refer to it as our model or baseline model.

The works in [68, 71] made us select the MobileNetV2 [78] (an upgrade on the original MobileNet) and VGG16 backbones to build other two models to test ours against. Using Keras, we load them except for the classification layers, which are substituted by ours.

For model training, we chose MAE and MSE for loss functions. We performed 5-fold cross-validation for each possible combination (model, loss functions and usage of data augmentation), with a batch size of 8 and an Adam optimizer with an exponential learning rate, starting with 0.001 and decaying 80% every 200 steps. We picked these values since they got the best results after several tests with our baseline model. For experiments with data augmentation, 350 epochs were used, as for the experiences without data augmentation, we used 250.

3.3.2 Parameters for Results Assessment

To compare the models' performances and this methodology against the current one, we used the Jaccard Index (JI), also known as Intersection over Union (IoU), between the segmentation mask and the respective ground truth annotations one. These masks are polygons drawn with the points.

$$JI = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

3.4 Perspective Correction

The original implementation uses the detected trap corners to compute the homography matrix, followed by a perspective transformation. Works in [67, 68, 70, 71] use a four-corner perspective transformation too. An example of a trap perspective correction using the corners can be seen in figure 3.6. We wanted to understand if using more than four corner points leads to less information loss and makes transformations more robust.

We resized the detected landmarks to their original image scaling before this procedure.

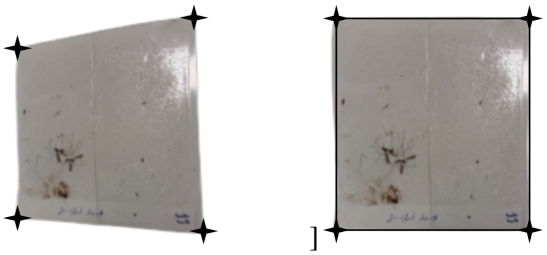
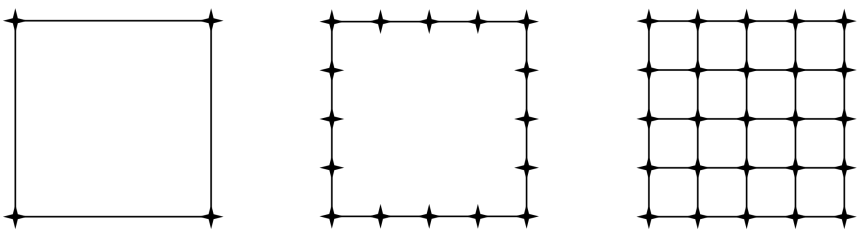


Figure 3.6: Four point mapping for perspective correction of a trap

3.4.1 Algorithms

We start by calculating the aspect ratio of the traps using the detected corners. We assume they always present an approximately square or rectangular shape. Used in the original implementation, the method used in [79], when applied to our purpose, assumes that the trap is projected in a plane perpendicular to the camera's pinhole, using two concurrent vectors. The aspect ratio is then used to get the final image dimensions, by fixing the height at 1500 px and calculating the width using the ratio.

Our strategy for point correspondence is based on destination grids with the desired locations for them, as seen in figure 3.7. The 25 point grid is obtained using the intersection of the lines between the correspondent points on each side, as seen in the figure. These grids' dimensions are the same as the final image.



(a) 4 points (b) 16 points (c) 25 points

Figure 3.7: Perspective correction destination grids

The project's original algorithm makes use of the OpenCV [54] function *getPerspectiveTransform*, which maps the corners of the trap to their desired position. However, it only accepts four pairs of points matches.

To make use of more correspondences, the function *findHomography* was used. This allows us to choose the method, having the least-squares and the RANSAC-based robust available.

Since the traps rarely present a perfect rectangular shape, there is always a distance error in the correspondences used in the transformation.

We compared methodologies using 16 and 25 points for the two mentioned algorithms. It is important to notice that using both methods with 4 points leads to the same results. We then compared these methods to only using the corners of the trap to see if there was any improvements.

3.4.2 Rejection of Demarked Transformations

We assume that images which need demarked perspective corrections will be impacted in terms of quality, regardless of the successful transformation in terms of points. It could eventually cause worse results in the following insect detection pipeline. Also, by constraining the capture angle, we force the user to use a more frontal view of the trap when capturing the photo. Therefore, a rejection module was built. We obtained rules for this rejection by labelling the dataset images (reject/accept) and training a Decision Tree Classifier (DTC) from the Scikit-learn [80] library using the g and h values of the homography matrix, seen in equation 3.2. These parameters are the ones in the homography matrix responsible for perspective distortion.

$$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \tag{3.2}$$

3.4.3 Parameters for Results Assessment

We used the Euclidean image distance, which in our case represents the mean squared distance between all the transformed image points and the correspondent ones in the grid. We always use the 25 points to measure this, even when using a 4 point homography, in order to understand how well our trap is centred in our destination grid.

The other used metric is the IoU between the area inside the destination grid for the landmarks and the trap segmentation mask area, to understand what is the trap portion we ignore after the transformation.

3.5 Implementation

The flow diagram for the developed algorithm can be seen in figure 3.8.

However, the project application pipeline is designed differently. It starts by validating several necessary conditions to assure that the image is suitable to be captured, including a successful

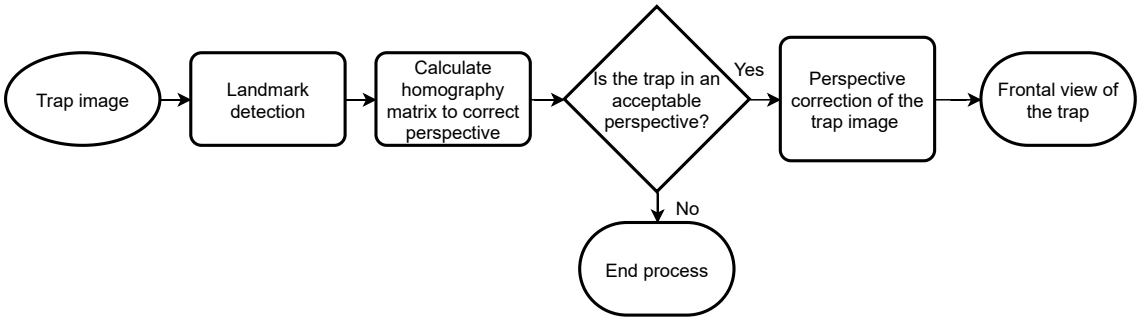


Figure 3.8: Our algorithm block diagram

segmentation of the trap in a preview step. Afterwards, it automatically acquires the image, again performing the trap segmentation, this time followed by its perspective correction. In order to replace the old algorithm with ours, we had to follow the original configuration, only introducing a step to reject the procedure in case of a demarked perspective transformation. We mainly replaced the original modules with ours, not changing the rest of the application. The application flow can be seen in figure 3.9.

