Automatic landmarking for 2D biological images : image processing with and without deep learning methods

van Linh Le

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

SPÉCIALITÉ: INFORMATIQUE

Par Van Linh LE

Marquage automatique des images biologiques 2D: traitement d'images avec et sans méthodes d'apprentissage profond

Sous la direction de: Marie BEURTON-AIMAR
Co-encadrement: Akka ZEMMARI

Soutenue le 19 Novembre 2019

Membres du jury :

M.	DESBARATS	Pascal	Professeur	Université de Bordeaux	Président
Mme.	BEURTON-AIMAR	Marie	Maître de conférence/HDR	Université de Bordeaux	Directrice de Thèse
M.	BRUN	Luc	Professeur	Université de Caen	Rapporteur
M.	ANDREY	Philippe	Directeur de Recherche	INRA	Rapporteur
M.	ZEMMARI	Akka	Maître de conférence/HDR	Université de Bordeaux	Examinateur
M.	RAGOT	Nicolas	Maître de conférence/HDR	Université de Tours	Examinateur
M.	PARISEY	Nicolas	Ingénieur de recherche/HDR	INRA	Invité

université de BORDEAUX

THESIS PRESENTED

TO OBTAIN THE DOCTORAL DIPLOMA OF

UNIVERSITY OF BORDEAUX

DOCTORAL SCHOOL OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALITY: COMPUTER SCIENCE

Presented by Van Linh LE

Automatic landmarking for 2D biological images: image processing with and without deep learning methods

Supervisor: Marie BEURTON-AIMAR
Co-supervisor: Akka ZEMMARI

Defended in November 19th 2019

Committee in charge :

M.	DESBARATS	Pascal	Professor	University of Bordeaux	President
Mme.	BEURTON-AIMAR	Marie	Associate professor/HDR	University de Bordeaux	Supervisor
M.	BRUN	Luc	Professor	University of Caen	Reviewer
M.	ANDREY	Philippe	Research director	INRA	Reviewer
M.	ZEMMARI	Akka	Associate professor/HDR	University de Bordeaux	Examiner
M.	RAGOT	Nicolas	Associate professor/HDR	University of Tours	Examiner
M.	PARISEY	Nicolas	Researcher engineer/HDR	INRA	Invite

Dedication

This dissertation is dedicated to my grandparents, my parents, my parents-in-law, my brothers, my sisters and my loving wife, VU Thi Phuong Thao, who has been a source of encouragement and inspiration to me throughout my life.

Acknowledgements

This thesis marks a new stage on the adventure of my education to look for the knowledge of humanity. I would like to acknowledge certain people who have followed me on this journey.

First and foremost, I would like to sincerely thank my supervisor, Assoc. Prof. BEURTON-AIMAR Marie, for her supports during the past three years. The supports are not only in invaluable ideas and discussions but also many things in my life.

I wish also to thank ZEMMARI Akka, my co-supervior and PARISEY Nicolas, my collaborator, for their helpful advices, their comments, and suggestions, especially on the explanations of the obtained results.

I would like to express to Assoc. Prof. BLANC Carole and Assoc. Prof. MAABOUT Sofian, who are my monitoring committees, for their concerns, pieces of advice during my thesis.

I would like to thank the French Ministry of Higher Education, Research and Innovation for funding three years of my studies.

I would like to send my sincere thank to the people in LaBRI, who have supported from technical to administration during my stay.

I would like to give a special thank to my wife, Thao, who has given up everything to stand behind me and to support me during my studies. I am also thankful to my parents, my parents in law, all my sisters and brothers in my family have always encouraged me during this experience.

From my heart, thank you all.

33400 Talence, France

November, 2019

LE Van Linh

Title: Automatic landmarking for 2D biological images: image processing with and without deep learning methods

Abstract: In various applications of different domains, e.g., biomedical or biological, many analyses have used landmarks as one of the input data. For example, biologists do not only use landmarks for measuring the form of the object, but also for determining the similarity between two objects. In biology, landmarks are usually used in the analysis to detect the variations of inter-organisms. Most often, the landmarks are manually provided. This operation is time-consuming and is pretty difficult to process on a large dataset. In recent years, several methods have been proposed to predict landmarks automatically. But, it exists of the hardness because these methods focused on the specific data. This thesis focuses on the automatic determination of landmarks on biological images, more specifically on two-dimensional images of beetles. In our research, we have collaborated with biologists to build a dataset including the images of 293 beetles. For each beetle in this dataset, 5 images correspond to 5 parts have been taken into account, e.g., head, body, pronotum, left and right mandible. Along with each image, a set of landmarks has been manually proposed by biologists. In the context of this work, these manual landmarks have been used as ground truth to evaluate the predicted ones. Firstly, we have brought a method which was applied on fly wings, to apply on our dataset with the aim to test the suitability of image processing techniques on our problem. Secondly, we have developed a method consisting of several stages, so-called Iterative Method to Estimate Landmarks (IMEL), to automatically provide the landmarks on the images. Our proposition is a pipeline of three steps: segmentation, registration of two images by using an iteration of transformation operations, and verification of estimated landmarks with the help of SIFT descriptors. These two first works have been done on the mandible images which are not so difficult to segment, to analyze by using the relevant image processing algorithms. As the results of the first work, the estimated landmarks on mandibles have been used to compute the centroid

size and the distances to the manual ones. These landmarks have been transferred to biologists, and they have confirmed that our prediction is good enough to use in the morphometry analysis instead of using manual positions. Based on the success of the method on the mandibles, we continue applying it to other parts of the beetle. Unfortunately, the other parts present the characteristics that reveal them out of reach for segmentation. It is why we have turned to Deep Learning methods. We have designed a new model of Convolutional Neural Network, named EB-Net, to predict the landmarks on remaining images. Our architecture is based on the concept of the Elementary Block. Each Elementary Block is a composite of one convolutional, one maximum pooling, and one dropout layer. For our objective, EB-Net consists of three blocks with different parameters of the layers at each block. In addition, we have proposed a new procedure to augment the number of images in our dataset, which is seen as our limitation to apply deep learning. As the first results on the three remaining parts, the predicted landmarks are good enough in the statistical points of view. However, they are required to improve when we display them on the images. Finally, to improve the quality of predicted coordinates, we have employed Transfer Learning, another technique of Deep Learning. In order to do that, we trained EB-Net on a public facial keypoints. Then, the parameter values have been transferred to fine-tune on the beetle's images. The final results have shown that we have improved the coordinates of landmarks with the fine-tuning process. They are more close to the manual ones. These results have been discussed with biologists, and they have confirmed that the quality of predicted landmarks is statistically good enough to replace the manual landmarks for most of the different morphometry analysis.

Keywords : Deep learning, Convolutional Neural Network, Morphometric analysis, Key points detection, Landmark, Probabilistic Hough Transform, SIFT descriptor.

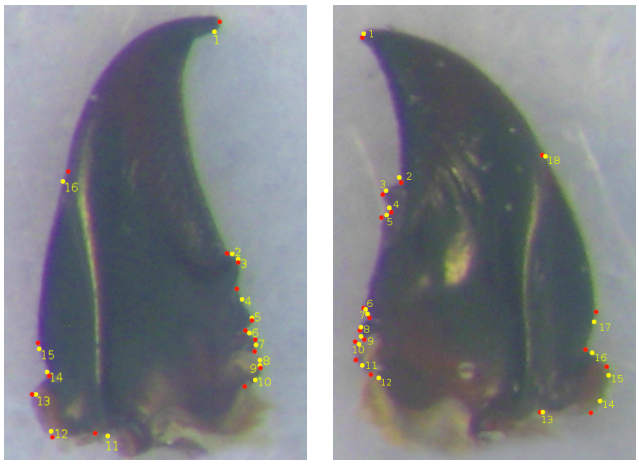
Titre : Marquage automatique des images biologiques 2D: traitement d'images avec et sans méthodes d'apprentissage pronfond

Résumé : Les points de repères ou *landmarks* sont utilisés dans les applications of différents domaines tel que la biologie ou la médecine. Ils ne sont pas utilisés pour uniquement mesurer la forme d' un objet, mais aussi pour évaluer une similarité/dissimilarité entre deux objets ou encore la variation inter-organismes en biologie, par exemple. Il est à noter que positionner des landmarks peut se réveler une tâche très lourde en fonction du nombre affecté à chaque objet d'étude, d'autant qu'ils le sont en générale manuellement. Si ces dernières années, plusieurs méthodes ont été proposées pour prédire automatiquement les landmarks, ces méthodes sont souvent spécifiques à un jeu de données et changer ces données suppose le plus souvent de refaire l'étude du modèle d' estimation des coordonnées des landmarks. Cette thèse porte sur la détermination automatique de landmarks sur des images biologiques, plus spécifiquement des images 2D de coléoptères. Dans le cadre de nos recherches, nous avons collaboré avec l'équipe Démécologie de l'INRA de Rennes qui disposait d'une collection d'images de 293 coléoptères, pour chacun d'eux 5 images ont été réalisées pour 5 parties différentes de l'animal: le pronotum, la tête, l'élytre et les mandibules droite et gauche (Figure 1). Pour chaque image, un ensemble de landmarks a été fixé manuellement par un biologiste. **Ces landmarks seront utilisés tout au long de cette étude comme vérité terrain pour évaluer la qualité des estimations automatiques.**

Figure 1: *Les images de cinq parties de notre ensemble de données. De gauche à droite: mandibule gauche, mandibule droite, pronotum, élytre et tête.*

Au cours de la première année de thèse, nous avons appliqué une méthode de la littérature (Palaniswamy et al., 2010) appliquée aux ailes de Drosophile, ceci dans le but de tester la faisabilité et la pertinence d' une approche automatisée dans notre contexte. Les premiers résultats étant encourageant, nous avons poursuivi le travail en proposant, notre propre pipeline de traitement d'images qui améliore les résultats obtenus précédemment par la méthode de Palaniswamy. Ce pipeline comporte une étape de segmentation, d'alignement par itération d'opérations de transformation et une vérification des coordonnées estimées grâce à des descripteurs SIFT.

Le pipeline a été appliqué aux images de mandibules et a fourni des résultats qui ont été considérés par les biologistes comme pouvant remplacer les landmarks manuels dans le calcul du centroide des mandibules et de leur taille, pour les analyses de morphométrie avec un degré de confiance satisfaisant, statistiquement parlant. Par rapport aux résultats obtenus par la méthode de Palaniswamy, notre pipeline a fourni des landmarks plus proches des landmarks manuels. Cependant, les résultats n'étaient pas au même niveau sur les mandibules gauche et droite. Une explication possible de la différence de ces résultats est une plus grande variabilité dans les formes des mandibules gauches. La figure 2 illustre les landmarks planifiés et manuels sur les mandibules.



a . Left mandible. b . Right mandible.

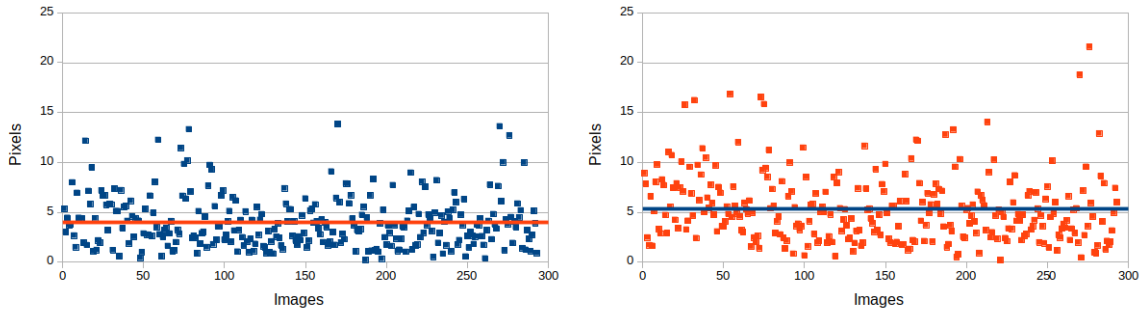
Figure 2: *Les landmarks prédits et manuels sur les mandibules gauche et droite. Les points rouge/jaune présentent les landmarks manuels et estimées.*

Ce travail a fait l'objet d'une présentation orale à la conférence internationale en Europe centrale sur l'infographie, la visualisation et la vision par ordinateur (WSCG-2017) ainsi que d'un poster à la conférence nationale GRETSI, 2017.

À partir des résultats satisfaisants sur les mandibules, nous avons appliqué notre modèle aux autres parties: pronotum, tête, élytre. Dès les premières analyses sur le pronotum, l'étape de segmentation s'est révélée être un goulet d'étranglement extrêmement difficile à dépasser. Nous avons donc décidé de nous tourner vers des modèles d'apprentissage profond ne nécessitant pas de segmentation préalable.

Nous avons proposé une architecture de réseau de convolution (CNN) à partir d'un *bloc élémentaire* (EB) répété plusieurs fois dans le réseau. Un bloc élémentaire est composé d'une couche de *convolution*, une de *max pooling*, et une de *dropout*. Ce réseau que nous avons nommé EB-Net contient 3 blocs avec différents paramètres pour chaque bloc. Nous avons également proposé une nouvelle procédure d'augmentation des données basées sur la sélection de canaux de couleur de l'image afin de produire un jeu de données consistant pour des méthodes d' apprentissage profond. Nous avons pu générer près de 2000 images pour l'entraînement du réseau. Ces résultats obtenus grâce à ce modèle d'apprentissage profond montrent que les landmarks estimés sur les 3 autres parties du coléoptère: pronotum, élytre et tête, sont statistiquement satisfaisants et peuvent être utilisés pour des analyses de morphomètrie. Toutefois, en terme de visualisation sur l'image, force est de constater que les landmarks estimés peuvent se situer parfois assez loin visuellement des landmarks manuels. Nous avons donc souhaité aller plus loin dans l'analyse en calculant la distance pour chaque landmarks entre la vérité terrain et l'estimation afin d'améliorer cet aspect de nos résultats. Nous avons illustré la distribution des distances aux landmarks 1 et 6 du pronotum. La figure 3 montre pour landmarks 1 et 6 du pronotum qu'effectivement la distribution des values peut être large.

Ce travail a fait l'objet d'une présentation orale à la conférence internationale sur l'analyse multimédia et la reconnaissance des formes (MAPR-2018).



a . The 1^{st} landmark

b . The 6^{th} landmark

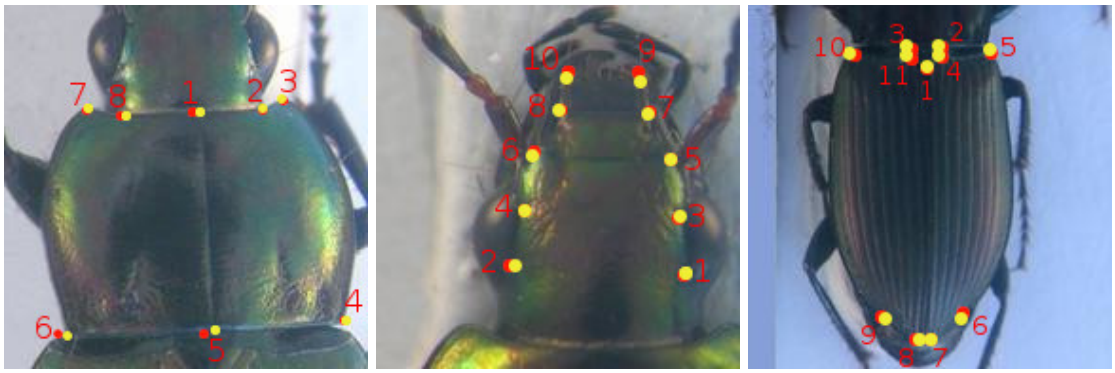
Figure 3: *La distribution des distances entre les landmarks manuels et estimées de toutes les images pour le meilleur* (1) *et le pire cas* (6).

Supposant que notre jeu de données était probablement trop petit nous nous sommes tournes vers l'ajout d'une étape de transfert d'apprentissage depuis une autre base de données. Nous avons sélectionné pour ce faire une base de données de landmarks sur des visages[1], c'est une application très connue et plusieurs modèles pour traiter cette base, sont disponibles de type AlexNet, VGG16, ... Mais les résultats issus d'un transfert de paramètres depuis cette base vers notre jeu de données avec ces réseaux se sont révélés insuffisants. En conséquence, nous avons décidé de customiser notre modèle, EB-Net, pour qu'il puisse apprendre les landmarks de la nouvelle base de données. Ensuite, de transférer les valeurs de paramètre pour affiner les images du coléoptère. Pendant le processus de réglage fin (*fine-tuning*), les paramètres des calculs sont réglés pour poursuivre l'apprentissage.

Les landmarks estimés sont ensuite évalués de la même manière que précédemment: premièrement, les distances moyennes des points de repère ont été prises en compte. On peut noter que toutes les valeurs moyennes ont été abaissées de 1 à $1,5$ pixels dans les trois séries d'images à l'aide du processus de réglage fin; ensuite, les distributions de distance à chaque position ont été évaluées. Ces distances ont été réduites, elles sont plus proches de la plage entre 0 et la valeur moyenne, en particulier pour la plupart

[1]https://www.kaggle.com/c/facial-keypoints-detection/data

des cas très éloignés des valeurs moyennes dans le précédent modèle. Afin d'établir une comparaison avec les méthodes de traitement d'images, le processus de réglage fin a été appliqué aux images des mandibules. Les résultats obtenus ont montré que les landmarks estimés sont plus stables à l'aide du processus de réglage fin. Remarquablement, il y a une forte amélioration pour les positions proche de zone difficile à segmenter.



a . Pronotum b . Head c . Elytra

Figure 4: *L'emplacement des points de repère prévus dans un cas de chaque partie. Les points rouge/jaune représentent les landmarks prévus/manuels.*

En nous basant sur le succès de la mise au point sur EB-Net, nous avons voulu vérifier si les résultats sont améliorés avec d'autres compositions d'Elementary Block et d'autres paramètres. Pour ce faire, nous avons ajouté un bloc élémentaire à notre configuration precedant et changé la fonction d'activation en fonction LeakyReLU. La nouvelle version d'EB-Net a été entraînée sur l'ensemble des données faciales avant d'être utilisé pour affiner le réglage sur toutes les parties des coléoptères. Les landmarks estimés ont ensuite été utilisés pour calculer la distance par rapport aux landmarks manuels et pour calculer la valeur moyenne à chaque position. Les scores ont été améliorés dans tous les cas de coléoptères. L'amélioration se situe plus généralement autour de $0,5$ pixels. En conséquence, les landmarks prédits sont plus proches des landmarks manuels, en particulier sur les cas qui sont difficiles à prédire. Les résultats finals ont été discutés avec des biologistes et ils ont confirmé que la qualité des landmarks prédits est suffisamment bonne sur le plan statistique pour remplacer les landmarks manuels pour la plupart

des analyses de morphométrie différentes. Comme le montre la figure 4 l'affichage des landmarks estimées apparaît aussi comme très proche des manuels.

Mots clés : Deep learning, Convolutional Neural Network, Morphométrique, Détection de points clés, Landmark, Probabilistic Hough Transform, SIFT descriptor.

UMR 5800 – Laboratoire Bordelais de Recherche en Informatique (LaBRI)

Université de Bordeaux

351, cours de la Libération – F-33405 TALENCE

Acronyms

1D one-dimensional. 54

2D two-dimensional. 31, 33, 36, 40, 48, 54, 55, 60, 68, 70, 88, 93, 103, 117, 120, 138, 173

3D three-dimensional. 31, 40, 93

CNN Convolutional Neural Network. 32, 33, 57, 88, 92, 93, 95, 97, 98, 101–103, 106, 112, 117, 120, 121, 138

CNNs Convolutional Neural Networks. 93, 98

CONV Convolutional. 93, 94

DNNs Deep Neural Networks. 92

DoG Difference of Gaussian. 63

DROP Dropout. 95

EB Elementary Block. 32, 110, 111, 132, 133, 135, 138

EB-Net Elementary Blocks Network. 22, 32, 33, 106, 111, 113, 114, 116, 117, 120–124, 129, 132–134, 136, 138, 139, 176, 177, 182, 185

EmotiW Emotion Recognition in the Wild. 102

FC Fully-connected. 95, 96

GUI Graphics User Interface. 174, 175

HT Hough Transform. 59, 60

IMEL Iterative Method to Estimate Landmarks. 25, 32, 33, 70, 71, 73, 74, 78, 79, 81, 82, 85

PCA Principle Component Analysis. 83

PGH Pairwise Geometric Histogram. 70, 165–167

PHT Probabilistic Hough Transform. 58, 60, 71, 167–169, 173

POOL Pooling. 94

RNNs Recurrent Neural Networks. 92

SFEW Static Facial Expression Recognition in the Wild. 102

SIFT Scale-invariant feature transform. 32, 63, 65, 66, 71, 76, 77, 79, 83, 85, 138

SURF Speeded up robust features. 63–65

List of Figures

List of Tables

Contents

Introduction

Motivation

A point of interest, or a *key point*, or a **landmark** is a point in an image that may contain useful information and be stable when the image changes. It is a candidate to be effective for many applications in various domains such as object recognition in computer vision or human face detection. In biology, the landmark is an essential type of data in morphometry analysis to evaluate the form/shape of the object; or in Procrustes analysis to identify the variation of anatomical features. From the set of landmarks, biologists can appreciate the morphology of an organism. It can be used to measure the influence of ecological factors and the development of the organism. Currently, the landmarks are manually given by biologists. This procedure is costly and impossible to achieve in good conditions when we work on a large dataset. Consequently, the development of methods that can automatically produce the landmarks in the biology domain is very noteworthy.

In computer vision, *key points detection* offers methods to automatically determine landmarks in two-dimensional (2D) or three-dimensional (3D) images. In which, most of the methods are focused on the landmarks which stayed on the contours of the object such as the leaf or wing contours; or located inside the object, for example, human facial key points (e.g., eyes, nose, eyebrows). In collaboration with biologists, we have used a dataset including 2D images of anatomical parts of beetle, e.g., head, pronotum, elytra, left and right mandibles, as experimental data. A set of landmarks has been set

manually for all parts and has been used as ground truth for this work. For each part, the number and the location of landmarks are different. *For example*, the left mandible has 16 landmarks, and they are located on the contour of the mandible; while pronotum has 8 landmarks and their positions are indicated both on contour and inside the object. In this project, our team focuses on the methods to automate such landmarks setting. This task is the main goal of this Ph.D. thesis.

This work has been done in two parts. The first year has been dedicated to classical image processing methods to experiment if they work on our data or not? These methods often require a step of shape segmentation to detect the features which are used in the following steps. In order to do that, we have studied and applied the method of Palaniswamy [PTK10] to work on the mandible images. Then, we have proposed a method, IMEL, which combines an iterative process of registration and verification of Scale-invariant feature transform (SIFT) descriptors to estimate landmarks. We will see that even IMEL has been successfully applied for the left and right mandibles images, it fails to work with the head, the pronotum, and the elytra. It explains why we have turned to another way to process our dataset.

In the second part of the work, we have applied Deep Learning methods, specifically Convolutional Neural Network (CNN), to predict the landmarks. A CNN combines a sequence of operations such as convoluting, down-sampling, non-linearity, ... These operations are organized into the layers to study different level features of the input before giving the output at the last layer. As usual, several models have been tried before obtaining a stable model for landmarking. The results of this studying lead us to define a concept of Elementary Block (EB), which is the base to propose Elementary Blocks Network (EB-Net) architecture. Finally, our proposed model has been completed by a *fine-tuning* module that returns statistically robust estimated landmarks to biologists.

Organisation of the thesis

Chapter 1 of the thesis gives the principal context of morphometry analysis, landmark and landmark detection which are focused through the thesis. This chapter is closed by introducing our dataset which will be used to evaluate the methods during the thesis.

Chapter 2 overviews several image processing techniques that can be combined together for the landmarking task. This chapter presents also some methods which can be applied to detect the landmarks on 2D images.

Chapter 3 presents the IMEL method to automate the landmark setting. This method includes several steps to extract the features before giving the coordinates of landmarks. The experiments are done on two sets of images: left and right mandibles.

Chapter 4 begins a second part of the thesis speaking about deep learning algorithms. This chapter begins with a short description of machine learning and deep learning context. Then, a brief overview of CNNs which are used to solve the problems on grid topology data will be presented. After that, a short preview of the transfer learning technique will be given.

Chapter 5 describes our first proposition of CNN, EB-Net architecture, to predict landmarks on beetle's images. It also presents a new procedure for data augmentation which was used to increase the number of items in the dataset. The performance of this method is evaluated on the remaining 3 parts of the beetle: head, pronotum, and elytra.

Chapter 6 presents a complementary strategy to apply deep learning for landmark detection by using transfer learning. Accordingly, the proposed CNN model in Chapter 5 is pre-trained on a facial key points dataset before transferring the parameter values to fine-tune on beetle's images. The evaluation is mainly done on three parts of beetles as in Chapter 5, but also on the mandibles to produce a comparison with the obtained results in Chapter 3. We also evaluate the feedback from some modifications of EB-Net model. Finally, we discuss the perspectives on our works before the conclusion.

Chapter 1

Landmark-based geometric morphometry

Contents

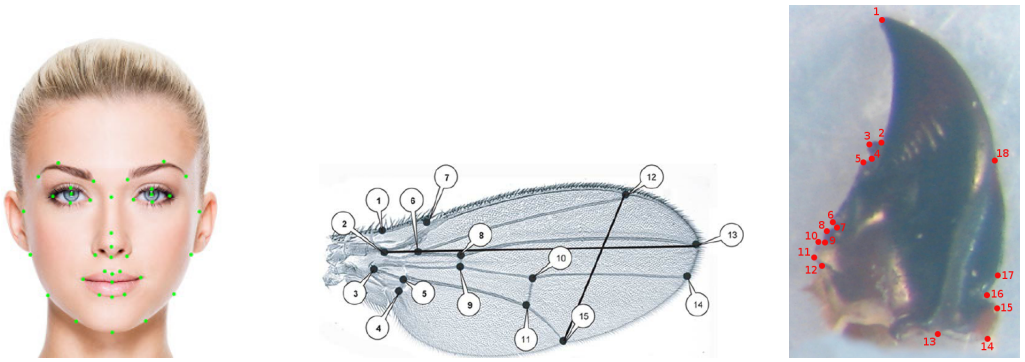
In this chapter, we first describe the context of landmarks and their applications. After that, we briefly introduce the facial landmarks which are linked to our problem. Then, we will present a short overview of anatomical landmarks which are our main studied objects in this thesis. Next, we expose the landmark detection problem in 2D images. Finally, we outline the experimental data which has been used in this work.

1.1 Context

In morphometry analysis, the term of landmarks is most often used to define anchor positions, control points on the object, or specific points on the contours. There are three basic types of landmarks: anatomical landmarks, mathematical landmarks, and pseudo-landmarks [DM92, ZSS12].

"Anatomical landmarks are points assigned by an expert that corresponds between organisms in some biologically meaningful way" [Bro12]. They are discrete anatomical pieces that have the same positions in all specimens in the study [ZSS12]. "Mathematical landmarks are points located on an object in accordance with some mathematical or geometrical properties" [Bro12], for instance, a high curvature point or/and extreme point. "Pseudo-landmarks are constructed points on an organism, located either around the outline or in between anatomical or mathematical landmarks" [Bro12]. A typical example is a set of points positioned equally between two anatomical landmarks to get more sample points on the shape. Pseudo-landmarks could also be useful during shape matching when the matching process requires a large number of points [DM92].

In an application, the number of landmarks and their definitions can vary and depend on the objectives of the studying. Figure 1.1 gives three examples of the landmarks in different studies, e.g., 40 landmarks on a human face, 15 landmarks on a fly's wing, and 18 landmarks on a right beetle's mandible. In the context of this thesis, we consider the anatomical landmarks provided by biologists. This dataset of landmarks will be considered as the ground truth that we want to achieve in all experiments.



a . Facial landmarks

b . Landmarks on a fly's wing

c . Landmarks on a mandible

Figure 1.1: The location of landmarks in various applications. Images from Facial technologies[1] and Giga Science [2]

Nowadays, landmarks are usually used in many applications of different domains such as computer vision [Lin15, MNR90, Can87], shape analysis of organisms in biology [HMGC03], face recognition [Z+14], and biomedical investigations [FP10].

In **computer vision**, landmarks are positions in the image that are invariant when the scene changes. The correctness of the landmark positions is essential and it greatly contributes to the accuracy of the methods. Several important types of research can be mentioned such as:

- Corner detection [MNR90]: trying to find the intersection point of two edges which have different directions in a local neighborhood of the points.
- Edge detection [Can87]: identifying the points in the image that have high brightness. These points are typically organized into a set of curves.
- Image matching [JC02, Lin15]: finding the key points in the images, then trying to identify the matches between two images.

In **biomedical examinations**, landmarks appear in the studies of human medical images, for example, X-ray or MRI images, to examine the possible signs of sickness or the

[1] http://www.arcsoft.com/technology/face.html
[2] https://academic.oup.com/gigascience/article/4/1/s13742-015-0065-6/2707551

symptoms of a medical disorder in their body. The applications could be cephalometric analysis [FP10], or brain registration [GBR+99]. In another approach, landmarks are also used in the **biometric systems**, which are the real-time systems to identify the unique characteristics for each person. In these systems, the particular characteristics of a person are extracted and compared to a library, which contains examples of many people, for a detection task. Landmarks have been identified on data as match points to compare the human attributes. We mention the applications: fingerprint verification [NB02, PPJ08], iris scanning [SK17, VCFR09], facial recognition [SWT13, Z+14]. The landmarks, which are used in the face detection tasks, are called facial landmarks. They will be described in Section 1.2.

In biology, providing landmarks to biologists is highly beneficial. They are useful inputs in analyses of *Procrustes* [Gow01, Dry14] or **morphometric analysis** [Boo97, WS10]. In the Procrustes, landmarks are most often used to analyze the shape for identifying the change of anatomical characteristics. In the morphometric analysis, landmarks are used to detect the impact of mutations, the changing in the body, or the effects of the environment on the shape [Boo97]. These analyses are usually focused on living objects, samples of the organism, or fossil records.

1.2 Facial landmarks

Facial landmarks are known as the specific landmarks for human face detection. These landmarks are usually located around facial components such as eyes, mouth, nose, etc. According to the application, the different number of facial landmarks are set, e.g., 5-points model, 8-points model, or 17-points model. However, whatever the number of positions, these points should cover several different areas on the face (e.g., eyes, mouth, and nose) because most of them carry essential information for diverse purposes. In practice, the process usually starts by determining the face region using a rectangle bounding box. Then, the searching process could be used to find the location of facial landmarks. It most often starts with an initialization, then moves to a better

position step by step until reaching convergence [WGT+18]. The facial keypoints detection methods could be divided into four groups: constrained local model (CLM)-based, active appearance model (AAM)-based, regression-based, and others [WGT+18]. CLM-based methods consist of a shape model and a number of local experts, each of which is utilized to detect a facial feature point [CILS12, LBL+12, AZCP13, BJKK13]. AAM-based techniques fit a shape model to an image my minimizing texture synthesis errors [ASWC13, MCB13]. Regression-based methods directly learn a mapping function from facial image appearance to facial feature points [MVBP12, BAPD13, YP13]. Besides three main categories, there are also other methods, such as graphical model-based methods [UFH12, ZSCC13], independent facial feature point detectors [SLBW13], and deep learning-based methods [SWT13, Z+14].

1.3 Anatomical landmarks

Anatomical landmarks or landmarks (for short) are the biologically meaningful points in an organism that can store important information about the object. According to biologists, the landmarks can be classified into 3 categories [Boo97, ZSS12]: (1) the landmarks are clearly and locally defined by particular structures close to the point, for example, the intersection between veins on the fly wings; (2) intermediate class, the landmarks are located at the local minima and maxima of curvature, such as a tip of the structure; (3) landmarks are not defined by any structure; instead, they are defined solely by being at an extreme distance from another point.

In practice, landmarks can be used to reconstruct the shape of an organism and to apply some morphometric analysis in the shape. For example, it is useful in biology to evaluate the evolution of species, as well as the influence of environmental factors on the development of the organism. The question to choose the number of landmarks and their positions is a difficult question. They are usually suggested by the biologist depending on which kind of application and the studied objects that they focus on.

Currently, landmarks in the biological applications have been manually determined

by experts. Firstly, the object is captured by a specific device such as a camera, trinocular magnifier, scanner, ... to obtain a digital image. Then, the landmarks are manually set on the digital images by using a particular program. For example, tpsDig [Roh04] is a software that able to display the image, to set the location of the landmarks manually, and to export the coordinates of the landmarks into a text file. However, this process meets some difficulties in real work, such as updating the coordinates of the landmarks, crashing software, or requiring many people to work on the data. These disadvantages make manually setting landmarks time-consuming and difficult to reproduce. So, the method which provides the landmarks automatically could be very interesting and this is the main goal of this thesis.

1.4 Landmark detection

Landmark detection refers to the methods which are used to determine the landmarks on 2D (or 3D) images. Based on the characteristics of the input images, the applied methods could be different, and they can be grouped into various categories. In the context of this thesis, we consider the methods belonging to two groups: in the first group, the images are analyzed by using classical image processing algorithms; while in the second group, the deep learning algorithms are applied for predicting the coordinates of the landmarks.

In the first group, the methods usually take into account the shape of the object or the relationship among the pixels. In order to detect the landmarks by considering the shape, the object shape is needed to be extracted first. Then, the list of points belonging to the shape is used to determine the location of the landmarks. In this case, the landmarks could be detected by measuring the curvature [TC89, RR92, Cor97, Wu03, MS04], or evaluating the feature spaces which are the transformation of the contours [MM92, Mok95, GGS+98]. In another approach, the descriptor for each pixel in the image can be also computed by using the relationship between it and its neighborhoods. Then, they will be used to predict the coordinates of

landmarks [Low04, BTVG06]. These methods will be presented in Chapter 2.

In the second group, Neural Networks are used with deep learning algorithms to consider the features of the images from the local level (e.g., pixels) to the abstract level (e.g., curve, edge). As usual, a set of different layers is used to compose a network for predicting the key points. Layers are ordered through the network to characterize the image features at different levels, and the final decision (key points coordinates) is given by the last layer [LYL+16, SWT13, Z+14, HGT17, SLJ+15, CQSA+16]. The methods of this group will be described in Chapter 4.

1.5 DevMAP Project

In collaboration with biologists at *Institute for Genetics, the Environment and Plant Protection (IGEPP), INRA-1349*, we work in the DevMAP project: *Développer la Morphométrie Automatisée pour aider à mesurer l'impact des Paysages anthropisés sur les traits de vie d'espèces invasives et d'auxiliaires des cultures*. The objective of this project is to quantify the mean and the variability of morphological traits of beetles in different agriculture contexts. The morphology was assessed on varied environmental contexts, especially with the agricultural system applied at the field and landscape scales. For this purpose, *Poecilus cupreus*, the beetle morphologies in conventional and organic fields have been considered. In order to do that, the Brittany lands (North-West of France) have been selected to collect the samples. After collecting in three months (from May to July 2013), a collection of 293 beetles (Figure 1.2) (147 males and 146 females/ 155 organic and 138 conventional), **DevMAP dataset**, has been established.

