Automatic landmark identification in digital images of Drosophila wings for improved morphometric analysis

Sebastian Bas
Abstract

Automatic landmark identification in digital images of Drosophila wings for improved morphometric analysis

Sebastian Bas

A considerable number of morphometric studies which are performed nowadays with fly wing images require manual annotation of landmarks or key-points. This work is tedious and time consuming for researchers. This is why there is interest in automating the process and therefore several approaches for this purpose have been developed. The problem with these methods is that they are difficult to use and are usually specific to a particular imaging format and species. This project’s objective is to develop two methods, one based on classic image analysis techniques, and another one based on machine learning algorithms. A comparison is made to understand the strengths of each approach and find a solution that is general and easy to use. The first method (classic) uses domain knowledge to extract features and match template structures to determine landmark locations. Every parameter is fine-tuned manually and requires a long time to develop. Nevertheless, the results achieve human-level precision. The second method uses deep learning algorithms to train 30 neural networks which divide the image into regions and extract the coordinates of the landmarks directly. The results obtained for the machine learning approach are similar (approximately 10-pixel precision for 2448 x 2048 size images), with the advantage that it does not require any domain knowledge and can be reused for any kind of format and species. A solution that combines the strengths of both methods seems to be the best path to find a fully automatic algorithm.
Acknowledgements

First of all, I would like to thank my teachers and supervisors Zorana Kurbalija Novičić, Nataša Sladoje, and Joakim Lindblad for their invaluable help throughout the whole process. This includes great guidance, feedback, ideas, and also much needed encouragement to keep going forward. Thank you so very much!

I would also like to thank my family for their continuous support, allowing me to chase dreams far away from home. This project would not have been completed without the many forms of inspiration and help they provide me.

This thesis project has resulted in a very interesting process of learning and confronting of challenges. I feel very grateful of having the opportunity to work in this subject and hopefully this document proves to be useful to the community.
# Contents

1 Introduction .............................................................................................................. 7
   1.1 Background .......................................................................................... 7
   1.2 Aims of the Project ............................................................................ 8
   1.3 The Problem ....................................................................................... 9

2 Related Work .................................................................................................. 12
   2.1 Related Algorithms and Concepts ........................................................ 12
      2.1.1 Vessel Enhancement Filter ......................................................... 12
      2.1.2 Binary Segmentation Methods .................................................. 13
      2.1.3 Skeletonization ........................................................................ 14
      2.1.4 The Coherent Point Drift Algorithm ........................................ 14
      2.1.5 Deep Learning ......................................................................... 16
   2.2 Classic Image Analysis ........................................................................ 18
      2.2.1 Feature Matching Approach ....................................................... 18
      2.2.2 B-splines Approach .................................................................. 20
   2.3 Machine Learning ................................................................................ 21
      2.3.1 VGG Approach .......................................................................... 22
      2.3.2 Cascaded Networks Approach .................................................. 24

3 Method ............................................................................................................ 26
   3.1 The Dataset ......................................................................................... 26
   3.2 Image Resolution ................................................................................ 26
   3.3 Imaging and Equipment ....................................................................... 28
   3.4 Specific Requirements .......................................................................... 28
   3.5 Classic Image Analysis ......................................................................... 28
      3.5.1 Feature Extraction ...................................................................... 29
      3.5.2 Alignment .................................................................................. 34
      3.5.3 Result Refinement ...................................................................... 37
   3.6 Machine Learning ................................................................................ 38
      3.6.1 Architecture ................................................................................ 38
      3.6.2 Data Augmentation .................................................................... 40
      3.6.3 Training ..................................................................................... 42
      3.6.4 Output ....................................................................................... 43

4 Results and Discussion ................................................................................. 44
   4.1 Manual Error ....................................................................................... 44
4.2 Classic Image Analysis ................................................................. 48
  4.2.1 Algorithm’s Error ................................................................. 48
  4.2.2 An Extra Dataset ................................................................. 55
4.3 Machine Learning ......................................................................... 56
  4.3.1 Algorithm’s Error ................................................................. 56
4.4 Algorithm Comparison ................................................................. 64

5 Conclusions .................................................................................... 69

References .......................................................................................... 71
1. Introduction

1.1 Background

Comparing the anatomical features of organisms has been a central element of biology for centuries. The taxonomic classification of organisms, and understanding the diversity of biological life, were both historically based on descriptions of morphological forms. During the early twentieth century however, biology began the transition from a descriptive field to a quantitative science, and the analysis of morphology saw a similar “quantification revolution” [1]. Morphological studies included quantitative data for one or more measurable traits, which were summarized as mean values and compared among groups [2]. By the mid-twentieth century, quantitative description of morphological shape was improved with the development of advanced statistical analysis which enables us to describe more precise patterns of size and shape variation of morphological traits. This is how the modern field of morphometrics began.

To a large extent, biological research has been focusing on genetics for many years now, and the study of an organism’s anatomical shape and its variation is part of studying gene function, its expression, and even its regulation [3]. This is what makes morphology important; the fact that a particular specie’s genetics can be studied indirectly by looking at its anatomy. Some traits are correlated with many different genes located in different chromosomes, producing a strong link between variability of the trait and variability in the organism’s DNA (e.g. fly wing shape) [4].

Within morphometrics appears the concept of geometric morphometrics, which is basically the use of mathematical tools to analyze size and shape of organisms and their constituents. One of the most widely used methods is to make landmark annotations on the different morphological features of organisms. This means that samples, which are usually 2D or 3D digital images, are examined and a series of landmark coordinates are extracted. These landmarks correspond to easily identifiable locations in the samples that have a strong relationship with the sample’s size and shape. The landmark extraction process has many names; we will refer to it as landmark-localization from this point on.

The work of landmark-localization is usually done manually by experts. This is time consuming and it wastes the experts’ resources; it prevents them...
from working on more relevant tasks. One famous study required morphologic information of 16615 flies [5]; it used specialized hardware and software but expert observers were still needed. In addition, the manual task is error-prone and is limited by a human’s precision. With today’s technology it is assumed that this type of task can be automated; it does not come without challenges though. Several methods have been designed and tested (e.g. [6]) to address this kind of tasks or issues, and they have been very effective at replacing manual work in specific cases. It is worth noting though that each case has its particular issues.

Image processing routines may vary widely depending on the input’s characteristics. Image datasets may have different characteristics influenced by the imaging technique used. Experts might use different cameras and prepare their samples in a specific way, producing images of a particular quality. Characteristics such as resolution and contrast have a strong influence in a processing algorithm’s design. Furthermore, samples within sets that have been imaged in a similar fashion may still be different from each other in terms of position, orientation, and scale. This variation is also a big factor to consider in the design of an effective image processing algorithm and it is the main driver of development for this project.