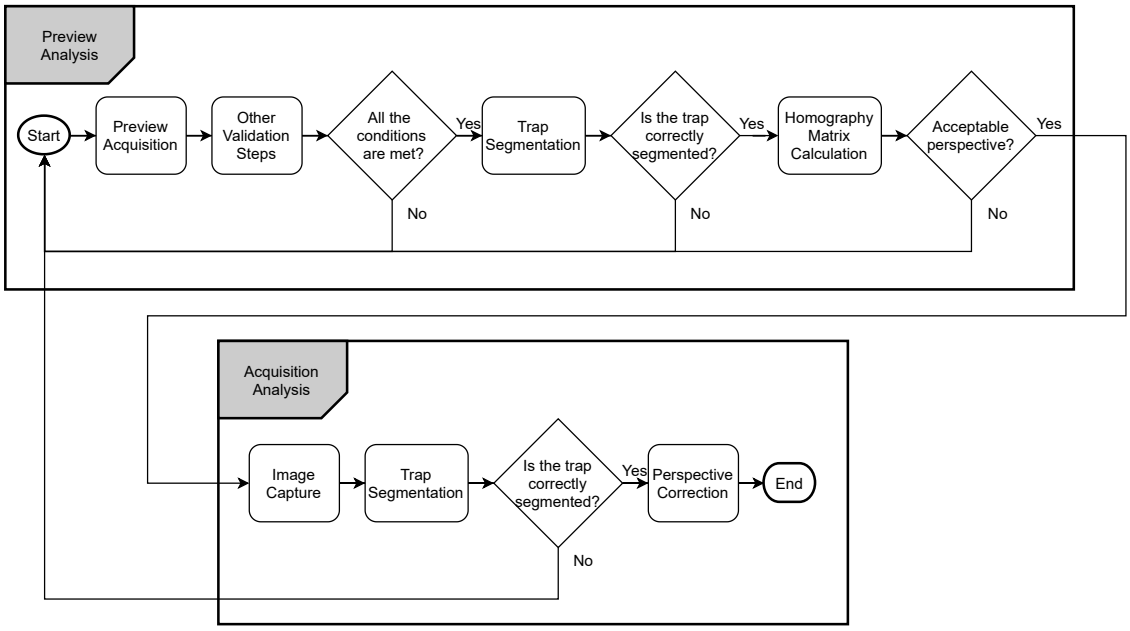


Figure 3.9: Application flow

The application was implemented on Android Studio using Kotlin and C++, this last one via JNI (Java Native Interface).

The trap segmentation module is then replaced by our landmark detection neural network, which is first converted to TensorFlow Lite in order to be used on mobile devices. We also changed the C++ functions to perform the perspective correction in order to implement our approach.

Chapter 4

Results

In this chapter, we display the results obtained with the previously mentioned approaches for trap segmentation and its perspective correction. In section 4.1, the procedure to select a model to perform the trap segmentation is presented by comparing results between them. We also checked the dataset constitution to understand the results we got. Section 4.2.2 presents the results for the perspective correction part, where we additionally show the rules established to reject transformations. Lastly, in section 4.3 we discuss and compare this new approach with the project's original one in terms of performance and inference time. Visual results of our algorithm results can be seen in section 4.3.4.

4.1 Trap Segmentation

4.1.1 Network Training and Selection

We started by comparing our model to MobileNetV2 and VGG16 architectures with the desired output size (32), after removing the original output layers (1000). Our model achieved the best results. However, after fine-tuning the others, adding a dropout layer equivalent to the one used in ours (between the convolutional layers and the fully connected layer) lead MobileNetV2 to achieve the best results, as seen in tables 4.1, 4.2. This architecture trained with MAE using data augmentation proved to be the best combination. We only used rotations and translations while comparing the models. Due to a good response of the best configuration to the use of this technique, we retrained it from scratch adding flips and zooms to the other transformations, which lead to even better results, seen in table 4.3.

During fine-tuning, increasing the number of dense layers worsen the performance of all models. Using only one lead to better results compared to two or more, as it is usually suggested.

We also observed that the results are always worse regarding yellow traps, which leads to the work done in the next section.

Model	Loss Function	Data Augmentation	JI	
			Mean	Std
Ours	MAE	No	0.954	0.031
		Yes	0.957	0.030
	MSE	No	0.942	0.041
		Yes	0.944	0.039
MobileNetV2	**MAE**	No	0.964	0.011
		Yes	**0.969**	**0.012**
	MSE	No	0.960	0.018
		Yes	0.963	0.020
VGG16	MAE	No	0.853	0.092
		Yes	0.862	0.079
	MSE	No	0.866	0.071
		Yes	0.843	0.078

Table 4.1: Model training results for white traps

Model	Loss Function	Data Augmentation	JI	
			Mean	Std
Ours	MAE	No	0.946	0.031
		Yes	0.951	0.050
	MSE	No	0.938	0.053
		Yes	0.942	0.048
MobileNetV2	**MAE**	No	0.957	0.032
		Yes	**0.963**	**0.030**
	MSE	No	0.950	0.043
		Yes	0.958	0.036
VGG16	MAE	No	0.866	0.085
		Yes	0.865	0.087
	MSE	No	0.837	0.084
		Yes	0.837	0.087

Table 4.2: Model training results for yellow traps

JI	Subset				
	All	Y-CE	Y-NCE	W-CE	W-NCE
Mean	0.975	0.974	0.977	0.975	0.975
Std	0.013	0.017	0.012	0.012	0.008

Table 4.3: Trap segmentation results for increased data augmentation for the MobileNetV2 model trained with MAE loss

4.1.2 Dataset Critical Analysis

All our models perform worse when handling yellow traps. By checking the histogram for the best model configuration (figure 4.1), we found out the reason why some of these images got worse results. Mainly for yellow traps, some images feature odd perspectives of the trap, such as the ones in figure 4.2.