As usual, images of beetles have been chosen to be studied instead of using real objects for practical reasons. To create the images, biologists have used a trinocular magnifier to capture the dorsal view of beetles. For each beetle, 5 images are available corresponding to 5 body parts: *head, pronotum, elytra, left and right mandibles*. One can note that the images of head, pronotum, and elytra have been made before dissection. Then, biologists have dissected the insects to separate mandibles from the beetle's body

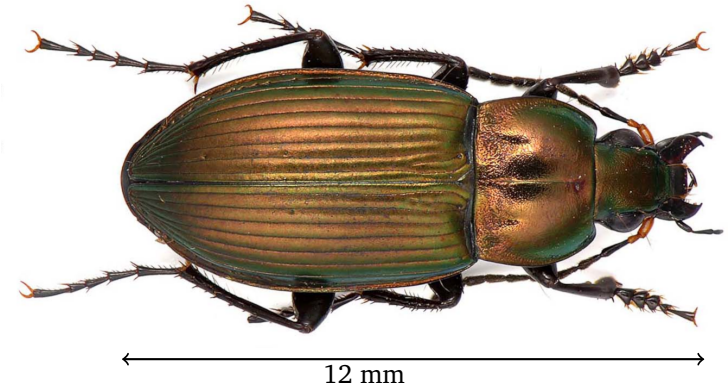


Figure 1.2: An illustration of the beetle.

before taking the photos. The images of all body parts of beetles are captured with an identical protocol by using the same camera (trinocular magnifier) with different resolutions for pieces to handle the difference between sizes of body parts. At the end of this process, all the images have been released in the RGB color mode (JPEG compression) with a size of 3264×2448 pixels. Unfortunately, biologists have not given in detail the different camera resolutions for each element. However, as we know the average size of beetles in nature ($\approx$ 12 mm). So, we can approximate the focus on each part of beetles, such as ≈ 300 pixels/$1mm$ for elytra, ≈ 600 pixels/$1mm$ for both pronotum and head, and ≈ 1500 pixels/ $1mm$ for mandibles.

The team of biologists who have initialized this project wants to work with anatomical landmarks to apply different morphometric analysis. Our goal in this context is to automatically identify the landmarks to replace the manual task of setting them. To support us in this work, they have defined a set of anatomical landmarks for each piece that they would like to obtain and providing as ground truth to evaluate our results. It is worth to note that the number of landmarks is different among the five portions of the beetle. Biologists have defined a set of manual landmarks with tpsDig2 software [Roh05]. They have set $8, 10, 11, 16,$ and 18 landmarks for each pronotum, head, elytra, left and right mandible image, respectively. Figure 1.3 shows the images and landmarks positions of each beetle's part in our dataset.

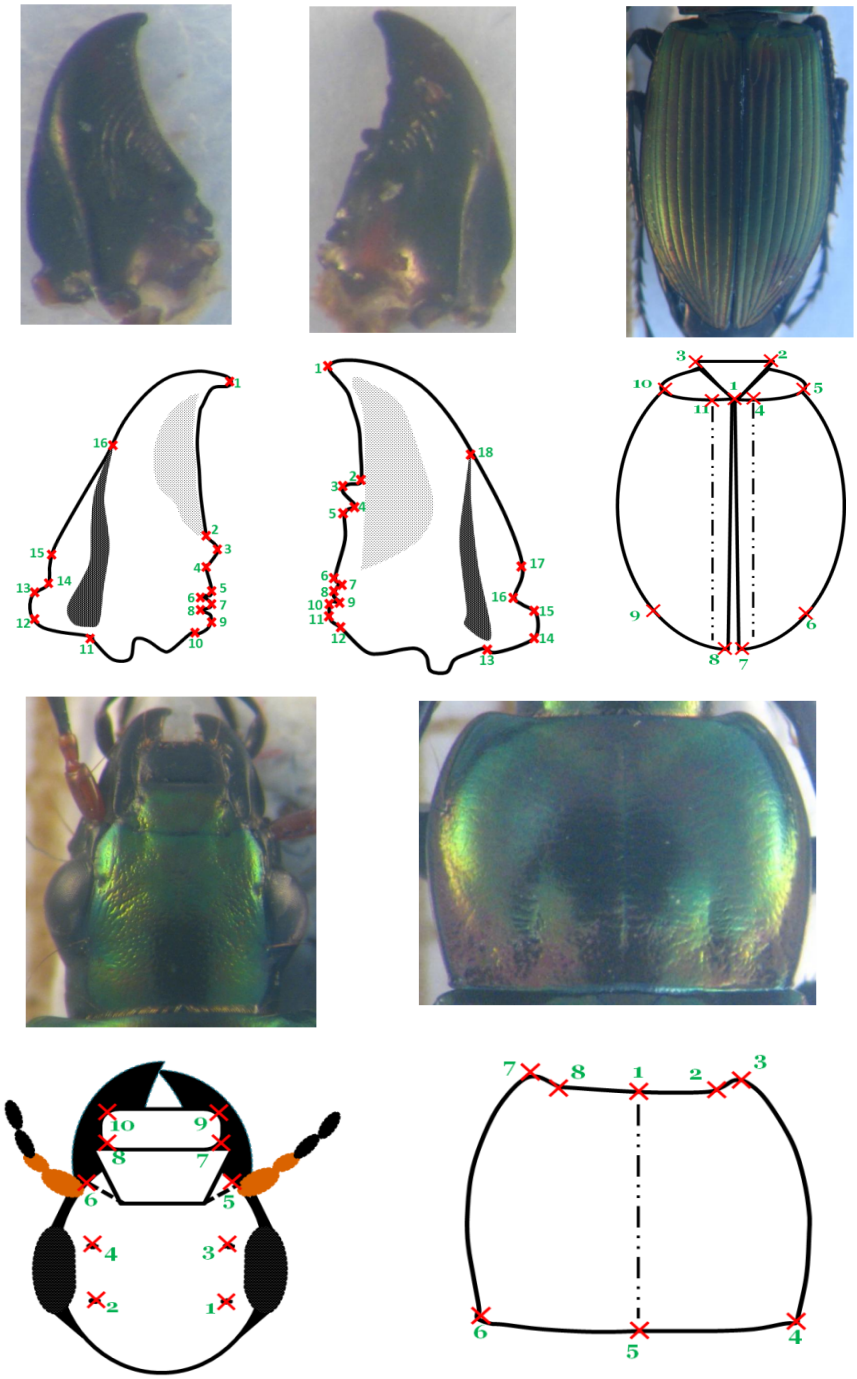


Figure 1.3: The sample images (1^{st} and 3^{rd} rows) and positions of manual landmarks (2^{nd} and 4^{th} rows) on each part in our dataset.

The beginning of this work has been analyzed an article proposed by Palaniswamy et al. [PTK10] to automatically predict landmarks on fly's wings. We have deeply studied their method and have first tried to reproduce the results on the mandible images belonging to our dataset. This work will be described in the next chapters, but we present firstly main steps of image processing related to the current work on landmarks in Chapter 2.

Part I

Automatize landmarks by using Image Processing Techniques

Chapter 2

Key points detection and Image processing methods

Contents

In image processing domain, a huge number of methods have been proposed to work on digital images and each of them performs a particular task such as feature extraction, projection, or key points detection, ... For a complex application, it is necessary to combine various algorithms for different steps, and selecting the algorithm for each step depends on the problem type, as well as the characteristics of the input images.

In this chapter, we describe the procedures that are usually used in the context of landmarks setting on 2D images: segmentation, registration, and landmarks detection. This chapter ends with a survey of some previous works that have applied these methods to identify landmarks automatically. The readers familiar with the image processing techniques can skip this chapter to go to Chapter 3 directly.

2.1 Segmentation

Segmentation is often the first step in the process of image analysis. It is used to change the representation of the image into other ones which are more meaningful and easy to analyze. This process assigns a label to each pixel of the image. Then, it outputs the sets of pixels with different characteristics. The segmented image is then used to extract features and/or to compute descriptors. Choosing a specific segmentation solution is a complex process, it depends on the objective of the application, as well as the required features for future treatments. Segmentation methods can be divided into three groups based on the considered data: *points, regions, and contours*. Nida et al. give a large list of these methods in [ZA15]. Besides the classical methods, neural networks have been applied for the segmentation task in recent years, especially on medical images. This section presents an overview of segmentation algorithms in all categories.

2.1.1 Points - based methods

The methods based on points directly consider each pixel in the image without taking care of the structure or topology of objects. These methods divide the pixels into different groups based on their properties.

Thresholding

The simplest method for segmentation uses a threshold value to turn a gray-scale image into a binary image or to change values of a color image by considering every channel [GW+02]. It is also possible to use multiple threshold values as it is in the case of hybrid thresholding.

Considering an image, $src(x, y)$, which contains a light object on a dark background and a threshold value, T. Any pixel (x, y) in the image that has the intensity $src(x, y) > T$ is called object point, otherwise it belongs to the background. The value of each pixel in the obtained image, $dst(x, y)$, is given by Equation 2.1.

$$dst(x, y) = \begin{cases} 1 & \text{if } src(x, y) > T \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

If the image contains several types of objects, multiple thresholds could be more useful. For example, if we consider an image containing two types of light objects on a dark background, it is necessary to use two threshold values for separating each object.

One can note that setting the threshold value, T, could be very tricky. Its correctness directly links to the quality of the results. When T is a constant, Equation 2.1 refers to *global thresholding*; when T changes over image, it is *variable thresholding*; and when T depends on the values of neighbor pixels, we have the *local* or *region thresholding*. In all cases, threshold values can be set both manually or automatically by analyzing the histogram of the image.

From the idea of thresholding, several methods have been proposed and widely used in practice including maximum entropy method [GS84], balanced histogram threshold-

ing [dAS08], hybrid thresholding [SKPS11], Otsu's method [LCC+01].

Clustering method

Clustering [DK32] is a task of gathering a set of objects where the objects in the same group are more similar than objects belonging to other groups. It is a classical technique to solve many tasks in different fields such as data analysis, machine learning, image analysis, data compression or computer graphics. Clustering uses distance metrics for presenting the differences between the objects, it packs the objects into the same clusters based on this distance value.

In image analysis, clustering is a frequent method for the segmentation task. It is used to partition off pixels into several clusters (groups) where the number of clusters is most often known in advance. For example, *K-means* [M+67] is one of the most well-known methods and widely applied for this task. It packs the pixels into K clusters by considering the difference between features at pixels and cluster centers. Firstly, cluster centers are randomly (or manually) initialized for all K clusters. Secondly, the differences (distances) in the feature between each pixel and cluster centers are computed. These could be the difference between pixel colors, intensities, or texture values. Thirdly, pixels are assigned to the nearest cluster (least difference). Finally, the cluster centers of K groups are re-computed based on the pixels belonging to them. The steps (from the second to the last step) are repeated until the distance (in the feature) between pixels in a cluster can not be minimized, and the distance between cluster centers cannot be maximized any more. The performance of this method depends on the K value and often on the initialization of clusters.

Mean-shift [FH75] is one of the most powerful clustering methods [Che95, HYZ07, Art08]. Let S is a finite data point in the n-dimensional Euclidean space, X. Let K is a flat kernel that the characteristic function of the λ in X is:

$$K(x) = \begin{cases} 1 & \text{if } \|x\| \leq \lambda \\ 0 & \text{if } \|x\| > \lambda \end{cases} \tag{2.2}$$

The sample mean at $x \in X$ is

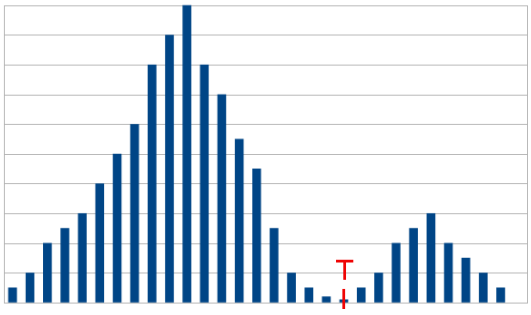
$$\text{m(x)} = \frac{\sum_{s \in S} K(s-x)s}{\sum_{s \in S} K(s-x)} \tag{2.3}$$

The difference $m(x) - x$ is called the *mean shift*. The repeated movement of data points to the sample means is called *mean-shift* algorithm. In each iteration of the algorithm, $s \leftarrow m(s)$ is performed for all $s \in S$ simultaneously [FH75, Che95].

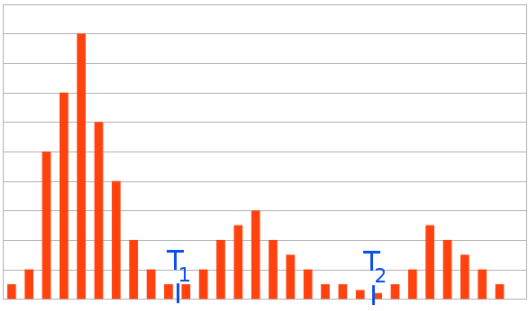
We can mention that mean-shift has been used in computer vision [CM99], e.g., image segmentation [CM97, HYZ07, KBA08], image filtering [CM99], visual tracking [Art08].

Histogram-based method

In the histogram-based method, a histogram is first computed from all pixels in the image by considering the color or intensity values; then, peaks and valleys of the histogram are used to find the groups of pixels. For example in Figure 2.1, the pixels of the image corresponding to the left histogram (Figure 2.1a) can be packed into 2 categories; and we can get 3 categories for the image which provide the right histogram (Figure 2.1b). This method can be adapted to be applied to multiple frames and is easy to implement. However, a disadvantage of this method is that it could be difficult to recognize which peaks and valleys are significant in the histogram. This point will be discussed in Section 3.2.



a . Histogram can be divided by a single threshold



b . Histogram can be divided by two thresholds

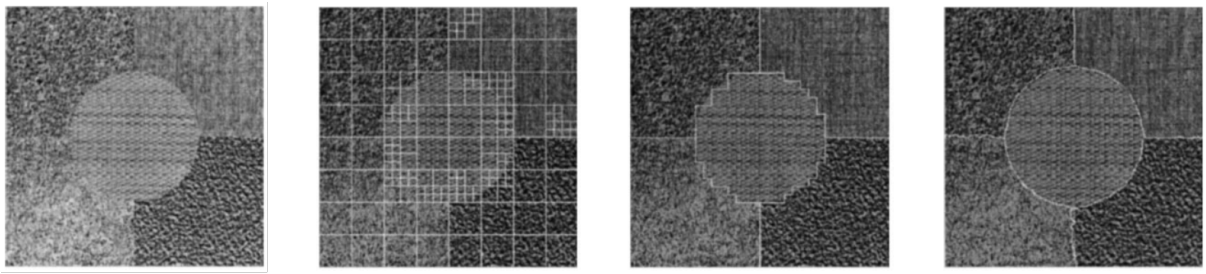
Figure 2.1: Intensity histograms of two different cases

2.1.2 Region-based methods

The region-detection based methods try to group the pixels which have similar properties to get a compact representation. In fact, methods in this category have an overlap with methods based on the pixels (points), especially clustering methods, because they consider the characters of the pixels to pack them into different groups. In this context, *region growing* [ZY96] is one of the simplest approaches, but several refinements have proposed to improve the achieved results in practice. Brice and Fenema [BF70] have developed a region-growing method based on a set of simple rules to enlarge a region as more as it is possible from an initial point chosen randomly. Yakimovsky [Yak73] has improved the region-growing concept by establishing merging constraints based on the estimation of Bayesian probability of the features of each region.

The *split and merge* [Fuk80, CP80, OP99] method is also a popular segmentation technique. It is based on a quadtree data representation whereby an image will be broken (split) into four parts if it has not the same attributes (non-uniform), e.g., color or texture. If four neighboring regions are found to be uniform, they are merged into a large square. In principle, the split and merge process could start at the full image level or at any area in the image. In this context, Fukada [Fuk80] has proposed the segmented variance as a uniformity measure. They try first to find the kernels of areas, then classify pixels into areas using these kernels. Chen and Pavlidis [CP80] suggested more complex statistical measures of uniformity. The segmentation areas are randomly initialized and checked for uniformity. If they do not respect the defined criteria of uniformity, these areas are sub-divided until they are not smaller than a given threshold. Next, the uniformed areas are analyzed to find the similarity for the merging process. Any areas remaining after this step are considered part of a boundary ambiguity zone. The location of the boundary is then estimated by interpolation between the existing uniform regions [CP80]. Ojala [OP99] uses the distribution of the local binary pattern and contrast pattern for measuring the similarity of adjacent areas. Firstly, they split the image into several areas and compute the descriptor for each areas (distribution of

the local binary pattern and contrast). Then, the descriptors of adjacent areas will be compared to make a base for merging them. Finally, the pixel-wise [SZW+18] classification is used to improve the localization of area boundaries. Figure 2.2 illustrates an example based on Ojala's method. The images from left to right present the different steps in their process: the image is first split into several pieces (Figure 2.2b); next, the descriptors are computed for each piece; then they are used to compare and to merge the adjacent pieces (Figure 2.2c) before using the pixel-wise classification to smooth boundaries (Figure 2.2d).



a . Original image | b . Splitted image | c . Merged regions | d . Segmented image

Figure 2.2: The sequence of Ojala's method. Image from semantic scholar [1]

In another approach, the graph cut optimization method can be used for image segmentation [YM12]. Let define an undirected graph $G =< V, E >$, where V is a set of nodes and E is set of the edges which connect every two neighbor nodes. V is composed of two different kinds of nodes: normal nodes that can connect to the others and terminal nodes which consist of s (source) and t (sink). In this graph, we assign a non-negative weight (cost) for each edge, denoted as w_e. A cut of the graph is a subset of edges E which can be denoted as C ($C \in E$). The cost of the cut C is the sum of the edge's weights in C, denoted $|C| == \sum_{e \in C} w_e$. A cut is a minimum when it has the minimum cost when finding the maximum flow [BJ01, BFL06].

In order to apply graph cut to segment the image, we can go back to the core of the image (pixels) which can be divided into two groups: pixels belonging to the object

[1] https://www.semanticscholar.org/

and the background's pixels. Each pixel corresponds to a node in the graph. We create an edge between two nodes if the two pixels are neighbors in the image. An energy function is defined as the sum of edge's weights. The segmentation can be achieved by minimizing the energy-function through the minimum graph cut [YM12].

Along with determining the pixels belonging to regions in the image, it is possible to segment regions by detecting the boundaries between them. This task is usually known as edge detection. The next section presents some methods belonging to this task.

2.1.3 Contours - based methods

Contours methods focus on the edges of the objects which are made up of the list of pixels located on the shape. They usually observe borders of the areas, i.e., where intensity value or gradient direction changes.

Basic edge detectors

One of the first methods uses the gradient vector to calculate the edge strength and edge direction at each pixel of the image. The edge strength is the gradient magnitude of the gradient vector. The edge direction at each pixel is considered as the direction that is perpendicular to the direction of the gradient vector. Equation 2.4 presents the way to compute the edge strength and edge direction at a pixel (x, y) in the image. G_x and G_y are gradients in x and y-direction, respectively.

$$\begin{aligned} G &= \sqrt{G_x^2 + G_y^2} \\ \phi &= atan2(G_y, G_x) \end{aligned} \tag{2.4}$$

Using the gradient of an image requires the computation of derivatives at every pixel in the image. In order to calculate the derivative, a sliding window (named *mask*) is applied to filter the image. If we just consider horizontal and vertical edges, the one-dimensional (1D) mask can be used. But, if taking into account also the diagonal edges, we need to use a 2D mask. The Roberts cross operator [Rob63] is one of the earliest

used 2D mask to detect the edge. He has proposed to use a mask of size 2×2 to compute the gradient of the image. However, this size of the mask is not useful for computing the edge direction on the areas of the image that are symmetric at the center point, for example, an area of size 3×3. The easiest solution is to increase the size of the mask to 3×3 as Prewitt [Pre70] or Sobel [Sob14] have done.

Figure 2.3 illustrates how to calculate the edge strength and direction at a pixel. Each square represents one pixel, the red square is the selected pixel where the edge strength and direction will be computed based on the gradient vector.

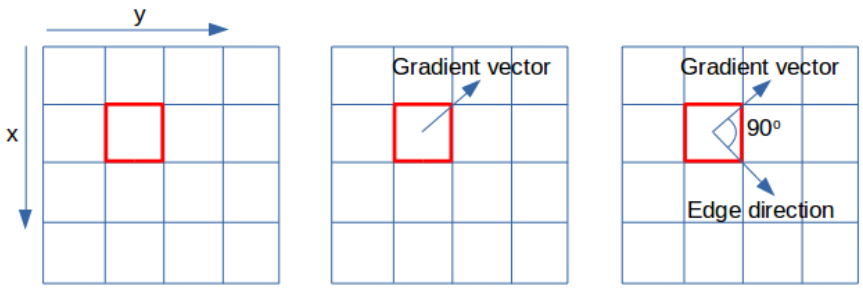


Figure 2.3: The detection of edge strength and direction at a pixel (red). Each square represents one pixel, the edge is perpendicular to the direction of the gradient vector.

Figure 2.4 presents a patch of an image that contains an edge. The pixels, which have the direction perpendicular to the gradient vectors (grey color with the blue border), are indicated as belonging to the edge.

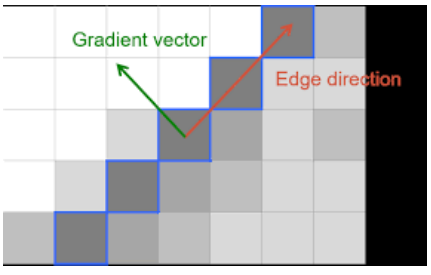


Figure 2.4: Illustration of the pixels (gris color) belonging to an edge.

Advanced edge detector

The previous methods are simply edge detectors based on filtering the image with several masks. These methods are useful when we consider the image without noise. In the case of complex images, the detectors need to be improved. Marr-Hildreth [MH80] has suggested convolving the image with the Laplacian of Gaussian function to reduce the noise and to extract the salient features, e.g., blur and sharp; then, the zero-crossing technique is applied to determine the edges. In another way, Canny [Can87] has proposed to use the gradient vector to determine the pixels belong to the edges. In this method, the noise is reduced by using a Gaussian filter before calculating the gradient vector for every pixel to survey the edge candidates. Then, the weak edges are eliminated by using the edge thinning technique [MZ92]. Finally, the double threshold is applied to determine the potential edges. Even if these methods can detect pixels that delimit the shapes of the objects, they do not give the segments of the shape. To obtain the pixels in the edges and to sort them following an order, S. Suzuki and K. Abe [S+85] have proposed a method to indicate the connected pixels belonging to them. Figure 2.5 shows the results of different methods on a right mandible of the beetle belonging to our dataset.

a . Original image b . Cluster c . Edge detector

Figure 2.5: An examination of segmentation techniques on a right mandible. From left to right: original image, cluster techniques, Canny edge detector, respectively.

2.1.4 Artificial Neural Network-based methods

Besides the segmentation methods that we have described previously, artificial neural networks have been used for segmentation tasks in recent years. A neural network is known to try to simulate the learning strategies of the human brain for decision making [DC00, TLL16]. It is usually made of a large number of connected nodes, and each connection has a particular weight. These nodes are organized into layers to study the image features from the basic level, e.g., pixels, edges, to the abstract ones, e.g., different objects. As usual, the network has been trained on a dataset. Then, it is used to provide the segmentation of the images in another set (testing set).

In the various types of neural networks, CNN [LBH15] is one of the architectures that most often used for the segmentation task. A CNN usually contains numerous convolutional layers and pooling layers associated to non-linear activation function, batch normalization [GWK+18]. These layers can be grouped into two phases, called encoder and decoder phases. In the encoder phase, we reduce the spatial size but increase the depth (channels) of the image by passing it through combinations of the convolution and down-sampling layers. At the end of this phase, it outputs the low-resolution tensors, which contain the high-level features (e.g., object) of the image [LSD15]. In the decoder phase, it takes into account the low-resolution tensors and provides the output as the high-resolution tensors with the label for each pixel in the image. To do that, we use the convolution layers coupled with upsampling layers to increase the size of the image and to decrease the depth of input tensors. From the first application to the present, various CNNs have been proposed, e.g., Fully Convolutional Networks [LSD15], ParseNet [LRB15], Encoder-Decoder [NHH15], UNet [RFB15], Mask-RNN [HGDG17], DeepLab [CPK+17]. The details of neural networks and CNN will be given in the next part of the thesis. We will figure out the components of a network model as well as their operations. Besides, we also introduce the other applications of neural networks, especially CNN.

2.2 Image registration

In image processing, image registration is one of the most necessary tasks to compare two or several images. The registration methods use one image as the source and the others as targets. Features in the pictures are used as inputs to calculate the registration values between the source and target. Based on the data as well as the characteristics of methods, we can divide the registration methods into several categories, e.g., intensity-based [Gos05], feature-based [Gos05], transformation model-based [Gos05, SDP13, Boo97], frequency-domain-based [Zok04, KEB91], similarity measures-based [Gos05, YH09, Bru09]. The readers can find the detail in the surveys [ZF03, SDP13, VMK+16]. In the context of this work, we will present in details only three methods that we have studied from the literature because the two first are referenced in the Palaniswamy's article [PTK10] that we have used to initialize our work, and the last is often cited in morphometry analysis. These are cross-correlation or template matching (similarity measures category), Probabilistic Hough Transform (PHT) (frequency-domain class), and Procrustes methods (transformation model-based).

2.2.1 Cross-correlation method

The cross-correlation method [YH09] or template matching [Bru09] deals to find the matching regions between two images. Usually, we extract a small window from the source image. Then, we slide this window through each pixel of the target image. At each sliding step, we calculate the correlation coefficient between the window and the area in the target image. Finally, the matching region on the target image is indicated as the position of the maximum correlation value with the sliding window.

The formula to calculate the correlation coefficient in the template matching technique between two images (source and target) by using Cross Coefficient is described as in Equation 2.5.

$$R_{corr}(x,y) = \sum_{x',y'}[S(x',y').T(x+x',y+y')] \tag{2.5}$$

Where:

- S, T: are source and target images, respectively.
- (x', y'): is selected coordinates in source image.
- (x, y): presents the coordinates of each pixel in target image.
- $(x + x', y + y')$: is selected coordinates in target image when the source image slides

Nevertheless, the correlation coefficient is affected by the difference which can be observed from the brightness of target and source images. The pixel values can be normalized before calculating to reduce the effect of the brightness by dividing a normalization coefficient (Equation 2.6).

$$Z(x,y) = \sqrt{\sum_{x',y'} S(x',y')^2 . \sum_{x',y'} T(x+x',y+y')^2} \tag{2.6}$$

Consequently, the correlation coefficient can be computed by using Equation 2.7.

$$R_{corr\_norm}(x,y) = \frac{R_{corr}(x,y)}{Z(x,y)} = \frac{\sum_{x',y'}[S(x',y').T(x+x',y+y')]}{\sqrt{\sum_{x',y'} S(x',y')^2 . \sum_{x',y'} T(x+x',y+y')^2}} \tag{2.7}$$

2.2.2 Probabilistic Hough Transform

Hough Transform (HT) [Hou62] has been introduced in the previous century but it remains a common method in image processing and computer vision. At the beginning, HT was used to detect lines in the image, however over time, it has been applied to detect image features, objects, or shape. Practically, HT is a process of summing up evidence for a shape by voting process. Following that, a parameter space, so-called an

accumulator, is created to carry in the votes of the corresponding features of the image. Each vote is explicitly constructed by the algorithm for computing the Hough transform. At the end of this voting process, the local maxima correspond to the instances of the shape. For example, in a line detection application, HT converts the space of the pixel coordinates to the space of the slope and y-intercept of the lines by voting, the vote is stored in the 2D accumulator. Each cell in the accumulator represents the equation of a line (which is presented by the slope and the y-intercept), then pixels vote for the bins that have a corresponding slope and y-intercept.

In order to speed up Hough Transform, PHT [KEB91] has been proposed. Different from Hough Transform, PHT considers a subset of dataset instead of processing on the whole dataset. In practice, PHT is used to predict the presence of an object (of an image) in another image. Firstly, we build the feature descriptors for each object. That is a table containing the geometric relationship (e.g., angle and perpendicular distance) among the object's lines. Secondly, we determine the best matching features between the two objects by applying PHT. Like Hough Transform, PHT uses the voting process to find the best matching features. Finally, the transform information of two objects has been determined by comparing the matching features. Then, these values are used to register the objects.

2.2.3 Procrustes analysis

Procrustes analysis (PA) [Boo97, GD+04] is a method that allows comparing the shapes of two or more objects. As usual, an object is selected and used as the source object to register other ones. PA is performed by optimally translating, rotating and scaling the target images to find the best match with the source. Practically, we consider each object made up from a finite number of points in n dimensions, called key-points. Then, these key-points are used to determine the transformation values to apply to the target.

Consider an example to register two objects, source and target. Each object is made from a set of k key-points: $((xs_1, ys_1), (xs_2, ys_2), \ldots, (xs_k, ys_k))$ and

$((xt_1, yt_1), (xt_2, yt_2), \ldots, (xt_k, yt_k))$ for source and target object, respectively. Now, we find the translation, rotation and scaling values to register target on the source.

In order to obtain translation value, we firstly determine the coordinates of the origin point for each object ($(\overline{xs}, \overline{ys})$, $(\overline{xt}, \overline{yt})$) by using Equation 2.8. Then, we calculate the distance between the two origin points and use it as the translation value. Finally, the target's points are moved to new positions.

$$\overline{x} = \frac{x_1 + x_2 + \ldots + x_k}{k}, \ \overline{y} = \frac{y_1 + y_2 + \ldots + y_k}{k} \tag{2.8}$$

Where:

- $((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k))$ are k key-points made up the object.
- $(\overline{x}, \overline{y})$ are the coordinates of the object's origin.

The scale value is considered as the ratio between the sizes of the two objects. Firstly, the size of each object has been calculated as the root mean square distance (RMSD) from the points to the origin by using Equation 2.9. Then, the ratio is computed and used as the scale to re-form the target object.

$$size\_s = \sqrt{\frac{(x_1 - \overline{x})^2 + (y_1 - \overline{y})^2 + \ldots}{k}} \tag{2.9}$$

The rotation operation is a little bit more complex than the two previous ones. Assuming that the target has been translated and scaled, we now rotate the target object around the origin until we find an optimum angle of rotation, θ, such as the sum of the squared distances between the corresponding landmarks of the two objects is minimized.

2.3 Landmark detection

As mentioned in Chapter 1, a landmark has a position in the image, which is invariant when the scene changes. Landmark detection refers to methods that can automatically

give their coordinates. These methods usually include several steps from extracting the object/region of interest at the segmentation step to figuring out the coordinates of the landmarks. In this section, we present algorithms that are usually used to do that.

2.3.1 Based on the curvature estimation

The methods based on curvature estimation assume that landmarks are on the contours. They usually concentrate at the corners of the shape or at the folding point of the edge. At each point on the curve, the local extreme curvature is computed and used to identify the landmark. In practice, many algorithms have been proposed to calculate the curvature at a point on the curve by using the information of the neighboring points which were indicated as the support region of a point (called *support region*).

Considering a contour C is a set of points:

$$C = \{P_i(x_i, y_i) | i = 1, 2, 3, ..., n\} \tag{2.10}$$

where n is the number of points, P_i is the ith point with coordinates (x_i, y_i). The curve C can be represented by using n Freeman's chain code [Fre61] and is denoted as $\{c_1, c_2, c_3, ..., c_n\}$. If $c_{i-1} = c_i$, P_i is a point in a linear edge, otherwise it is a break point and is a candidate to become a landmark.

Teh [TC89] has first presented a parallel algorithm for detecting key points on a digital closed curve without any input parameters. It was able to work with multiple sizes (lengths) of the digital curve. The procedure first determines the region of support for each point based on its local properties (e.g., Freeman chain code), then computes measures of relative significance curvature of each point, and finally using non-maxima suppression [NVG06] to detect key points. From the method, many other ones based on measurement curvature were suggested to detect the key points. Basically, these methods all rely on finding support regions before applying different techniques to measure characteristics in order to indicate the key points: Ray and Ray [RR92] used the support region and the measure of the significance of each point, which has been computed by using relation with the neighboring pixels, to predict the key points; Cornic [Cor97]

suggested computing left and right support regions, then the key points were detected by a logical function relied on these support regions; Wu [Wu03] preferred to detect the key points with local maximum smoothing bending value; Marji and Siy [MS04] proposed to find the endpoint of each support region before ranking and using them for detecting the key points; Carmona-Poyato et al. [CPFGMCMC05] calculated the adaptive blending value to indicate the location of the key points.

2.3.2 Based on considering the descriptors

The descriptors based methods are mostly focused on the invariant property of the landmarks, e.g., if the scene's point of view is changed by scaling or translating the objects. Accordingly, the methods consider different scales of the input image. In the first step, the descriptors of the image have been computed for each scale. Then, the properties of descriptors are considered to indicate the location of the key points. The methods that should be mentioned in this group are Scale-invariant feature transform (SIFT) which has been proposed by D. Lowe [Low04], and Speeded up robust features (SURF) method of Herbert Bay et al. [BTVG06].

In SIFT, the key points are considered as invariant to the transformation of the image. The method mainly includes 4 steps: (1) scale-space extrema detection which considers all scales and orientation of input image to produce the key-point candidates, (2) keypoint localization to refine the key point candidates by suppressing the points which have the low contrast or are poorly localized along an edge, (3) orientation assignment to calculate the orientation and gradient magnitude of key points by considering their 4-neighborhoods, and (4) keypoint descriptor to build the descriptor for each key point based on the orientation and gradient magnitude values.

In practice, a Difference of Gaussian (DoG) [WD] is first applied to identify the interest points at all scales of the image. The key points are indicated as the maximal and the minimal of the DoG function results. However, this process produces a lot of key point candidates with some of them unstable (they are candidates in some scales

but not in other ones). In the second step, the key point candidates are localized and refined by suppressing the ones which have low contrast or far away from the objects. In the third step, the orientation and gradient magnitude of key points are calculated by considering their 4-neighborhoods. Finally, the descriptor is computed for each key point based on the orientation and gradient magnitude values of a 16×16 region around the key point. The descriptor is presented as a 4×4 histogram, each element includes 8 bins corresponding to 8 ranges of gradient orientations (45^o for each range).