1.2 Aims of the Project

The main objective of this project is to develop a useful and efficient software which enables us to replace manual landmark-localization with an automatic process. Secondly, the objective is to test and evaluate existing automatic landmark-localization methods on a particular dataset, select the best performing one(s), and adapt/change it/them to improve performance and ease of use. Ideas from different methods may be incorporated into the algorithm in order to achieve the best results possible for the project’s dataset. Even though the applicability of the method to other datasets is desirable, efforts are concentrated on its usability with the project’s own image data.

Test and evaluation of methods are carried out for two distinct groups of techniques: Classic Image Analysis and Machine Learning. Based on these approaches, two methods are developed and compared to each other taking into consideration speed, precision, accuracy, robustness, and ease of use. The comparison includes, when possible, the capability of generalization over different datasets.
1.3 The Problem

This project is inspired in evolutionary biology research that makes use of *Drosophila* wings as the samples that need to be analyzed. The manual landmark-localization process is widely used to analyze size and shape of different *Drosophila* wings, which leads to an indirect study of the organisms’ genes. Furthermore, the evolutionary history of a fly specie can be studied with this technique through several generations, measuring for example their extinction risk and morphology stability among other characteristics [7].

There is no standard way of imaging *Drosophila* wings. Researchers make use of their own independent methodologies and produce datasets that fit their analysis tools. Only after the datasets have been passed through a usually manual landmark-localization process they can be compared to each other. The main dataset used in this project was processed manually so there is a clear need of an automatic method that can reliably perform landmark-localization on these samples. Software packages that address landmark-localization problems already exist, but none of them are suitable for the project’s dataset; much less for general use. Existing solutions are very specific and lack the user-friendliness that allows for ease of use as an effective research aid. Thus, a prototype utility software will be developed to be able to process the data in the easiest way possible.

Choice of landmarks for the project’s dataset has already been set by experts based on biological arguments. In this case, 13 landmarks are required to be located in each sample. Figure 1.1 presents a model sample with the corresponding annotated landmarks.

The research question for this project is the following:

**Is manual annotation replaceable by automatic landmark-localization?**

The question has many aspects that need to be addressed and clarified, since replaceability can be interpreted in various ways. In essence what is meant by having an algorithm that replaces the manual annotations is that the bulk of the work is done by a computer. Human work is still needed to set parameters, choose models, and prepare the samples in a way that they fit the algorithm’s input requirements.

To be able to answer the research question, manual annotations are analyzed and a performance profile is determined. If the algorithm performs in a similar or better way in terms of accuracy and precision, the automatic procedure is considered to be successful in replacing the manual work. It is important to mention that both approaches have innate weaknesses and strengths that can-
not be avoided. Therefore, replaceability is in many senses just partial, and the algorithm ultimately serves as a tool that can save time, correct certain types of mistakes, and obtain better results if specific conditions are met.

A performance profile consists in statistical information about the landmark-localization process. The main variables that are considered in the profile are the mean error and the standard deviation of the error. This means that for each sample, an error is calculated, measuring how far are the annotated landmarks from the expected value. Then an average and a standard deviation are calculated for each of the landmark positions.

In this project, the concept of stability refers to the sensitivity of a process under changes in the samples or parameters. For example, if the image samples have different scales or lighting conditions, the process should remain stable and robust; that is to say, measurements should remain the same independent of variations in the imaging conditions. Reproducibility in this context is understood as being able to repeat results if the process is applied on the same dataset and similar conditions or parameters. It is desirable that the solution to be proposed is both stable and reproducible.

Figure 1.2 shows an overview of how to go from manual to automatic landmark-localization. It is a simple and effective workflow.

In addition, the project should include:
Software development using 93 wing image samples which have been manually labelled.

A test of the produced software by landmarking a sample of 1000 *Drosophila* wings.

Evaluation of the methods by comparison of the automatically generated landmarks with the ground truth (manual annotations produced by several different operators/researchers).

Evaluation of the software effectiveness in terms of speed, precision/accuracy, and user-friendliness.

Strong efforts towards the utility of the software in generalizing over different datasets (fly species).
2. Related Work

This section describes some of the algorithms and approaches that have been implemented to solve the problem of automating landmark-localization. The solutions come mainly from two broad groups of techniques; these will be referred to as Classic Image Analysis and Machine Learning. The former group consists of typical image-analysis techniques with explicitly programmed algorithms that are designed to perform very specific tasks. The latter group consists of techniques that can be thought of as optimization problems in which a model is built automatically by experience (e.g. trained with data). This model can be for example an algorithm to predict the weather or one that does landmark-localization. A famous general explanation was provided by Tom M. Mitchell: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E” [8].

2.1 Related Algorithms and Concepts

2.1.1 Vessel Enhancement Filter

A popular filter used on images produced through research in the discipline of angiography was proposed in 1998 [9]. This filter has been the base of various applications [10] and further developments, and that is why it became the first step of the process. What it does is to highlight or extract the vessel-network-like structures in an image.

The original filter was designed to enhance the vascular tree of an angiogram using a multi-scale approach. Many of the proposed methods were developed for a fixed scale of features; this means that images that contain blood vessels with varying widths cannot be segmented effectively. This is why the filter combines results from a range of scales that can be chosen depending on the input.

Images are represented as discrete 2D functions or 2D matrices, so it is useful to analyze local behavior by calculating the discrete second-order Taylor expansion of the image. This can be done effectively by convolving the image with derivatives of Gaussians. The concept comes from linear scale-space theory, and, therefore, takes scale into account [9]. The objective is to look into
the second-order derivatives (Hessian matrix) and extract information that describes the local structure at a particular scale. By calculating the eigenvalues of the Hessian matrix and their corresponding eigenvectors, the geometrical properties of the image can be extracted. The eigenvectors point in the directions of minimum and maximum intensity variation; they can be thought of as the axes of a tube or an ellipsoid, shapes that resemble the features that need to be segmented. The corresponding eigenvalues can therefore be used to create a criterium for filtering, since they are related to an ellipsoid’s geometry and can describe desirable characteristics of local neighborhoods in the image.

In the end, calculations are carried out for a set of scales and then the results are combined to obtain the final output of the filter.

2.1.2 Binary Segmentation Methods

The methods discussed here are useful when the goal is to obtain an image with two classes of pixels, the ones that belong to an object, and the ones that are part of the background [11]. This is why they are called binary segmentation methods. The objective is usually to extract one object from the image and separate it from the background.