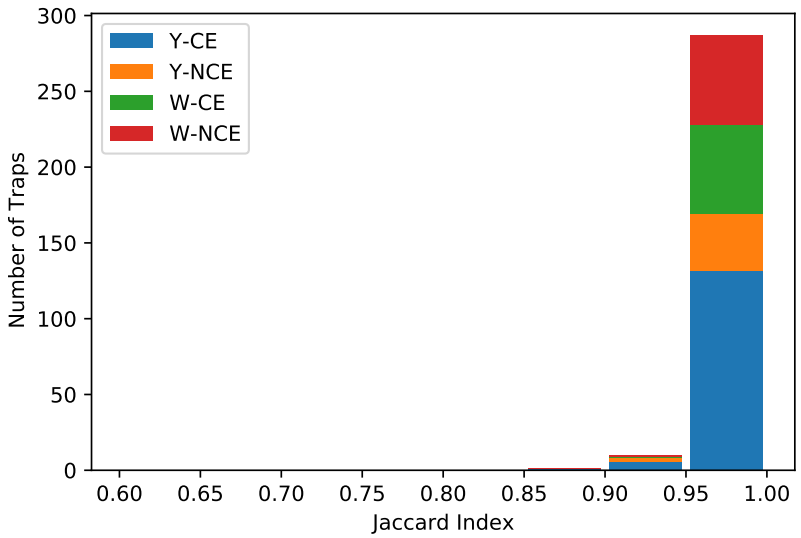


Figure 4.1: Landmark detection JI distribution for our best model results

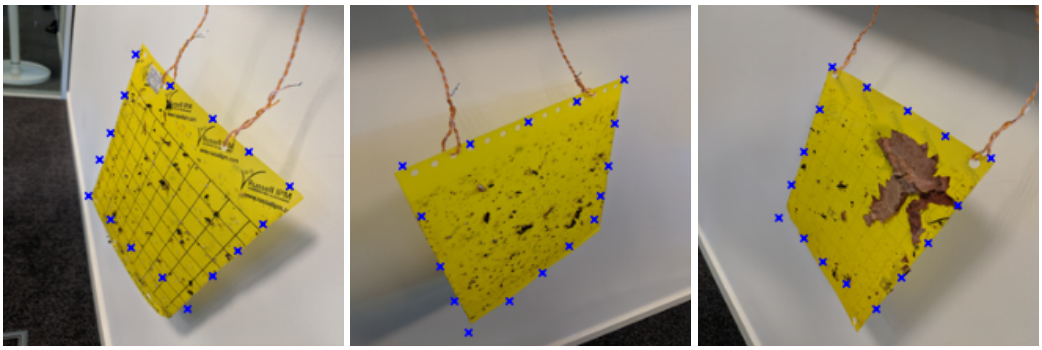


Figure 4.2: Images with weaker results for landmark detection

Through visual analysis of the images in the controlled conditions subset, many of them presented these trap views that are not usual in the normal image acquisition procedure by the end-users. Although it is good for training purposes, as it represents the different possible situation the algorithm might come up against, it is not representative of the correct use of the application.

4.2 Perspective Correction

4.2.1 Methodology Selection

We proceeded to test our algorithms for perspective correction using the product of landmark detection.

RANSAC-based methods presented an unpredictable behaviour, even after testing with different threshold values for outlier classification. Some transformations produced weak results.

Using the least-squares method to compute the homography matrix achieved consistently better outcomes, as it does not rely on a more random selection of points. Also, this correction part's good results will always depend on an accurate point detection by our model. Since it performed well, we chose to stick with the 16 points approach as it presents the biggest percentage of trap inside the cropping frame (grid) for the final image, as seen in table 4.4. This method avoids loosing too much area of the trap when the corners present deformation, as it represents better the shape of the trap However, 16 and 25 points approaches were similar, since they are technically related due to the way the 25 point grid is calculated.

Method	Number of Points	E		% SAIF	
		Mean	Std	Mean	Std
LSM	4	1445.91	2663.97	0.988	0.015
	16	**513.94**	**746.34**	**0.990**	**0.008**
	25	463.32	646.21	0.988	0.010
RANSAC	16	7372.47	26842.32	0.969	0.057
	25	3124.84	7918.05	0.970	0.038

Table 4.4: Evaluation metrics results for the perspective correction methodologies using the whole dataset landmark detection results. SAIF - Segmented Area Inside the Frame

4.2.2 Demarked Corrections Rejection

After training the DTC, we extracted rules, seen in figure 4.3 to discard transformations such as the ones needed for the traps in figure 4.2. The tree starts in first node, and if its associated conditions is true, it goes to the left child node, if false, to right child node.

From all the dataset, 76 transformations were discarded, 73 from the controlled environment subset. We fixed the depth of the tree in order to obtain maximum and minimum thresholds for both variables.

4.3 Current Approach Comparison

In this section, we compare our new methodology that features the approaches which gave the best results with the project's original algorithm, in terms of trap segmentation results, perspective correction and inference time. The visual improvements can be seen in figure 4.5.