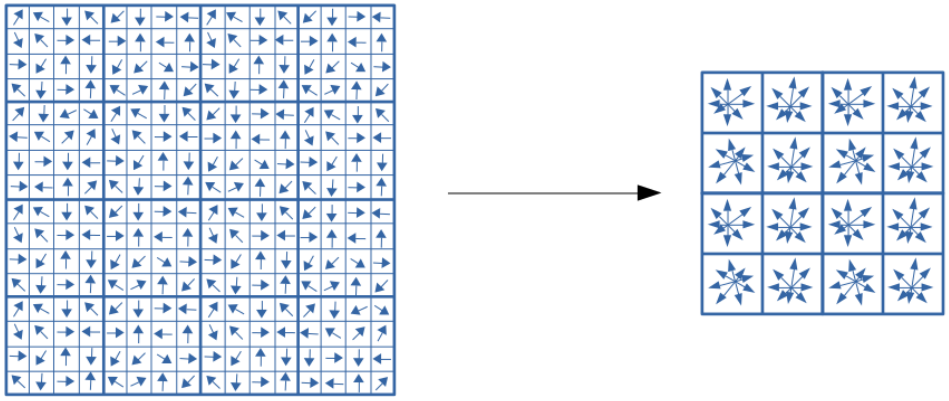


Figure 2.6: SIFT descriptor of a 16×16 patch.
Left: the gradient at each pixel of the patch. Right: Key point descriptor

Figure 2.6 illustrates the process to compute the descriptor for a detected feature. Firstly, a window of 16×16 pixels around the feature is extracted. Secondly, the gradient angle (orientation) and gradient magnitude are computed for each pixel. Then, the pixels in the window are divided into 4×4 grid (left figure). Finally, the descriptor of the region is built by combining the orientation histogram of all regions 4×4. For each region 4×4, an orientation histogram is created by considering the angle and the magnitude of the gradient. The x-axis of the histogram includes 8 bins covering the 360^o ranges of gradient angles (each bin is 45^o). The y-axis of the histogram presents the gradient magnitude. During the histogram construction process, if the gradient angle at a pixel is appropriate with the presented angle of a bin, its gradient magnitude will be added to the corresponding bin of the histogram.

In the same context, Herbert Bay et al. [BTVG06] presented the SURF method. It

has the same principles as SIFT but details at each step are different. This algorithm mainly has three steps: (1) key points detection, (2) local neighborhood description, and (3) matching. Different from SIFT, SURF uses a blob detector based on the Hessian matrix to find the key points. The Hessian matrix is used as a measure of local change around the point and chooses the points where this determinant is maximal. Then, the descriptors are computed around the key points by describing the intensity distribution of keypoint's neighborhoods.

2.3.3 Based on measuring the similarity

Besides using to register two images, template matching could be used to detect the landmarks in the image. To do that, we use an image and its manual landmarks as the reference. Then, we try to estimate these points on another one, the target image. For each source landmark, we extract a small window centered at that point. Then, we slide the window on the target image to find the best matching region with the window by calculating the cross-correlation score (see Section 2.2.1).

As an application of PHT and template matching, Palaniswamy et al. [PTK10] have proposed a method to predict the landmarks on wing images of Drosophila fly [SVP+15]. Their proposition uses a source image and its manual landmarks to estimate landmarks on a target one. The method is a pipeline of four steps: segmentation, registration, estimation, and verification. Firstly, the shape contours of source and target images are extracted and saved as a list of points by using the Canny algorithm [Can87]. Then, they have converted the list of points toward approximated lines [TRY95] for the next step. Secondly, the geometric relations between the lines (angle and perpendicular distances) have been used to encode the features into the invariant form by applying the Pairwise Geometric Histogram [ETM93]. After that, the matching features between the two objects are determined by computing the Bhattacharyya score [Bha43]. Thirdly, Probabilistic Hough Transform [KEB91] is applied to register two objects and to set the hypothesized coordinates of estimated landmarks on the target image. To verify the

position of an estimated landmark, they have extracted the small patch centering at the source's landmark (P), and another region centered at the corresponding estimated landmark on the target image (R). It is worth to note that the size of the patch P is smaller than R. Then, the template matching [Bru09] is used to find the best match position of the patch P in the region R.

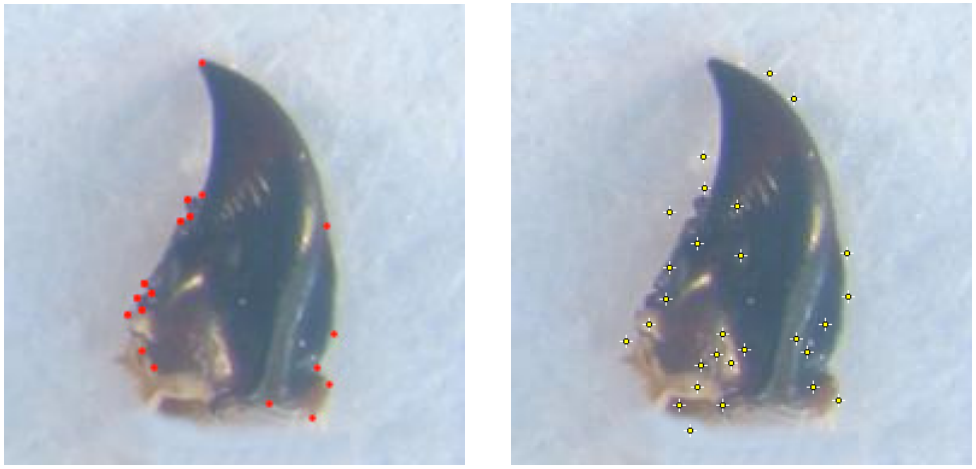
As the preliminary work to test the possibility offers by image processing techniques, we have implemented their method and applied it to mandible images. The implementation of the algorithm is available on our framework [LVBAKP17]. All the details of this process will be described in Appendix A.

2.4 Testing landmark detection methods on our dataset

As described in the previous section, several methods have been proposed in the literature to predict the coordinates of landmarks. These methods can be divided into two groups: the first group includes the methods that can directly provide landmarks by studying features of the image without a-priori information; the programs of the second group input pre-defined landmarks (of a source image), use them as references, and predict their locations in another image (target image). In this section, we examine the performance of two methods belonging to the two groups on our dataset: SIFT and template matching. Accordingly, we have used the two functions corresponding to two methods provided by Fiji library [SACF+12] to do a quick test.

Figure 2.7 shows the best result that we have obtained with the SIFT method. Figure 2.7a illustrates a target image with the waited landmarks. Figure 2.7b shows the estimated landmarks on the target given by the SIFT method of the Fiji library. We can see that SIFT has produced too many key points on the image. In these points, some of them are expected, while others are errors.

Figure 2.8 shows the best example with template matching. Figure 2.8a shows a source image with a patch around the landmark (green region) that we would like to predict in the target image. Figure 2.8b represents the last result of the template

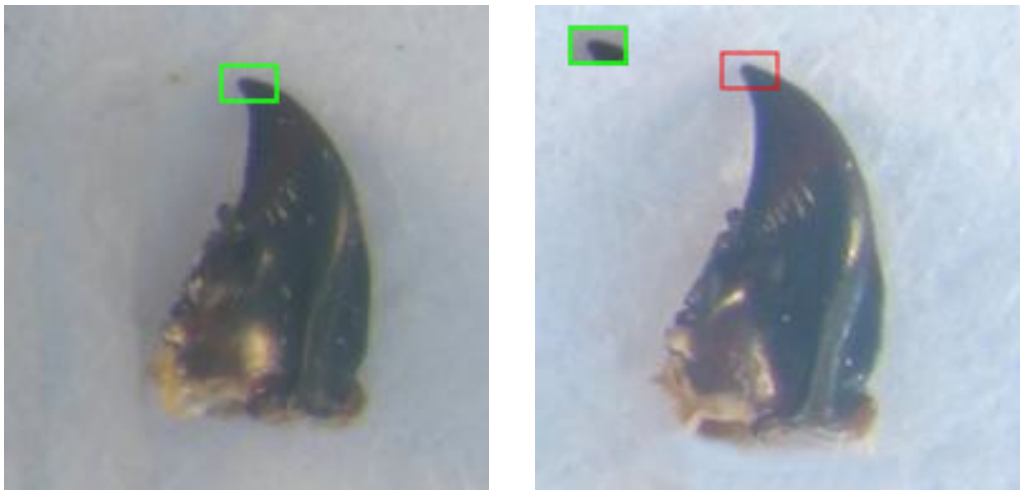


a . Target image and expected landmarks (red points)

b . Landmarks provided by SIFT (yellow points)

Figure 2.7: A result on right mandible with SIFT method [Low04]. From left to right: the mandible image with manual landmarks, the estimated landmarks provided by SIFT, respectively.

matching process. The red region is indicated as the position of the source's patch on the target image.



a . Source image and searching region (green one)

b . Template matching's result

Figure 2.8: The result of the template matching technique. Left: the source image with a patch centering at expected landmark (green area). Right: the small green patch represents the source's patch, the red area is the corresponding region of the source's patch on the target image.

From these two samples, we can tell that SIFT results need to be improved, for example, by adding specific information. The template matching is able to detect correctly the region to push predicted points. We have followed this track to propose a pipeline to predict landmarks on our dataset.

2.5 Conclusion

In this chapter, we have focused on methods for segmentation task which is often the first and important step in the image processing methods. Then, we have also described the techniques to register objects. We have also presented the different methods to determine the landmarks/key points in 2D images. Most of these methods are not only focused on the landmarks located on the contours of the object but also concentrated on the landmarks within the objects. Although each proposed method is applied to a specific problem but in general, it is possible to combine several image processing algorithms to create a new method for landmarking. As an application of these steps, we introduce a proposition for automatic landmarks prediction on beetle images in the next chapter.

Chapter 3

MAELab: a framework to automatize landmark estimation

Contents

The previous chapter has given an overview of several well-known image processing techniques that can be applied for landmark detection. It has also described various studies focused on automatizing key points setting on 2D images. Usually, these methods combine some image processing algorithms to perform the feature extraction before predicting the coordinates of landmarks. The choices of algorithms depend on the characteristics of the image, as well as the type of landmarks.

This work has been initialized by studying the article presented by Palaniswamy et al. [PTK10] which was designed to predict the landmarks on Drosophila wings [HMGC03] during my Master's degree internship. It has been done to test the reproducibility of his method on our dataset. It is worth to note that the article gives steps of the method, but it does not mention how to realize them. To test the suitability of their processes with our application, we have followed and found down solutions to implement the steps in their method. At the beginning of this thesis, we have adjusted some elements in his procedure to improve the results. Details of these realization steps will be presented in Appendix A.

This chapter presents the works that we did during the first year of the Ph.D. We have proposed to modify or to replace some steps of the Palaniswamy method to predict the landmarks in the mandible images (left and right). We have called it, Iterative Method to Estimate Landmarks (**IMEL**). In the same context as Palaniswamy, our method uses a *source image* and *its manual landmarks* to estimate the landmarks in another one, named *target image*. Before going to the details, we outline the steps of our proposition in the next section.

3.1 Overview of IMEL method

In the method of Palaniswamy, an image needs to pass through four steps before providing the coordinates of estimated landmarks, in which two processes require the matrices computation. These calculations could request numerous memory resources and be time-consuming, e.g., to build a Pairwise Geometric Histogram (PGH) and to reg-

ister two objects by using PHT (more details in Appendix A.III .). Their method has provided more than 90% of predicted landmarks which have less far 2 pixels from the manual ones. However, these results have not been at the same levels when we have applied it to mandible images. A quick analysis has been done to point out the differences between the two applications: first, a portion of the mandible is very noisy. It will affect the process to predict the landmarks. More, the landmarks on the fly wing are positioned at the intersection between veins, which are not difficult to identify. Last, mandibles present a variety of sizes that could be more complex than in the fly wings.

Before going to the details of IMEL, Figure 3.1 illustrates the three steps in our pipeline: features extraction, shape registration, and landmarks refinement. Firstly, extracting the mandible shapes (source and target) and saving as the lists of contour points. Then, using lists of contour points to register two mandible shapes by applying an iteration of transformation operations. Finally, the coordinates of estimated landmarks are refined by extracting and comparing the local descriptors around the landmarks (SIFT descriptor).

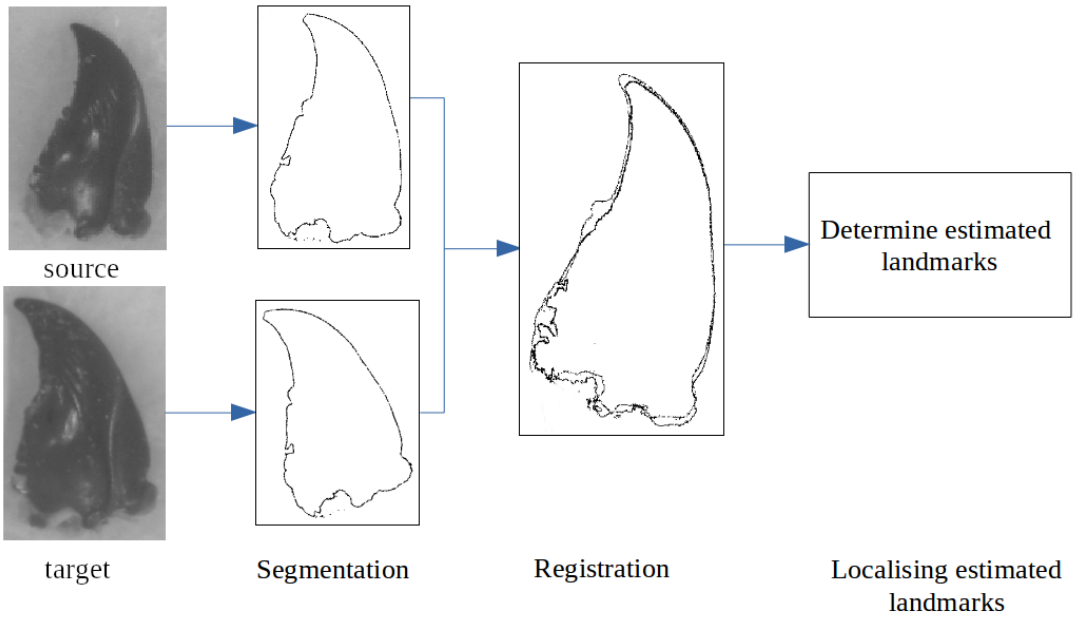


Figure 3.1: Illustrates the steps in our proposition.

It is worth to note that our proposition is different from Palaniswamy's method.

Firstly, it is not necessary to convert the contour points to lines to register two mandible shapes. Instead, we directly use the coordinates of contour points. This makes the computation of our method simple and does not require to build large matrices. Secondly, we have used the local descriptors to refine the locations of predicted landmarks instead of considering the pixel's values (template matching). All the next sections will detail the steps of our proposition.

3.2 Mandible extraction

The beginning of the procedure, extracting mandible shape, is the same as in Palaniswamy's method [PTK10]. To do that, two threshold values are needed to determine the potential edges as it is mentioned in Section 2.1.3. These values could be extracted from the image's histogram [LVBAS+16]. Fortunately, in the case of mandible images, each image only presents a single object onto a good-looking background. It promises that the histogram of the image will exhibit only two regions. Consequently, the histogram exhibits only two peaks and a valley. The threshold values are easy to extract from this histogram. We have firstly converted all the pixels of the color image to the gray-scale mode by using Equation 3.1. Then, the histogram of the gray-scale image has been built and analyzed to determine the threshold value.

$$Gray = 0.299 * R + 0.587 * G + 0.114 * B \tag{3.1}$$

Where:

- $Gray$: is output gray-scale value.
- (R, G, B): is the values at red, green, and blue channels of the considering pixel.

Figure 3.2 shows the gray-scale histogram of a right mandible image. To determine threshold value from the histogram, we have firstly detected the positions of the two peaks and the valley on the histogram: The first peak and the valley have been identified as the highest and the lowest values from the first to the median values in the histogram. The second peak is the highest value after the median. Secondly, two center positions

have been indicated for two regions: The first region begins from the first peak to the valley, whereas the second region starts from the valley to the second peak. Finally, the threshold value is indicated as the average value of the two center positions.

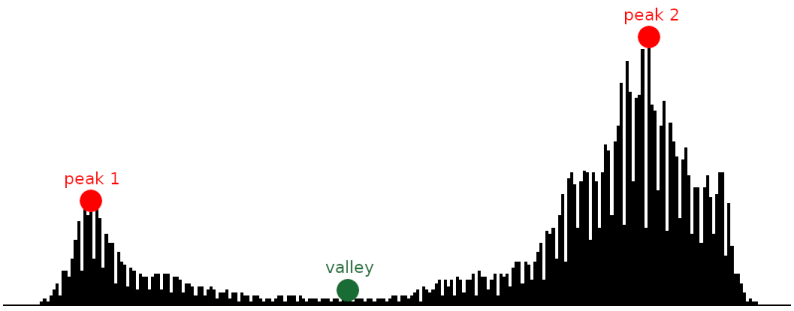


Figure 3.2: The histogram of an gray-scale right mandible image in our dataset. The peaks and valley are illustrated on the figure.

By using these thresholds, the lists of contour points have been extracted as in Palaniswamy method [PTK10] and they are used as the input to register the two mandibles in the next step.

3.3 Shape aligment

As mentioned in Section 1.5, the biologists have provided us images with the same size and same resolution for all mandibles, but mandible objects could have different sizes because of the various sizes of the beetles. Moreover, the position and the orientation of mandibles could vary during the acquisition process. In this section, we describe the procedure that we have used to register two mandible shapes.

As mentioned, IMEL differs a little bit from Palaniswamy's method from this step. We have chosen to register the source and the target shapes from the list of points instead of using the list of lines. The fact is justified that some landmarks are positioned into a noisy area of the mandible, and they are not on line intersections as in the case of fly wings. We can hope that using a list of points could reduce the error range in the shape description. In this case, Procrustes could be a candidate for this step.

But, we have decided to use another way, and we have kept this kind of analysis for other objectives after achieving the landmarks for every mandible. So, IMEL has been designed to registers the source and the target shapes by iterations of transformation operations.

Firstly, we determine the center point and the axes for each mandible shape (both source and target) from the list of contour points. The center point corresponds to the mean coordinates of all contour points. The first axis is the line which connects the center point and a point on the contour, and has the minimum average distance to other contours points, we named it the *Axis with the Min Distance (MDAxis)*. The second axis is perpendicular to the first one. Algorithm 1 describes the process to find the MDAxis.

Algorithm 1: Algorithm to find the MDAxis from a list of contour points

Input : Centroid c, list of contour points l
Output: MDAxis a

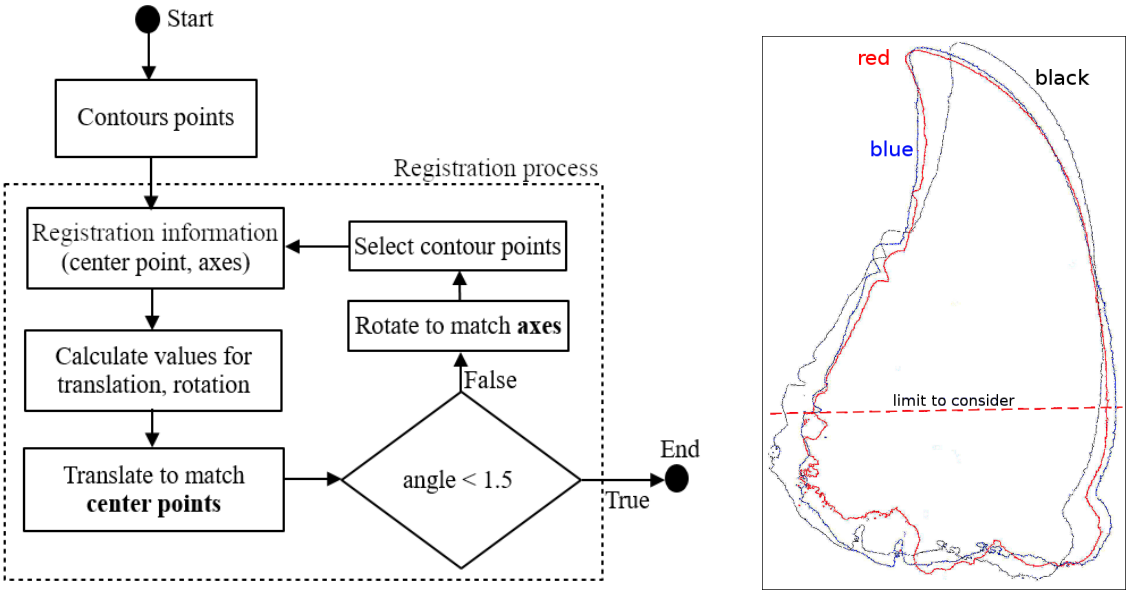
```
for all points p_i in l do
    for all points p_j in l do
        if p_i ≠ p_j then
            Compute the orthogonal distance d_ij between the line (c, p_i) and p_j.
        end
    end
    Compute d_mean as the average distance of all d_ij distances.
    if d_mean is minimal then
        p_min = p_i;
    end
end
The axis is: a = (c, p_min).
```

Secondly, the parameter values of the transformation operations are defined. The translation value is the distance between two center points; the rotation value is the different angle between the axes of the two images. Then, the center point of the target shape will be translated to match with the center point of the source shape.

The matching between the two shapes is evaluated by re-computing and comparing the registration (center point and principal axes) information of two objects. It is worth to note that the registration information can be imprecise if the list of points is not cor-

rectly positioned on the contour. It is mainly the case of the points belonging to the base part of the mandible. To reduce the effect of this problem, we have sorted the contour points according to their y values. Then, we have only taken into account a subset of contour points (the upper part) for computing. After several tests, we have fixed the upper part containing the points that have y-coordinate larger than one-quarter of shape height. This part is used to compute the center point and axes in the iterations.

One can note that the coordinates of the points on the contour will be updated after each registration step. It makes some points that have y-coordinate smaller than the limit value could be moved to the upper part and vice-versa. Consequently, we will most often have a new subset of points to compute the registration information. These steps will be repeated until we satisfy the conditions to have an angle of rotation is less than 1.5 degrees. In all experiments, the range of the number of useful iterations is between 3 and 5 repetitions to reach the best registration.



a . Process allows to refine the rotation angle between the source and the target

b . Obtained results after several registration steps

Figure 3.3: The registration process and its result on a right mandible.

Figure 3.3a illustrates the steps in the process to register the two mandible shapes. Figure 3.3b shows a sample of the process to register two mandibles: the red contours

are the source's contours, the black outlines are the target's contours after one iteration of the process, the blue ones are the final results of registration process.

At the end of this step, the source's manual landmarks are set on the target image. They are considered as the theoretical positions of predicted landmarks on the target mandible. By experiments, we have observed that some predicted landmarks could stay a little bit far from the ground truth. So, we have introduced a last step in the procedure to refine the results.

3.4 Refinement of estimated landmarks

As we mentioned, some landmarks are close to manual ones, but some others are far away, and even sometimes they are not on the contour. To improve the accuracy of prediction, we have added a new step in our pipeline to refine the position of estimated landmarks by using the SIFT descriptor [Low04]. Commonly, the SIFT descriptor is computed on the whole image. However, we have modified some aspects of the original version to define a specific process for our identification. We have chosen to re-use the information coming from the previous step to define a small area, e.g., a patch, around the source and predicted landmarks, and to extract the SIFT descriptors from these patches.

As usual, the computed SIFT descriptor consists of computing the orientation and the gradient magnitude of each pixel belonging to the patch. The SIFT method takes into account eight bins to cover all directions of the gradient orientation: 45 degrees for each direction class. Additionally, the feature vectors are normalized to reduce the influence of illumination. In the original SIFT method, the patches of 16×16 pixels are taken into account. However, we have chosen the size 9×9 for the patch in the source image after testing several different sizes of patches such as 16×16, 18×18, 36×36 or 54×54. In the target image side, we select the region with a size of 36×36 (defined by experiments) to search the best match with the source's patch. Figure 3.4 shows the orientation and the gradient magnitude of each pixel in a patch of 9×9 centered at a

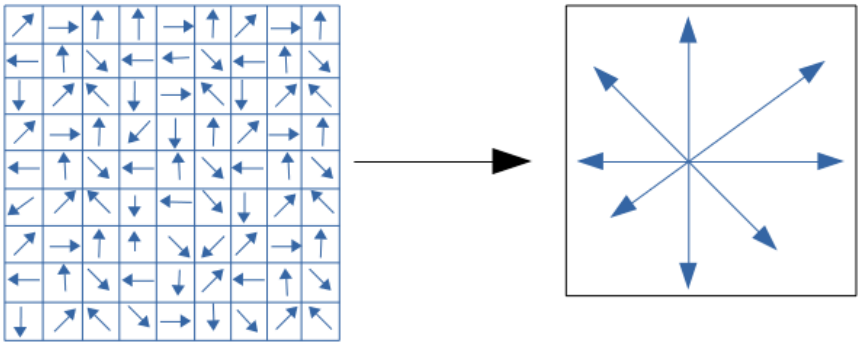


Figure 3.4: The SIFT descriptor of a patch. In the left figure, the orientation and gradient magnitude of each pixel in the patch. In the right figure, the arrow length corresponds to the sum of gradient values in each direction.

landmark (left) and its SIFT descriptor (right).

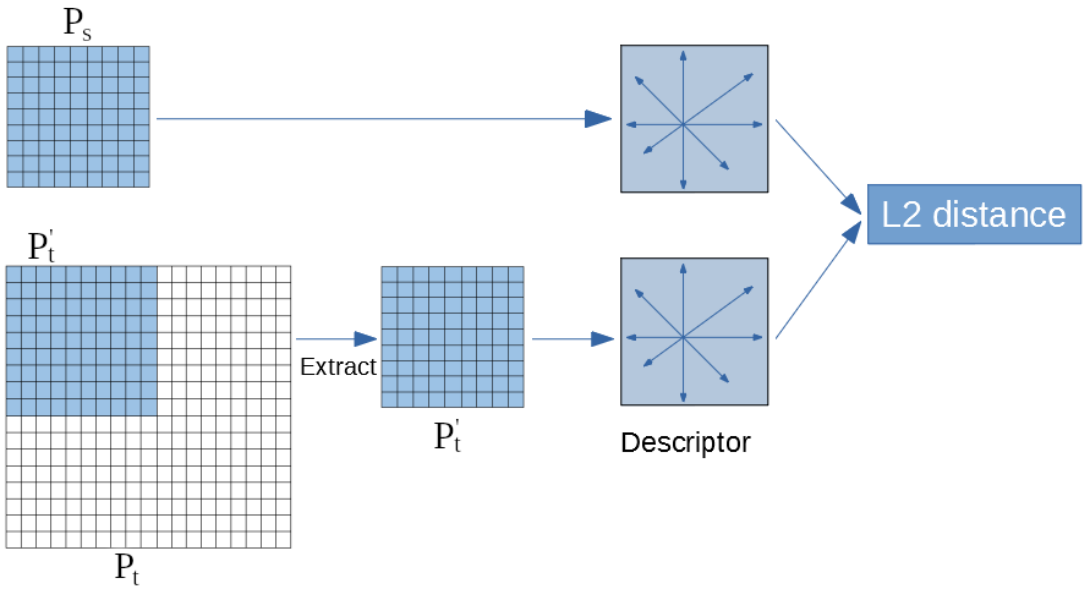


Figure 3.5: Illustration of the process to apply SIFT method in our approach.

Concretely, a patch P_s centered at the source's manual landmark (9×9) and a patch P_t centered at the estimated landmark (36×36) on the target image are initialized. Then, the SIFT descriptor is computed for P_s. For each pixel in P_t, a sub-patch P'_t is extracted with the same size as P_s and the SIFT descriptor is built. In order to compute the agreement between two patches, their SIFT descriptors are compared by computing

the $L2$-distance (Equation 3.2). This process is repeated until all pixels in P_t are considered. The new predicted coordinates of the landmark will be set at the center of the sub-patch P'_t which has the smallest distance L to P_s (Figure 3.5).

$$L(D1, D2) = \sum_{i=0}^{n} \sqrt{(D1_i - D2_i)^2} \tag{3.2}$$

Where:

- n is the number of directions
- $D1$ and $D2$ are two descriptors of size n
- $D1_i$, $D2_i$ are the i^{th} value in descriptor $D1$, $D2$

3.5 Results

All the steps of the IMEL have been also implemented in MAELab framework[1] and have been verified on two sets of images: left and right mandibles. After verifying the images, it remained $286/290$ workable images of the left/right mandible. A set of $16/18$ landmarks for each left/right mandible is considered as ground truth. In order to provide the estimated landmarks, we have randomly chosen an image in each set (left and right mandible) and used it as the source to predict the landmarks on others images considered as the targets. For example, the *Md28* and *Mg52* images have been randomly selected as the source images for right and left mandibles in our experiment, respectively.

After checking these results, we have noticed that it remains some estimations that are a little bit far from the ground truth, and we have wanted to go deeply to the analysis. As we have discussed in Section 3.3, mandible shapes could show different sizes. We have found that our method is sensitive to this parameter. To improve the results, a pre-processing process has been inserted to estimate the scale between the

[1]The functionalities of MAELab are described in Appendix B.

source and the target image before computing the SIFT descriptors. According to that, the bounding boxes around the source and the target mandibles have been computed. Then, the scales on x and y dimensions have been determined by computing the ratio between the corresponding sides of the bounding boxes, and the target contours have been scaled to fit the source contours.

We have evaluated our results at two scales: first to verify when we use the predicted landmarks to compute the centroid size, do we obtain similar results than the obtained results from ground truth landmarks, and do we satisfy the requirements of biologists analysis? Secondly, are the predicted landmarks good enough to be displayed in the place of manual ones in an user interface?

The first evaluation is to compare the results provided by IMEL to the results obtained from the use of the Palaniswamy method (Appendix A). The biologists have suggested computing the difference between the two centroid sizes (manual and estimation). The error has been calculated following Equation 3.3, which has been done with Palaniswamy results.

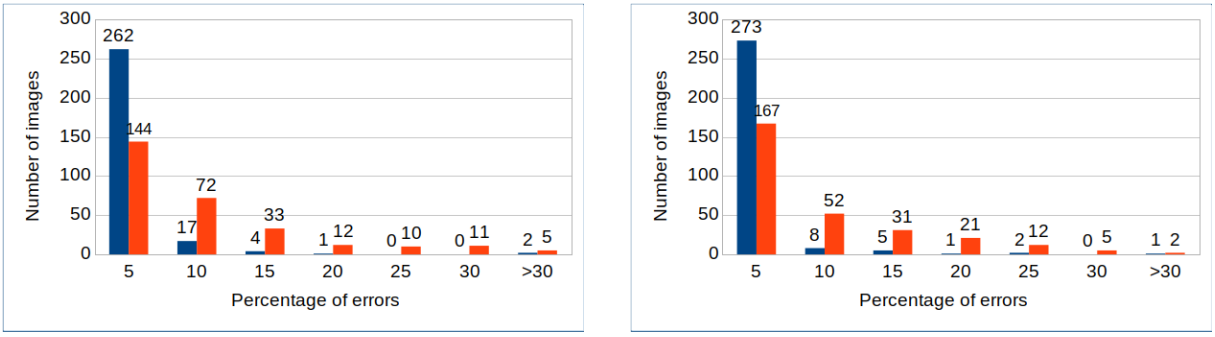
$$Percent\_Of\_Error = \frac{100 * |(Original\_Size - Estimated\_Size)|}{Original\_Size} \tag{3.3}$$

Where:

- Original_Size: is the centroid size calculated by using manual landmarks (ground truth).
- Estimated_Size: is the centroid size computed by using predicted landmarks.

Figure 3.6 shows this comparison. The orange columns remind the results from Palaniswamy method, while the blue columns represent the IMEL ones. To remind the previous results, their method applied to the beetle's mandible datasets provided 150 right and 140 left mandibles with an error of less than 5% of the difference between the centroid size of the manual and estimated mandible. Our proposition has provided more precisely the coordinates of predicted landmarks than the Palaniswamy one. We

have obtained 273 right and 262 left mandibles with the same error threshold (less than 5% of errors).



a . Left mandible

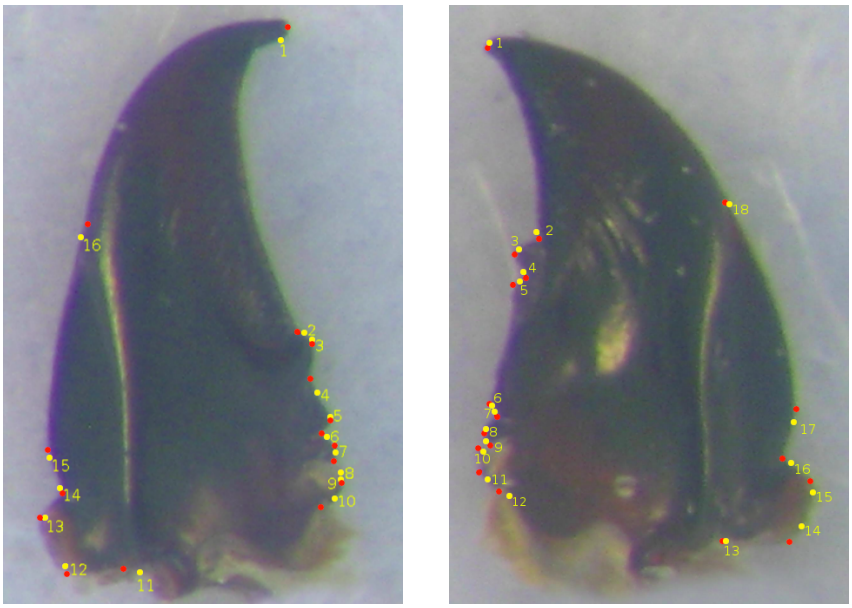
b . Right mandible

Figure 3.6: The frequencies of images at each percentage of error between the manual and estimated centroid sizes.

Figure 3.7 shows the location of manual and estimated landmarks on an example of a left and a right mandible. First of all, the estimated landmarks are quite near manual ones. However, as it has been shown in Figure 3.6, we can see that it exists a small difference in the prediction on two sets of images: the predicted coordinates in the right mandibles are closest to the manual landmarks than the left ones. In section 3.3, we have also discussed that it could exist of variations in the left mandible size than the right. This hypothesis has been arisen by biologists, but it has not been clear when we have tested the dataset.

In the second attempt, we are also interested in the accuracy of each estimated landmark. We have calculated the distances between the manual landmarks and the corresponding predicted ones. Then, we provide the percentage of proportions based on the distance between manual landmarks and estimated ones. These proportions have been calculated based on several ranges of pixels, as mentioned in Palaniswamy's article.

As mentioned in Appendix A, we have compared the distances (between the manual and estimated ones) on mandibles and fly wing (in the article) provided by Palaniswamy's method. To deal with the difference of image sizes, we have scaled



a . Left mandible b . Right mandible

Figure 3.7: The manual (in red) and estimated landmarks (in yellow) on a left and right mandible.

the distances on mandibles. So, to be able to compare with previous results, we have normalized the distances that we have obtained by the IMEL method. To remind the values that we gave in Chapter 1, the initial resolution for mandibles is 1500 pixels/mm. As we have normalized the distances in the following tables, the considered resolution is 600 pixels/mm.

Tables 3.1 and 3.2 show the proportion of accuracy at several ranges of errors (in pixels) that we have considered on the left and right mandibles. First of all, the obtained scores with our proposition are better than Palaniswamy's ones. For example, for both left and right mandibles, we obtained more or less 50% of the dataset which exhibits an error of distance less than 20 pixels (Tables 3.1b and 3.2b), which is far from the performance of the Palaniswamy method (Tables 3.1a and 3.2a).