**Otsu’s Thresholding Method**

This method is a simple but very effective way of choosing a threshold $k$ to binarize a gray-scale image. The approach is based on the assumption that the classes of the pixels in the image can be well separated by using their gray level values. An image’s gray level histogram is modelled as a probability distribution in which there are two classes of pixels, the ones below threshold $k$ and the ones above. By means of statistical techniques [12], the search for a good threshold $k$ is reduced to an optimization problem, in which the within-class variance is minimized and the between-class variance is maximized simultaneously.

**Adaptive Thresholding Method**

This binarizing method deals with cases such as uneven illumination, in which a global threshold will not be able to separate the two classes of pixels in the correct way (object and background). The approach is to divide the image into several smaller regions and choose a separate threshold for each of them (adaptive) [11]. The challenge is to choose the right region size and threshold search algorithm.
2.1.3 Skeletonization

The skeleton of an object in a binary image is a thin representation that can be produced in different ways with varying results. The process can be expressed as a series of morphological operations for finding a set of points that fulfill certain geometric requirements [11]. The algorithm usually runs iteratively until the object is transformed into the skeleton, which can be thought of as a smaller and simpler object that preserves some of the original characteristics such as topology.

In a more intuitive description let us say that an object of interest is represented with white pixels in an image and the background is represented with black pixels. The idea is to methodically remove white pixels and make them part of the background so that eventually a new representation of the object is obtained. The objective is to have a representation that preserves the object’s topology with a 1-pixel width representation. This is what is called the skeleton of the object. Implementation details can be found extensively in the literature, and there are multiple algorithm variations [11].

An example of a wing’s vessel-network skeleton is shown in Figure 2.1. This image results as an output of the actual method used in the project.

2.1.4 The Coherent Point Drift Algorithm

Point set registration is a process in which a point set (or point cloud) is matched with another point set. The result of the process is a correspondence
between the two point sets and a transformation that maps points from one set to the other set. A great tool for this purpose is the Coherent Point Drift (CPD) algorithm. The main idea of the algorithm is to treat the point set alignment as a “probability density estimation problem” [13], which involves the use of Gaussian mixture models (GMM) to represent the points in the source set. These points then need to be fitted to the data or target set; that is the objective.

The algorithm is separated into three cases depending on the type of transformation: similarity (rigid plus scaling), affine, and deformable point set registration. The algorithm is constructed and initialized differently for each case; then the algorithm proceeds by alternating two calculation steps until convergence (EM algorithm). The E-step is the first step and it comprises the generation of the GMM centroids for the source point set, followed by an estimation of the point correspondence probability with the target point set. Then comes the M-step which constitutes the calculation of the case parameters and its transformation (mapping between sets). After each iteration, new parameter values are obtained, which can be used to start a new iteration. The process is repeated until a convergence condition (tolerance) is met or a maximum number of iterations are completed.

The similarity and affine cases are very similar and their difference lies in the optimization constraints. The similarity registration case tries to find a transformation of the form \( T(y; R, t, s) = sRy + t \), where \( s \) is a scaling parameter, \( R \) is a rotation matrix, \( y \) represents the GMM centroids (source set), and \( t \) is a translation vector. There are restrictions for the rotation matrix: \( R^T R = I \) and \( \det(R) = 1 \). For the affine registration, the transformation takes the form \( T(y; B, t) = By + t \), where \( B \) is an affine transformation matrix.

Deformable registration is a bigger challenge since the transformation has more degrees of freedom than in the two previous cases. The transformation is of the form \( T(Y, v) = Y + v(Y) \), where \( Y \) is the initial position of the source set points, and \( v \) is a displacement function. To find \( v \), techniques from a math branch called calculus of variation are used. This leads to a formulation of the optimization problem that includes \( v \) as part of a regularization term in the objective function to be minimized, ensuring that the displacement function is smooth. Iteration of the E and M steps is carried out analogously.

Regarding tuning parameters, all cases share the variable \( w \) as an assumed value that expresses the presence of noise and outliers. If \( w \) is close to 0, this means that there is little noise and a few outliers. On the other hand, if \( w \) is close to 1, the assumption is that the point sets are very noisy and full of outliers. The non-rigid case requires the setting of two more parameters. The first parameter \( \beta \) controls the width of the Gaussian filter contained in the regular-
Figure 2.2. Illustration of a deep neural network for image processing. Source: [15]

ization term, and the second parameter $\lambda$ is the weight of the regularization term. Both parameters represent the “amount of smoothness regularization” [13] and a discussion about the differences between them can be understood by studying The Motion Coherence Theory [14].

Each case with its corresponding optimization-procedure results in the obtention of a transformation mapping and a correspondence between the source point set and the target point set.

2.1.5 Deep Learning

Within the field of machine learning, deep learning is an approach that uses complex neural networks with many layers. These networks are essentially a combination of many functions that operate over an input and give out an output or a new representation of the data [15]. Figure 2.2 shows a simple diagram of how a neural network could be used to process an image and output an abstract quality of the object.
**Convolutional Neural Network**
A convolution is a linear mathematical operation that is largely used to filter images and similar grid-structured data. A convolutional neural network has certain stages or sub-networks that implement that mathematical operation and act as trainable filters. Moreover, these networks usually include a so called activation function, which introduces a nonlinearity in the network operations, and pooling stages, which reduce the dimensionality of the data by replacing output regions with a statistic summary of the neighborhood [15]. This network architecture has been very effective in practical applications.

**Training**
Training deep neural networks is a difficult task that has benefited immensely from the increase of computational power. Nowadays there are many frameworks and services that allow for a relatively easy training process. An example of this would be to use the Tensorflow framework in a computer provided by a cloud service.

The training process is a complicated optimization problem that is constructed to find the best values for the network’s parameters (weights and biases). That is to say, the values that minimize an error function (usually). Since it is a difficult problem, there are many different ways to solve it, and the choice of method depends on the data. Nevertheless, the basic training involves processing some samples, calculate an error, and update the weights and biases so that the error is minimized. This is done iteratively until a condition is met.

Even though it is not the purpose of this section to explain the whole training process, there are two essential parameters that need to be mentioned. These variables are relevant for every training process. The first one is the batch size; a parameter that controls how many samples should be processed before calculating the error and updating the network’s weights and biases. The second is the learning rate, which controls the size of the update steps; in other words, it influences how much are the weights and biases going to change on each update. The learning rate usually needs to decrease with each iteration so that the training remains stable and converges to a solution.

**Data Augmentation**
Training a neural network usually requires a huge amount of data which is most often not readily available, or it is difficult to collect. Data augmentation is a technique that addresses this problem by producing artificial data based on real data. The objective is to expand our training set in a way that it contains...
as many variations as possible that the network might encounter processing future data.

2.2 Classic Image Analysis

The following subsections provide descriptions of the main approaches found in the literature and used in this project. Their essential ideas are presented here together with the algorithms that are used in the proposed method, which are described in more detail.