4.3.1 Trap Segmentation

Comparing the JI results for the original approach and ours in the table 4.5 we conclude that this new DL methodology produces less disperse results and perform better under in-the-wild conditions. For a fair comparison, we only considered the corners of the segmentation mask and the ground-truth, since the original approach only uses these points. Also, in order to understand what portion of the trap is ignored when using only the corners for segmentation purposes, for the 16 points ground-truth, we compared the JI obtained using only the corners points to using all of

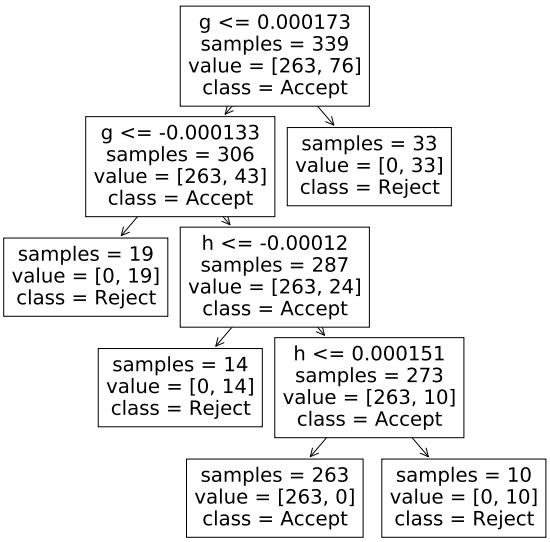


Figure 4.3: Decision tree for rejection of perspective correction

them. Results in table 4.6 show that this difference is small, with the bigger increase in white traps under non controlled conditions, more suitable to deformations.

Method	JI	Subset				
		All	Y-CE	Y-NCE	W-CE	W-NCE
Ours	Mean	0.975	0.974	0.977	0.975	0.974
	Std	0.015	0.018	0.013	0.014	0.011
Original	Mean	0.967	0.989	0.936	0.990	0.936
	Std	0.076	0.029	0.108	0.012	0.109

Table 4.5: Old versus new approach trap segmentation results using only the corners

Method	JI	Subset				
		All	Y-CE	Y-NCE	W-CE	W-NCE
16 points	Mean	0.975	0.974	0.977	0.975	0.975
	Std	0.013	0.017	0.012	0.012	0.008
4 points	Mean	0.972	0.973	0.975	0.968	0.969
	Std	0.014	0.017	0.012	0.015	0.010

Table 4.6: Difference on using 16 and 4 points for the segmentation of the trap

4.3.2 Perspective Correction

Results obtained using a least-squares method approach with 4 and 16 points were similar, as seen in table 4.4. However, visual results show that the biggest difference between approaches happen

when dealing with deformed traps (cases where the 16 points approach works the best) and odd perspectives (4 points approach taking advantage).

4.3.3 Inference Time

After deciding what methods to use for the segmentation and perspective correction parts, we proceeded to replace these modules in the project's application.

Our goals with this were to:

- Test the modified application
- Evaluate the inference time

The algorithm performed as expected, being able to reproduce the results of the experiments. As for the inference time, it decreased a lot when compared to the original approach, as seen in the tables 4.7 and 4.8. The time difference between both steps in the old methodology is related to using different images resolutions images. Also, due to some operations done in the old methodology being dependent on the trap type, it causes inference time for both types to be different, which does not happen in ours.

The experiments were done by performing 10 runs of each process with high-end and low-end smartphones, Samsung S10 and S6, respectively. We were able to reduce acquisition analysis time. However, the preview analysis time is similar.

Time	Preview Analysis				Acquisition Analysis			
	Y		W		Y		W	
	HE	LE	HE	LE	HE	LE	HE	LE
Mean (s)	0.556	1.508	0.802	1.784	5.091	14.922	21.409	44.581
Std (s)	0.036	0.183	0.135	0.195	0.592	1.574	12.055	7.665

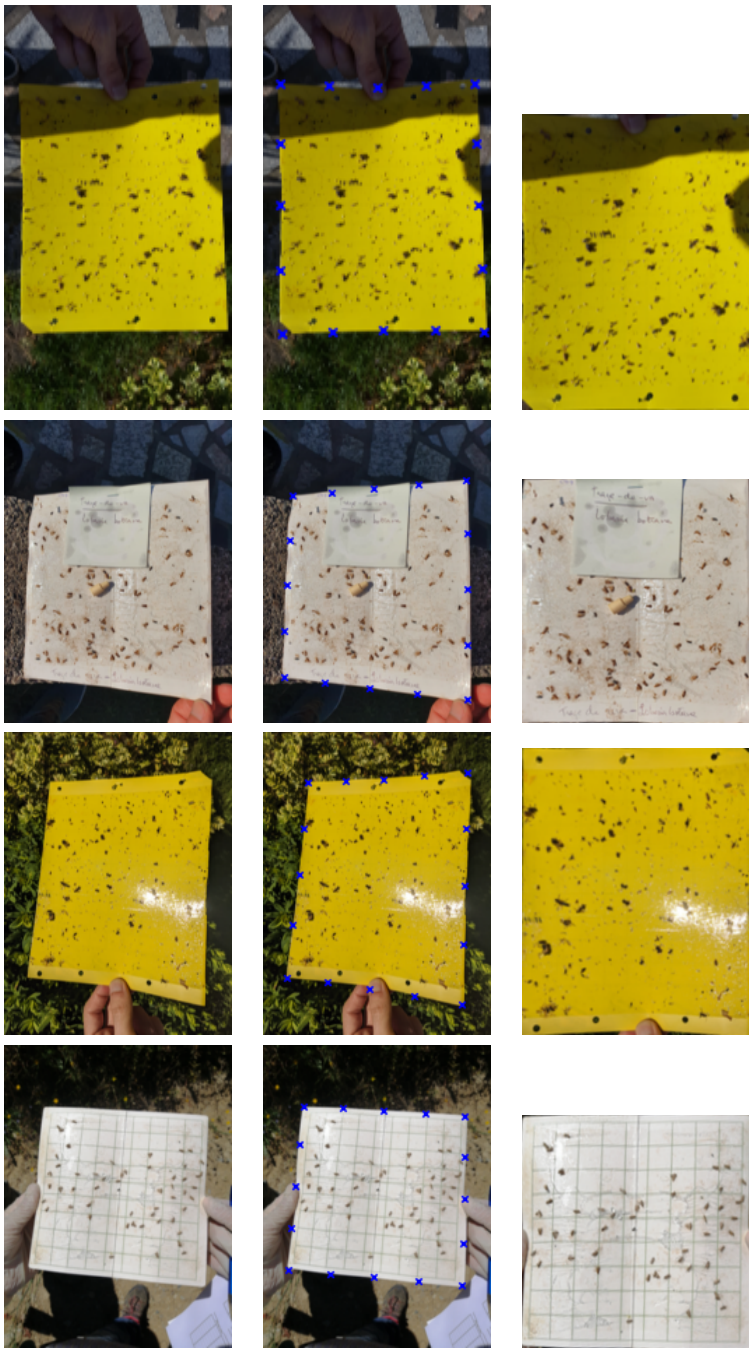
Table 4.7: Original algorithm inference time. HE - High-End, LE - Low-End