As the last evaluation of results, we have considered the average distance obtained with IMEL at each landmark position. Figure 3.8 shows the average values at each landmark position on each set of mandible images. In this figure, the blue/green curve represents the values of left/right, respectively. For the right mandible, the lowest/highest

Pixel range	No. of points	Proportion (%)
≤ 5	36	0.784
≤ 10	106	2.308
≤ 15	156	3.397
≤ 20	252	5.488
≤ 30	426	9.277
≤ 50	789	17.182

a. With Palaniswamy's method

Pixel range	No. of points	Proportion (%)
≤ 5	411	8.982
≤ 10	948	20.717
≤ 15	1411	30.835
≤ 20	1860	40.647
≤ 30	2779	60.73
≤ 50	3734	81.6

b. With IMEL

Table 3.1: The proportion of several ranges (in pixel) on the left mandibles set. Comparison between the two methods: Palaniswamy method [PTK10] and IMEL.

Pixel range	No. of points	Proportion (%)
≤ 5	71	1.36
≤ 10	170	3.257
≤ 15	306	5.862
≤ 20	424	8.123
≤ 30	717	13.736
≤ 50	1236	23.678

a. With Palaniswamy's method

Pixel range	No. of points	Proportion (%)
≤ 5	598	11.616
≤ 10	1482	28.788
≤ 15	2280	44.289
≤ 20	2950	57.304
≤ 30	3833	74.456
≤ 50	4533	88.054

b. With IMEL

Table 3.2: The proportion of several ranges (in pixel) on the right mandibles set. Comparison between the two methods: Palaniswamy method [PTK10] and IMEL.

value is $13.55/46.30$ pixels corresponds to $1.3\%/4.8\%$ of the image's size. In the case of the left mandible, these values are $12.75/52.75$ pixels agree to $1.3\%/5.5\%$ of the image's size, respectively. As we have mentioned before (Figure 3.3b), the segmented contours at the base are noisier than on the tip of the mandible. It explains why the accuracies at the 11^{th}, 12^{th} landmarks on the left mandible and the 13^{th}, 14^{th} landmarks on the right mandible are less than the others. Again, the left mandibles have more variety of sizes than the right ones. It illustrates the accuracy of the right part is always better than the left in all experiments.

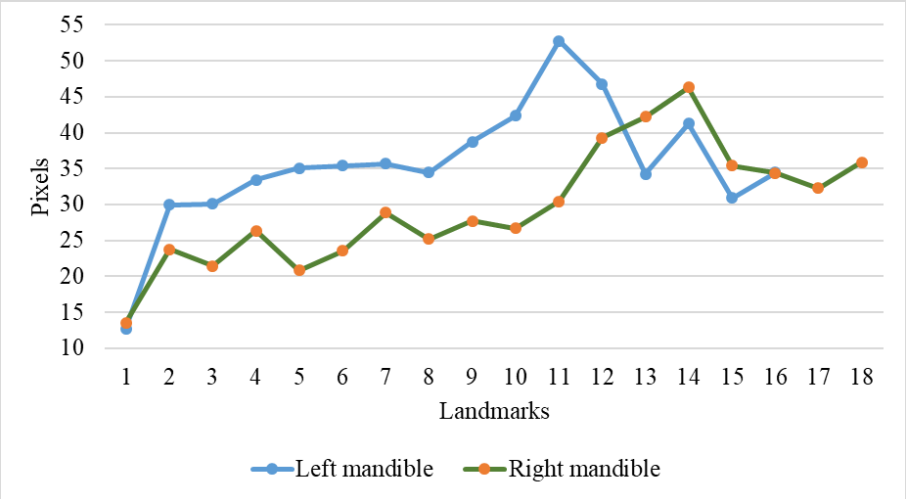


Figure 3.8: The average distance at each landmark of two sets of images. The blue/green represents the mean distances on left/right mandible images.

3.6 Discussion

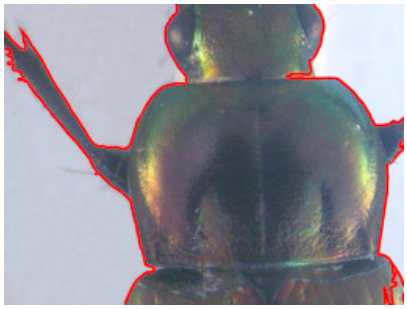
In the same context to identify the landmarks on biological images, D. Houle et al. [HMGC03] have described a method based on analyzing the curves belonging to the wing shape of the fly. The method has been evaluated on 535 Drosophila wing images (12 landmarks on each wing) and the obtained mean proportion of well-estimated landmarks is 82%. Ke. Yan et al. [KS+04] have proposed to combine SIFT descriptor and Principle Component Analysis (PCA) for characterizing and identifying the matching points between the images in Graffiti dataset. The method firstly finds the landmarks in each image. Then, they are used to determine the matches between the pictures. Their process has obtained an accuracy close to 95%. One can note that MAELab[2] with all the program dedicated to the automatic identification of landmarks on 2D images of mandibles are available from now.

The results of this section have been presented as:

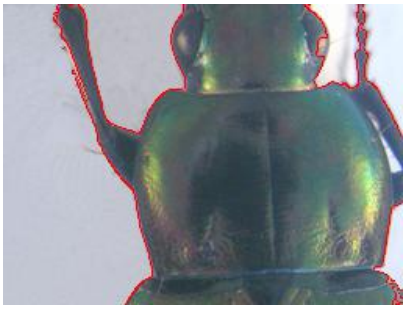
- a paper [LVBAKP17] presented at the international Conference on Computer Graphics, Visualization and Computer Vision 2017 (WSCG-2017).

[2]It is freely available on Github at the address: https://github.com/linhlevandlu/MAELab2019

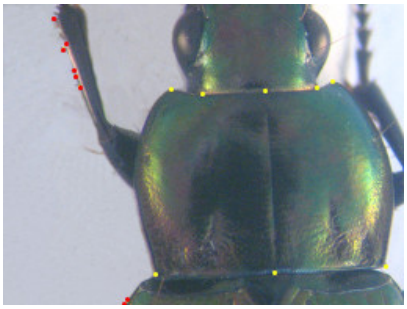
- a poster presented at GRETSI, 2017.



a . Segmentation of source image



b . Segmentation of target image



c . Landmarks on target image. Red is prediction, yellow is manual.

Figure 3.9: The results of two steps (segmentation and landmarks detection) on a pronotum image.

To see how our proposition works on other parts of beetle, we have first applied the steps on pronotum images. Figure 3.9 shows the results that we have obtained on a pronotum. As we have mentioned before, the pronotum image is different from the mandible because it contains not only the studied object but also other anatomical parts, e.g., legs and parts of the head or elytra. The results have shown that the pronotum image is quite segmented (Figure 3.9b), but the predicted landmarks are totally incorrect (Figure 3.9c), and really far from the manual ones. Clearly, the appearance of noisy elements has caused trouble for the treatments.

During the last ten years, new methods have been proposed to solve different problems in computer vision without using image processing techniques, e.g., SVM [CYSC03, LV12], random forest [Bre01, GEW06], neural network [LKF10, LBH15]. These methods have appeared in different computer vision applications such as image registration [ZGS17, YKSN17, WQL+18], object detection [GDDM15, GG15, ZCG+19], face recognition [LZS16, WGT+18], or automatically identify landmarks on the images [SWT13, Z+14, CQSA+16, WGT+18]. Those methods have also opened new directions for landmarks detection by using machine learning methods. In our case, we have chosen to turn to Deep Learning methods without the necessity of the segmentation step in

order to check if this way can provide an efficient solution to estimate our landmarks.

3.7 Conclusion

In this work, we have presented a pipeline, IMEL, for the automatic estimation of landmarks on mandible images. The proposed method includes a step of segmentation to extract the shapes of the mandibles. Then, an iteration of transformation operations is applied to register two forms before predicted the hypothetical landmarks on the target object. Finally, the SIFT descriptor is used to improve the estimated coordinates. The method has been implemented in MAELab framework and has been evaluated on two sets of mandible images. For this dataset, we have shown that we can provide a set of landmarks that can be used to compute the centroid size or to apply other kinds of analysis on the mandible instead of the manual ones, for example, in Procruste analysis. Moreover, a set of refinements as computing the SIFT descriptor on a selected patch around the first estimation has reduced the distance between the estimated coordinates and the ground truth. It makes our proposal satisfactory for biologists as a possibility to replace manual settings.

From now, the next step consists in switching to analyze other beetles parts and to remove the manual interventions. However, these images are more complex than mandible ones. It is not so easy to extract the studied objects for the registration step. Unfortunately, this is an essential step in our proportion to align two objects and to set the landmarks. The experiments on pronotum have shown that these steps are the bottleneck of this kind of method. The segmentation that has implemented in IMEL can be considered as not really expert, and efforts could be made to improve this step with newer methods. But, we have preferred to investigate deep learning methods, especially Convolutional Neural Network, which has risen in image processing recently, to try to pass this difficulty.

Part II

Deep Learning for automatic identification landmarks

Chapter 4

Deep learning and landmark detection

Contents

In the first year of this work, the image processing techniques have been studied and applied to determine the landmarks on mandible images. This chapter turns to another approach of landmark detection, using machine learning algorithms, specifically deep learning, to study the other anatomical parts of the beetles.

This chapter begins with an overview of some machine learning and deep learning algorithms. Then, we present Convolutional Neural Network (CNN), a specific variant of deep learning for computer vision task, and its components. Next, we will turn to transfer learning, a complementary approach in deep learning. Finally, the chapter ends with some applications which have used CNN to analyze 2D images.

4.1 Machine learning and neural network

4.1.1 Machine learning

Machine learning [Sam88, MST+94] refers to teaching computer the abilities which are mostly done by humans. It addresses the question: "How to make the machine learns better in the future based on current or past experiences?" [Sam88]. The answer to this question could be to create a process that can learn directly through experiences or observe the behaviors of an algorithm on a dataset. An algorithm which is built for tasks of a machine learning system and able to learn from data is called a *machine learning algorithm* [MST+94]. Nowadays, machine learning algorithms are widely used in various applications. Depending on the approach, type of input/output data, and kind of tasks to achieve, the machine learning algorithms can be categorized into three categories: *supervised or semi-supervised learning*, *unsupervised learning*, and *reinforcement learning*.

In *supervised learning* [RN16], the algorithms learn from examples. In the training process, each pair of input data and its ground truth label is given to the model. The algorithm analyzes the input data and tries to optimize its parameter values. Then, the model is used to find the label of new data. As supervised learning, *semi-supervised learning* develops the model for the incomplete training data, which includes a portion

of data that does not label.

In *unsupervised learning* [Sam88, Fri98] the algorithms track operations to describe the structure of unlabelled data. *For example*, clustering analysis is a branch of this group that proposes to classify the unlabeled data. The algorithm tries to identify the common features of data belonging to a group. When a new piece of data appears, it will be assigned to the group which exhibits the same common features.

Reinforcement learning [KLM96, SB+98] concerns how to map situations to actions so as to maximize a numerical reward signal. In an essential way, reinforcement learning can be seen as a *close-loop* because the learning system's actions influence its later inputs. At each round, the learner discovers the actions which can bring the best reward by trying them out. Reinforcement learning is different from supervised or unsupervised learning: it tries to maximize the reward signal instead of indicating the correct/cluster category of the situation.

From the first appearance till now, many machine learning models have been introduced, e.g., Decision Tree [Qui87], Clustering [Llo82], Support Vector Machine [CV95], Neural Network [MP43]. In these models, Neural Network is different from these models. It is a combination of hundreds of unit process to solve a problem. Each unit learn to perform the tasks by considering examples without programming of any task-specific rules. Nowadays, Neural Network becomes very popular for regression and classification problems, but over time they have been transformed to adapt with all manner of problem types such as object recognition or speech recognition.

4.1.2 Neural Network

Neural Network [LBH15] is a computing system based on a collection of connected units, called *neurons*, which are inspired by the biological neural network. In the biological model, each neuron hires an "activation function" to decide which action will apply to the input. An *axon*, is a connection between two neurons, to transmit the signals. Each axon is partnered with a weight to adjust the "importance" of the inputs. It

is used to increase or to decrease the strength of the sending signal to the next unit. Figure 4.1 illustrates the exchange signals between two human neurons. The signals will be processed, converted and passed through the axon to the second ones. At the second neuron, the signals can be accepted or rejected depending on their strength. The signals are then processed at the second neuron and sent to the next neuron.

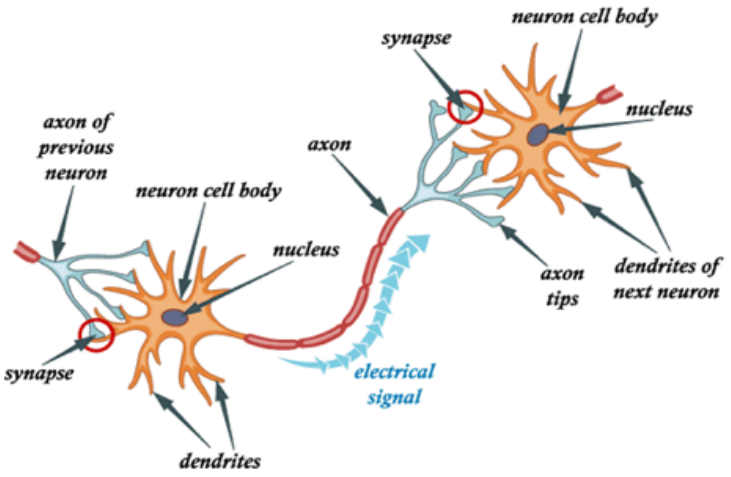
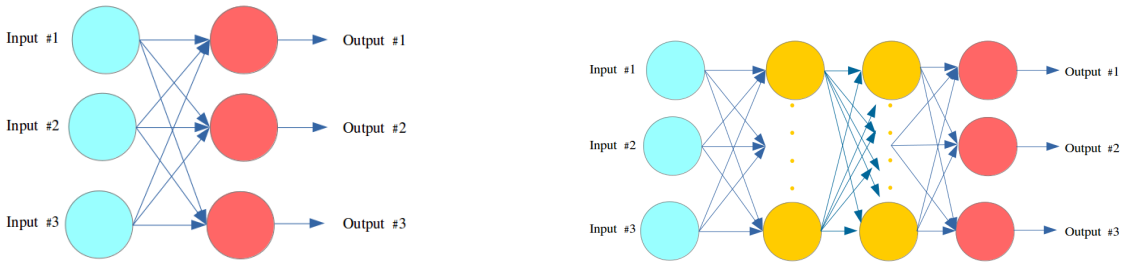


Figure 4.1: A drawing of a biological neuron[1].

In practice, the neurons are grouped to create the layers and a neural network is built from a set of layers. The layers in the neural network can be classified into three types: input layer, hidden layer, and output layer. The input layer receives input data, then passes to the hidden layers before returning to the output. In a neural network, the number of input data for each training time is variable and depends on the capacity of resources. However, even different kinds of data in various applications, e.g., image or text, these data have to be converted to vectors before providing to the network. In the same situation, the number of hidden layers and the number unit in each layer is also the model's hyper-parameters. These layers can be inserted into the model to improve the power of the network. In practice, these values are specified through the experiments. However, it depends on the computing capacity, as well as the complexity of the problem that we need to solve. For example, we can use a model of two layers

[1]Image source: https://towardsdatascience.com

with several dozens of units for a classification task. The number of units at the output layer is usually corresponding to the goal of the problem, e.g., it will have two units for a binary classification task. Figure 4.2 illustrates two different architectures with and without hidden layers (Figure 4.2a/ Figure 4.2b). A network that contains a number of hidden layers is named Deep Neural Network, and this is the first idea of Deep Learning.



a . Neural network without hidden layers

b . Neural network with hidden layers

Figure 4.2: The examples of neural network without and with hidden layers

A neural network model is not a machine learning algorithm, but a framework for many different machine learning algorithms to work together on complex input data. The original objective of neural networks is to solve the problems in the same way as the human brain. It has been used on various tasks over time, including computer vision, speech recognition, machine translation. As described in chapter 3, we have designed a specific process based on the image processing techniques for predicting the landmarks. We have tested the suitability of our proposition on other parts of beetles (e.g., pronotum and head), but obtained results were not satisfying. To examine the agreement of other answers with our problem, we have chosen deep learning to analyze the images of other parts of beetle. The context of deep learning will be detailed in the following section.

4.2 Deep Learning

Deep learning algorithms have been introduced as a modern update on Neural Network in the previous century with the composition of multiple layers with non-linear functions to learn the representations of data at various levels [LBH15, GBC16]. In the organization of a model, each level of representation is corresponding to the different levels of abstraction. The lower layers (closed to the input) take into account the local features, and the higher layers (closed to the output) enlarge the aspects of the input which are important to discriminate and to suppress irrelevant variations. At the first ages, deep learning encounters several problems to take into account real-world cases because of the limitation of the memory size or computing power before the $2000s$. Recently, programming on the Graphics Processing Unit (GPU) has fastly grown up because of the improvement of computing capacities, both in memory size and computing time. It has offered deep learning a new perspective to deal with the problems.

In a deep learning model, the computation can be divided into two phases: *forward phase*, where the layers take the input from the previous layer, compute new representation and send to the next layer; and *backward phase*, where backpropagation algorithms are applied to compute the updated values for the parameters at each layer.

From its first appearance to nowadays, many variant architectures of deep learning have been proposed, and they have found great success in different domains. For example, Deep Neural Networks (DNNs) to solve the classification or text analysis problems [H$^+$12, M$^+$11]; CNNs to deal with the problems in computer vision such as image classification [SLJ$^+$15, KSH12], document recognition [LBBH98, HHW19], object detection [FCNL13, LLS$^+$15]; Recurrent Neural Networks (RNNs) to analyze data under time sequence, such as language and text processing [JCMB14, SVL14, CWB$^+$11]. In the next section, we focus on CNN, a specific variant of deep learning models for grid topology data.

4.2.1 Convolutional Neural Network

In recent years, deep learning has risen strongly as a method to solve many difficult tasks in different domains. In serval model variants of deep learning, Convolutional Neural Networks (CNNs) are known as good solutions to solve the problems on grid topology data such as 2D/3D images, or video sequences. A CNN is a feedforward network that takes the information following one direction from the input layer to the output layer. Different CNNs architectures have been proposed. But generally, they consist of several types of layers: **convolutional** and **pooling** layers which are stacked together to convolve and to down-sample the inputs. Then, they are followed by one or more **fully connected layers** to achieve the output of the network. Sometimes, the dropout layers are added, for example, between the FC layers to prevent the overfitting of the network.

Figure 4.3 shows a classic example of a CNN, the network inputs directly an image to several stages of **convolutional and pooling** layers. Then, the representation is passed to three **fully-connected** layers. A **dropout** layer is inserted after the second fully-connected layer (it is represented by some blue nodes). Finally, the last fully-connected layer provides the label of the input image. This architecture could be seen as the most popular one. Now, we will describe the different types of layers that could be used to build a CNN architecture.

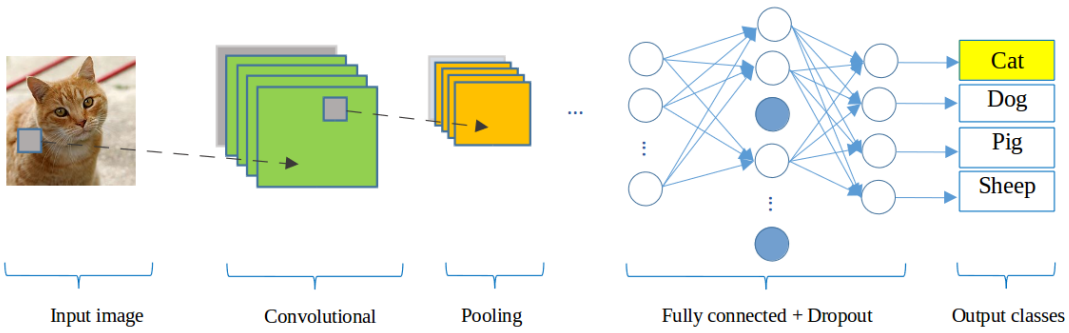


Figure 4.3: A CNN network for classification problem

Convolutional (CONV) layer is used as a feature extractor by applying some learn-

able weights (filters) on the input image. The input image is convolved with the filters in order to compute the new feature maps [DV16]; then, the convolved results are passed through a nonlinear activation function before sending it to the next layer. In the CONV layer, the neurons are arranged into groups, and each group's size is equivalent to the size of the filters. All neurons in a group have the same weights, however different groups in the same CONV layer will have different weights. So, several features can be extracted at each location of an input image.

In an application, the number of convolutional layers, as well as parameters of each are variables. The answer depends on a few factors, for example, the capacities of resources or the complexity of the problem. In practice, these values are usually indicated by the experiments. As usual, depths of the layers are set to increase from the lower to the higher story. The filter matrices should be designed with a small size, for example, 2×2 or 3×3 as mentioned in [SLJ+15].

Pooling (POOL) layer is mostly used to down-sampling the size of inputs with the purpose to reduce the spatial resolution of the feature map and so to reduce the computation cost. Initially, it is a common practice to use average pooling to propagate the average of all the inputs to the next layer. However, in more recent models [KSH12, CMS12, LLS+15], the maximum pooling function has been preferred. It propagates the maximum values of the inputs to the next one.

In practice, choosing pooling layers in the model depends on features that we would like to extract, for example, max-pooling is preferred to extract features like edges, whereas average-pooling is favored to extract the global objects. Additionally, the size of the filter (F) and the stride value (S) are also important. These parameter values affect its output features which are the summary of characteristics of the previous layer. There are two common variations of pooling in the practice: overlapping pooling which uses three as the filter's size (F=3) and two as the stride value (S=2); and non-overlapping pooling where the filter size equals the stride value (F = S = 2).

Figure 4.4 illustrates the differences between the maximum and average pooling: Giving an input image of size 4×4 and considering a filter with a size of 2×2 and

a stride of 2. If we apply to the yellow region an average pooling, the output will be 36.25. Otherwise, if we apply a maximum pooling, the output value will be 122. These operations are the same in other color regions.

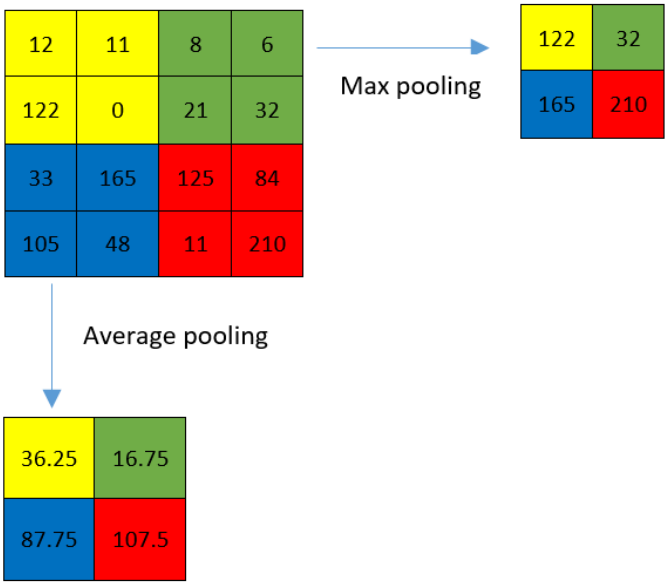


Figure 4.4: The results of different pooling

Dropout (DROP) [SHK+14] is a technique used to prevent over-fitting during the training. The term dropout mentions the exclusion of some output units with their connections (incoming and outgoing) belonging to a layer in the network. The units are dropped randomly with a probability p. When applying the dropout technique, the network becomes a collection of thinned networks [SHK+14]. So, training a neural network with dropout looks like training a collection of thinned networks. The dropout layers are most often placed after the fully connected layers, but it is possible to use them after the pooling layers to create some kinds of image noise augmentation [SHK+14]. In practice, the dropout layer could be considered as a kind of method to reduce the over-fitting in the training step.

Fully-connected (FC) layer usually follows the group of convolutional and pooling layers to extract the abstract feature representations of the image. A CNN may have one or several FC layers. They explain the feature presentation (their input) and perform a

function of high-level reasoning. In practice, the last FC layer produces the outputs of the network. Depending on the problem, an activation function can be added to promote the output values, for example, it uses the linear values for a regression problem, but it is necessary to use the Sigmoid activation function [HM95] in the case of binary classification.

The four types of layers that we have only mentioned are not the only ones that exist. A large list of layer kinds can be found in the related documents of deep learning [LBH15, GWK+18]. Deep learning addresses a lot of different application domains, such as image classification [KSH12, CMS12], object recognition [FCNL13, LLS+15], semantic segmentation [LSD15, RFB15, NHH15, CPK+17, BKC17], image creating/editing [GPAM+14, RMC15], where it has encountered impressive results.

LeNet [LBBH98] model is considered as the first architecture of CNN. LeCun et al. [LBBH98] have used it to classify the handwritten digits in cheques. LeNet exhibits a standard architecture of a CNN that consists of two convolutional layers, pooling layers, followed by two fully connected layers. But to be applied to realistic problems, this model requires huge computation capacities and a large amount of training data which were hardly available in the early 2000s. In the last ten years, the computing capabilities have drastically improved, while at the same time, a huge amount of data became available, new models of neural networks appeared well adapted to this new environment. One of the first ones is AlexNet [KSH12], which is similar to LeNet [LBBH98] but with a deeper structure: LeNet has two convolutional layers and 1 fully connected layer while AlexNet has five and three, respectively. Furthermore, in AlexNet the activation functions have been changed, and dropout layers have been added to prevent the over-fitting. AlexNet won the famous ImageNet Challenge2 in 2012. From the success of AlexNet, a lot of different models have been proposed to improve the performance of CNN; one can cite ZFNet [ZF14], GoogLeNet [SLJ+15], VGGNet [SZ14], or ResNet-50 [HZRS16]. The main difference between these networks is that their architectures became deeper and deeper by adding more layers, e.g., ResNet-50, which won the champion of ILSVRC 2015, is deeper than AlexNet around 20 times. Besides

applying to solve tasks in classification or recognition objects, CNNs have been used in the applications of key points detection too. The research activities in this field will be described in Section 4.2.3. The next section mentions another essential component in deep learning method, data augmentation.

4.2.2 Data augmentation

From AlexNet to ResNet-50, the obtained success stories [KSH12, HZRS16] have proved that CNN models produce better results on a large dataset but to use this technique, the size of dataset remains a bottleneck and our own some hundred of images are considered as modest for these models. So, it is important to be able to provide a large dataset in order to learn more cases and to improve the learning ability of the network. Unfortunately, in some application domains as this work in biology, providing a large dataset is too costly. For this reason, one way to solve this problem is to create misshapen data from real data and to add them to the training set. This process is known as data augmentation.

In deep learning domain, the image augmentation algorithms could be grouped into two main groups. The first group includes the methods based on the classical transformations of the image, e.g., rotation, translation [KSH12, HZRS16]. In the second group, the algorithms use the neural network to augment the dataset. These models could be the adversarial network, the generative adversarial network [GPAM+14], or neural style transfer [GEB15]. For more details about the methods in the two groups, the readers could find in [SK19].

Practically, choosing the augmentation method depended on the task of the problem that we solve. For example, the classical image transformations like rotating, translating, cropping, zooming could be used to increase the number of samples in a dataset for an image classification task [KSH12, DT17, SLJ+15]. However, they could not fit other tasks, for example, object detection task or keypoint detection task because re-labeled data is more costly [GG15, ZCG+19]. In our case, we have preferred to work on the

color channels of the image. All details of the process will be discussed in Section 5.1.

4.2.3 Landmark detection using Deep Learning

In the context of applying CNN for keypoints detection task, Liu et al. [LYL+16] have presented a method to predict the positions of functional key points on fashion items such as the corners of the neckline, hemline, and cuff. Yi Sun et al. [SWT13] have proposed a CNNs cascade to predict the facial points belonging to the human face. Their model contains several CNNs which are linked together in a list as a network cascade. Three levels of the cascade are set to recognize the human face from the global to local view with the objective to increase the accuracy of predicted key points. Zhanpeng Zhang et al. [Z+14] have proposed a Tasks-Constrained Deep Convolutional Network to join facial landmarks detection problem with a set of related tasks, e.g. head pose estimation, gender classification, age prediction, or facial attribute inference. In their method, the input features have been extracted by 4 convolutional layers, 3 pooling layers and 1 fully connected layer which is shared by multiple tasks in the estimation step. Shaoli Huang et al. [HGT17] have introduced a coarse-fine network to locate key points and to estimate human poses. Their framework consists of the base convolutional layers shared by two streams of key point detectors: the first stream, named coarse stream, includes 3 detector branches (3 stacks of Inception modules [SLJ+15]) which are used to focus on capturing local features and modeling spatial dependencies between human parts. The second one, named fine stream, receives the features which are concatenated from the coarse stream and provides accurate localization. Cintas et al. [CQSA+16] have introduced an architecture that enables one to recognize 45 landmarks on human ears. Their model includes 3 times repeated structure. This structure consists of 2 convolutional layers, 1 pooling layer, and 1 dropout layer, to extract the features. These structures are followed by 3 fully connected layers.

In the same context of key point detection, we have developed a CNN model to automatize landmarks on beetle's anatomies. The details of this model will be presented

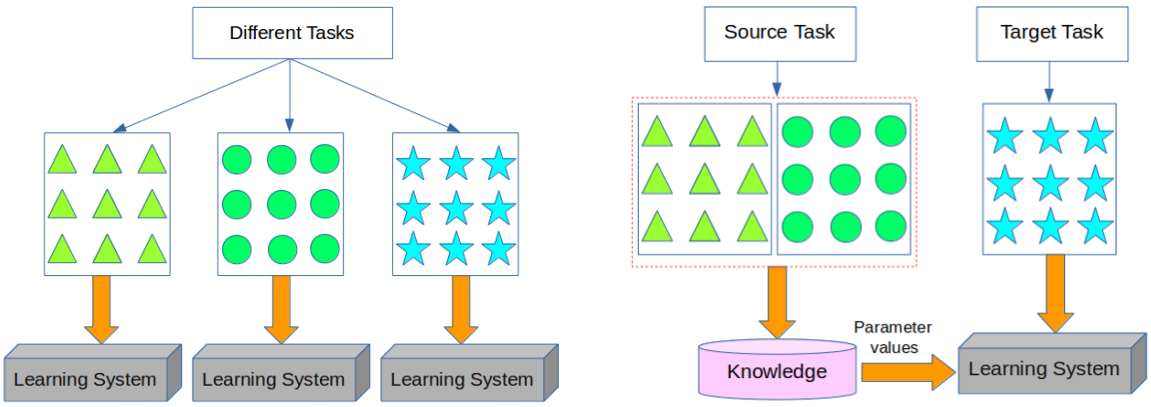
in chapter 5.

4.2.4 Transfer learning

The most popular way to solve a problem by applying deep learning is training from scratch. In this strategy, all layer parameters are randomly initialized. During the training, these parameter values will be updated in the backward phase to provide the best accuracy of the model. Using the random parameter values is extremely risky. It makes the model need a long learning time to optimize these values. Moreover, if we train the model on a dataset that could not provide enough samples for the training process, the model could often fall into the local minima. These problems can be escaped by training the model on a large dataset, which includes several thousand samples. However, this condition is hard to meet in practice because of collecting examples in the real world is costly and time-consuming. Fortunately, we can reuse the parameter values from another experiment to initial the parameter values when two tasks have a relation. This strategy is known as a **transfer learning**, another way to apply Deep Learning.

Transfer learning is a deep learning technique where a model trained on one task (called *source task*) is re-purposed on a second related task (called *target task*) [TS09, YCBL14]. It is the improvement of learning in the target task by transferring the knowledge, which has been learned from the source task. One of the most famous works in transfer learning is using ImageNet dataset [DDS+09] to train different popular models, e.g., AlextNet [KSH12], VGG [SZ14], ResNet [HZRS16]. Then, the parameter values of these models are provided to use as a pre-trained model to solve various problems. Figure 4.5 shows a comparison between 2 strategies to apply deep learning. In the traditional strategy, each model is built and trained for each specific task (Figure 4.5a). On the opposite side, the transfer learning strategy uses the knowledge on the source task to use in the learning system to solve the target task where the source and target tasks have a relationship (Figure 4.5b).

Based on the different situations between the source (dataset/task) and target



a . Traditional learning b . Transfer learning

Figure 4.5: Different processes between traditional learning and transfer learning

(dataset/task), transfer learning can be categorized into three groups [PY10]: *inductive transfer learning*, where the target task is different from the source task, no matter the datasets (source/target) are the same or not; *transductive transfer learning*, where the source and target tasks are the same, while the source and target dataset are different; and *unsupervised transfer learning*, which is similar to inductive transfer learning, but it focuses on supervised learning tasks.

In transfer learning, three main research questions are taken into account: (1) what to transfer? (2) how to transfer? and (3) when to transfer? [YCBL14]. *"What to transfer¿"* mentions the part of knowledge that can be transferred across the tasks. In a learning system, some knowledge is specific for source task, but some others may be common between different tasks. In this case, reusing common knowledge can help improve the performance of the target task. *"When to transfer¿"* asks in which situations, the transferring should be used or should not be used. In some cases, when the source and the target task are not related, transfer learning may be unsuccessful or be lead to unexpected results, for example, it may even hurt the performance of learning in the target task, which is known as the negative transfer. For example, if we transfer the knowledge of a model that is designed to solve a classification problem to apply on a regression problem, the model could make more errors than training from scratch. *"How*

to transfer¿‘ refers to the computation of learning algorithms, which use to transfer the knowledge. Most of the work in transfer learning focuses on a pair of two questions *“What to transfer?”* and *“How to transfer?”* by assuming the relation between source and target domain. However, the answers to another pair of questions: *“When and how to transfer?”*, is also important.

In practice, transfer learning is mostly targeted on 2 scenarios:

- **Use CNN as a fixed feature extractor**: Take a CNN pre-trained on a large dataset, then remove the last fully-connected and use the rest layers of CNN as a fixed extractor for the new dataset.

- **Fine-tuning a CNN**: This situation is the same as the first one. However, it does not only replace and retrain the last layer but also fine-tunes the weights of the pre-trained model by extending the backpropagation. One can note that to reuse a pre-trained model, the parameters have been adapted between two tasks. These parameters could be the size of input images, the number of outputs, or the parameters of each layer. As usual, the parameter values at each layer, e.g., padding or stride values, are selected to change their values to declare the differences between the two tasks.

As it is explained in [TS09, PY10], choosing a transfer learning strategy depends on various factors, but the most important is the size of the target dataset (small or big) and its similarity to the source dataset. The characteristic of a CNN is the generic features extracted by the early layers while the specific features are given by the later layers. Therefore, there are four major scenarios:

1. *Target dataset is small and similar to source dataset*. It is not a good idea to fine-tune the CNN due to over-fitting concerns because the data is small. Since the data is similar to the original data, we expect higher-level features in the CNN to be related to this dataset as well. So, the best idea could be to train a linear classifier on the CNN features.