2.2.1 Feature Matching Approach

The Feature Matching Approach [16] consists of the following four main steps: extraction of the wing’s vessel network, measurement of geometric features using shape descriptors, estimation of the pose, and a refining template matching step. The essential idea is to extract features from the samples and match them to a pre-defined model, a technique which is used broadly in many applications [17]. The dataset used in this [16] approach consists of 856 grayscale fly wing images of size 1280x1022 pixels.

The first stage introduces a very important concept which is ridge detection. In contrast to edge detection, which deals with borders, ridge detection helps with the problem of finding regions in an image that look like mountain ranges (geographical ridges) or valleys. This is key because the wing’s structure is mainly composed of veins that fit with the ridge/valley feature description.

An intuitive approach to the concept of ridge or valley is to think of a “sequence of connected pixels” [18] that share a similar intensity value and are surrounded by neighboring pixels with relatively higher or lower intensities. A technical description makes use of directional derivatives. If we were to traverse the highest points of a ridge or the lowest points of a valley in an image, we need to follow a direction that maximizes the magnitude of the second order derivative. Zero crossings of the first order derivative along the path represent either ridge peaks or valley bottoms.

There is not a unique way of identifying ridges and valleys. The Feature Matching method [16] suggests using a Difference of Gaussians (DoG) filter to extract them, which in essence is a band-pass filter and it needs to be set to fit the size of the features in the particular dataset. The filter did not prove
to be robust enough for the project’s dataset since the varying size of features was difficult to capture. Nevertheless, a DoG filter was good enough for the suggested method’s dataset [16]; it exposes the veins and allows an easier feature extraction that ends up with a skeleton representation of the wing’s vessel network.

The representation of the vessel network is made up of segments that are treated as features. Pairs of segments are used to construct shape descriptors through pairwise geometric histograms; a method that measures geometric relationships using angles and perpendicular distances. This technique is known to be robust and therefore effective at dealing with imperfections coming from the first step of ridge detection.

Shape description is done beforehand for a set of training/reference/model images, which are used to carry out a histogram matching process. In this way, features of the model images are matched with features of the input samples. The matching results are used to estimate the position, scale, and orientation of the test samples using the probabilistic Hough transform. Specific shapes extracted from the model images are used as the parametric curves to be detected by the transform. Groups of features that relate to each landmark can be used to locate them. An approximation of the landmark positions is obtained since the landmark locations are known for the model images.

A more accurate localization of the landmarks is still needed for them to be useful in research studies. Therefore, a template matching step is added to refine the position of the landmarks. The matching is done on a small region around the estimated landmark locations of the previous step; using the filtered versions of the model and sample images (DoG filtered).

To evaluate the system, the whole dataset [16] was processed. Multiple model images were used to locate the landmarks and the output information was combined in a way that produced a consistent result. The method obtains very good results; 98.36% of the landmark coordinates are located within five pixels of the manually annotated ones. Nevertheless, it is important to mention that usability of the results still requires the removal of outliers, which correspond to about 1% of the results. It is not completely clear how these outliers are recognized without previous knowledge; there needs to be a way to identify a sample that has been processed wrongly.

The article mentions that open-source code would be readily available, but a software package was not easy to find. The available software (TINA 6) was installed and tested but it was not usable at all for an external user, and the landmark-localization algorithms did not seem to be included. The available
tools were not easily adaptable to this project’s particular case.

The suggested method’s way of using model images to match features and perform landmark-localization seemed very cumbersome and complex. There are certainly many aspects that could not be tested for the project’s dataset, but the idea of using models with known information and relate them to the input samples was a key idea that permeated into the project’s actual method.

2.2.2 B-splines Approach

The article that proposes this approach introduces the reader to a rather different way of obtaining and analyzing samples [19]. A small vacuum pump is used to suck in a fly’s wing into an apparatus which allows to fix the fly onto the imaging microscope’s stage. It is worth noting that flies are anaesthetized before being handled with this so called “wing grabber”, and they recover unharmed from the procedure. After positioning the fly, a short video is recorded with frames of a maximum size of 632x480 pixels. A sample of one of these frames is shown in Figure 2.3.

The obtained images are then processed in a semi-automatic way. Two landmarks need to be annotated manually for further automatic analysis to work. The algorithm goes through the following basic steps: the mentioned semi-automatic pre-process, generation of a skeleton representation as in the previous method, matching of features to a model, and fitting of the model.

The model is constructed by a combination of B-splines, which are functions piecewise defined by other functions (e.g. polynomials). These are useful to fit the curves to the fly wing’s structure, which was obtained from the skeletonization process. A total of nine curves are fitted to the skeleton and the resulting output consists of 100 parameters. These can be used later to extract any kind of information regarding the fly wing’s shape.

Results for this method are presented for 12 landmarks annotated on a dataset of 535 images. This set is composed of 87 male flies and 92 female flies that have been imaged an average of 3.3 and 2.7 times respectively. In the worst case scenario, landmarks are correctly located an average of 82%. Combining several parameter configurations, the average goes up to 93%. Still, some landmarks have a low percentage of correct localization (close to 72%), which limits the usefulness of the data to comparisons among species and other types of classifications.
The proposed methodology requires a big amount of parameters to be set by the operator, and this is a trial and error endeavor that can waste a lot of time and eventually not be applicable to different datasets. The software was available for download ¹, but it required tweaking and it was not easily usable for this project’s dataset.

2.3 Machine Learning

The following subsections describe different approaches based on machine learning techniques; more precisely, neural networks, which have been effectively used to address the task of landmark-localization for several different cases. Solving the challenge involves dealing with specific characteristics of the data and finding the right network architecture, among other things. The approaches here presented are quite different and influenced the proposed method in various ways.

¹The software is available on this link: http://www.bio.fsu.edu/~dhoule/software.html.
2.3.1 VGG Approach

This approach [6] was designed to locate four landmarks on images that come from video footage of a flying hawkmoth. The head, abdomen tip, left wing tip, and right wing tip of the moth are the four landmarks that need annotation, and the task is carried out by using deep neural networks. These networks are called deep because they can have many layers of neurons, sometimes being made of several types of neural networks. In this case the model is essentially based on convolutional and fully-connected neural networks.

The dataset was generated using two cameras recording at 400 frames per second with a resolution of 600 by 800 pixels. Manual landmark localization was done for 800 frames (2 seconds), producing the ground truth for the dataset.

A research group at the University of Oxford called Visual Geometry Group (VGG) created a deep neural network architecture that performed extremely well (7.4% classification error) in the ImageNet challenge of 2014. This challenge involved classification and object detection on millions of images containing a thousand different classes of objects. This architecture has therefore inspired lots of research and projects including the approach being described.