Time	Preview Analysis				Acquisition Analysis			
	Y		W		Y		W	
	HE	LE	HE	LE	HE	LE	HE	LE
Mean (s)	0.892	1.973	1.086	2.116	0.330	0.549	0.435	0.598
Std (s)	0.093	0.245	0.167	0.235	0.092	0.085	0.178	0.120

Table 4.8: Our approach inference time

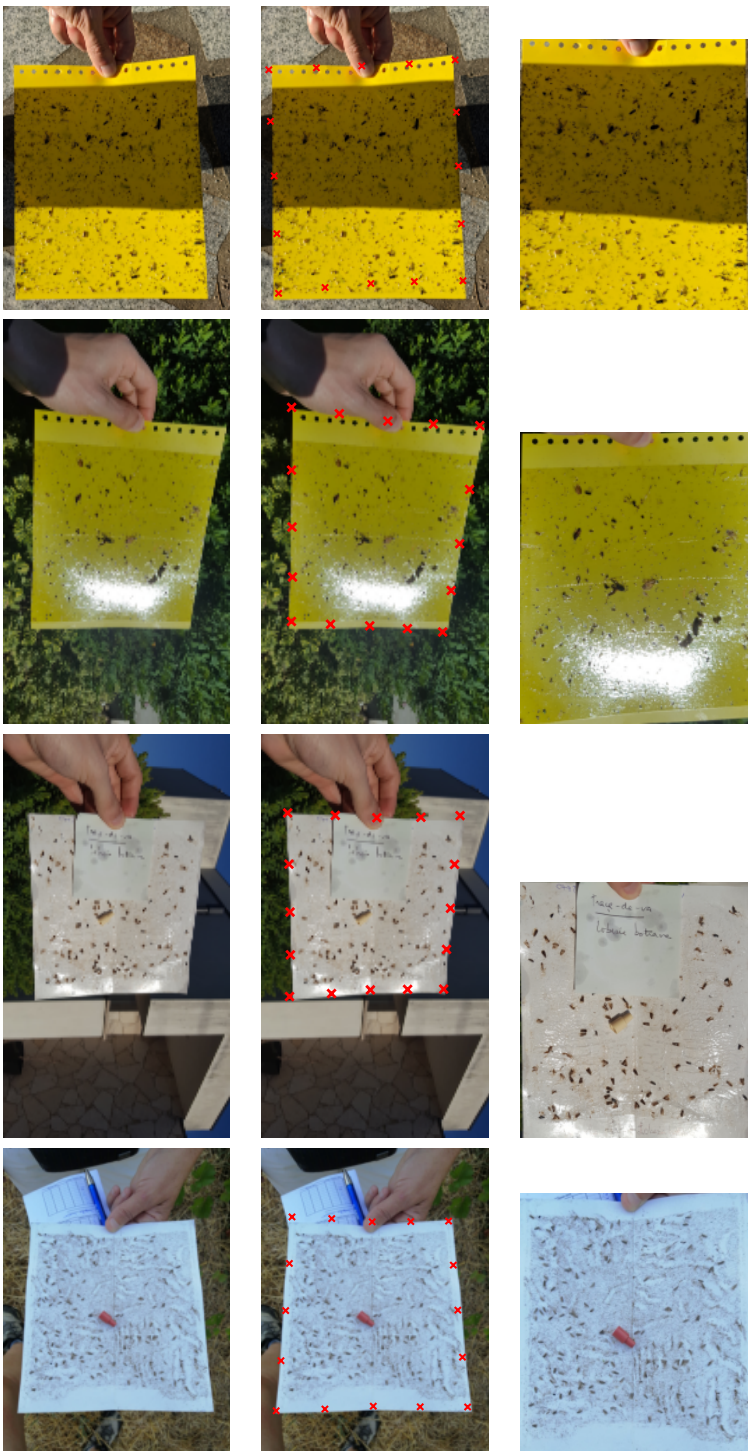
4.3.4 Visual Results

In this section we present examples of the product of our algorithm. Table 4.4 contains some cases of the procedure to correct the view of the traps, while table 4.5 shows the performance improvement on the cases in table 2.14



a) b) c)

Figure 4.4: Our algorithm results. a) original image b) segmented trap c) image after perspective correction



a) b) c)

Figure 4.5: Examples of better results when working under in-the-wild conditions. Comparison with table 2.14. a) original image b) segmented trap c) image after perspective correction

Chapter 5

Conclusions and Future Work

5.1 Objectives Satisfaction

Overall, the achieved results were satisfactory. We improved the performance of the system by using a DL-based landmark detection with 16 points to draw the contour of the trap to segment it from the background. The biggest improvement in this methodology was seen in images taken under non-controlled environments, in which our algorithm performed much better when handling complex backgrounds and lighting conditions. Unfortunately, we achieved slightly worse results for images in controlled environments. We analysed the cases where our algorithm performs worse and noted that this approach does not work so well when dealing with odd perspectives of the trap that are far from a frontal view. However, our module for rejection of these demarked corrections softens this issue, by forcing the capture angle to be more frontal, as desired. From the whole dataset, rejecting 73 images from this subset in a total of 76 rejected images denotes the difference between the two capture environments subsets, being the indoors one not very representative of the correct use of the application.