2. *Target dataset is large and similar to source dataset*. Since we have more data, we can believe that we will not overfit when we fine-tune through the full network.

3. *Target dataset is small and very different to source dataset*. It seems that training the classifier on the top of the network is not the best way instead of the network should be trained from somewhere earlier in the architecture.

4. *Target dataset is large and very different to source dataset*. Since we have a huge amount of data, we can completely train CNN from scratch. However, it is beneficial to initialize the weights from a pre-trained model to boost the learning process and to save time. Thus, we should have enough data and confidence to fine-tune through the entire network.

In practice, transfer learning has been applied in different applications: Ng. Hong-Wei et al. [NNVW15] have applied transfer learning in two stages to recognize the emotions on the human faces. Starting from a generic pre-training process of two CNN architectures (AlexNet [KSH12] and VGG [SZ14]) on the ImageNet [DDS+09] dataset, the first stage fine-tunes the pre-trained models on a facial expression dataset [GEC+13]. The second stage then takes place based only on training part of the Emotion Recognition in the Wild (EmotiW) dataset, adapting the network weights to the characteristics of the Static Facial Expression Recognition in the Wild (SFEW) sub-challenge. Their experimental results have shown that the cascading fine-tuning approach achieves better results than the single stage of fine-tuning on the combined datasets with an improvement of up to 16%. Girshick et al. [GDDM15] have fine-tuned the pre-trained model of AlexNet on the PASCAL dataset [EVGW+10] to perform object detection and segmentation tasks. The obtained results have proved that using transfer learning significantly improved over the other methods without CNN. Shin. Hoo-Chang et al. [SRG+16] has applied deep CNN on medical images to detect the thoracoabdominal lymph node and to classify the interstitial lung disease. They have chosen the two famous models: AlexNet and GoogLeNet. Their studies focused on three important factors of employing a deep CNN to solve a problem: training from scratch, using "off-the-shelf" CNN, and

transfer learning. In learning from scratch, all the parameters of CNN models are randomly initialized and trained on the dataset. In transfer learning, they have followed a hypothesis that: "despite the disparity between natural images, CNNs comprehensively trained on the large scale well-annotated ImageNet may still be transferred to make medical image recognition tasks more effective". Therefore, they have used the pre-trained of AlextNet[KSH12] and GoogLeNet [SLJ+15] models, to fine-tune on their medical image dataset. The performance of using "off-the-shelf" was also considered by using the pre-trained model of AlexNet as a feature extractor and training only the final classifier. At the end of the processes, they have found that the transfer learning strategy yields the best performance results. S. Lin et al. [LZS16] have proposed transfer and specialized net (TS-Net) which fuses the general and specialized knowledge by combining a Transfer FaceNet and a Specialized FaceNet. The former is obtained by fine-tuning the pre-trained GoogleNet network to transfer object-recognition knowledge to face recognition, and the latter is trained on global and local face patches to provide the discriminative specialized knowledge for face recognition.

In order to evaluate different strategies of deep learning and to improve the results, we have tried to transfer the knowledge of AlexNet [KSH12] on the ImageNet dataset [DDS+09] to fine-tune on our images but the results have not been satisfying. However, we believe that transfer learning is a good solution for our application based on what we have studied. To keep the idea of transfer learning, we have chosen another scenario to apply it. All details of this process will be presented in chapter 6.

4.3 Conclusion

In this chapter, we have presented some concepts of machine learning which can be applied to solve problems on 2D images: neural network, deep learning, and CNN. We have also described different strategies to apply a deep learning model in practice: training from scratch or transfer learning. Due to the availability of current resources for computing, deep learning is nowadays applied with success in a lot of image processing

activities.

As we have mentioned in the first part of this document, the classical image processing methods have been difficult to apply on some parts of the beetle's dataset. It explains why we have turned to deep learning to automatically set the landmarks on images without pre-processing steps as segmentation.

In the next chapters, we present the architecture that we have developed to predict the landmarks on beetles images. We will point out also how we have used transfer learning to improve the results.

Chapter 5

Landmarks prediction using Convolutional Neural Network

Contents

In chapter 3, we have described the results on the left and right mandibles images based on image processing algorithms. In mandible cases, images are pretty easy to segment and landmarks are mostly stayed on contours of shape, using image processing algorithms is appropriate, and this way has given good enough results (Section 3.6). However, the method has encountered difficulties when applying to other parts, e.g., pronotum, elytra, and head. Those images do not only contain the studied object part but also some noisy ones, for example, the legs in the case of pronotum images. Additionally, location of landmarks could be both on contours and inside the object.

In this chapter, we present the architecture of the CNN that we have designed to solve the task to automatically determine landmarks on the beetle's images. Accordingly, we have proposed a new composition of layers to build network architecture, called **EB-Net**. This model has been used to analyze the three other parts of beetles: pronotum, elytra, and head. Before to present our model, the first section describes the data management procedure that we have applied to augment the dataset size.

The first results obtained with EB-Net have been presented as a full paper [LBAZP18] presented at The First International Conference on Multimedia Analysis and Pattern Recognition 2018 (MAPR - 2018).

5.1 Data management

The fundamentals of deep learning algorithms are to train models on the dataset repeatedly in order to reach the best accuracy. So, providing a large dataset asserts to learn more cases and clearly improves the learning ability of the network. Unfortunately, in some application domains as in biology, providing a huge dataset is costly and could take a long time to be achieved (several months/years). For this reason, one way to solve this problem is to create misshapen data from real data and to add them to the training set. Figure 5.1 shows the images of three remaining beetle parts in our dataset. As mentioned before, we have only 293 images for each part and this number is modest to apply deep learning method. To supply that, we have augmented the number of

images in each set of images by applying two specific procedures.

Figure 5.1: The images in three remaining parts of beetle. From left to right: pronotum, elytra, and head

Most often in image processing, dataset augmentation [DT17, ZCG+19] uses operations like translation, rotation or scaling which are well known to be efficient to generate a new version of existing images. Using these techniques to augment the number of data has achieved good results in many applications [LKF10, KSH12, GG15]. But as we have mentioned in Section 4.2.2, choosing the augmentation techniques depends also on the application. In our case, we have made several tests by moving the object in the images, but each time, we have quickly observed over-fitting in the training step. It is why we have followed another direction to produce misshapen pictures from existing ones by working on color channels.

Our image set is in the RGB color map, the first procedure consists of changing the value of one of the three channels in the original image to generate a new image. A constant value is sampled in an uniform distribution $\in [0, 255]$ to obtain a new value capped at 255. For example, Figure 5.2 illustrates a pronotum image and three misshapen versions generated by adding a constant $c = 10$ to each channel of the original image. Following this way, three new versions are generated from one image.

In the second procedure, each channel is extracted from the image and the corresponding gray image is generated (Figure 5.3). Consequently, we have obtained 3 new images (one per channel) from an original one. At the end of the process, 6 versions of an original image are made. In total, the new dataset contains $293 \times 7 = 2051$ images

for each anatomical part of the beetle (an original image and six misshapen ones).

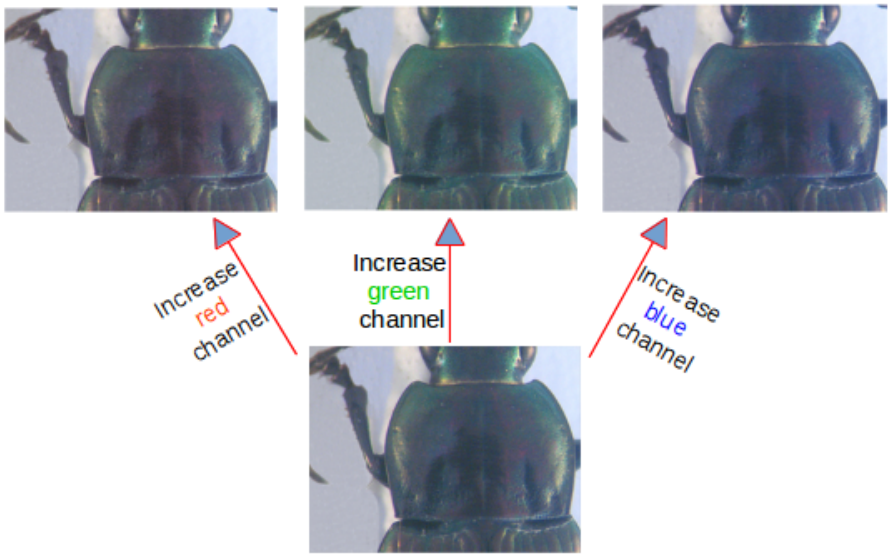


Figure 5.2: A constant $c = 10$ has been added to each channel of an original image

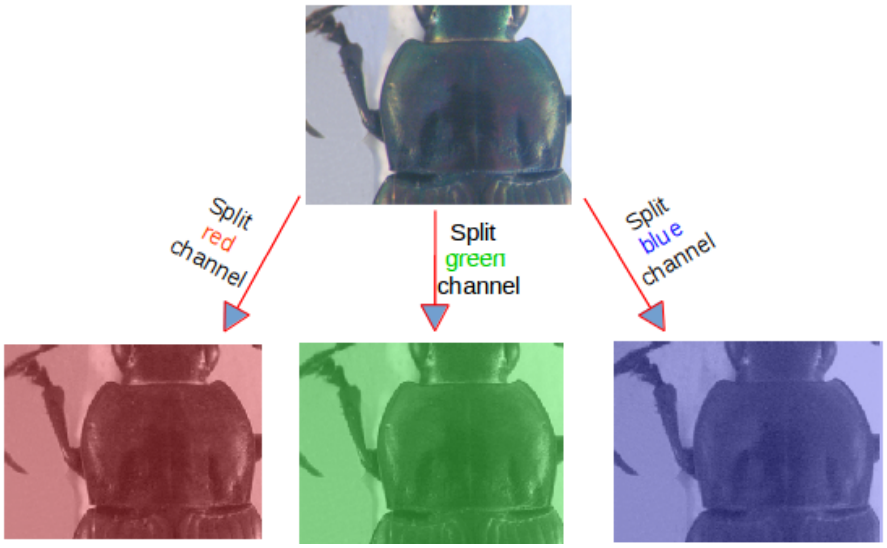


Figure 5.3: Three channels (red, green, blue) are separated from original image

In the objective to perform the learning process, we have observed the size of the input image in some CNN models [CQSA+16, LKF10, KSH12, SWT13] and we have noticed that the size of the input image is limited to 256×256 pixels in most of CNNs for computing reason. It is worth to note that the size of our images is 3264×2448 as

mentioned in Section 1.5. It seems that this size is a little bit heavy for training the network. Consequently, all images have been down-sampled to a new size of 256×192 to respect the ratio between width and height; of course, the coordinates of manual landmarks have been also scaled to fit with the new size. This operation causes a loss of information into the image, but using the original size cannot be foreseen without massive computing resources.

5.2 EB-Net architecture

As we have presented previously, some CNN architectures are available from literature and tools libraries. It is always possible to adapt them to a specific application by changing the parameter values or by modifying the arrangement of layers. Accordingly, several trials have been done before obtaining a satisfying model dedicated to landmarks estimation. In this section, we present the three versions of the model that we have designed to solve this task.

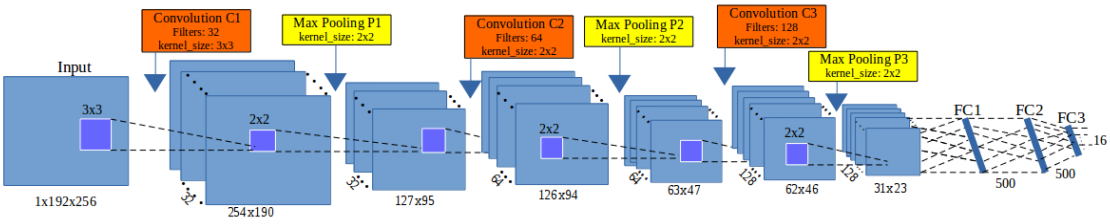


Figure 5.4: The architecture of the first model

The first architecture is a very classical one inspired by AlexNet [KSH12] architecture (Figure 5.4). It receives an input image with a size of $1 \times 192 \times 256$, then it is composed of three repeated structures of a convolutional (CONV) layer followed by a maximum pooling (POOL) one. In most of CNNs, the parameters of CONV layers are set to increase the depth of the images from the first to the last layer. This is done by setting the number of filters at each CONV layer. In this first model, the depths of the CONV layers increase from $32, 64,$ to 128 and with different size of the kernels: 3×3, 2×2 and 2×2, respectively.

Inserting POOL layers after a convolutional layer is usually done. The POOL layer helps to reduce the spatial size of the representation to decrease the number of parameters and the computation time, as well as to control over-fitting. The operations of POOL layers are independent of each depth slice of their inputs. In our model, we have used the common form for a POOL layer: a filter with a size of 2×2 and a stride of 2 pixels. At the end of the model, three fully-connected (FC) layers are added to extract the global relationship between the features and to proceed with the outputs. The first two FC layers take into account all possible features from convolutional and pooling layers. Then, the features are passed through the activation functions before sending them to the third FC layer (output layer). The number of outputs at each FC layer is $500, 500$ and 16. The 16 outputs of the last FC layer corresponds to the coordinates (x and y) of 8 landmarks which we would like to predict on pronotum. Nevertheless, the obtained results with this architecture have not been considered as enough good to continue to use it. One of the main problems is the presence of over-fitting during the training process: detailed results will be discussed in Section 5.4.

The second model has kept the same architecture as the first one. But the number of outputs at the first two FC layers has been increased to 1000. Increasing the value at FC layers allows getting more features from the CONV layer with limited requirements of new computing resources. However, the obtained results are still not satisfactory, it will be discussed also in the result section (Section 5.4).

To build the third architecture, we have defined a new concept: ***Elementary Block (EB)***. An *EB* is defined as a sequence of a CONV (C_i), a maximum POOL (P_i) and a dropout (D_i) layer (Figure 5.5). The Dropout layer has been added to prevent over-fitting [HSK+12, KSH12, SHK+14, SWS17, HLVDMW17]. As mentioned in Section 4.2.1, adding Dropout layers is like creating some kinds of image noise augmentation. It provides more studied cases to the network model. Additionally, the dropout layer will randomly drop some connections in each training iteration. This work makes the network thinner than the original one, and training a network with the dropout layer is equivalent to train a set of thinner models. It significantly reduces overfitting and gives

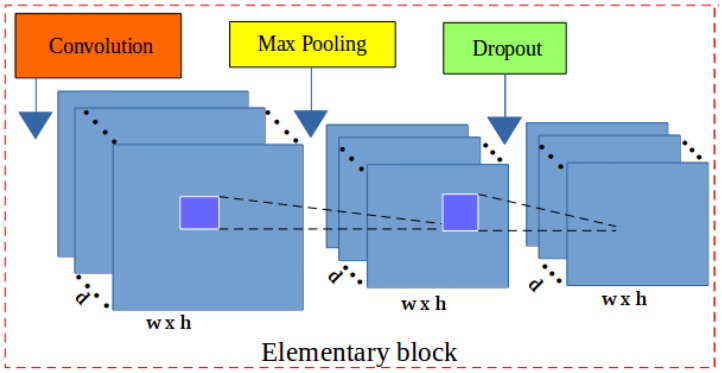


Figure 5.5: The layers in an elementary block. It includes a CONV layer (red), a maximum POOL layer (yellow) and a DROP layer (green).

more major improvements than other regularization methods [SHK+14]. Most often, adding the Dropout layer is seen only before the FC layers. In our case, it is inserted in the core of the EB. The probabilities applied for different dropouts are set to increase depending on the step (details are given in Appendix C). The final architecture, EB-Net, is a composition of elementary blocks, named Elementary Blocks Network.

Figure 5.6 describes the EB-Net architecture. For our purpose, we have assembled **three Elementary Blocks** initially. The parameters for each layer in each elementary block are detailed in Appendix C. Followed the Elementary Blocks, three FC layers have been added. The parameters for each FC layer remain the same as in the second architecture: FC1 and FC2 have 1000 outputs; the last FC layer (FC3) has 16 outputs. As usual, a dropout layer is inserted between FC1 and FC2 with a probability equal to 0.5.

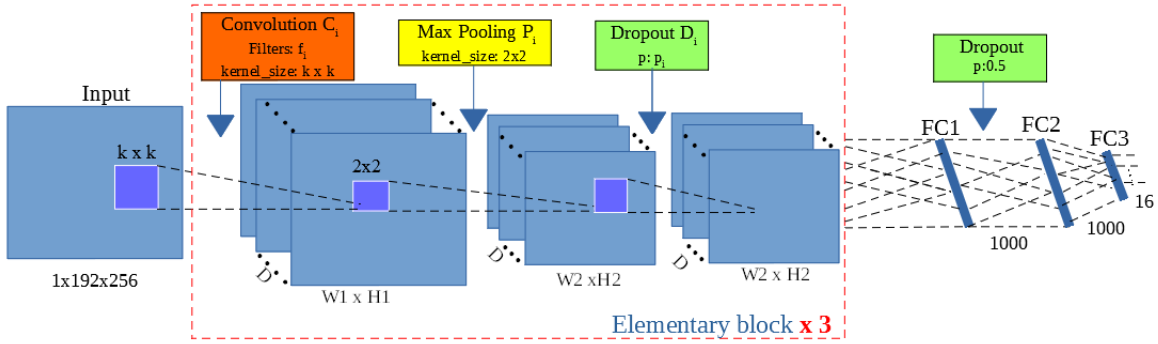


Figure 5.6: The architecture of the third model

5.3 EB-Net hyperparameters

In a CNN model, the hyperparameters can be divided into two categories: model-specific and optimizer hyperparameters [GBC16, BB12, BBBK11]. The model-specific hyper-parameters involve the structure of the network, for example, the number of hidden layers or the method to initialize the parameter values, which are determined during the designing process. The optimized hyper-parameters are the variables that are used to train a network model, for example, choosing the loss function, initializing the values of the learning rate, batch size, number of epochs, etc.

In practice, hyper-parameter values depend on the task and the dataset. They are usually determined empirically. There are many optimized algorithms in deep learning, but gradient descent is well known as a good choice to reduce the loss in the neural network. The core of gradient descent [Ama93] is to follow the gradient until reaching a minimum of the cost function. Consequently, we have chosen the gradient descent as the optimization algorithm with a learning rate initialized at 0.03 and to stop at 0.00001; while the momentum rate is updated from 0.9 to 0.09999. During the training, the values of learning and momentum rates are updated to fit with the number of epochs by applying parameter adjustments. Additionally, we have chosen the Root Mean Square Error (RMSE) as the loss function because it is usually used for regression problems where the outputs are not discrete values as in the case of landmark coordinates.

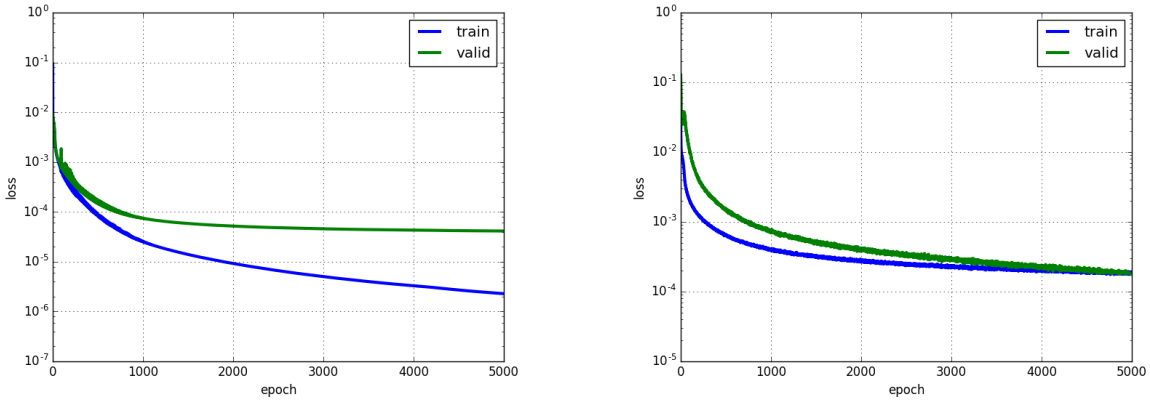
5.4 First results

The network architecture has been implemented by using Lasagne framework [D$^+$15] and trained in 5000 epochs on Linux OS by using NVIDIA TITAN X GPU card.

To predict landmarks for all pronotum images, we have applied the cross-validation procedure to select the test images, we will call a selection step is a round. For each round, we have decided to choose 33 images for testing. The 260 remaining pictures will be used to train and to validate the model. Of course, the set of 260 images has

been augmented as described in Section 5.1 to provide 1820 images for these 2 steps. To achieve the cross-validation steps, we have to do 9 rounds in total to predict all landmarks. It is worth to note that we have used the down-sampled images (256×192, resolution is approximated to 47 pixels/mm).

During the training and validation step, 1820 images are randomly separated into two sets with a ratio of $60\% : 40\%$ (training: validation). In the training step, each pair of an *image* and *its manual landmarks* are inputted to train the network model. For each pronotum picture, a set of 8 manual landmarks is available. They are considered as ground truth to evaluate the predicted ones.



a . The 1^{st} model

b . EB-Net (The 3^{rd} model)

Figure 5.7: The losses (training and validation) of the 1^{st} model and EB-Net

As mentioned in Section 5.2, the first and second architectures exhibited over-fitting behavior. Figure 5.7a shows the different curves of the losses during the training and the validation step of the first architecture. The blue curves present the RMSE error of training processing, while the green curves are the validation errors. We can see that the training loss is able to decrease along with the number of epochs, but the validation loss is stable. Clearly, the over-fitting has appeared in the first model. In the second model, no concrete change appears in the curves even the parameters of fully-connected layers have been modified. On the opposite side of the first two models, the losses of the third model, EB-Net, become close after several hundred epochs and the over-fitting

disappears as shown in Figure 5.7b. Therefore, we can assume that the addition of the *dropout* sequence inside the elementary block works well to prevent over-fitting and improves the accuracy of the model precisely.

Table 5.1 resumes the losses of 9 rounds when we trained EB-Net on pronotum images. For each round, the training and the testing images are different. However, we can observe that differences (of the training/validation losses) among rounds are not so high, they are tiny and stable.

Round	Training loss	Validation loss
1	0.00018	0.00019
2	0.00019	0.00021
3	0.00019	0.00026
4	0.00021	0.00029
5	0.00021	0.00029
6	0.00019	0.00018
7	0.00018	0.00018
8	0.00018	0.00021
9	0.00020	0.00027

Table 5.1: The losses during the training of the third model on pronotum images

After considering these results, we have focused on the location of predicted landmarks comparing to the ground truth. So, we have calculated the distances (in pixels) between coordinates of manual landmarks and predicted ones for all images. Then, the average distance between manual and estimated landmarks has been computed for each landmark position.

Table 5.2 shows the average distances by landmarks of all images in the pronotum dataset. With the image's size of 256×192, we can consider that an error around 1% (corresponding to 2 pixels) could be an acceptable error. Unhappily, our results exhibit an average distance of 4 pixels in the best case, the 1^{st} landmark and more than 5 pixels in the worst case, the 6^{th} landmark. The values of other cases are approximate to 4.3 pixels.

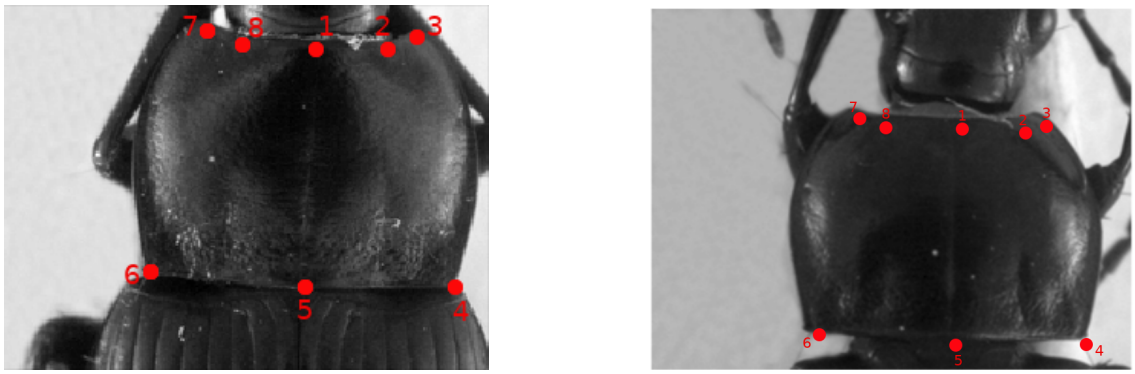
Important notice: The distances are given in pixels. We want to remind you that after down-sampling, the resolutions for pronotum, head, and elytra are 47 pixels/mm,

47 pixels/mm, and 23.5 pixels/mm, respectively. We will keep pixel unity because we consider the image on the screen and not on the real insect.

Landmark	**Distance** (in pixels)
1	**4.002**
2	4.4831
3	4.2959
4	4.3865
5	4.2925
6	**5.3631**
7	4.636
8	4.9363

Table 5.2: The average distances on all images per landmark on pronotum images.

To illustrate this point, Figure 5.8 shows the predicted landmarks in two random cases in our test images. One can note that even some predicted landmarks are close to the manual ones in most of the cases (Figure 5.8a), we have also some predicted coordinates that are far from the expected results (Figure 5.8b).



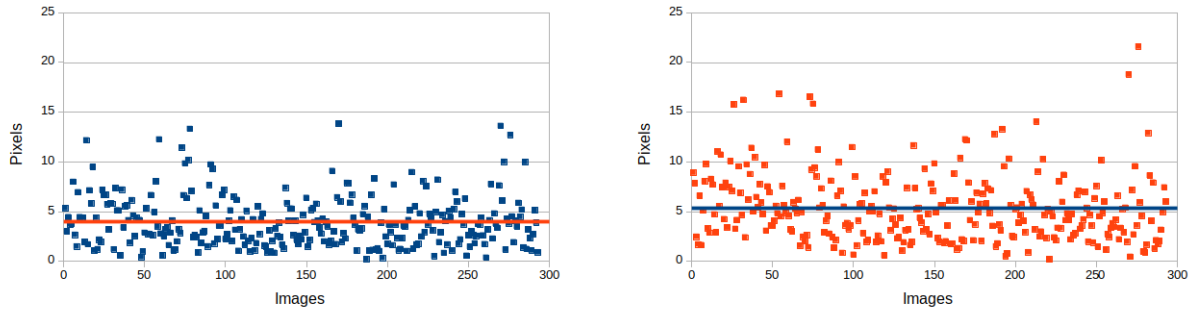
a . Image with well-predicted landmarks

b . Image with inaccuracy landmarks

Figure 5.8: The predicted landmarks, in red, on the images in test set.

Figure 5.9 shows the distribution of distances between manual and predicted landmarks on all images for the best and the worst cases. Each point presents the distance between keypoints (manual and predicted one) of an image. The lines are mean values of all distance values. It is worth to note that the mean value could reflect two different

cases: a lot of values closed together (small dispersion) or two sets of values very far (large dispersion). The first our results seem to exhibit a small dispersion even some points are still far away from the mean value (the 6^{th} landmark). We will discuss some examples of this miss estimation in the next section and how we have improved the global results in the next chapter.



a . The 1^{st} landmark

b . The 6^{th} landmark

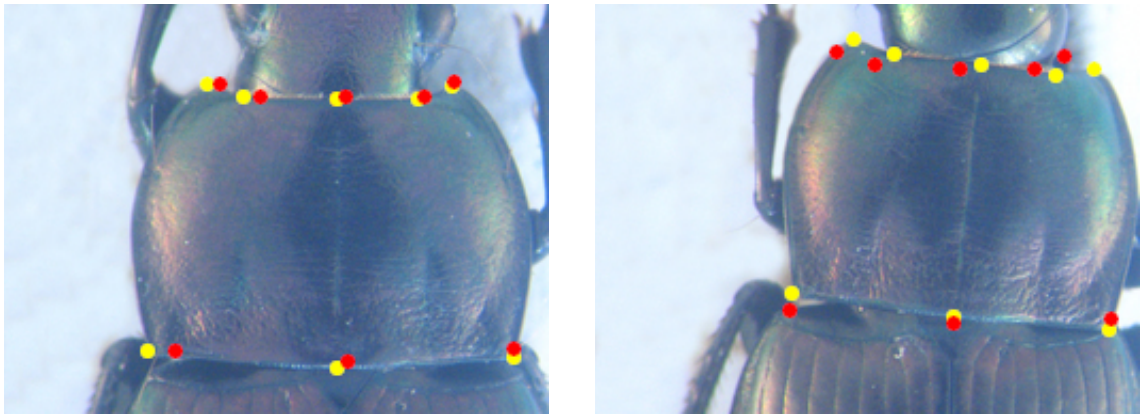
Figure 5.9: The distribution of distances between manual and corresponding landmarks of all images for the best (1^{st} landmark) and the worst case (6^{th} landmark)

To be more confident about our conclusion, the same procedure has been applied to two other parts of the beetle: head and elytra. The results are statistically similar (calculating the average distances and illustrating the distribution of distances). The results of these parts will be detailed in Appendix C.

5.5 Limitation of the model

The EB-Net has achieved good outcomes in most cases. However, it exists several difficult images in our dataset that the network could not recognize. We can see clearly from Figure 5.9 that we have several distances far away from the average values, both in the best and in the worst case. Figure 5.10 shows some examples of the pronotum that we have extracted from our dataset. The reasons have been found down by checking the images. In the pronotum 282, the head shape has misled the estimation on the left side of the upper part, and the same situation has been done by the leg's shadow at the lower

part. The pronotum 285 exhibits good coordinates in the bottom part, but it seems that the procedure has preferred to approximate the upper shape by following the wrong line on the right side. Improving training can help to outreach the troubles on difficult images. One way to do is to re-use the parameters coming from a pre-trained model on a huge dataset. This is the purpose of the next chapter.



a . Pronotum 282 b . Pronotum 285

Figure 5.10: The difficult cases (pronotum images) to provide the estimation landmarks. The yellow/red points represent the manual/estimated landmarks.

5.6 Conclusion

In this chapter, we have presented a new CNN model, EB-Net, to predict landmarks on 2D images of beetles. EB-Net is composed of Elementary Block three times repeated. An Elementary Block consists of a convolutional layer, a maximum pooling layer, and a dropout layer. Then, three fully-connected layers and a dropout layer achieve to compute landmarks coordinates. We have also proposed a strategy to augment the number of images in the dataset by modification of their color channels.

The network has been trained several times with different selections of training data (round). After training with the manual landmarks given by the biologist, the network was able to predict the landmarks on the set of unseen images. Then, the

predicted landmarks have been evaluated by calculating the distance between them and corresponding manual ones. The average of distance errors on each landmark has been also considered. The results have shown that using the convolutional network to predict the landmarks on biological images has promised good results in the case of the pronotum, head, and elytra. The quality of the prediction allows using automatic landmarks to replace manual ones for statistical analysis. However, if we consider the end-user point of view, the results are still needed to be improved. The last chapter will present how we have used transfer learning to solve this problem.

Chapter 6

Transfer learning and final results

Contents

In the previous chapter, we have proposed a new architecture of Convolutional Neural Network (CNN), EB-Net, to predict the landmarks on the three sets of 2D beetles images: elytra, pronotum, and head, which have exhibited shape characteristics remained the awkward problematic of segmentation. The results have figured out that using CNN to predict landmarks on these 2D images is relevant and the obtained results are statistically good enough to replace the manual ones. However, the predicted landmarks remain far from the manual ones in some cases, and we are looking for some improvements.

As mentioned in Section 4.2.4, working with Deep Learning requires not only to design a good architecture but also to provide a large dataset to train and to test the model. This is a potential problem in some application domains, as in biology. In Section 5.1, we have presented a way to augment the number of images in our dataset. However, our number is far away from several thousands even we have augmented our dataset. In this case, knowledge transfer or transfer learning between tasks could be advisable [TS09, YCBL14] as we have described in Chapter 4.

In this chapter, we present the last step of our workflow, which is a fine-tuning process. At this step, we have trained EB-Net on another large dataset, e.g., facial keypoints dataset, to obtain training parameter values and then to re-use them to fine-tune the model on beetle's images.

6.1 Fine-tuning design

As we have mentioned in Chapter 4, transfer learning is another strategy to use deep learning methods in the case we have a limited dataset to train the model. Fine-tuning is a scenario of transfer learning, and it is widely used in practice to boost the efficiency of a model. Technically, we usually fine-tune the weights of a CNN by continuing the backpropagation. It exists two ways to perform fine-tuning: *frozen* and *unfrozen*. With the *frozen* scenario, the parameters of several lower layers (close to the input layer) will be fixed, and we fine-tune only the higher ones (close to the output layer). On the

opposite side, *unfrozen* allows continuing updating the parameter values of all layers in the model. The hyperparameters of the network, e.g., learning rate, have also been adjusted. As usual, a small learning rate is recommended during the fine-tuning process to assume that the pre-trained weights are good enough and we do not wish to disfigure them too quickly and too much. For example, the learning rate of the fine-tuning process could equal to 1/10 time of the pre-training one.

In this context, ImageNet [DDS+09], a well-known dataset with more than one million images labeled in $20,000$ categories, has been used to train many famous CNN architectures [KSH12, SZ14] with success as we have mentioned in Section 4.2.4. The pre-trained models on ImageNet have been then shared in the deep learning community as a source to re-use the features of ImageNet dataset. Unfortunately, some preliminary tests have shown that re-using ImageNet features is not relevant for our application because ImageNet features mainly concern the detection of the global shape of the objects whereas landmarks can be considered as local features [LZS16]. Luckily, searching for landmarks is well defined in other applications as face recognition and facial keypoints detection, and we can consider that these applications present similarities with our problem. Consequently, we have decided to train EB-Net with facial key points dataset and then to transfer the trained parameters to fine-tune it for beetle's images. All the processes are detailed in the next sections.

6.2 Pre-trained EB-Net on Facial Keypoint datasets

A **Facial Keypoints dataset** has been published for a competition in the Kaggle community [1]. It includes $2,140$ human face images with a size of 96×96 pixels. Each image contains 15 landmarks on the face: 6 landmarks for eyes, 4 landmarks for eyebrows, 4 landmarks for the mouth, and 1 landmark for nose tip. Figure 6.1 shows four face images in the dataset and the landmarks on each face.

At the pre-training step, EB-Net has been trained with this dataset, the first objective

[1] https://www.kaggle.com/c/facial-key points-detection

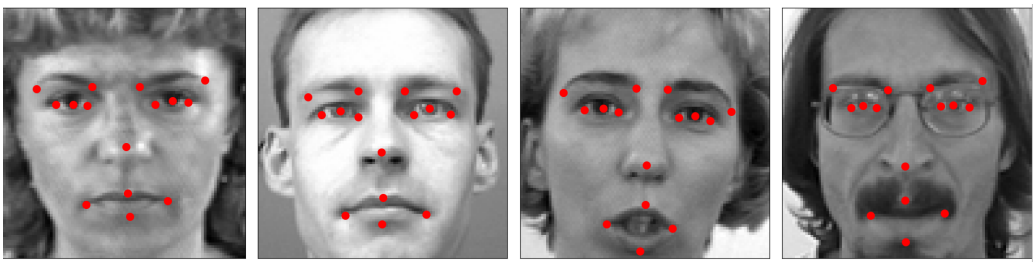


Figure 6.1: Four face images in the dataset and ground truth position of the landmarks.

of this task was to evaluate and to compare the effectiveness of EB-Net with other published results in the Kaggle challenge. At this step, we have adapted the parameters of the input and the output layers of EB-Net to match the requirements of the challenge. The new parameter values are 96×96 for the input's size value, and 30 for the output of the last FC layer. Considering the EB-Net hyperparameters, the learning rate and momentum remained the same but the number of epochs has been increased to 10000.