The VGG architecture shown in Figure 2.4 has 16 neural network layers; 13 are convolutional and 3 are of the fully-connected type. There are also some max pooling layers which reduce the dimension of the incoming data. The last layer named Soft Max in the diagram translates the output vector into probabilities of belonging to a class, and it is not used when the objective is to obtain landmark coordinates.

Taking the VGG architecture as a base, different variations were evaluated by discarding more or less layers. The default architecture is called “VGG7 +
“FC8” which is obtained by discarding every layer after the seventh convolutional neural network (counting from the input layer), keeping 3 max pooling layers, and adding a fully-connected layer of 8 neurons (one for each landmark’s coordinate). This is how the output is changed to produce coordinate values.

The network can be initialized and trained in many different ways. In virtue of being brief and efficient, only the default and essential case will be described. This consists of a network that is initialized with the original values obtained from training the VGG 16 network on the ImageNet dataset. Doing this is referred to as using a pre-trained network, a technique that saves training time by reusing parts of other networks. In contrast, the fully-connected layer is the only one being trained and it is initialized with the well known Xavier initialization method [20].

As mentioned before, the dataset counts with 800 frames that contain the hawkmoth and the target landmarks. This is not nearly enough to train such a large network, so it is necessary to perform data augmentation. The idea is to aim at representing all the possible variation of images that can be encountered as input by the network. By taking different crops of each frame, a translation of the hawkmoth is simulated and the training set can be grown to have around 200,000 samples.

Two parameters are the most influential on the convergence and results of training, the learning rate \( \lambda \) and the batch size. Usually one needs to experiment with these values to find the best combination. The default values used were \( \lambda = 10^{-10} \) and a batch size of 32 samples per iteration. Training was run commonly for 10,000 iterations, making sure that each sample would be seen at least once.

The first 400 frames were used for training and the rest for testing. To evaluate the resulting network, a loss function is calculated as follows:

\[
loss = \frac{1}{n} \sum_{i=1}^{n} \left\| y_i - y_i^* \right\|^2
\]  

(2.1)

where \( n \) is the number of samples from the test set, \( y_i \) is the output of the network, and \( y_i^* \) is the ground truth vector of coordinate values.

It is important to mention that the results for this approach were calculated with respect to the processing scale; that is to say, the samples are input at a resolution of 224 by 224 pixels and the landmark locations are compared at
this scale. This introduces some distortion on the results, making them harder to compare with other approaches. Nevertheless, the test loss for the default case \cite{6} was close to 150, which translates to an average distance between ground truth and prediction of 12.2 pixels.

2.3.2 Cascaded Networks Approach

This approach obtains great results on locating face landmarks by using several neural networks organized in a three level process \cite{21}. Focusing on the architecture of individual networks and combining their results into a robust approximation is the main objective of this idea.

For each level, a number of different neural networks operate over regions or crops of the input image. In the first level, image regions are chosen based on a set of rules; for the second and third levels, the regions are selected based on the previous output (landmarks) of the networks. A neighborhood around the landmarks is passed on to the next level, reducing its size progressively. In addition, at each level the networks locate one or more face landmarks simultaneously and merge their results by averaging the output coordinates. This reduces the variance considerably.

Predictions at the first level are taken as the base values for which the following levels introduce adjustments. This means that each level’s output is corrected by the next one according to the following expression:

\[
x = \frac{x_1^{(1)} + \ldots + x_{l_1}^{(1)}}{l_1} + \sum_{i=2}^{n} \frac{\Delta x_1^{(i)} + \ldots + \Delta x_{l_i}^{(i)}}{l_i}
\]  

(2.2)

where \( n \) is the number of levels and \( l_i \) is the number of predictions at each level. The superscript denotes the level at which the prediction was made.

Neural networks at the first level use nine layers; four convolutional layers, three of them followed by a max pooling stage, and two fully-connected layers. At levels two and three, neural networks are implemented on a shallower architecture with six layers; two convolutional layers followed by max pooling stages, and two fully-connected layers. The input image size varies from either 39 by 39 pixels or 31 by 39 pixels for the first level, and 15 by 15 pixels for the rest.
The training and validation sets consist of 5590 images from the LFW database and 7876 images gathered from the web, giving a total of 13466 face images. Of these, 10000 were augmented with small translations and rotations at each level, and subsequently used for training.

Performance was evaluated in a way so it is easier to compare with other research. The error is calculated as follows:

\[
err = \frac{1}{l} \sqrt{(x - x')^2 + (y - y')^2}
\]

where \( l \) is the width of the face’s bounding box, \((x, y)\) are the ground truth coordinates, and \((x', y')\) are the predictions. Dividing by \( l \) allows to express the results in terms of percentage, independently from the input image’s resolution.

The use of a cascade of neural networks is proved to be very effective and the average error after three levels lies around 1.1%. Some landmarks perform worse (mouth) and others perform better (eyes).
3. Method

3.1 The Dataset

Digital image samples have been produced by our collaborators at the Evolutionary Biology Centre of Uppsala University. They are interested in the automation of landmark-localization for *Drosophila* wings. The details about sample collection can be found in [7]. Landmark-localization for these samples was done manually by 8 individuals so that a ground truth is produced and the human error can be measured, which will be explained in a later section. The manual annotations consist of 13 landmarks that were collected using a software called *TpsDig2 ver. 2.26* ¹ [22], and expressed as X and Y coordinates in Cartesian space. Figure 3.1 shows a raw sample image of the project’s dataset provided by researchers.

The dataset consists of 93 images similar to the one shown on Figure 3.1. The variation of the samples involves translations, rotations, and a small amount of scaling. Some of the images are of low contrast and contain noise, small obstructing objects, and wing damage. In some cases, a part of a second wing is also included in the frame. All of these characteristics need to be dealt with and discussed so that the imaging process is standardized, and new input images fall within the expected variation. Examples of the mentioned sample variations are shown in Figure 3.2.

3.2 Image Resolution

All the samples in the dataset are 8-bit RGB images that have been stored in the *TIF* format. The size of the images is 2448x2048 pixels.

The goal is to be able to fully understand how the sample images relate to real world dimensions. Furthermore, it can be crucial to know the amount of detail that is present in the images, so that the results make sense.

¹The software is only available for Windows and can be found on this link: http://life.bio.sunysb.edu/morph/.
Figure 3.1. A *Drosophila* wing sample from the project’s dataset.

(a) Second wing.  
(b) Obstructing object.