When it comes to our choices after the experiments, a MobileNetV2 architecture proved to be the best one for our work. The fact that this model is highly optimized to run on mobile devices is a bonus. Fine-tunning experiments showed that increasing the number of dense layers in all our models lead to worse results. We only used one to regress our 32 desired output values. Also, adding a dropout layer of 0.5 improved the performance of every network. Additionally, using data augmentation led to an increase of 1.8% overall in the mean JI results for our best model.

Considering 16 contour points proves to be a better approach than using only the corners for segmentation, as it increased the segmented area, and for perspective correction, mainly due to the unpredictable variability of deformations in the corners of the traps the algorithm might encounter. Cases such as broken corners and severe border deformations were dealt better when compared to the original approach. Regarding the homography estimation algorithm, RANSAC-based methodologies showed some unpredictable results in some cases, leading to bad transformations overall. Since our contour detection achieved good results, we implemented the 16 point method using a least-squares estimation for the final algorithm since it reports the best and more consistent SAIF

metric. This metric represents the area of the trap that is transmitted to the insect detection pipeline in the application.

Besides the main objectives, the most promising methodology was implemented in android, to infer its usability through the algorithms inference time. When compared to the previous approach, we are able to segment the trap quicker, leading to a better usability of the app. This approach is also trap type independent, unlike the former one

In terms of innovation, our work shows a revolutionary use of DL techniques in the field of trap analysis, as it is the first application to use a neural network for this trap perspective correction procedure. Document perspective correction works proved to be useful for our purpose, as our methodology is similar to the ones presented in these papers. Also, since landmark-based segmentation is not usual outside the facial and human pose estimation domain, it represents a contribution to the research field, showing one more task in which this technique can be used.

5.2 Future Work

Testing the algorithm under real-time outdoor working conditions would be the first thing to do, as it was only evaluated in the laboratory.

Regarding the dataset quality, improvements could be done, such as not giving so much emphasis to odd perspectives of traps while capturing images and balancing the trap type distribution. It is good to have a collection of data that has a wide variety of cases our networks can see, but it should always represent the reality it might face. We observed that for images taken in the laboratory, yellow traps make 70% of the total, while for images taken in the work field, it makes 28%. There is also a low number of examples under extreme environmental conditions. The orientation of the border deformations could vary too, as these appear always on the side margins of trap in this dataset, but when using the application, there is no control over it. We chose to resize all the images to squares and use rotation transformations to tackle this issue. Furthermore, as this is a deep learning application, acquiring more data is recommended. Increasing the variety of backgrounds is an option. However, it would be interesting to test taking the segmented trap images from the existing data and creating an artificial dataset by placing it over different textures and landscapes backgrounds [67].

A semantic segmentation approach forward by a corner detection using a CNN would be interesting to explore, as it could lead to a more accurate segmentation of the trap and consequently a more tailored and fine perspective correction.

The methods referred in section 2.5.2 might be useful for the implementation of a solution as a unique DL architecture. However, it would require an unfolded frontal view image of the trap to work with, meaning discarding all the existent data.

Appendix A

A.1 Architecture of our Network

Model: "model_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 256, 256, 3)	0
conv2d_1 (Conv2D)	(None, 256, 256, 32)	2400
re_lu_1 (ReLU)	(None, 256, 256, 32)	0
batch_normalization_1 (Batch	(None, 256, 256, 32)	128
conv2d_2 (Conv2D)	(None, 256, 256, 32)	25600
re_lu_2 (ReLU)	(None, 256, 256, 32)	0
batch_normalization_2 (Batch	(None, 256, 256, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 128, 128, 32)	0
conv2d_3 (Conv2D)	(None, 128, 128, 64)	51200
re_lu_3 (ReLU)	(None, 128, 128, 64)	0
batch_normalization_3 (Batch	(None, 128, 128, 64)	256
conv2d_4 (Conv2D)	(None, 128, 128, 64)	102400
re_lu_4 (ReLU)	(None, 128, 128, 64)	0
batch_normalization_4 (Batch	(None, 128, 128, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 64, 64, 64)	0
conv2d_5 (Conv2D)	(None, 64, 64, 96)	55296
re_lu_5 (ReLU)	(None, 64, 64, 96)	0
batch_normalization_5 (Batch	(None, 64, 64, 96)	384

conv2d_6 (Conv2D)	(None, 64, 64, 96)	82944
re_lu_6 (ReLU)	(None, 64, 64, 96)	0
batch_normalization_6 (Batch	(None, 64, 64, 96)	384
max_pooling2d_3 (MaxPooling2	(None, 32, 32, 96)	0
conv2d_7 (Conv2D)	(None, 32, 32, 128)	110592
re_lu_7 (ReLU)	(None, 32, 32, 128)	0
batch_normalization_7 (Batch	(None, 32, 32, 128)	512
conv2d_8 (Conv2D)	(None, 32, 32, 128)	147456
re_lu_8 (ReLU)	(None, 32, 32, 128)	0
batch_normalization_8 (Batch	(None, 32, 32, 128)	512
max_pooling2d_4 (MaxPooling2	(None, 16, 16, 128)	0
conv2d_9 (Conv2D)	(None, 16, 16, 256)	294912
re_lu_9 (ReLU)	(None, 16, 16, 256)	0
batch_normalization_9 (Batch	(None, 16, 16, 256)	1024
conv2d_10 (Conv2D)	(None, 16, 16, 256)	589824
re_lu_10 (ReLU)	(None, 16, 16, 256)	0
batch_normalization_10 (Batc	(None, 16, 16, 256)	1024
max_pooling2d_5 (MaxPooling2	(None, 8, 8, 256)	0
conv2d_11 (Conv2D)	(None, 8, 8, 512)	1179648
re_lu_11 (ReLU)	(None, 8, 8, 512)	0
batch_normalization_11 (Batc	(None, 8, 8, 512)	2048
conv2d_12 (Conv2D)	(None, 8, 8, 512)	2359296
re_lu_12 (ReLU)	(None, 8, 8, 512)	0
batch_normalization_12 (Batc	(None, 8, 8, 512)	2048
dropout_1 (Dropout)	(None, 8, 8, 512)	0
flatten_1 (Flatten)	(None, 32768)	0
dense_1 (Dense)	(None, 32)	1048608