Team	Olegra 1^{st}	Trump 2^{nd}	Enes 3^{rd}	Our
RMSE score (in pixels)	1.2824	1.4004	1.4026	**1.497**

Table 6.1: RMSE comparison between our score and top three of challenge.

After training, we have tested EB-Net with 100 images. Then, the distances between the predicted and manual ones have been computed to finally calculate the RMSE score. The obtained value is 1.497. Comparing with other results in the Kaggle challenge (Table 6.1), three models present results better than us but very closed when considering that it is pixel value. In our opinion, the RMSE score around the 1 pixels is not so far if we would like to display the landmarks on the images. Consequently, we have the base to believe that EB-Net is still good in any way, and we have decided to re-use the pre-trained parameter values to fine-tune the model for beetle images.

6.3 Fine-tuning on beetle's images and results

One can note that the sizes of images between the two datasets are different, the beetle images have a size of 256×192 pixels; whereas the facial images are 96×96. Therefore, adjustments are needed to match the two tasks.

First of all, reducing the resolution of the beetle images to 96×96 could be lead to a loss of essential information. As our images contain a background band that is easy to suppress with a pre-processing operation, we have chosen to remove the background region instead of down-sampling our pictures. Moreover, removing the background pixels can limit the effect of un-useful image areas.

We have also respected the form of images that they are square. The new beetle images are finally set to 192×192 pixels. The EB-Net parameters will be settled to take into account these values between the pre-training and fine-tuning steps. To declare the modification, we mention in a stride value of the first convolutional layer equals to 2 (as the usual way to do [YCBL14]).

To evaluate this process, the parameters of the pre-trained model are transferred to separately fine-tune on three sets of images: pronotum, head, and elytra. We present all the obtained results in the same way as in Chapter 5 to have a clear comparison. We have also applied this approach to the mandible images in order to compare the obtained results with the results provided by image processing techniques (Chapter 3).

6.4 Results

Important notice: In all result tables, we use the pixel as unity. We want to remind that the pronotum and head have the same resolution (47 pixels/mm), and different than the elytra (23.5 pixels/mm) as specified in Chapter 5.

6.4.1 Distance between manual and predicted landmarks

The distances (in pixel) between predicted landmarks and corresponding manual ones have been computed, as we have done previously (Chapter 5). Then, these distances are used to compute the average value for each position on all images. Tables 6.2, 6.3, and 6.4 resume the results on the pronotum, head, and elytra: The first column presents the landmark number; **From scratch** column reminds the previously average distances when EB-Net was trained from scratch; **Fine-tune** column presents the new average distances; **the fourth** column presents the improvement percentage between the two processes. The green and red values are respectively the best and the worst values in each column. All distances are given in pixel unity.

First of all, to check if average distances are significantly different, we have computed the p-values from the average values of two processes (training from scratch and fine-tuning). These p-values are $0.057, 0.005, 0.031$ on pronotum, head and elytra, respectively, that is statistically significant.

One can note that all average distances have decreased between 1 and 1.5 pixels both in three sets of images. From tables 6.2, 6.3, and 6.4 we can see that sometimes the points with the best-predicted coordinates have changed, but we prefer to notice that it exists a group of well-predicted landmarks where the differences between the average distances are less than 0.5 pixels, such as:

- For pronotum: the 7^{th}, 3^{rd}, and 1^{st} landmark.
- For head: the 8^{th}, 6^{th}, 7^{th}, and 10^{th} landmark.
- For elytra: the 3^{rd}, 1^{st}, 4^{th}, 2^{nd}, 11^{th}, 5^{th}, and 10^{th} landmark.

In another way, we can examine the worst cases. They remain in the same positions as previously in both 3 datasets:

- For pronotum: the 6^{th} remains isolated as a bad result.
- For head: a group of 2 landmarks (1^{st} and 3^{rd}) obtained more than 4.5 pixels.

LM	From scr.	Fine-tune	% of impr.
1	**4.00**	2.99	25.25
2	4.48	3.41	24.01
3	4.30	2.98	30.56
4	4.39	3.54	19.18
5	4.29	3.37	21.55
6	**5.36**	**4.06**	24.28
7	4.64	**2.93**	36.85
8	4.94	3.64	26.16

Table 6.2: Average distances comparison on pronotum images

LM	From scr.	Fine-tune	% of impr.
1	**5.53**	**4.82**	12.83
2	5.16	4.21	18.43
3	5.38	4.73	12.15
4	5.03	4.11	18.42
5	4.84	4.18	13.69
6	**4.45**	3.50	21.43
7	4.79	3.92	18.29
8	4.53	**3.40**	24.94
9	5.14	4.17	18.88
10	5.06	3.94	22.01

Table 6.3: Average distances comparison on head images

#LM	From scratch	Fine-tune	% of improvement
1	**3.87**	3.21	17.04
2	3.97	3.28	17.34
3	3.92	**3.20**	18.36
4	**3.87**	3.22	16.61
5	4.02	3.31	17.66
6	4.84	4.21	13.13
7	5.21	4.54	12.82
8	**5.47**	**4.76**	12.96
9	5.27	4.55	13.69
10	4.07	3.39	16.68
11	3.99	3.29	17.54

Table 6.4: Average distances comparison on elytra images

- For elytra: it exists a group of 3 points which are close to bad result (7^{th}, 8^{th}, 9^{th}).

When we consider each landmark, the percentage of enhancement is different depending on the difficulty of its position. However, all cases have been improved, even they are the worst or best ones. With the help of fine-tuning, most of the predictions have been gained from 36.85%/ 24.94%/ 18.36% to 19.18%/ 12.15%/ 12.82% on pronotum, head, and elytra, respectively.

To look for additional improvement, we have studied other statistical indicators, e.g., standard deviation, median, minimum, and maximum values. These statistical

values are presented in Appendix D. From tables, the median values, which separate data samples into two parts, are smaller than mean ones. Moreover, they are close to the minimum, as well as so far from the maximum scores. This reveals the presence of some isolated cases, which are far away from the mean values, has affected the overall outcome. To deeply understand this problem, we continue with the distribution of distances on each landmark.

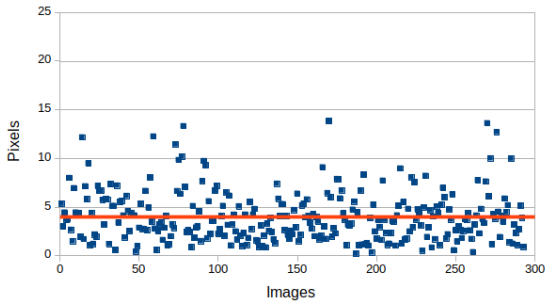
6.4.2 Distribution of distances

The average distance between manual and predicted landmarks could be not the only way to appreciate the obtained results. As we have mentioned, the mean value can hide different situations when deeply going to the analysis. Again, it could exist of two cases for an average value: all distance values are very close to the mean value, or they are widely widespread around the mean one. In this case, distribution of distances could be taken into account to characterize this situation.

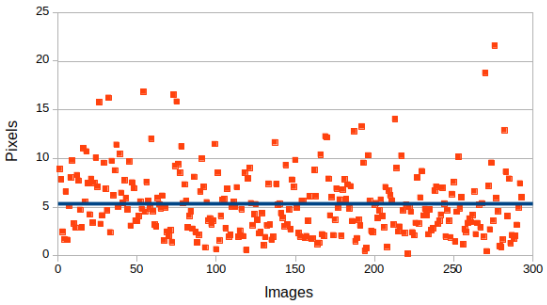
Figure 6.2 illustrates the distribution of the distances for two example cases: the 1^{st} and 6^{th} landmarks of all pronotum images. In the same meaning as figure 5.9 (Chapter 5), the *x-axis* and *y-axis* present the image number and the distance (in pixel), respectively. Each point in the chart represents a distance between manual and predicted landmarks. The blue/red lines in charts present the average values. We can observe from the figure that the distance values are very different from the two processes. With the help of fine-tuning, these distances have been reduced and more close to the range between 0 and average value.

6.4.3 Illustration of landmarks on image

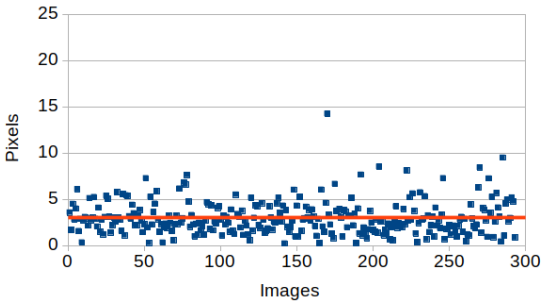
To illustrate the results, Figure 6.3 shows both the predicted (in red) and the manual (in yellow) landmarks on three random images of the three beetle parts. For example, we have obtained seven well-predicted landmarks on the head images. The three remaining positions are a little bit far from the manual ones.



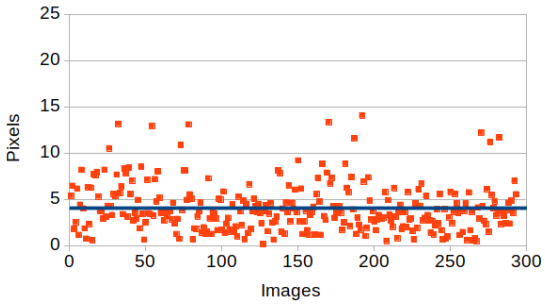
a . 1^{st} landmark (from scratch)



b . 6^{th} landmark (from scratch)

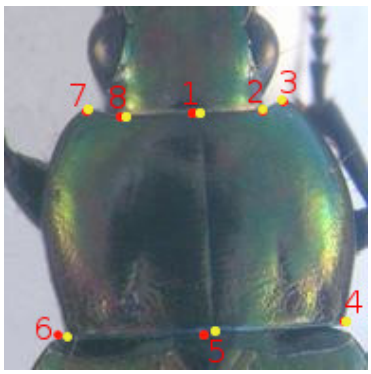


c . 1^{st} landmark (fine-tuning)

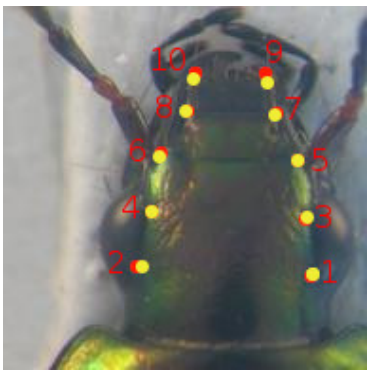


d . 6^{th} landmark (fine-tuning)

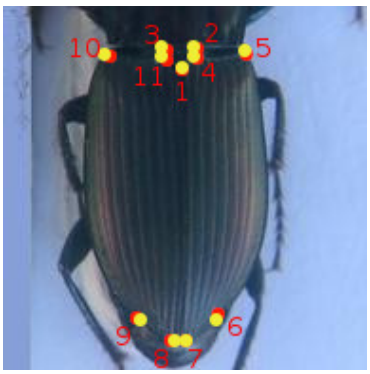
Figure 6.2: A comparison of distances distribution of the 1^{st} landmark and the worst case (6^{th} landmark) when applying two processes.



a . Pronotum



b . Head



c . Elytra

Figure 6.3: The location of predicted landmarks in one case of each part. The red/yellow points represent the predicted/manual landmarks.

We wanted to display an illustration of all the computing on a generic image of pronotum. Figure 6.4 displays mean coordinates on each landmark of all pronotum

pictures on a pronotum's shape[2]. The 3 colored points correspond to:

- Red points are the mean coordinates of all manual landmarks.
- The yellow squares present obtaining outcomes when we train the model from scratch.
- The green boxes show the results of the fine-tuning process.

One can note that the center of each square is the mean coordinates of all predicted landmarks in each process (from scratch and fine-tuning). The length of the square equals to double of average distance at each landmark position. We can observe that the distance of the average distances between two processes on each landmark is not so large. The overlap between the yellow and green squares shows the similar/different values of the distances. Moreover, it exists a consistency between the mean of manual coordinates and the average coordinates of predicted landmarks because the manual landmarks are always inside the range of the predicted ones (green rectangles).

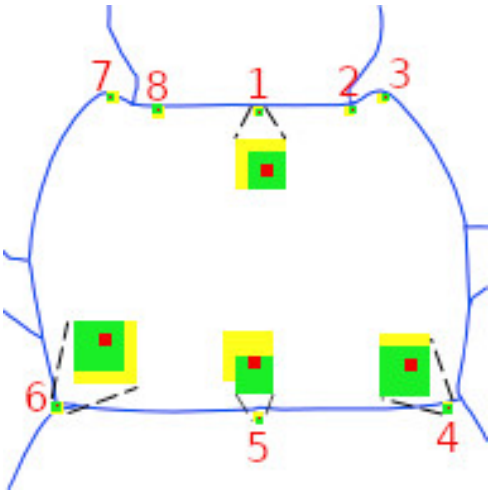
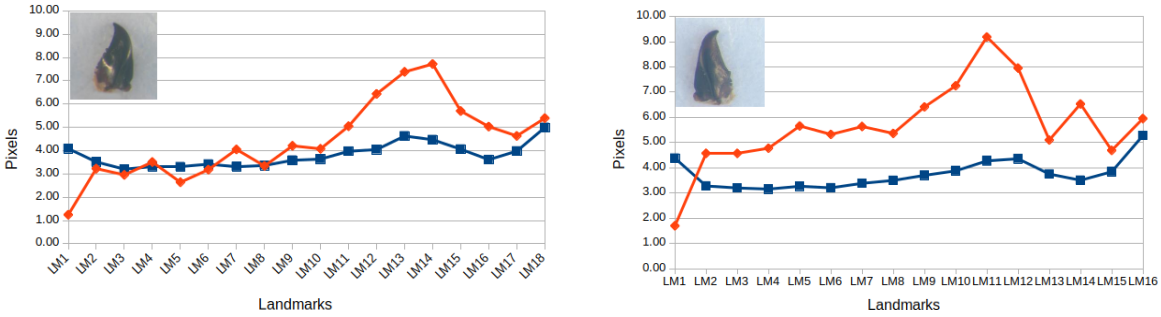


Figure 6.4: The location of average coordinates. The Zoom-in operation has been effected on several landmarks.

[2]No generic model of pronotum shape is available, we have randomly chosen one in the dataset to figure out it.

6.4.4 A comparision with Image Processing Techniques method

Mandible images have not been processed with the EB-Net model in the previous chapter. However, we have applied the fine-tuning on mandible images in the same way that we have done on other parts. Then, the results have been used to have a comparison between the two methods (deep learning and image processing methods). Figure 6.5 shows these results. The red curves illustrate the average distances which have been obtained from image processing technique while the blue curves present the results of fine-tuning process. To reach the comparison, the obtained values from Chapter 3 have been re-computed to match the new size of the images (192×192) by scaling these errors (distances) with the same ratio that we have used to down-sample images. Then, the average value has been computed for each landmark's position.



a . Right mandible

b . Left mandible

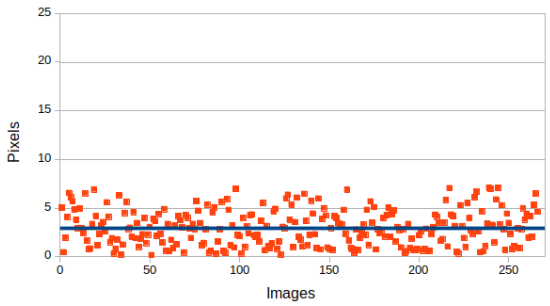
Figure 6.5: These charts show the average distance on each landmark of all mandibles images. These values have been down-sampled from the last results of Chapter 3 to fit with the new size of images. The red, blue lines present the results from image processing and fine-tuning process, respectively.

For the right mandible (Figure 6.5a), with image processing techniques, some first landmarks (from 1^{st} to 6^{th}) have been well-predicted, which illustrated by small average values in the chart. However, these values begin to increase from the 7^{th} landmark. The reason is that the first group of events is mainly concentrated on the tip of the mandible where we can obtain the precise segmentation, while others are on the base where we can meet some difficulties to get the segment contours. For the fine-tuning process, the obtained values are more stable. Although some first values can be approximate or

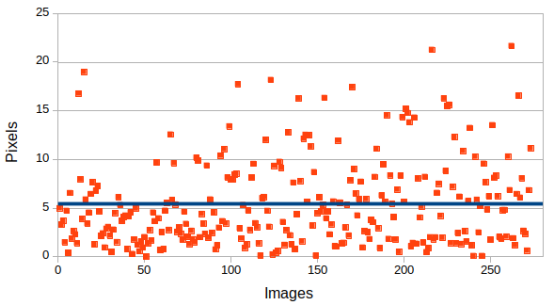
larger (from 2^{nd} to 7^{th} positions), other ones are better than the previous results (after 7^{th} position). Remarkably, the fine-tuning has a great improvement at the positions located at parts that are hard to extract the contours.

For the left mandible (Figure 6.5b), it is worth to note that they have been more difficult to process than the right ones by using image processing techniques, as we have discussed in Chapter 3. However, with the help of fine-tuning, the results are almost the same as the right ones. They are stable and present significant improvement in most of the landmarks.

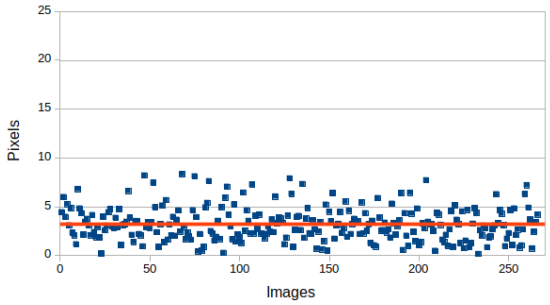
To achieve the analysis, we have also displayed the distribution of distances as we have compared in previous parts but by image processing techniques. Figures 6.6, 6.7 show these distributions in two cases: the best and the worst average distances (with the fine-tuning process). From the diagrams, we can observe that left mandibles exhibit a large dispersion with image processing. The different charts clearly illustrate the improvement of the fine-tuning process.



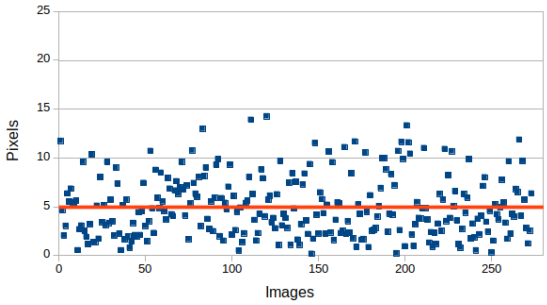
a . 3^{rd} landmark (image processing)



b . 18^{th} landmark (image processing)

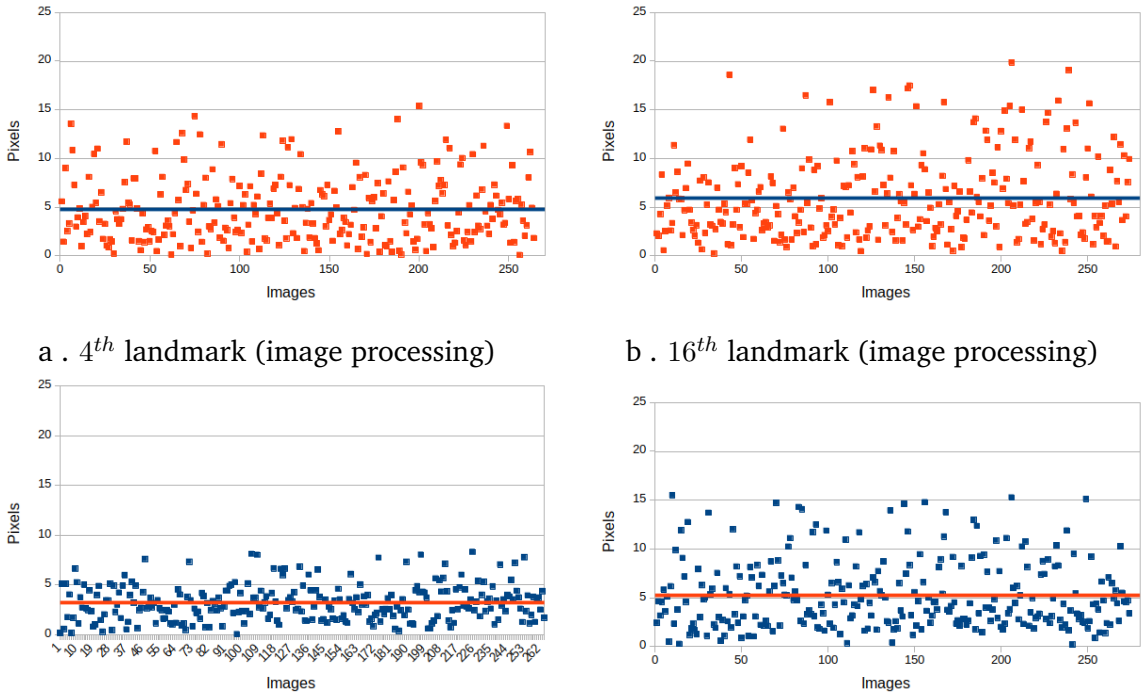


c . 3^{rd} landmark (fine-tuning)



d . 18^{th} landmark (fine-tuning)

Figure 6.6: The distribution of average distances of all right mandible images.



a . 4^{th} landmark (image processing)

b . 16^{th} landmark (image processing)

c . 4^{th} landmark (fine-tuning)

d . 16^{th} landmark (fine-tuning)

Figure 6.7: The distribution of average distances of all left mandible images.

From the achieved results, we give a short comparison between the two methods, Classical Image Processing (CIP) and Deep Learning (DL). As usual, CIP uses a set of techniques to achieve the last results. The algorithms in the method are easy to find in available libraries. In our application context, it does not require so many computing resources to apply it, and we can use it to process on a small dataset. However, a lot of features of the image could be considered as useful features for analyzing, choosing the proper ones becomes a hard problem in the CIP methods, and we need to try a long trial to decide which are best to describe the landmarks. Moreover, we have a link between the results of the steps in the method. If we fail at any level, it will affect the final result.

Deep Learning provides an embeded process to analyze the image. We only give a dataset of images with a set of manual landmarks to the model. Thereby, we train the network on the given data. It will automatically discover the particular features for each key point. At the end of the process, it will provide the coordinates of estimated

landmarks. From the comparison, the outputs are more stable and better than with CIP. However, we need to take into account some other aspects with DL methods such as the number of data, designing model architecture, or computing resources which can be difficult to satisfy. Along with the development of computing resources, deep learning methods could be a good choice for landmark detection on 2D images.

6.5 Feedback from modifications of EB-Net

In the deep learning community, the last ten years have seen a kind of competition about the size of the networks. The success studies have proved that adding hidden layers and increasing the depth could be beneficial for a complex task, e.g., multi classes classification or landmark detection as demonstrated [SZ14]. In our case, EB-Net has a very modest size. To check if we can obtain an improvement of the EB-Net results, we have tested by enlarging the size of the model. It is worth to note that we do not join to a breakneck quest for huge deep network size, but we want to appreciate the influence of the number of Elementary Block (EB) inside our model. To do that, we have decided to add only one EB to the previous architecture. In this block, the depth of the convolutional layer is larger than others to make the system to become more powerful to learn complicated patterns in data [KSH12]. The whole set of parameters of the layers in the new EB are set as follows:

- **Convolutional** layer: depth $= 256$, filter $= (2, 2)$, stride $= 1$
- **Maximum pooling** layer: filter $= (2, 2)$, stride $= 2$
- **Dropout** layer: probability $= 0.4$

Another tested hypothesis was to change the activation function. In the list of activation functions, ReLU was known to be more efficient than other ones as Sigmoid, Softmax, TanH, ... (more detail in [GBC16]). Technically, ReLU function outputs all the positive values and zero for all negative values. Therefore, the last version

of EB-Net has used ReLU as non-linear functions as usual in the most architectures [KSH12, HZRS16, LSD15] of the deep convolutional neural network. However, it is worth to note that ignoring negative output values by setting to zeros is equivalent to stop some neurons in the network model. Such neurons are not playing any role in the forward and backward phases, and the network can lose information in the computation. To tackle this problem, we have decided to replace the ReLU by the Leaky ReLU function which has a small slope to process the negative values instead of completely replaced them by zero.

6.5.1 New results from the modification

After modification, the new EB-Net has been applied to our dataset in the same protocol as the last work. The trial has been lead on all parts of the beetle. Tables 6.5, 6.6, 6.7, 6.8, 6.9 present the comparison of the obtained results from the two models. The first column indexes the landmark number. Column **A** reminds the previous average distances (from the fine-tuning process with the model of three EBs). Column **B** presents the new obtained mean values. Column **C** shows the improvement percentages of the new results. The green/red numbers represent the best/worst values in each column. First of all, the results have been improved in all cases. Sometimes, the gain is pretty 1 pixel and more generally 0.5 pixels. Of course, the enhancement of each position is different, but we can see that the improvement most often occurs at the positions which are considered as being a little bit difficult to predict (following the previous results). For example, the new composition performs to almost positions of head images. It has also affected to the 6^{th}, 7^{th}, 8^{th} landmarks on pronotum; the 6^{th}, 7^{th}, 8^{th}, and 9^{th} landmarks of elytra; the 2^{nd}, 14^{th} positions on left mandible; and the 4^{th}, 18^{th} positions of the right mandible.

Finally, to be completed, we illustrate the distribution of the distances obtained from the new version of EB-Net. As usual, we consider the representation of the best and worst cases. One can note that all the distances between the manual and predicted

LM	**With 3 EBs (A)**	**With 4 EBs (B)**	**% improv. (C)**
1	2.79	2.67	4.32
2	3.19	3.00	5.88
3	**2.69**	2.61	2.93
4	3.28	3.24	1.06
5	3.07	2.92	4.75
6	**3.74**	**3.39**	9.29
7	2.72	**2.43**	10.82
8	3.35	2.97	11.36

Table 6.5: Pronotum images.

LM	**3 E-Blocks (A)**	**4 E-Blocks (B)**	**% impr. (C)**
1	**4.62**	**4.30**	6.85
2	3.86	3.38	12.15
3	4.42	3.94	10.90
4	3.68	3.15	14.41
5	3.94	3.51	11.03
6	3.06	2.66	13.01
7	3.66	3.14	14.31
8	**3.03**	**2.64**	13.14
9	3.89	3.52	9.52
10	3.50	3.14	10.29

Table 6.6: Head images

LM	**3 E-Blocks (A)**	**4 E-Blocks (B)**	**% impr. (C)**
1	2.70	2.47	8.65
2	2.74	2.55	6.85
3	2.74	2.67	2.38
4	**2.68**	**2.43**	9.44
5	2.87	2.75	4.12
6	3.80	3.39	10.81
7	4.09	3.66	10.48
8	**4.33**	**3.87**	10.54
9	4.12	3.65	11.38
10	2.78	2.86	-3.08
11	2.73	2.66	2.51

Table 6.7: Elytra images

landmarks have been improved (reduction) in the new composition of EB-Net. Particularly, we can observe from the charts that the weak points (far from the mean line) in the previous results are improved, they are more close to the average values in the new results. These distributions of all pieces are presented in Appendix D.

6.5.2 Future works

From these comparisons, we can observe that changing the activation function and adding one elementary block has improved the obtained results even the increase of the network's size is not large. As we have mentioned, we did not want to join a disputa-

LM	**3 E-Blocks (A)**	**4 E-Blocks (B)**	% **impr. (C)**
1	4.38	4.43	-1.02
2	3.28	3.03	7.61
3	3.20	3.03	5.42
4	**3.16**	**2.94**	6.86
5	3.27	3.12	4.42
6	3.21	3.14	1.98
7	3.39	3.28	3.20
8	3.50	3.36	3.94
9	3.70	3.52	4.89
10	3.88	3.77	2.69
11	4.28	4.04	5.58
12	4.36	4.13	5.25
13	3.75	3.52	6.23
14	3.51	3.14	10.33
15	3.84	3.62	5.84
16	**5.28**	**4.94**	6.48

Table 6.8: Left mandible images.

LM	**3 E-Blocks (A)**	**4 E-Blocks (B)**	% **impr. (C)**
1	4.09	4.02	1.73
2	3.52	2.98	15.3
3	**3.2**	**2.87**	10.41
4	3.31	2.87	13.36
5	3.31	2.95	10.93
6	3.42	3.16	7.73
7	3.31	3.13	5.43
8	3.36	3.11	7.58
9	3.59	3.38	5.89
10	3.64	3.42	6.14
11	3.97	3.77	5.11
12	4.05	3.94	2.68
13	4.63	**4.52**	2.39
14	4.47	4.33	3.09
15	4.06	3.83	5.8
16	3.61	3.31	8.52
17	3.98	3.68	7.5
18	**5.00**	4.29	14.19

Table 6.9: Right mandible images.

The comparison of average distances on each landmark of beetle's parts. The results are obtained from different processes on two versions of EB-Net.

tion between the size of the network and the achieved results. However, changing the composition of the network by adding a new EB in our case has given better results to us. The coordinates of predicted landmarks are more close to the manual ones. As a perspective, we would like to test that could we achieve better results when adding one or two new blocks if we have the computing resources.

It is worth to note that our block is a generic one. It is easy to add it, easy to test the model, even to remove a block from the model if the results are not satisfying. The users can choose suitable blocks for their applications. Choosing the number of blocks depends on the computing resources, as well as the expected results. As usual, we need to note that it exists a balance between the depth of the model, the accuracy of outputs and the cost to compute.

As perspectives, we would like to introduce our model for applying on other applications, e.g., different applications in biology or predicting landmarks for human pose detection which are studied topics in our team. Moreover, the pre-trained model of EB-Net is existed from now, available freely on Github [3], the users can use it to apply on other applications of landmarking.

6.6 Conclusion

In this chapter, we have presented a complementary approach of deep learning, transfer learning for the landmarking problem. We have also described a process to apply fine-tuning, a specific strategy of transfer learning, for improving the performance of automating landmarks on beetles images. In this step, we have pre-trained our network, EB-Net, on a facial key points dataset before transferring the parameters to fine-tune on beetle images.

We have examined another composition of EB-Net by adding a new elementary block. We have also replaced the activation functions to LeakyReLU. The modification of the model has achieved our works on beetle images. We are now able to provide a set of predicted landmarks, which is good enough to replace manual ones in the statistical point of view, and to display them on the images in a user-end application. So, we have distributed the final results to the biologists, and they have confirmed that our estimation is good enough to use in the morphometry analysis.

[3] https://github.com/linhlevandlu/CNN_Beetles_Landmarks

Conclusion

This Ph.D. research has figured out the methods to automatically predict landmarks in 2D biological images. The application has been made on beetle images coming from a dataset of the Demecology team (INRA, Rennes). In the first step, the studies focused on the beetle's mandibles. In this context, we have first studied and tested solutions found in the literature, as the detection of landmarks on wing images of Drosophila wings with the help of Probabilistic Hough Transform computation associated with a template matching process. The first results with this method attest the feasibility of this research, but the outputs need to be improved. We have proposed another pipeline of operations composed of a segmentation step and an iteration of registration step. The estimations of landmarks have been then achieved by applying a SIFT method. The results have been considered good enough to use these landmarks for morphometric operations required by biologists. Unfortunately, running our pipeline on the three other parts of the beetle: pronotum, head, and elytra, has come out the poorest results. The pictures of these parts have been taken before dissection, and they stay stuck together. To deal with this problem, we have chosen to turn to another approach coming from deep learning algorithms, which do not need the segmentation to process these parts.

Accordingly, a CNN model has been proposed, named EB-Net. The EB-Net architecture is a combination of three Elementary Blocks, EB. Each Elementary Block is a composite of three classical layers: convolutional, maximal pooling, and dropout layers. After several working processes with EB-Net architecture, the final results have been obtained with the help of a fine-tuning process, i.e., transfer learning from a facial

landmarks analysis. The outcomes have provided the estimated landmarks, which are most often in an area more or less than three pixels around the manual ones.

The final results have been delivered to biologists. They have confirmed that the quality of predicted landmarks is statistically good enough to replace the manual landmarks for the different morphometry analysis, and to be displayed them on an end-user interface. In order to apply our framework to a large set of problems in morphometry, we plan to test different datasets owned by the INRA team. All the implementations, both IMEL and EB-Net, are available freely on the Github website. It is possible to reuse EB-Net parameters for another landmark setting application and to apply transfer-learning. We plan also to export EB-Net architecture to other application domains: MRI images analysis, pose identification, . . . , studied in our team.

Bibliography

[Ama93] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[Art08] Nicole M Artner. A comparison of mean shift tracking methods. In *12th Central European Seminar on computer graphics*, volume 23. Citeseer, 2008.

[ASWC13] Robert Anderson, Bjorn Stenger, Vincent Wan, and Roberto Cipolla. Expressive visual text-to-speech using active appearance models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3382–3389, 2013.

[ATRB95] Anthony Ashbrook, Neil A Thacker, Peter I Rockett, and CI Brown. Robust recognition of scaled shapes using pairwise geometric histograms. In *BMVC*, volume 95, pages 503–512, 1995.

[AZCP13] Akshay Asthana, Stefanos Zafeiriou, Shiyang Cheng, and Maja Pantic. Robust discriminative response map fitting with constrained local models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3444–3451, 2013.

[BAPD13] Xavier P Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark estimation under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1513–1520, 2013.

[BB12] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[BBBK11] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[BF70] Claude R Brice and Claude L Fennema. Scene analysis using regions. *Artificial intelligence*, 1(3-4):205–226, 1970.

[BFL06] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[Bha43] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[BJ01] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[BJKK13] Peter N Belhumeur, David W Jacobs, David J Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2930–2940, 2013.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[Boo97] Fred L Bookstein. *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press, 1997.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[Bro12] Bruce Brown. Permutation tests for complex data: Theory, applications and software by f. pesarin and l. salmaso. *Australian & New Zealand Journal of Statistics*, 54(1):126–127, 2012.

[Bru09] Roberto Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[Can87] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[Che95] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[CILS12] Tim F Cootes, Mircea C Ionita, Claudia Lindner, and Patrick Sauer. Robust and accurate shape model fitting using random forest regression voting. In *European Conference on Computer Vision*, pages 278–291. Springer, 2012.

[CM97] Dorin Comaniciu and Peter Meer. Robust analysis of feature spaces: color image segmentation. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 750–755. IEEE, 1997.

[CM99] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1197–1203. IEEE, 1999.