Figure 3.2. Examples of sample variation.
3.3 Imaging and Equipment

Frozen flies (−20°C) were used for slide preparation. The right wings of the flies were gently dissected at the point of articulation with the thorax, and mounted between microscope slides and coverslips using double sided tape. The sides (left versus right) of individuals were designated by placing them on the ventral side with the heads facing forward. The right wing of each fly was digitally photographed (40x magnification) by using a camera (INFINITY2-5C CCD) attached to a microscope (LEICA M165C).

3.4 Specific Requirements

A series of minimum requirements have been set as main priority for the software tool. The following lists express some of the most important needs that a user has. These are the main characteristics of the software utility.

Regarding data handling, the software should allow a user to:

- Preselect a number of landmarks to be detected for the analysis (in this case 13 landmarks are used).
- Load and process a large number of images (batch process).

Regarding automatization, the software should be able to:

- Locate the landmarks automatically and in the identical order for each image in the dataset (e.g. from 1 to 13).
- Create a text file output data with the X and Y coordinates for each landmark on each image.

3.5 Classic Image Analysis

This section describes the development of a process based on techniques from what is called the Classic Image Analysis group (Section 2). The process is presented in a toolchain form, with each section explaining a step of the procedure and leading into the next stage.

The approach explained here was developed entirely with MATLAB 2016b; which contains an extensive toolbox of various image-processing algorithms.
3.5.1 Feature Extraction

It was clear from studying the previous work in Section 2.2 that a good starting point to solve the problem was to extract the structure of the wing’s veins. The objective is to produce a skeleton of the wing’s vein/vessel network; something similar to what was presented in Figure 2.1. Since the majority of the landmarks are located on wing veins, it is a key step to extract them.

During the study of previous work, the concept of ridge detection became really important since it seemed to point in the right direction. Following this trail led to several studies about blood vessel detection and feature enhancement; applied for example to retinal and placental blood vessels [10]. The vein networks of fly wings look similar to human blood vessel structures, so it was reasonable to think that feature extraction methods used on the latter could also be applied for this project’s image data.

Vessel Network Filter

To extract the vessel network from the wing, the images of the dataset are first converted to grayscale (MATLAB: rgb2gray) and then resized to 1224 pixels wide and 1024 pixels high (MATLAB: imresize) for easier and faster handling. Then, the vessel enhancement filter [9] described in Section 2.1.1 is applied using parameter values that were chosen experimentally. In the 2D case there are three parameters that need to be set; the scale values, and two constants that act as thresholds controlling the sensitivity of the filter to specific geometric measurements. These measurements are based on the eigen values of the Hessian matrix. The following expression represents the output of the filter:

\[
V_o(s) = \begin{cases} 
0 & \text{if } \lambda_2 > 0, \\
\exp\left(-\frac{R_B^2}{2\beta^2}\right)(1 - \exp\left(-\frac{S^2}{2c^2}\right)) & \text{otherwise.}
\end{cases}
\] (3.1)

Here, \(R_B\) and \(S\) are the measurements calculated using the two eigen values of the Hessian matrix. Therefore, these measurements depend on the scale \(s\) which is present in the definition of the Gaussian function used to differentiate the image in the first steps of the algorithm. The eigen value \(\lambda_2\) is the largest in absolute value of the two. The threshold constants \(\beta\) and \(c\) are set to 0.5 and 15 respectively. The scale values are chosen in the range \([3, 12]\) with a step size of 1.5 between values. To combine the results from all the scales, the maximum value of the filters is chosen for each pixel of the image.

Figure 3.3 shows what the filter produces after applying it to one of the dataset’s samples. The presented enhancement has been assessed to be effec-
Binary Segmentation

Prior to producing the desired skeleton representation of the vessel network, the image needs to be binarized, i.e., converted to an image that only has two possible pixel values (e.g. black and white). A binary image can be produced in several different ways; the important part is to make sure that the features of interest are well segmented. In this case, what needs to be binarized is the output of the vessel filter, determining which pixels belong to the vein network. This step requires some experimentation and choice of parameters. Figure 3.4 shows some of the binarization results obtained in this trial and error phase.

The objective here was to try different binarization methods and evaluate which one preserves the vessel network features better. The methods that were tested are described in Section 2.1.2. The result of the first method tested is shown in Figure 3.4a and it corresponds to a global binarization using Otsu’s method, which was described in Section 2.1.2. The other tested methods correspond to variations of the adaptive binarization algorithm.

It is usually difficult to extract the lower edges of the sample wings. The feature is very thin compared to other structures so it gets easily fragmented in the binarization step. The issue persists even if the scales used in the vessel filter are reduced, since the edge gets mixed up with other small veins, ruining the whole extraction. Figure 3.4b shows the best results in terms of conserving the lower edge, so the default adaptive binarization algorithm was selected.

An extra step was considered to improve the segmentation of the lower edges. A concept called hysteresis thresholding, which is used in the well
Figure 3.4. Results of different binary segmentation algorithms.
known Canny edge detection algorithm [23], proved to be useful. The idea is to generate two vessel-filtered images, one with a lower scale range than the other. The scale values used were the default ones for a high scale image, and values in the range $[0.5, 11]$ with a step size of 1.5 for a low scale image. These images are binarized with the default adaptive algorithm and slightly thickened with a dilation operation that uses a diamond structuring element of radius of 1 pixel. Then the hysteresis thresholding can take place. Figure 3.5 shows the two images; Figure 3.5a contains low scale features and Figure 3.5b contains high scale features. The hysteresis step takes the image with high scale features and includes low scale features that are connected to the former. This means that only the details that are related to the core features are added to the output. Figure 3.6 shows the result after performing hysteresis thresholding. It can be observed that the lower edge has improved connectivity.

**Skeletonization**

The process of skeletonization is described in Section 2.1.3, and it is applied to the output of the hysteresis thresholding stage by using MATLAB’s `bwmorph` function together with the `thin` operation [24]. The `skel` operation achieves similar results.

As can be seen in Figure 3.6, the target object (vessel network) that needs to be segmented is surrounded by smaller blob objects that should be discarded. The skeletonization process is not able to discard them, therefore, a simple algorithm was developed to solve the problem. The procedure lists all the segments of the image and orders them by length. Then the largest segment is chosen as a base and a set number (default is 10) of smaller segments are added depending on the Euclidean distance between their endpoints. If a smaller segment has an endpoint which is close enough (default distance is 70
Figure 3.6. A wing image filtered and binarized with hysteresis thresholding.
Figure 3.7. Phases of the vessel-network skeletonization.

pixels) to one of the base segment’s endpoints, then it is included in the output. Figure 3.7 shows an example of a skeleton before and after discarding small objects.