Total params: 6,058,880
Trainable params: 6,054,528
Non-trainable params: 4,352

References

[1] Pedro Faria, Telmo Nogueira, Ana Ferreira, Cristina Carlos, and Luís Rosado. Ai-powered mobile image acquisition of vineyard insect traps with automatic quality and adequacy assessment. *Agronomy*, 11(4), 2021. URL: https://www.mdpi.com/2073-4395/11/4/731, doi:10.3390/agronomy11040731.

[2] Douro valley. Douro valley. http://www.dourovalley.com/PageGen.aspx?WMCM_PaginaId=79565. Accessed: 2021-01-30.

[3] João A. Santos, Helder Fraga, Aureliano C. Malheiro, José Moutinho-Pereira, Lia-Tânia Dinis, Carlos Correia, Marco Moriondo, Luisa Leolini, Camilla Dibari, Sergi Costafreda-Aumedes, Thomas Kartschall, Christoph Menz, Daniel Molitor, Jürgen Junk, Marco Beyer, and Hans R. Schultz. A review of the potential climate change impacts and adaptation options for european viticulture. *Applied Sciences*, 10(9), 2020. URL: https://www.mdpi.com/2076-3417/10/9/3092, doi:10.3390/app10093092.

[4] FAO. Plants vital to human diets but face growing risks from pests and diseases. http://www.fao.org/news/story/en/item/409158/icode/. Accessed: 2021-07-23.

[5] A. Thorat, S. Kumari, and N. D. Valakunde. An iot based smart solution for leaf disease detection. In *2017 International Conference on Big Data, IoT and Data Science (BID)*, pages 193–198, 2017. doi:10.1109/BID.2017.8336597.

[6] Shanwen Zhang, Haoxiang Wang, Wenzhun Huang, and Zhuhong You. Plant diseased leaf segmentation and recognition by fusion of superpixel, k-means and phog. *Optik*, 157:866–872, 2018.

[7] Kuei-Chung Chang, Po-Kai Liu, Zi-Wen Kuo, and Sin-Huei Liao. Design of persimmon growing stage monitoring system using image recognition technique. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2. IEEE, 2016.

[8] J. Zhang, Z. Guo, S. Chen, B. Shao, and Y. Wang. Iot-based detection for tropical flower. In *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, pages 219–222, 2016. doi:10.1109/ISAI.2016.0054.

[9] ForesightIPM. Foresightipm. https://foresightipm.com/. Accessed: 2021-01-30.

[10] Metos. Metos. https://metos.at/insect-monitoring/. Accessed: 2021-01-30.

[11] Trapview. Trapview. https://www.trapview.com/v2/en/. Accessed: 2021-01-30.

[12] Sharada P. Mohanty, David P. Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7:1419, 2016. URL: https://www.frontiersin.org/article/10.3389/fpls.2016.01419, doi:10.3389/fpls.2016.01419.

[13] EyesOnTraps. Eyesontraps. http://www.eyesontraps.pt/. Accessed: 2021-02-02.

[14] ADVID. Advid. https://www.advid.pt/pt/home. Accessed: 2021-07-29.

[15] GeoDouro. Geodouro. https://www.geodouro.pt/. Accessed: 2021-07-29.

[16] Simão Felgueiras., João Costa., João GonÇalves., and Filipe Soares. Mobile-based risk assessment of diabetic retinopathy using a smartphone and adapted ophtalmoscope. In *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies - HEALTHINF,*, pages 168–175. INSTICC, SciTePress, 2018. doi:10.5220/0006599701680175.

[17] José Faria, J. Almeida, M. Vasconcelos, and Luís Rosado. Automated mobile image acquisition of skin wounds using real-time deep neural networks. In *MIUA*, 2019.

[18] José Alves, Dinis Moreira, Pedro Alves, Luís Rosado, and Maria João M. Vasconcelos. Automatic focus assessment on dermoscopic images acquired with smartphones. *Sensors*, 19(22), 2019. URL: https://www.mdpi.com/1424-8220/19/22/4957, doi:10.3390/s19224957.

[19] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292, 2019.

[20] Uchenna Odi and Thomas Nguyen. Geological facies prediction using computed tomography in a machine learning and deep learning environment. 07 2018. doi:10.15530/urtec-2018-2901881.

[21] London and Zafeirios Fountas. Imperial college spiking neural networks for human-like avatar control in a simulated environment. 06 2021.

[22] Md. Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Nasrin, Mahmudul Hasan, Brian Essen, Abdul Awwal, and Vijayan Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8:292, 03 2019. doi:10.3390/electronics8030292.

[23] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[24] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. 01 2020.

[25] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi:10.1109/TSMC.1979.4310076.

[26] David H. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122, 1973.

[27] Suo Xuesong, Liu Zi, Sun Lei, Wang Jiao, and Zhao Yang. Aphid identification and counting based on smartphone and machine vision. *Journal of Sensors*, 2017:1–7, 08 2017. doi:10.1155/2017/3964376.

[28] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 309–314, New York, NY, USA, 2004. Association for Computing Machinery. URL: https://doi.org/10.1145/1186562.1015720, doi:10.1145/1186562.1015720.

[29] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972. URL: https://doi.org/10.1145/361237.361242, doi:10.1145/361237.361242.

[30] Guotai Wang, Wenqi Li, Sébastien Ourselin, and Tom Vercauteren. Automatic brain tumor segmentation using cascaded anisotropic convolutional neural networks. *Lecture Notes in Computer Science*, page 178–190, 2018. URL: http://dx.doi.org/10.1007/978-3-319-75238-9_16, doi:10.1007/978-3-319-75238-9_16.