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[Cor97] Philippe Cornic. Another look at the dominant point detection of digital curves. *Pattern Recognition Letters*, 18(1):13–25, 1997.

[CP80] PC Chen and T Pavlidis. Image segmentation as an estimation problem. *Computer Graphics and Image Processing*, 12(2):153–172, 1980.

[CPFGMCMC05] A Carmona-Poyato, Nicolás Luis Fernández-García, Rafael Medina-Carnicer, and Francisco José Madrid-Cuevas. Dominant point detection: A new proposal. *Image and Vision Computing*, 23(13):1226–1236, 2005.

[CPK+17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[CQSA+16] Celia Cintas, Mirsha Quinto-Sánchez, Victor Acuña, Carolina Paschetta, Soledad De Azevedo, Caio Cesar Silva de Cerqueira, Virginia Ramallo, Carla Gallo, Giovanni Poletti, Maria Catira Bortolini, et al. Automatic ear detection and feature extraction using geometric morphometrics and convolutional neural networks. *IET Biometrics*, 6(3):211–223, 2016.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CWB+11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[CYSC03] Shantanu Chakrabartty, Masakazu Yagi, Tadashi Shibata, and Gert Cauwenberghs. Robust cephalometric landmark identification using support vector machines. In *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, volume 3, pages III–429. IEEE, 2003.

[D+15] Sander Dieleman et al. Lasagne: First release., August 2015.

[dAS08] António dos Anjos and Hamid Reza Shahbazkia. Bi-level image thresholding. *Biosignals*, 2:70–76, 2008.

[DC00] Stanislas Dehaene and Jean-Pierre Changeux. Reward-dependent learning in neuronal networks for planning and decision making. In *Progress in brain research*, volume 126, pages 217–229. Elsevier, 2000.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[DK32] Harold Edson Driver and Alfred Louis Kroeber. *Quantitative expression of cultural relationships*, volume 31. University of California Press, 1932.

[DM92] IL Dryden and KV Mardia. Size and shape analysis of landmark data. *Biometrika*, 79(1):57–68, 1992.

[Dry14] Ian L Dryden. Shape analysis. *Wiley StatsRef: Statistics Reference Online*, 2014.

[DT17] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[DV16] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[ETM93] Alun Evans, Neil A Thacker, and John EW Mayhew. The use of geometric histograms for model-based object recognition. In *BMVC*, volume 93, pages 429–438, 1993.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[FCNL13] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[FH75] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

[FP10] Leila Favaedi and Maria Petrou. Cephalometric landmarks identification using probabilistic relaxation. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 4391–4394. IEEE, 2010.

[Fre61] Herbert Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, (2):260–268, 1961.

[Fri98] Jerome H Friedman. Data mining and statistics: What's the connection? *Computing Science and Statistics*, 29(1):3–9, 1998.

[Fuk80] Youji Fukada. Spatial clustering procedures for region analysis. *Pattern Recognition*, 12(6):395–403, 1980.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GBR+99] Igor D Grachev, Dmitriy Berdichevsky, Scott L Rauch, Stephan Heckers, David N Kennedy, Verne S Caviness, and Nathaniel M Alpert. A method for assessing the accuracy of intersubject registration of the human brain using anatomic landmarks. *NeuroImage*, 9(2):250–268, 1999.

[GD+04] John C Gower, Garmt B Dijksterhuis, et al. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004.

[GDDM15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[GEB15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[GEC+13] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning:

A report on three machine learning contests. In *International Conference on Neural Information Processing*, pages 117–124. Springer, 2013.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[GG15] Jian Guo and Stephen Gould. Deep cnn ensemble with data augmentation for object detection. *arXiv preprint arXiv:1506.07224*, 2015.

[GGS+98] A Garrido, M Garcia-Silvente, et al. Boundary simplification using a multiscale dominant-point detection algorithm. *Pattern Recognition*, 31(6):791–804, 1998.

[Gos05] Arthur Ardeshir Goshtasby. *2-D and 3-D image registration: for medical, remote sensing, and industrial applications*. John Wiley & Sons, 2005.

[Gow01] John C Gower. Procrustes analysis. 2001.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[GS84] Stephen F Gull and John Skilling. Maximum entropy method in image processing. In *IEE Proceedings F (Communications, Radar and Signal Processing)*, volume 131, pages 646–659. IET, 1984.

[GW+02] Rafael C Gonzalez, Richard E Woods, et al. Digital image processing [m]. *Publishing house of electronics industry*, 141(7), 2002.

[GWK+18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei

Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[H+12] Geoffrey Hinton et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[HGT17] Shaoli Huang, Mingming Gong, and Dacheng Tao. A coarse-fine network for keypoint localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3028–3037, 2017.

[HHW19] Yuehui Han, Yusheng Hao, and Weilan Wang. Research on the character recognition of tibetan ancient document based on cnn. In *Journal of Physics: Conference Series*, volume 1237, page 032052. IOP Publishing, 2019.

[HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[HM95] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[HMGC03] David Houle, Jason Mezey, Paul Galpern, and Ashley Carter. Automated measurement of drosophila wings. *BMC evolutionary biology*, 3(1):25, 2003.

[Hou62] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[HSK+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[HYZ07] Yiping Hong, Jianqiang Yi, and Dongbin Zhao. Improved mean shift segmentation approach for natural images. *Applied mathematics and computation*, 185(2):940–952, 2007.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JC02] Hans J Johnson and Gary E Christensen. Consistent landmark and intensity-based image registration. *IEEE transactions on medical imaging*, 21(5):450–461, 2002.

[JCMB14] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[KBA08] Jens N Kaftan, Andre A Bell, and Til Aach. Mean shift segmentation-evaluation of optimization techniques. In *VISAPP (1)*, pages 365–374, 2008.

[KEB91] Nahum Kiryati, Yuval Eldar, and Alfred M Bruckstein. A probabilistic hough transform. *Pattern recognition*, 24(4):303–316, 1991.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[KS+04] Yan Ke, Rahul Sukthankar, et al. Pca-sift: A more distinctive representation for local image descriptors. *CVPR (2)*, 4:506–513, 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LBAZP18] Van-Linh Le, Marie Beurton-Aimar, Akka Zemmari, and Nicolas Parisey. Landmarks detection by applying deep networks. In *2018 1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, pages 1–6. IEEE, 2018.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LBL+12] Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.

[LCC+01] Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung, et al. A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17(5):713–727, 2001.

[Lin15] Tony Lindeberg. Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, 52(1):3–36, 2015.

[LKF10] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS),*

Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[LLS+15] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[Low87] David G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial intelligence*, 31(3):355–395, 1987.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LRB15] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[LV12] Thai Hoang Le and Truong Nhat Vo. Face alignment using active shape model and support vector machine. *arXiv preprint arXiv:1209.6151*, 2012.

[LVBAKP17] Linh Le Van, Marie Beurton-Aimar, Adrien Krahenbuhl, and Nicolas Parisey. Maelab: a framework to automatize landmark estimation. 2017.

[LVBAS+16] L Lê Vãnh, M Beurton-Aimar, Jean-Pierre Salmon, Alexia Marie, and N Parisey. Estimating landmarks on 2d images of beetle mandibles. 2016.

[LYL+16] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Fashion landmark detection in the wild. In *European Conference on Computer Vision*, pages 229–245. Springer, 2016.

[LZS16] Shilun Lin, Zhicheng Zhao, and Fei Su. Homemade ts-net for automatic face recognition. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 135–142. ACM, 2016.

[M+67] 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.

[M+11] Tomáš Mikolov et al. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011.

[MCB13] Pedro Martins, Rui Caseiro, and Jorge Batista. Generative face alignment through 2.5 d active appearance models. *Computer Vision and Image Understanding*, 117(3):250–268, 2013.

[MH80] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[MM92] Farzin Mokhtarian and Alan K Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):789–805, 1992.

[MNR90] Rajiv Mehrotra, Sanjay Nichani, and Nagarajan Ranganathan. Corner detection. *Pattern recognition*, 23(11):1223–1233, 1990.

[Mok95] Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):539–544, 1995.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MS04] Majed Marji and Pepe Siy. Polygonal representation of digital planar curves through dominant point detection—a nonparametric algorithm. *Pattern Recognition*, 37(11):2113–2130, 2004.

[MST+94] Donald Michie, David J Spiegelhalter, CC Taylor, et al. Machine learning. *Neural and Statistical Classification*, 13, 1994.

[MVBP12] Brais Martinez, Michel F Valstar, Xavier Binefa, and Maja Pantic. Local evidence aggregation for regression-based facial point detection. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1149–1163, 2012.

[MZ92] Stephane Mallat and Sifen Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):710–732, 1992.

[NB02] Kenneth Nilsson and Josef Bigun. Prominent symmetry points as landmarks in fingerprint images for alignment. In *Object recognition supported by user interaction for service robots*, volume 3, pages 395–398. IEEE, 2002.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the*

IEEE international conference on computer vision, pages 1520–1528, 2015.

[NNVW15] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*, pages 443–449. ACM, 2015.

[NVG06] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[OP99] Timo Ojala and Matti Pietikäinen. Unsupervised texture segmentation using feature distributions. *Pattern recognition*, 32(3):477–486, 1999.

[PPJ08] Unsang Park, Sharath Pankanti, and Anil K Jain. Fingerprint verification using sift features. In *Biometric Technology for Human Identification V*, volume 6944, page 69440K. International Society for Optics and Photonics, 2008.

[Pre70] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[PTK10] Sasirekha Palaniswamy, Neil A Thacker, and Christian Peter Klingenberg. Automatic identification of landmarks in digital images. *IET Computer Vision*, 4(4):247–260, 2010.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[Qui87] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[RN16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[Rob63] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[Roh04] FJ Rohlf. Tpsdig. *Department of Ecology and Evolution, State University of New York, Stony Brook, NY*, 2004.

[Roh05] F James Rohlf. tpsdig, digitize landmarks and outlines, version 2.05. *Department of Ecology and Evolution, State University of New York at Stony Brook*, 2005.

[RR92] Bimal Kr Ray and Kumar S Ray. Detection of significant points and polygonal approximation of digitized curves. *Pattern Recognition Letters*, 13(6):443–452, 1992.

[S+85] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[SACF+12] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source

platform for biological-image analysis. *Nature methods*, 9(7):676, 2012.

[Sam88] Arthur L Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. In *Computer Games I*, pages 366–400. Springer, 1988.

[SB+98] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[SDP13] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable medical image registration: A survey. *IEEE transactions on medical imaging*, 32(7):1153, 2013.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[SK17] John Howison Schroeder and Alexander D Kiderman. Method of precision eye-tracking through use of iris edge based landmarks in eye geometry, May 23 2017. US Patent 9,655,515.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[SKPS11] Vavilis Sokratis, Ergina Kavallieratou, Roberto Paredes, and Kostas Sotiropoulos. A hybrid binarization technique for document images. In *Learning Structure and Schemas from Documents*, pages 165–179. Springer, 2011.

[SLBW13] Xiaohui Shen, Zhe Lin, Jonathan Brandt, and Ying Wu. Detecting and aligning faces by image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3460–3467, 2013.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[Sob14] Irwin Sobel. History and definition of the sobel operator. *Retrieved from the World Wide Web*, 1505, 2014.

[SRG+16] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[SVP+15] Anne Sonnenschein, David VanderZee, William R Pitchers, Sudarshan Chari, and Ian Dworkin. An image database of drosophila melanogaster wings for phenomic and biometric analysis. *GigaScience*, 4(1):25, 2015.

[SWS17] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[SWT13] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3476–3483, 2013.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[SZW+18] Tiezhu Sun, Wei Zhang, Zhi-Jie Wang, Lin Ma, and Zequn Jie. Image-level to pixel-wise labeling: From theory to practice. In *IJCAI*, pages 928–934, 2018.

[TC89] C-H Teh and Roland T. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on pattern analysis and machine intelligence*, 11(8):859–872, 1989.

[TLL16] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2759–2764. IEEE, 2016.

[TRY95] Neil A Thacker, PA Riocreux, and RB Yates. Assessing the completeness properties of pairwise geometric histograms. *Image and Vision Computing*, 13(5):423–429, 1995.

[TS09] Lisa Torrey and Jude Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[UFH12] Michal Uřičář, Vojtěch Franc, and Václav Hlaváč. Detector of facial landmarks learned by the structured output svm. *VIsAPP*, 12:547–556, 2012.

[VCFR09] M Villalobos-Castaldi and Edgardo M Felipe-Riverón. Fast automatic retinal vessel segmentation and vascular landmarks extraction method for biometric applications. In *IEEE International Conference on Biometrics, Identity and Security*, volume 54, 2009.

[VMK+16] Max A Viergever, JB Antoine Maintz, Stefan Klein, Keelin Murphy, Marius Staring, and Josien PW Pluim. A survey of medical image registration–under review, 2016.

[WD] M. Abramowitz W. Davidson. Molecular expressions microscopy primer: Digital image processing - difference of gaussians edge enhancement algorithm. *Olympus America Inc., and Florida State University Michael W*.

[WGT+18] Nannan Wang, Xinbo Gao, Dacheng Tao, Heng Yang, and Xuelong Li. Facial feature point detection: A comprehensive survey. *Neurocomputing*, 275:50–65, 2018.

[WQL+18] Shuang Wang, Dou Quan, Xuefeng Liang, Mengdan Ning, Yanhe Guo, and Licheng Jiao. A deep learning framework for remote sensing image registration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:148–164, 2018.

[WS10] MARK Webster and H David Sheets. A practical introduction to landmark-based geometric morphometrics. *The Paleontological Society Papers*, 16:163–188, 2010.

[Wu03] Wen-Yen Wu. Dominant point detection using adaptive bending value. *Image and Vision Computing*, 21(6):517–525, 2003.

[Yak73] Yoram Yakimovsky. Scene analysis using a semantic base for region growing. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1973.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[YH09] Jae-Chern Yoo and Tae Hee Han. Fast normalized cross-correlation. *Circuits, systems and signal processing*, 28(6):819, 2009.

[YKSN17] Xiao Yang, Roland Kwitt, Martin Styner, and Marc Niethammer. Quicksilver: Fast predictive image registration–a deep learning approach. *NeuroImage*, 158:378–396, 2017.

[YM12] Faliu Yi and Inkyu Moon. Image segmentation: A survey of graph-cut methods. In *2012 International Conference on Systems and Informatics (ICSAI2012)*, pages 1936–1941. IEEE, 2012.

[YP13] Heng Yang and Ioannis Patras. Sieving regression forest votes for facial feature detection in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1936–1943, 2013.

[Z+14] Zhanpeng Zhang et al. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[ZA15] Nida M Zaitoun and Musbah J Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015.

[ZCG+19] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

[ZF03] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[ZGS17] S Kevin Zhou, Hayit Greenspan, and Dinggang Shen. *Deep learning for medical image analysis*. Academic Press, 2017.

[Zok04] Siavash Zokai. *Robust image registration using log-polar transforms*. PhD thesis, City University of New York, 2004.

[ZSCC13] Xiaowei Zhao, Shiguang Shan, Xiujuan Chai, and Xilin Chen. Cascaded shape space pruning for robust facial landmark detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1033–1040, 2013.

[ZSS12] Miriam Leah Zelditch, Donald L Swiderski, and H David Sheets. *Geometric morphometrics for biologists: a primer*. Academic Press, 2012.

[ZY96] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):884–900, 1996.

Appendix A

Landmarks estimation by Probabilistic Hough Transform

As mentioned in Chapter 3, this work was initialized from my Master's degree internship by studying a method of Palaniswamy [PTK10] and applying to our application. This is a method including four steps: (1) segmentation to extract the contour; (2) conversation to convert the features to an invariant form for comparing the objects; (3) registration to find the match between two objects and prediction of hypothesized landmarks; (4) verification to verify the coordinates of estimated landmarks. The thesis begins by adjusting their steps to improve the results. In this chapter, we present the implementations of steps of the method that we have studied.

As usual, segmentation is the first step of the method to extract the object's features. In this case, the contour points have been taken into account. In order to select the shape of mandibles, we have decided to apply a combination of binary threshold and Canny algorithm [Can87]. The details of segmentation step are described in Section 3.2.

I . Features conversion

As mentioned in [PTK10], the geometric relationship between lines is a useful feature for shape representation. Therefore, lists of points come from the segmentation step are turned to sets of approximated lines by using a recursive algorithm [TRY95], that is a new improved version of Lowe's method [Low87]. In [TRY95], the algorithm stops when the relative between measured distances (d_{max} and the length of line l) is less than an approximate threshold. However, to save computing time and to implement easily, we have modified this condition by only comparing the maximum distance (d_{max}) to the threshold λ received from the experiments. Steps of the recursive algorithm are described as follows:

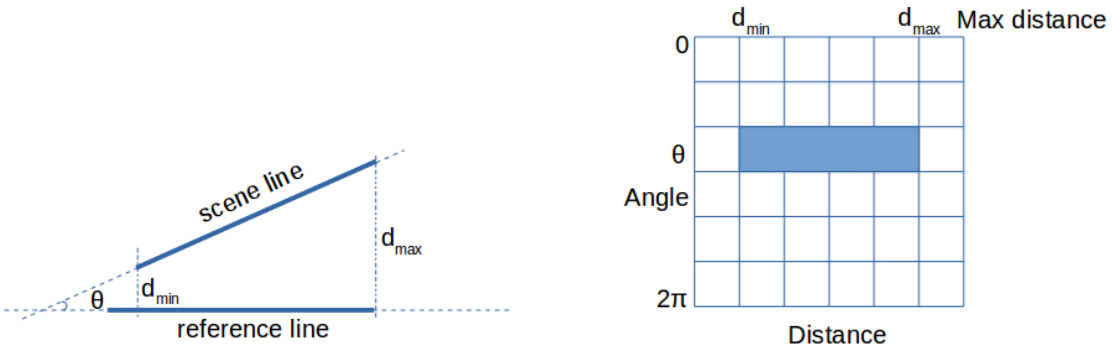
1. Creating a straight line l between two endpoints of the edge
2. Calculating the perpendicular distance from each point in the edge to the line l. Then, identifying the point p_m which has the maximum distance (d_{max}) to l.
3. If the perpendicular distance from p_m is greater than a threshold value ($d(p_m, l) > \lambda$), then the edge is split at this point (p_m) into two parts, and the procedure continues on both two split parts. Otherwise, the edge can be represented by l.

II . Pairwise Geometric Histogram

In order to compare objects, it is necessary to encode their features into the compact and invariant forms, e.g., Pairwise Geometric Histogram (PGH). It is worth to note that the shape of any object can be represented by a set of approximated lines, then the geometric relationships between lines can be used to built PGH. In this case, the relative angle and the perpendicular distances are chosen to provide an efficient description of the object: the relative angle is defined by the angle between two lines, and perpendicular distances are the distances from two endpoints of a line to the extension of another

one (Figure A.1a). Notably, these features are based on absolute distance and they are sensitive to the scale. Fortunately, this problem could be solved by normalizing the perpendicular distances.

In practical, a PGH is represented as a two-dimensional matrix. One dimension presents the relative angle $(0 - 2\pi)$, and other dimension outlines the perpendicular distance (Figure A.1b). Each PGH stores the geometric relations between a line, called *reference line*, and other lines presented the object, named *scene lines*. Firstly, the relative angle and the perpendicular distance between reference and a scene line are computed. Then, these values will be recorded in the PGH matrix. This process is repeated until all scene lines have been considered. At the end of the process, the blur of the entries in the PGH histogram presents the true position and orientation of object's lines. An object, therefore, has a set of PGHs associated with it, one histogram for each line [ETM93].



a . Geometric features. b . Pairwise geometric histogram.

Figure A.1: Geometrics features between two lines and their pairwise geometric histogram. Left figure: the relative position between two lines; right figure: the PGH of two lines in the left figure.

In image processing, shape recognition mentions identifying the similarity of two images features. This work can be achieved by comparing their histograms. Palaniswamy has chosen Bhattacharya metric [Bha43] to measure the similarity of two histograms. This metric provides a quantitative estimation of the likeness between the source and the target features as a dot product correlation of the histogram of lines. The form of

Bhattacharya is following.

$$d_{Bhatt}(H_i, H_j) = \sum_{\theta}^{2\pi} \sum_{d}^{d_{max}} \sqrt{H_i(\theta, d).H_j(\theta, d)} \tag{A.1}$$

In which:

- H_i, H_j: are the histograms of image i and image j
- θ: is angle value ($\theta \in [0, 2\pi]$)
- d: is the perpendicular distance
- $H(\theta, d)$: is an entry at row θ, column d in histogram

In our application, the feature conversion step has outputted lists of lines presented the mandible object. In this step, these lines are used to build the PGHs for the object as described. Then, the PGHs are used to examine the correspondence between two mandibles by calculating the Bhattacharya metric.

III . Probabilistic Hough Transform

At this step, Probabilistic Hough Transform (PHT) [KEB91] is applied to discover the presence and the location of the target object in the source image, as well as find out the hypothetical coordinates of target's landmarks. Applying PHT can be divided into two steps: firstly, confirming the appearance of the target object in the source image by finding a pair of target lines which is the best matching with a couple of source lines; secondly, indicating the pose of the target object on the source image. At this moment, the positions of the source's landmarks are considered as the hypothetical landmarks of the target image.

Firstly, the relative information tables for both the source and the target object have been created, namely T_s and T_t, respectively. To build a related information table for an image, an arbitrary point (in the image) is selected and considered as the origin point. Then, for each time we examine a couple of lines, the angles and the perpendicular

distances between lines and the origin point are computed: the angles are equals to the angles of two lines and the horizontal axis which begins from the origin point; the perpendicular distances are the distances from the origin point to each line. Then, the information is recorded in the table. This process is repeated until considering all distinct pair of object's lines.

After that, to find the best matching of couples of lines between two objects, an accumulator is created as a two-dimensional matrix represents the angle and the perpendicular distance information. All the cells in the accumulator are set to zero. For each pair of source's lines, if existing a couple of target's lines that corresponds to position, orientation, selecting their information (angle and perpendicular distance) from the table T_t. Then, a vote is carried out on the accumulator at the corresponding cell with the selected information by adding one to the cell's value. At the end of the voting process, couples of lines (both in source and target image) which have the highest value will be kept.

The next step concerns retreiving the presence of the source's origin point in the target image by using the relative information of the best matching couples of lines: firstly, extracting the relative information of the pair of source's lines from T_s. Then, indicating the source's origin point on the target image by extending lines which are perpendicular with the two of target's lines at the appropriate position [ATRB95].

After obtaining the location of the source's origin in the target image, it is necessary to register two origin points. Then, the source's manual landmarks are set by using the relatedness among them and the source's origin point. These points are assumed to be the hypothetical positions of the predicted landmarks of the target's image. Besides, the angle deviation between the two images is also recorded to use in the next step.

IV . Template Matching

The PHT step has computed the estimated landmarks on target image based on the corresponding features of global shape. To refine the location of estimated landmarks,

a cross-correlation method [Bru09], *template matching*, is used for the refinement of the coordinates of predicted landmarks.

As mentioned in Section 2.3.3, the template matching process needs *a template* and *a search image*. In this case, a template is a small patch around the landmark on the source image and the search image is the target image. However, the number of candidates in the target image is numerous. It is necessary to limit the search region for saving the computing time. It is worth to note that the previous step provided the estimation of landmarks positions in the target image. These positions can be used to reduce the searching areas correspondence with the patches. As mentioned in Section 1.5, the mandibles could be placed in different poses during the process to capture the images. So, to obtain the complete matching, the target image is rotated to match with the source image by using the angle which was extracted from the PHT step. This process finishes when all estimated landmarks are refined.

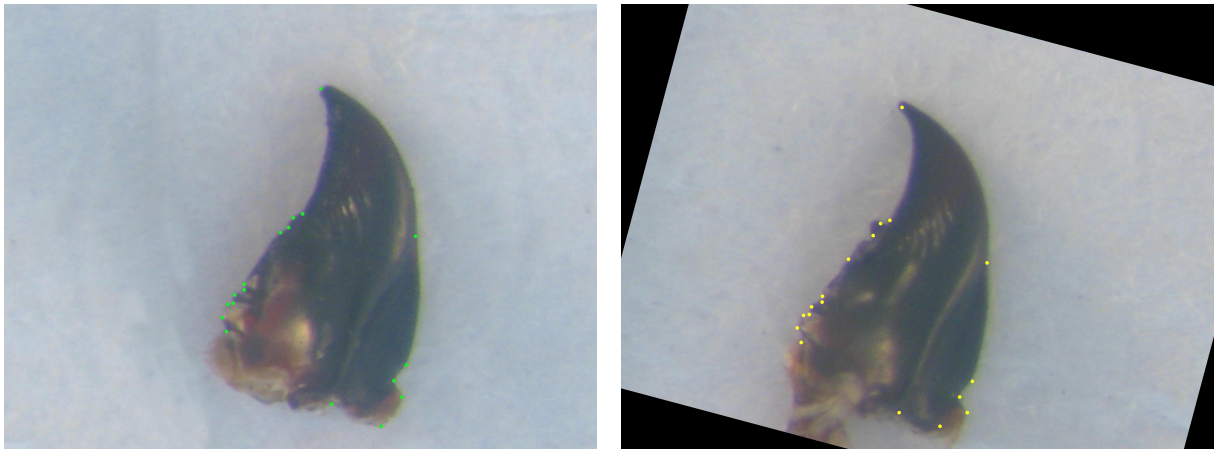
V . Results

As we have mentioned, this first work has focused on mandible parts. The two sets of mandible images have been selected to evaluate the accuracy, reliability, and robustness of the Palaniswamy's method. After verifying the correctness of the images, it remains 290 right and 286 left mandible image by removing the pictures which are empty or contains broken objects.

All the steps of the method have been implemented in MAELab framework [1]. The program is designed to extract 18 landmarks on right mandible images and 16 landmarks on left mandible images. The automated landmarks are indicated based on the learning from the manual landmarks of source image which have been provided by biologists. Figure A.2 presents a prediction sample on the right mandible. The left image (Figure A.2a) shows the source model and its manual landmarks (cyan points); the right image (Figure A.2b) illustrates the target image and its automated landmarks

[1] MAELab is a free software written in C++. The source code is available on Github.

(yellow points). The target image has been rotated to match with the model image before setting predicted landmarks.



a . Source image and its manual landmarks. b . Target image and its automated landmarks.

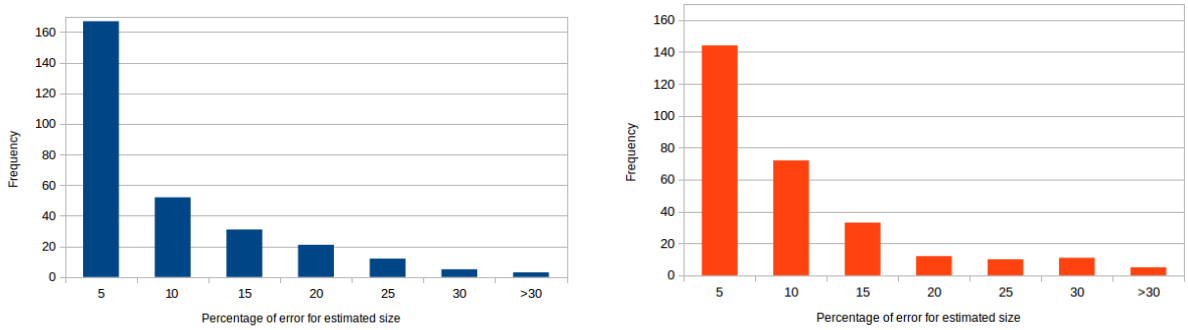
Figure A.2: The automated landmarks on a target image which are estimated from a source image and its manual landmarks.

To figure out the results, biologists have chosen to use centroid size to measure the mandibles. Firstly, indicating the center point for each mandible. The coordinates of the center point are considered as the mean values of the coordinates of all landmarks. Then, calculating the size measure of the mandible as the sum of all square distance from each event to the center point. In that way, we have compared the centroid sizes which are computed from manual landmarks and the corresponding from estimated landmarks. The percentage of errors has been evaluated as Equation A.2:

$$Percent\_Of\_Error = \frac{100 * |(Original\_Size - Estimated\_Size)|}{Original\_Size} \quad \text{(A.2)}$$

Figure A.3 shows the percentage of error from estimated landmarks on both left and right mandible images. For right mandibles, we have obtained more than 160 images of less than 5% of errors between manual and estimated sizes. Only two right mandibles are more than 30% of errors. These cases could be considered as the wrong predictions in the right mandibles (Figure A.3a). In the part of left mandibles, we have got more than 140 images of less than 5% of errors. Only five pictures are more than 30% of errors

(Figure A.3b). Finally, 90% of images (both in left and right mandibles) have less than 10% of errors in their size computing.



a . Right mandible. b . Left mandible.

Figure A.3: Percentage of error in computing centroid size from estimated landmarks.

To compare the performance of Palaniswamy's method on our mandibles with their data (fly wing). We have computed the distances between manual landmarks and predicted ones. Then, we have considered the proportion of prediction with several ranges of acceptable distances. One can note that the size of the images is different between the two datasets. To be able to compare, the distance on each landmark of the mandible has been scaled to size provided by the article (1280×1022).

Pixel range	**No. of points**	**Proportion (%)**
≤ 5	36	0.784
≤ 10	106	2.308
≤ 15	156	3.397
≤ 20	252	5.488
≤ 30	426	9.277
≤ 50	789	17.182

a. Left mandible

Pixel range	**No. of points**	**Proportion (%)**
≤ 5	71	1.36
≤ 10	170	3.257
≤ 15	306	5.862
≤ 20	424	8.123
≤ 30	717	13.736
≤ 50	1236	23.678

b. Right mandible

Table A.1: The proportion of several ranges (in pixel) on the left mandibles set provided by Palaniswamy's method.

Table A.1 shows the results that we have obtained on the left and right mandibles, respectively. We can see that the results are at different levels on two datasets. Even

if we accept a distance of 50 pixels between the estimated and manual landmarks, we cannot reach to the results shown in the article. We have taken a look, analyzed the images, and seen that we have a big difference between two kinds of images. Landmarks on fly wings have mostly stayed at the intersection of veins, whereas these are not the same in mandible cases. It seems that the method is more suitable for wing fly than the mandibles.

Palaniswamy's method includes a sequence of algorithms, in order to improve the global result, we need to improve the results of each step in the process. Additionally, the values of the parameters could affect the results, for example, the window size during the template matching process. If the window size is small, the result of the verifying step has not a significant variation. In the opposite side, a larger window will influence the speed of the algorithm, which is important when processing the large numbers of images.

The obtained results have shown that the estimated landmarks are accurate enough to compute centroid sizes of mandibles. They have been also evaluated by comparing to the manual ones. However, these positions are not satisfied to display them on a user interface. The quality of predicted coordinates needs to improve both to optimize the computation time and to provide more realistic landmarks.

The obtained results in this section were published in a poster [LVBAS+16] and presented at the International Conference on Computer Graphics, Visualization and Computer Vision 2016.

Appendix B

MAELab and its functionalities

As mentioned in Chapter 3, all the methods to provide the automated landmarks on mandible images have been implemented in MAELab framework. This appendix will describe the modules, as well as the functions of MAELab.

I . Software architecture

MAELab software mainly provides the functionality for landmarking on 2D beetle's images, but it includes also the helper functions for other processes in image processing such as segmentation, binary operations. The main modules are displayed in Figure B.1. The functionality of each module is describing as follows:

- **Qt Framework** module: contains the classes inherited from Qt Framework, which provide the graphics interface to software.
- **pointInterest** module: includes the classes for automatic landmarking.
- **pht** module: realizes operations for the PHT process.
- **segmentation** module: implements the segmentation algorithms, for example threshold, Canny, Suzuki, and line segment algorithm.

- **imageModel** module: includes the classes to represent the data structure of image such as matrix, point, line.
- **io** module: contains the input and output functions of software. The input functions read the image into the matrix for computing. The output operations convert/save the results into output image. The capacities of this module are helped from LibJpeg module.
- **LibJpeg** module: uses to decode and to encode (read) JPGE image. This is a free library.

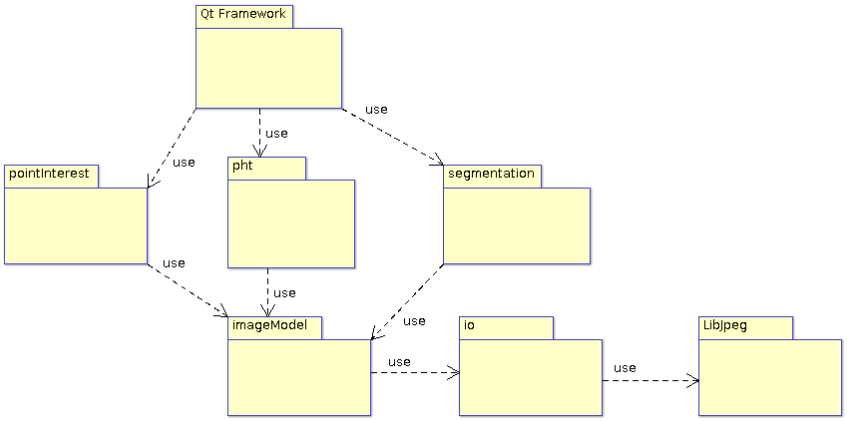


Figure B.1: The packages of MAELab software

II . MAELab interface

MAELab has been developed under two distributions: Graphics User Interface (GUI) and library package. The GUI version allows users to directly manipulate the functions of the software and see the results. Whereas, the users can use the library package as an add-on to a C++ application. All the versions are distributed on Github[1].

[1]https://github.com/linhlevandlu/MAELab2019

Figure B.2 shows the main window of MAELab framework in GUI mode. The main functions of MAELab are described as follows, but the users can find the demos on the web page of the project[2].

- Menu **File** contains the functions to open the source image, print the image, save the image and close the program.
- Menu **View** contains the functions to change the view modes of image in program.
- Menu **Segmentation** contains the functions to segment image.
- Menu **Process** provides the functions to filter image. It contains also the binary operations on image.
- Menu **Landmarks** provides the operations to automatically determine landmarks.
- Menu **Help** describes the information about the software.