3.5.2 Alignment

At this point, the skeleton has been extracted from a sample image. At the beginning of the development of the following stage, it was thought that the skeleton’s branch points would match some of the landmarks. This was effective to find some of the landmarks but it was not reliable enough. Concepts of image registration led to the alignment stage. The idea of using reference images with annotated landmarks and projecting that information onto other samples was the key to develop the following steps.

Aligning a model skeleton with a target skeleton is the objective of the steps to be described in this section. The alignment would allow to find an approximate location of the landmarks by using one or several model samples as reference. The concept of image registration deals with this problem, and there are various algorithms that can be used. In a typical procedure, feature points from a reference image are matched with feature points of the image that is to be aligned, also referred to as the target. This leads to the calculation of a transformation between points of the reference image and points of the target image.

Sampling

It is common that groups of points are necessary to carry out the transformation calculations. This is why the skeletons needed to be sampled and trans-
formed into point clouds. To ensure that the whole structure is preserved, a point cloud sample is produced by randomly selecting a predefined number of points (default is 1000 points) from the skeleton and registering the corresponding 2D coordinates.

**Affine Transformation**

Many algorithms were tested for the purpose of finding a transformation between two point clouds. Among these were variations of the Iterative Closest Point (ICP) algorithm and one called Kernel Correlation [25]. None of these achieved good results on the dataset, especially when compared with the Coherent Point Drift (CPD) algorithm described in Section 2.1.4. This algorithm performed surprisingly well, even when outliers form a large part of the point cloud.

Since the point clouds of the skeletons do not have a true correspondence, and therefore finding a transformation would require a very robust algorithm, the Coherent Point Drift algorithm was used effectively by setting the parameter $w$ to a value of 0.9. This allows the algorithm to perform well with noisy samples containing many outliers.

The performance of the algorithm can be assessed visually on Figure 3.8. Two sampled skeletons are shown before and after affine transformation alignment. The process finds an affine transformation that can be used to locate the landmark location neighborhoods.
Deformable Transformation

The Coherent Point Drift algorithm allows for deformable transformations, which have extra degrees of freedom compared to the previous affine transformation. This makes it harder for it to converge and find a transformation between point clouds, but it can be calculated right after an affine transformation, making the convergence a lot easier. The Coherent Point Drift algorithm also outputs a correspondence vector field that relates each point of one cloud with a point of the other. The Euclidean distance between these corresponding points is calculated to detect outliers. A point is discarded from the target point-cloud if it has no correspondence with a point in the model point-cloud, or if the distance between the corresponding points is over 40 pixels. This procedure was developed to ensure that the deformable transformation was estimated correctly. After discarding outliers, the deformable transformation is calculated using parameter values $w = 0.9$, $\beta = 2$, and $\lambda = 5$, effectively obtaining a more precise fit as shown in Figure 3.9.

Affine and deformable transformations were both calculated using the default values for the maximum iterations (150) and tolerance ($10^{-5}$) parameters.

The deformable transformation improves upon the affine transformation, but the results are not accurate enough. This problem led to a series of experiments and eventually the next step was developed.

Resampling

The idea of resampling comes from trying to align the skeletons by incorporating the new acquired knowledge regarding the landmark location neighborhoods. It seems reasonable that if the skeleton sampling is denser in the
regions around each landmark, the fit will be more accurate around these regions. This was the argument behind the experiment and it proved to be effective.

The algorithm that was developed for the resampling purpose works in the following way. First, the affine transformation is applied to the model landmarks. Next, since the deformable transformation calculation outputs a translation vector, an extra step needs to be developed. This translation vector gives a specific translation value for each point of the cloud, so how can we apply this to the landmark location estimates? The solution was to take each landmark-location-estimate and look for the 100 closest points in the cloud. A weighted average of the 100 translations corresponding to these points is calculated by using the distance from the landmark-location-estimate to each point as a weight and dividing by the sum of the distances. The result is then used to correct the landmark’s estimated location. This is done for every landmark-location-estimate, and then the resampling takes place. For each landmark, 100 random points are chosen at random from the vicinity (radius of 40 pixels). The resampling is done for the model skeleton as well, and then with these two new point clouds, both transformations are calculated again in the same way. Figure 3.10 shows the results of the transformations for the resampled skeletons.

3.5.3 Result Refinement

To refine the result of the described process, five manually marked wings are taken and registered to the wing sample which is being processed. The wing image sample is processed five times with each of these different models and the landmark outputs are combined to obtain a more robust measurement.
Once registered, the landmark locations are transformed from the five wings to the geometry of the observed sample. Each of the five processes gives a number of coordinates that are weighted using the error of the alignment obtained from the CPD algorithm; then they are summed up together to form a weighted average. Doing this has proven to reduce the testing error considerably.

3.6 Machine Learning

This section describes the development of a method based on techniques from what is called the Machine Learning group (Section 2). Based on the research done on existing methods, it was decided that the best option would be to use convolutional neural networks described in Section 2.1.5. The method was developed using Python and Tensorflow, including libraries such as OpenCV and SciPy. Some parts of the implementation correspond to re-purposed code that was found in different collaboration sites.

3.6.1 Architecture

The first step in developing a convolutional neural network approach is to find an architecture that is suitable to solve the problem at hand. The base architecture for this approach was adapted from a convolutional network implementation that achieved great results in a facial keypoints-detection (same as landmark-localization) online challenge. This base architecture is made of 3 convolutional layers, all followed by a max pooling stage, and 3 fully-connected layers including the output. By base it is meant that the architecture represents one unit of the larger network. It is the basic building block of a cascade network inspired in the work presented in Section 2.3.2. Figure 3.11 shows a representation of this base architecture.

The convolutional layers have a depth of 32, 64, and 128 filters starting from the input. Their kernel sizes are 3, 2, and 2 respectively. The three max pooling stages use filters of size 2 and stride 2, therefore since the input image size is 96 by 96 pixels, the subsequent layers receive an input which is 48 by 48 and 24 by 24 pixels. The first two fully-connected layers have 1000 neurons each and the last layer has a number of neurons which is twice the number of landmarks to be output (26 for 13 landmarks). All of the layers except the last one use rectifiers (ReLU) as an activation function \( f(x) = \max(0, x) \).

This tutorial was very helpful: https://navoshta.com/facial-with-tensorflow/.
Next, the base neural network is repeated and arranged as a so-called cascade, in an effort to tackle the issue with the high resolution of the input images. The cascade is divided into 4 levels: the first one acts over the whole input image, locating the 13 landmarks at a very low resolution. Using these localizations, 3 regions/crops are selected from the original image and processed with 3 new networks that find 7, 3, and 3 landmarks respectively. Each of these 3 networks is then divided into one network per landmark, acting over an even smaller region. Now at the fourth and last level, each landmark is located by one network that processes a very precise crop, obtaining the desired accuracy. This means that levels 1, 2, 3, and 4 have 1, 3, 13, and 13 networks respectively. Figure 3.12 presents an abstraction of the level progression.