[31] Yading Yuan, Ming Chao, and Yeh-Chi Lo. Automatic skin lesion segmentation using deep fully convolutional networks with jaccard distance. *IEEE Transactions on Medical Imaging*, PP:1–1, 04 2017. doi:10.1109/TMI.2017.2695227.

[32] Nianfeng Liu, Haiqing Li, Man Zhang, Jing Liu, Zhenan Sun, and Tieniu Tan. Accurate iris segmentation in non-cooperative environments using fully convolutional networks. pages 1–8, 06 2016. doi:10.1109/ICB.2016.7550055.

[33] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation, 2015. arXiv:1505.04366.

[34] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016. arXiv:1511.00561.

[35] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation, 2020. arXiv:1909.11065.

[36] UNI Freiburg. U-net: Convolutional networks for biomedical image segmentation. https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/. Accessed: 2021-01-30.

[37] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation, 2016. arXiv:1606.04797.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. arXiv:1311.2524.

[39] Ross Girshick. Fast r-cnn, 2015. arXiv:1504.08083.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. arXiv:1506.01497.

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. arXiv:1703.06870.

[42] COCO. Coco. https://cocodataset.org/#home. Accessed: 2021-02-05.

[43] Neeraj Kumar, Peter Belhumeur, and Shree Nayar. Facetracer: A search engine for large collections of images with faces. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 340–353, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[44] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[45] Z. Huang, X. Zhao, S. Shan, R. Wang, and X. Chen. Coupling alignments with recognition for still-to-video face recognition. In *2013 IEEE International Conference on Computer Vision*, pages 3296–3303, 2013. doi:10.1109/ICCV.2013.409.

[46] Yue Wu and Qiang Ji. Facial landmark detection: A literature survey. *International Journal of Computer Vision*, 127(2):115–142, May 2018. URL: http://dx.doi.org/10.1007/s11263-018-1097-z, doi:10.1007/s11263-018-1097-z.

[47] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 International Conference on 3D Vision (3DV)*, pages 484–494, 2018. doi:10.1109/3DV.2018.00062.

[48] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[49] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Fashion landmark detection in the wild. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 229–245, Cham, 2016. Springer International Publishing.

[50] Y. Wu and Q. Ji. Robust facial landmark detection under significant head poses and occlusion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3658–3666, 2015. doi:10.1109/ICCV.2015.417.

[51] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3483, 2013. doi:10.1109/CVPR.2013.446.

[52] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition, 2017. arXiv:1603.01249.

[53] Erjin Zhou, Haoqiang Fan, Zhimin Cao, Yuning Jiang, and qi Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. pages 386–391, 12 2013. doi:10.1109/ICCVW.2013.58.

[54] Opencv. Opencv. https://opencv.org/. Accessed: 2021-02-05.

[55] Anubhav Agarwal, C Jawahar, and P Narayanan. A survey of planar homography estimation techniques. 07 2008.

[56] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22:1330 – 1334, 12 2000. doi:10.1109/34.888718.

[57] Antonio Criminisi, Ian Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[58] H-Y Shum and Richard Szeliski. Construction of panoramic image mosaics with global and local alignment. In *Panoramic vision*, pages 227–268. Springer, 2001.

[59] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 251–258, USA, 1997. ACM Press/Addison-Wesley Publishing Co. URL: https://doi.org/10.1145/258734.258861, doi:10.1145/258734.258861.

[60] M. Brown, A. Majumder, and R. Yang. Camera-based calibration techniques for seamless multiprojector displays. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):193–206, 2005. doi:10.1109/TVCG.2005.27.

[61] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[62] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[63] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[64] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[65] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2018. arXiv:1712.07629.

[66] Rui Zeng, Simon Denman, Sridha Sridharan, and Clinton Fookes. *Rethinking Planar Homography Estimation Using Perspective Fields*, pages 571–586. 05 2019. doi:10.1007/978-3-030-20876-9_36.

[67] Syed Ammar Abbas and Sibt ul Hussain. Recovering homography from camera captured documents using convolutional neural networks. *CoRR*, abs/1709.03524, 2017. URL: http://arxiv.org/abs/1709.03524, arXiv:1709.03524.

[68] Lorand Dobai and Mihai Teletin. A document detection technique using convolutional neural networks for optical character recognition systems. In *ESANN*, 2019.

[69] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. arXiv:1704.04861.

[70] Khurram Javed and Faisal Shafait. Real-time document localization in natural images by recursive application of a cnn. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 105–110, 2017. doi:10.1109/ICDAR.2017.26.

[71] Anna Zhu, Chen Zhang, Zhi Li, and Shengwu Xiong. Coarse-to-fine document localization in natural scene image with regional attention and recursive corner refinement. *International Journal on Document Analysis and Recognition (IJDAR)*, 22, 09 2019. doi:10.1007/s10032-019-00341-0.

[72] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[73] K. Ma, Z. Shu, X. Bai, J. Wang, and D. Samaras. Docunet: Document image unwarping via a stacked u-net. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4709, 2018. doi:10.1109/CVPR.2018.00494.

[74] Sagnik Das, Ke Ma, Zhixin Shu, Dimitris Samaras, and Roy Shilkrot. Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 131–140, 2019.

[75] TensorFlow. Tensorflow. https://www.tensorflow.org/. Accessed: 2021-02-05.

[76] Keras. Keras. https://keras.io/. Accessed: 2021-06-27.

[77] Kaggle. Kaggle. https://www.kaggle.com/. Accessed: 2021-02-05.

[78] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. arXiv:1801.04381.

[79] Zhengyou Zhang and Li-wei He. Whiteboard scanning and image enhancement. *Digital Signal Processing*, 17:414–432, 03 2007. doi:10.1016/j.dsp.2006.05.006.

[80] Scikit-learn. Scikit-learn. https://scikit-learn.org/stable/. Accessed: 2021-06-27.