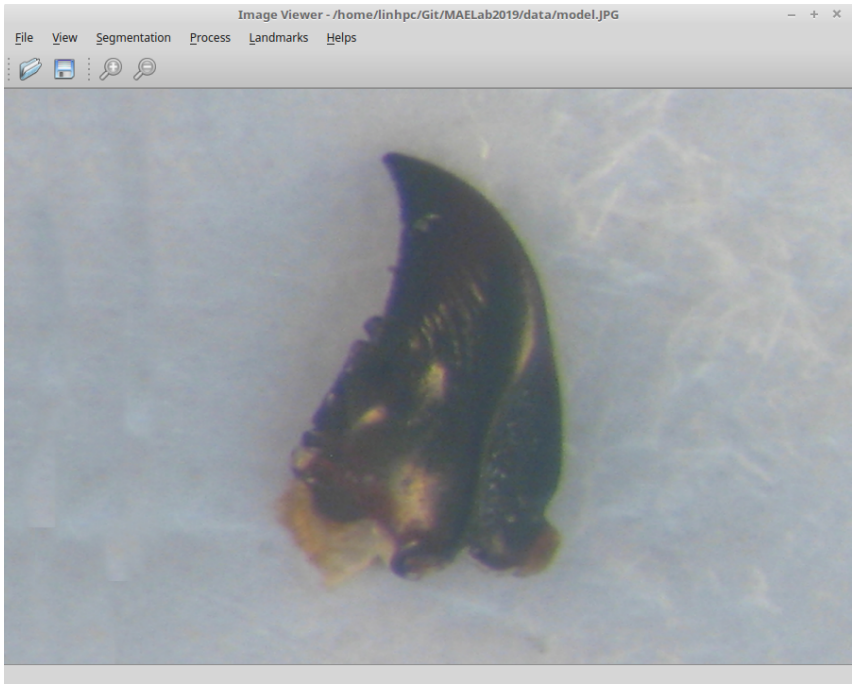


Figure B.2: The GUI of MAELab software

[2]https://morphoboid.labri.fr/devmap.html

Appendix C

Training EB-Net on remaining datasets: head and body

Chapter 5 has presented the results when we trained the EB-Net on pronotum images. These following sections describes the results on two remaining sets of images: head and elytra images, respectively. This chapter firstly figures out the parameters of the layers in the elementary blocks. As mentioned in Chapter 5, we have applied the cross-validation technique to select data. This makes we need to train EB-Net in several times, named rounds, with different subsets of data. So, we will present the losses of the rounds during the training process in the next part. Then, the mean distances will be discussed. It finishes with the illustrations of predicted landmarks on the images. This presented structure is applied to both head and elytra images.

I . The detailed parameters in EB-Net

As mentioned in Chapter 5, EB-Net is a combination of three elementary bocks, followed by 3 fully-connected layers and a dropout layer has been inserted between the first two convolutional layers (Figure C.1). In this architecture, the order of layers in the elementary blocks is the same, but their parameter values are different in each block.

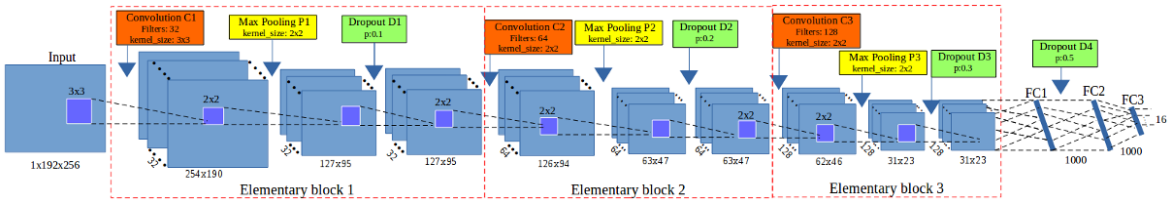


Figure C.1: The architecture of the third model

The parameter values for each layer are detailed as below, the list of values follow the order of the blocks ($i = [1..3]$):

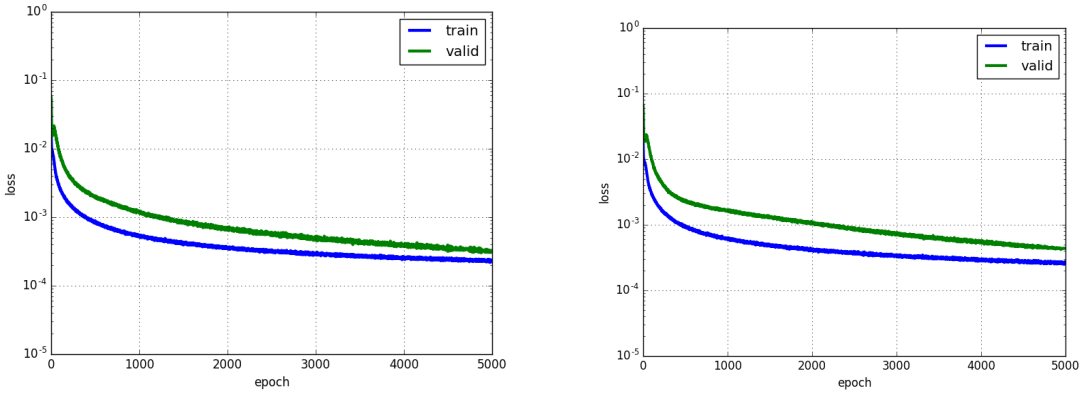
- CONV layers:
  - Number of filters: $32, 64$, and 128
  - Kernel filter sizes: $(3 \times 3), (2 \times 2)$, and (2×2)
  - Stride values: $1, 1$, and 1
- POOL layers:
  - Kernel filter sizes: $(2 \times 2), (2 \times 2)$, and (2×2)
  - Stride values: $2, 2$, and 2
- DROP layers:
  - Probabilites: $0.1, 0.2$, and 0.3

II . The results on Head images

The losses during training process

Like the processes on pronotum images, the losses are stables when we train EB-Net on head images. The average losses among the processes are $0.00026/0.00041$ for training/validation. Figure C.2 illustrates the losses of two rounds. In these images, the blue

lines present the training losses and the green lines symbolize the validation losses. Although training on other subsets of data, we can see that the losses are similar between two rounds. The losses (train and validation) have a little bit different at the beginning, but they are more closely at the end of the process.



a . 1^{st} round b . 2^{nd} round

Figure C.2: The losses of two rounds during training EB-Net on head images

The average distances

Table C.1 shows the average distance on each landmarks of all head images. The green and red numbers represent the best and the worst distances in each case. First of all, the predictions are quite stable. We do not have a big difference between the values. In this case, the 6^{th} landmark has the best prediction with 4.45 pixels; whereas the worst-case belongs to the 1^{st} position with 5.53 pixels. If we consider an error of 3% of the image's size (256×192), these values are acceptable. However, it is still high in the point of view of the biologists when we display the landmarks on the images.

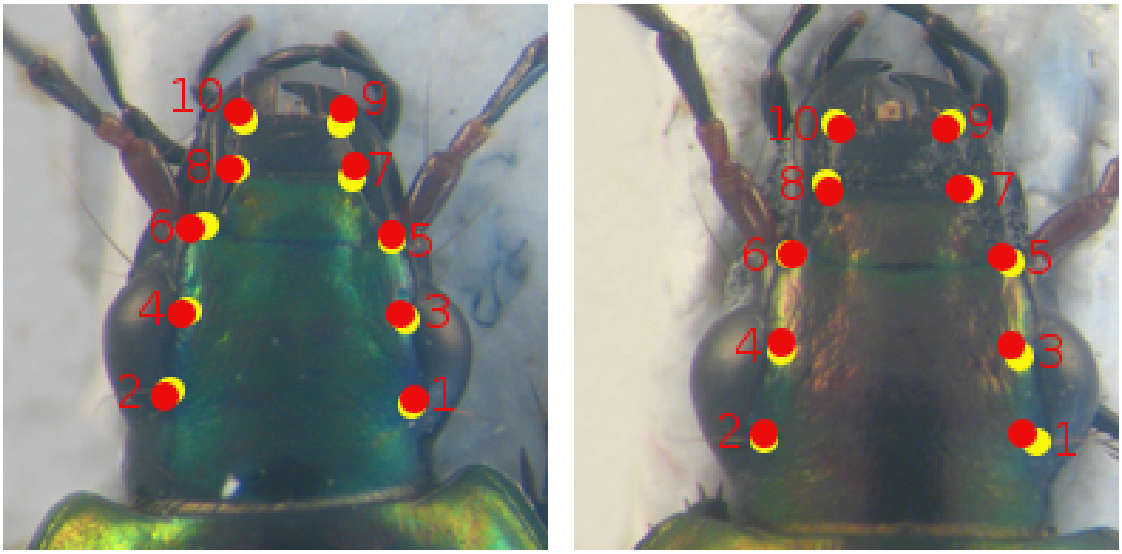
Landmarks displaying

Figure C.3 shows the locations of predicted landmarks on two head images. The images are randomly chosen from our dataset. The red/yellow points illustrate the pre-

Landmark	LM1	LM2	LM3	LM4	LM5	LM6	LM7	LM8	LM9	LM10
Mean distances	5.53	5.16	5.38	5.03	4.84	4.45	4.79	4.53	5.14	5.06

Table C.1: The average distance on each landmark of all head images.

dicted/manual landmarks. One can note that predicted points are not so far from the manual ones. On each image, they can be assembled into two groups: the first group includes the cases close to the manual; while the prediction in the second group is a little bit far from manual ones.



a . The 6^{th} image

b . The 55^{th} image

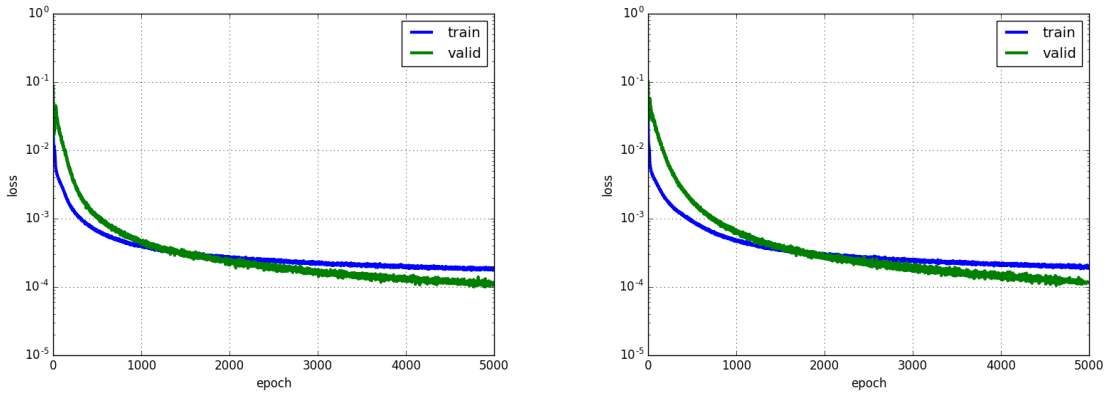
Figure C.3: The predicted and manual landmarks on head images. The red/yellow points illustrate the predicted/manual landmarks

III . The results on Elytra images

The losses during training process

Figure C.4 shows the losses of two rounds during the training process. We can observe from the figure that the distributions of the losses are nearly the same: from the be-

ginning, we have a little bit difference between training and validation losses, but after 1000 epochs, the validation losses have significantly decreased and more close to the training loss. In the case of elytra images, the mean losses values (of 9 rounds) are $0.00018/0.00012$ for training/validation.



a . 1^{st} round b . 2^{nd} round

Figure C.4: The losses of two rounds during training EB-Net on elytra images. The blue/green curves present to the training/validation losses.

The average distances

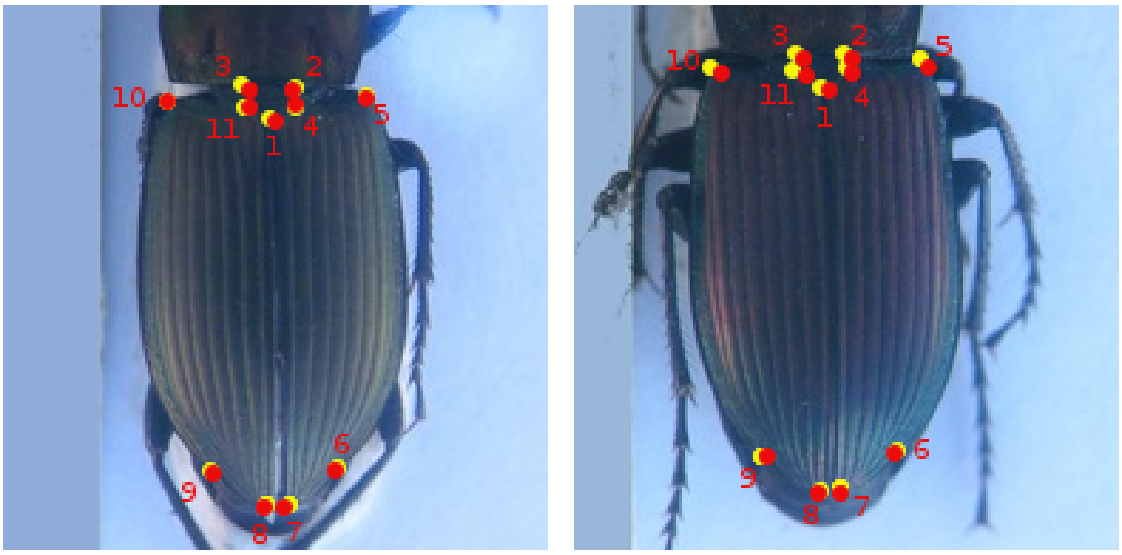
Table C.2 shows the average distance on each landmark of all elytra images. The green and red numbers represent the average distances for the best (1^{st} landmark) and the worst (8^{th} landmark) cases. With the image size of 192×256, the worst distance (5.47 pixels) is corresponding to 2.5% of error. Comparing with other parts, the average range of elytra images is higher but not so much (from 3.87 to 5.47 pixels). These distances could be separated into two groups: around 4 pixels (1^{st} - 5^{th}, 10^{th}, 11^{th}) and nearby 5 pixels (6^{th} - 9^{th}). These results are considered reasonable because the landmarks from 6^{th} to 9^{th} are considered difficult to predict in the case of elytra.

Landmark	LM1	LM2	LM3	LM4	LM5	LM6	LM7	LM8	LM9	LM10	LM11
Mean distances	3.87	3.97	3.92	3.87	4.02	4.84	5.21	5.47	5.27	4.07	3.99

Table C.2: The average distance on each landmark of all elytra images.

Landmarks displaying

Figure C.5 shows the location of the predicted/manual landmarks of two sample images in red/yellow color, respectively. One presents to the good prediction (Figure C.5a), another illustrates to an estimation with less accuracy on some landmarks (Figure C.5b). However, one to note that in both cases, the position of predicted landmarks are very closed to manual ones.



a . The 7^{th} image

b . The 22^{nd} image

Figure C.5: The predicted and manual landmarks on elytra images. The red/yellow points illustrate the predicted/manual landmarks

Appendix D

Fine-tuning EB-Net on beetle's images: supplements

As mentioned in Chapter 6, this chapter supplements the results of the fine-tuning process on three parts of beetle. Firstly, we present other statistical indicators of the fine-tuning process on pronotum, head, and elytra. Secondly, we illustrate the distribution of distances on the best and the worst case of all head and elytra images. To test other combinations of Elementary Block, we have modified the structure of EB-Net by adding more one block. The last section shows the comparison of the fine-tuning process on two versions of EB-Net.

I . Other statistical indicators

Table D.1, D.2, and D.3 display the statistical values on each part: pronotum, head and elytra, respectively. The green and red numbers represent the best and the worst values on each statistical indicator, respectively. First of all, all the values have been decreased from training from scratch as mentioned in Chapter 6. Secondly, the median rates which are separated the data into two parts, are reduced and smaller than the mean values. Additional, these values close to the minimum values and far from the maximum rates.

It means that most of the landmarks are well-predicted. It only exists some difficult cases to predict.

#LM	Mean	Stand. Dev.	Median	Minimum	Maximum
LM1	2.9914	1.808	2.7031	0.23	14.2496
LM2	3.4066	2.235	2.9626	0.175	18.4053
LM3	2.9829	2.063	2.5864	0.216	19.2092
LM4	3.5449	2.433	3.117	0.1638	22.8899
LM5	3.3675	2.272	2.9741	0.101	17.4586
LM6	4.0611	2.588	3.5733	0.1733	14.0745
LM7	2.9274	1.984	2.5703	0.2263	14.092
LM8	3.6448	2.483	3.0116	0.1647	15.4585

Table D.1: The statistical indicator values on pronotum images

#LM	Mean	Stand. Dev.	Median	Minimum	Maximum
LM1	4.8185	2.925	4.2951	0.3732	21.1819
LM2	4.2098	2.936	3.7484	0.2072	23.9351
LM3	4.7286	2.918	4.3991	0.2719	19.12
LM4	4.1071	2.912	3.6232	0.1942	21.6451
LM5	4.1769	2.645	3.7967	0.2683	20.2307
LM6	3.4976	2.837	2.9338	0.2384	22.6836
LM7	3.9168	2.529	3.4284	0.2134	21.0319
LM8	3.402	2.544	2.7877	0.1478	21.233
LM9	4.1703	2.536	3.7181	0.4441	22.0267
LM10	3.9433	2.695	3.4147	0.152	20.7223

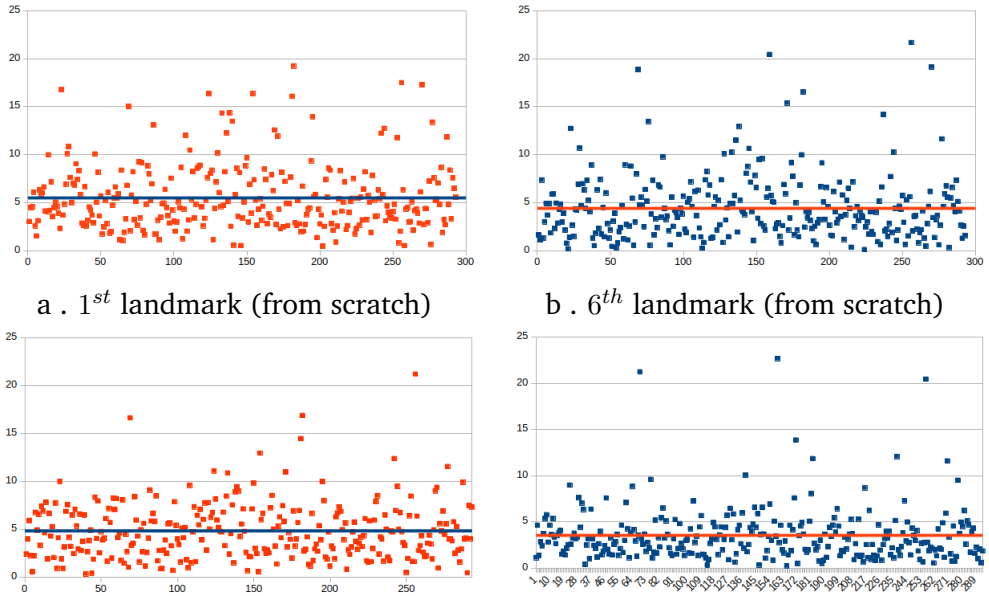
Table D.2: The statistical indicator values on head images

#LM	Mean	Stand. Dev.	Median	Minimum	Maximum
LM1	3.2081	3.064	2.6311	0.1265	32.6688
LM2	3.2842	3.204	2.5934	0.1607	33.9982
LM3	**3.1975**	**3.004**	2.5412	0.0763	31.0928
LM4	3.225	3.102	**2.479**	0.1485	33.1458
LM5	3.3062	3.200	2.606	0.1187	**35.7959**
LM6	4.2069	3.350	3.578	0.2149	35.3037
LM7	4.5445	**3.507**	4.0792	0.3454	34.7368
LM8	**4.7596**	3.454	**4.3057**	**0.4697**	32.1749
LM9	4.548	3.279	3.9626	0.2711	**28.3484**
LM10	3.3918	3.033	2.7726	0.1799	29.9211
LM11	3.2897	3.019	2.7064	**0.0527**	32.3641

Table D.3: The statistical indicator values on elytra images

II . The distributions of the distances

Figures D.1 and D.2 illustrate the distribution of distances on the landmarks (the best and the worst case) between two processes (training from scratch and fine-tuning) on each part: head, and elytra, respectively. The lines in charts represents the mean values in each case. In these charts, the horizontal axis presents the number of images in the dataset. The vertical axis illustrates the distances in pixels between manual landmarks and predicted ones. From the charts, most of the distances between predicted and manual landmarks have been reduced in both two cases (the best and the worst). In which, we make a note of the distances above the average values. These values have been decreased, and they are more close to the average lines by helping of fine-tuning.



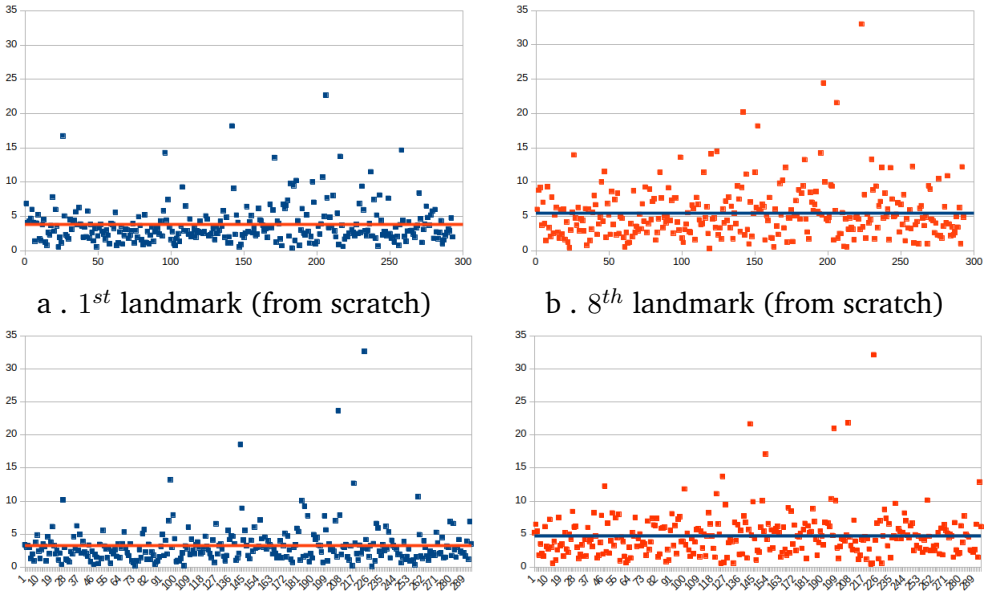
a . 1^{st} landmark (from scratch)

b . 6^{th} landmark (from scratch)

c . 1^{st} landmark (fine-tuning)

d . 6^{th} landmark (fine-tuning)

Figure D.1: The distribution of average distances of all head images on 1^{st} landmark and 6^{th} landmark.



a . 1^{st} landmark (from scratch)

b . 8^{th} landmark (from scratch)

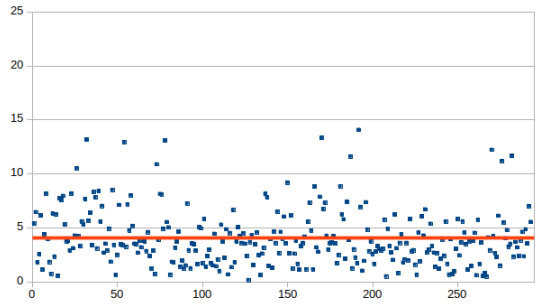
c . 1^{st} landmark (fine-tuning)

d . 8^{th} landmark (fine-tuning)

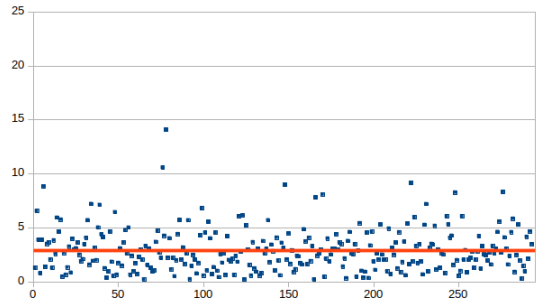
Figure D.2: The distribution of average distances of all elytra images on 1^{st} landmark and 8^{th} landmark.

III . The novel results on modifications of EB-Net

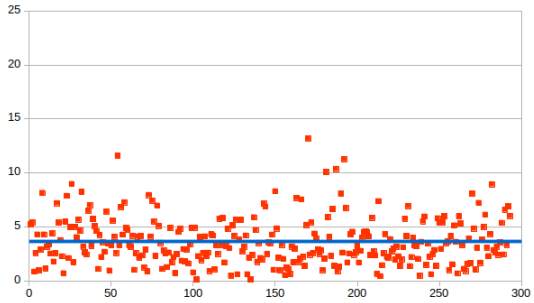
This section shows the distribution of distances between the manual and predicted landmarks at the best and the worst-case on five parts of beetle. The results have been obtained by applying the fine-tuning process with the new version of EB-Net (4 blocks). It illustrates the distances given by the previous version (3 blocks). In Figures D.3, D.4, D.5, D.6, and D.7, the top row displays distribution of the previous process, and the bottom row presents the new ones. The figures show that the new results have been improved, the distances are more convergence to the mean values (the lines in the charts), especially, the cases stay far way the mean lines. In these charts, the horizontal and vertical axes have the same meaning than in II .: horizontal axis - number of images, the vertical axis - the distances in pixels.



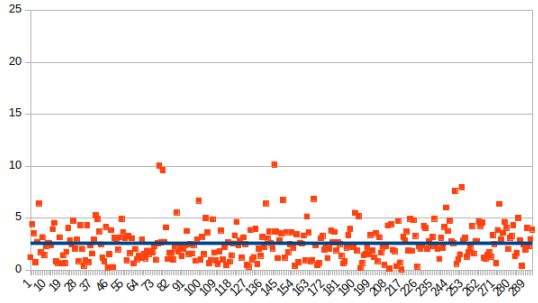
a . 6^{th} landmark (3 Elementary Blocks)



b . 7^{th} landmark (3 Elementary Blocks)

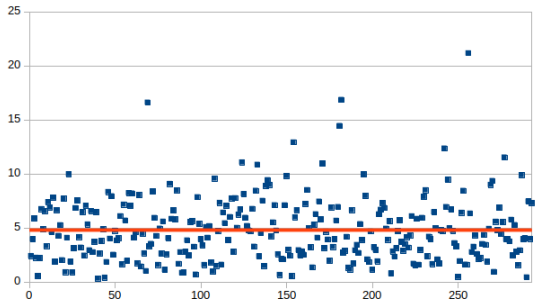


c . 6^{th} landmark (4 Elementary Blocks)

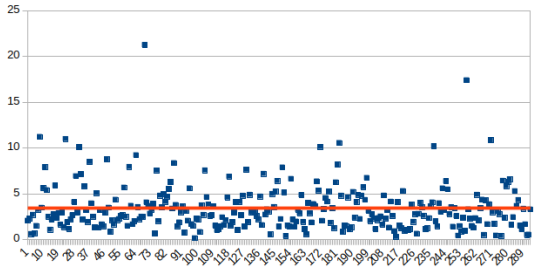


d . 7^{th} landmark (4 Elementary Blocks)

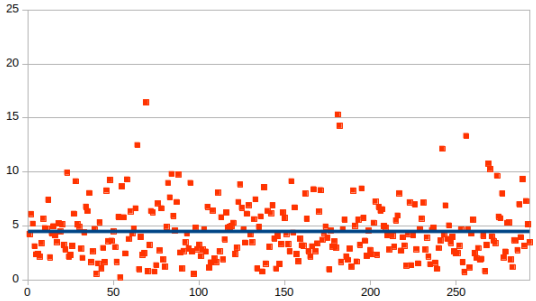
Figure D.3: The distribution of distances on two positions of pronotum images.



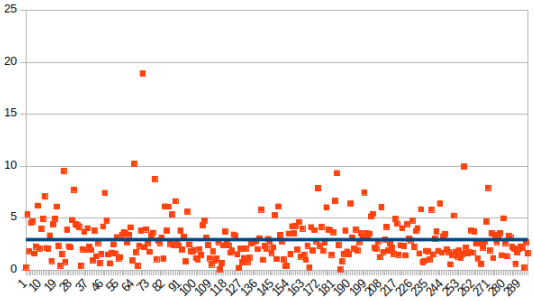
a . 1^{st} landmark (3 Elementary Blocks)



b . 8^{th} landmark (3 Elementary Blocks)

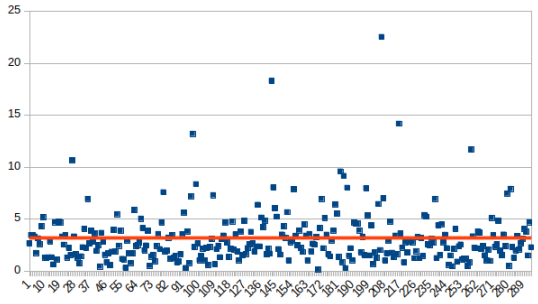


c . 1^{st} landmark (4 Elementary Blocks)

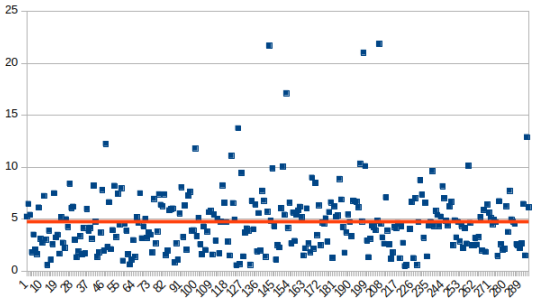


d . 8^{th} landmark (4 Elementary Blocks)

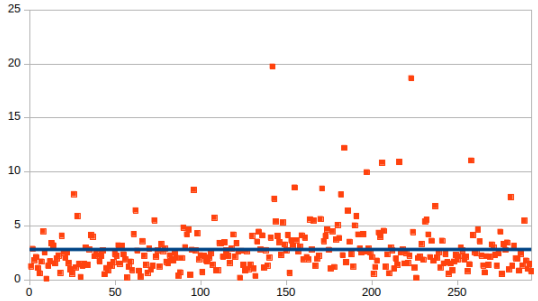
Figure D.4: The distribution of distances on two positions of head images.



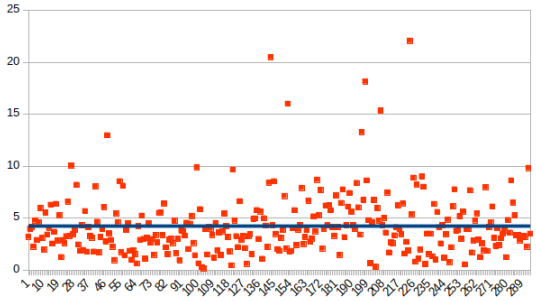
a . 4^{th} landmark (3 Elementary Blocks)



b . 8^{th} landmark (3 Elementary Blocks)

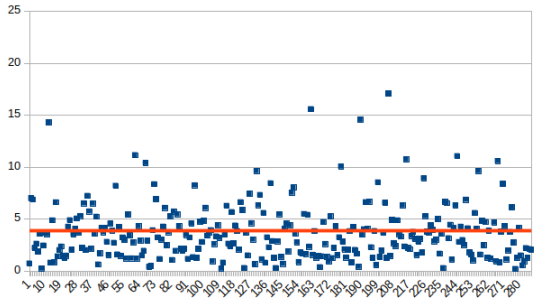


c . 4^{th} landmark (4 Elementary Blocks)

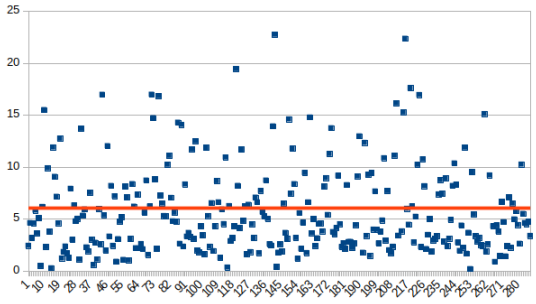


d . 8^{th} landmark (4 Elementary Blocks)

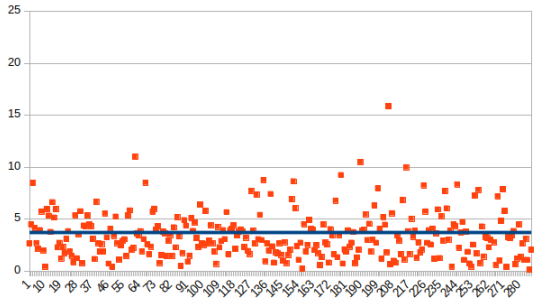
Figure D.5: The distribution of distances on two positions of elytra images.



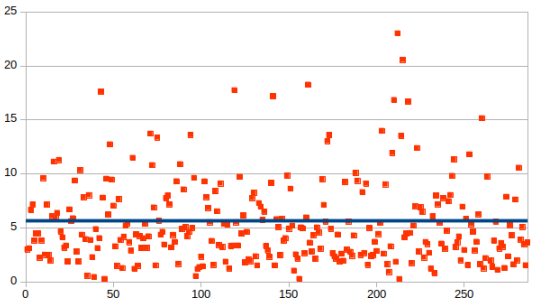
a . 5^{th} landmark (3 Elementary Blocks)



b . 16^{th} landmark (3 Elementary Blocks)

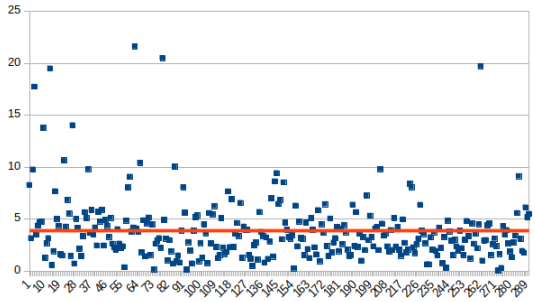


c . 5^{th} landmark (4 Elementary Blocks)

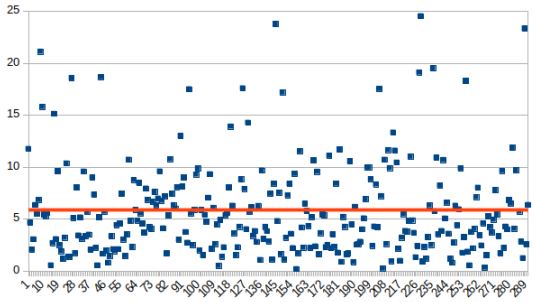


d . 16^{th} landmark (4 Elementary Blocks)

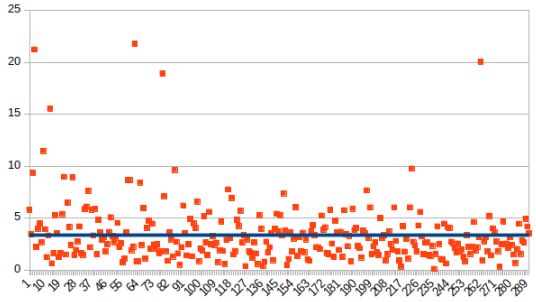
Figure D.6: The distribution of distances on two positions of left mandible images.



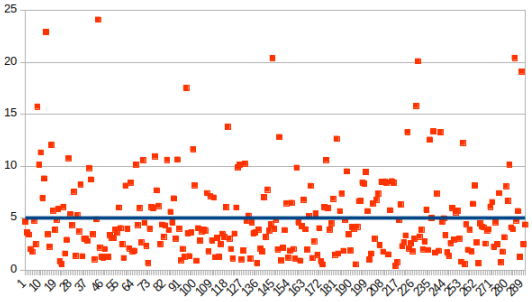
a . 4^{th} landmark (3 Elementary Blocks)



b . 18^{th} landmark (3 Elementary Blocks)



c . 4^{th} landmark (4 Elementary Blocks)



d . 18^{th} landmark (4 Elementary Blocks)

Figure D.7: The distribution of distances on two positions of right mandible images.