It is important to mention that the architecture described above is the result of a trial and error process. Starting from the simple base architecture, different experiments and theoretical ideas came together to improve the architecture.
3.6.2 Data Augmentation

Since the project’s database has only 93 wing image samples and a typical convolutional neural network training uses tens of thousands of images, data augmentation needs to be done to create the training set. In addition, validation and test sets were also produced using this technique. As a rule of thumb, around 80% of the dataset is used for training and the rest is split in two for the validation and test sets. The first 19 samples were used for validation and testing and the following 74 samples were used for training.

All sets were augmented in the same way. First, an original sample is cropped and centered. Then it is rotated with a random angle taken from a specific interval (chosen by looking at the dataset’s extremes) and cropped randomly making sure all the landmarks are contained. The process guarantees random rotations, translations, and scales, with some limitations. This is repeated 40 times per sample to create a training pack which has 2960 (74*40) samples. Validation and test packs are produced by repeating the process 20 times, so their size is 380 (19*20) samples. The concept of pack is used in this project for practical reasons. It was easier to have packs with specific sizes that allow fast experimentation and trying various training settings.

Data augmentation is done for every level of the cascaded network, making sure the crops correspond to the regions being processed and that they contain the required landmarks. Examples of the augmented images obtained for sample 20 are shown in Figure 3.13 for level 1, Figure 3.14 for level 2, Figure 3.15 for level 3, and Figure 3.16 for level 4.
Figure 3.13. Examples of data augmentation in level 1.

Figure 3.14. Examples of data augmentation in level 2.

Figure 3.15. Examples of data augmentation in level 3.
The output of each level determines the input for the next level. The coordinates obtained from each network are used to determine a crop of the original image to be resized into a 96 by 96 pixels image and then fed to the next level. These crops have different sizes; level 1 uses the whole image, level 2 uses a crop of 768 by 768 pixels, level 3 uses a crop of 384 by 384 pixels, and level 4 uses a crop of 192 by 192 pixels. All of these crop sizes correspond to the average size of the crop, since it actually is chosen to be a random size which is centered on these values. The ratio between the maximum size and the minimum size is always 1.5.

3.6.3 Training

Training is done in the same way for each of the 30 neural networks. A training pack of images is fed to the network using a learning rate of 0.01 and exponential decay rate of 0.03. The batch size is 36 images and the number of epochs is 1001. This guarantees that every image is processed at least once and some of them twice. The optimization method is called stochastic gradient descent with momentum, which is designed to accelerate the learning process by introducing a variable of velocity that controls the way in which the solution is searched for in parameter space [15]. The value of this parameter variable increases linearly with the epochs from 0.9 to 0.99. In addition, dropout is used to avoid overfitting in all convolutional layers and the first fully-connected layer. This means that each of these 4 layers has a probability of dropping neuron connections in each iteration. The probabilities used starting from the input layer were 0.9, 0.8, 0.7, and 0.5. Initialization of weights was done using the Xavier initializer, an algorithm that helps the process converge by keeping values from exploding or vanishing.

After a pack has been used to train, the validation pack is used to check if the error is low enough. If it is not, a new training pack is generated and fed to
the network. This is done until the error does not go lower or if it has reached an acceptable value which is comparable to manual landmark-localization.

3.6.4 Output

Each of the 30 networks is trained independently and then put together in a cascade as explained earlier. The output comes from the 13 networks in level 4, and each of them gives a pair of coordinates which make reference to a position in the 96 by 96 pixel image that was processed. The coordinates need to be rescaled to the original size and then compared with the ground truth to calculate the real error.
4. Results and Discussion

4.1 Manual Error

Manual landmark annotations for the samples have been provided as described in Section 3.1. The 8 mentioned researchers carried out the landmark annotations on the same samples; each with his/her own technique and precision. For this reason it is interesting to study the differences among the manual landmark-localization procedures; it provides an estimate of the tolerance that is currently accepted in these types of studies, and it also gives a target maximum error for the automatic algorithm.

The first step consisted in loading the manual annotation data into MATLAB so that it could be analyzed. Already at this point, many benefits of using an automatic approach were exposed, since the data contained several errors that a computer can easily avoid. The problems encountered included: missing annotations for some samples, missing landmarks, and mislabelled files among other issues regarding data format consistency. These issues sum up to a certain percentage of missing information that was calculated with respect to the expected amount of data. A set of 93 samples was provided and each one of them was expected to have 13 associated landmarks from 8 individual researchers, that is to say, a total of $93 \times 13 \times 8 = 9672$ landmarks (two-dimensional coordinates). Out of the expected 9672 landmarks, 255 were missing, of which 188 belonged to one specific researcher and the rest were more or less evenly spread out. This means that 2.64% of the information was missing. Since the data was easier to analyze as a whole, it was completed by filling in the missing landmarks using the mean location of the other annotations for each specific sample. In other words, if a landmark was missing for one of the researchers in a specific sample, it was replaced with the average of the annotations provided by the other researchers for that same landmark and sample. The completion of the data certainly introduces some distortions on the results. Nevertheless, the objective is to obtain an estimate of the manual error so as to decide on a soft threshold for the algorithm’s acceptable error.

After the data was completed, researcher annotations were used as references with which one can compare and measure the error. A researcher-reference is compared with all the other researcher annotations, and some statistics are calculated to describe the manual error. The measurement consisted in calculating the Euclidean distance between each manually annotated
landmark and the researcher-reference. For each sample and non-reference researcher (651 total sample annotation sets), a Root Mean Square (RMS) value was calculated as follows:

$$RMS = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\Delta x_i^2 + \Delta y_i^2)}$$  \hspace{1cm} (4.1)

where $n$ is the number of landmarks, $\Delta x_i$ is the mentioned Euclidean distance for the first dimension, and $\Delta y_i$ is the one for the second dimension. This number gives a rough idea of a sample’s annotations error, and it will be useful to compare with the automatic approach’s landmark-localization procedure.

Figure 4.1 presents two histograms (two researcher-references) which show the sample count corresponding to each level of RMS error in pixels, together with a legend showing the mean, standard deviation, maximum, and minimum. The mean RMS error lies around 9 pixels. To check if this value was representative and not just linked to the choice of researcher-reference, the statistics were generated using every researcher as a reference. The mean RMS error would always stay between 8 and 11 pixels and the standard deviation around 3, except for one case in which it reached 4.26.

In an effort to show the spread of the error among the different manual annotations, Figure 4.2 presents two stem plots in which the RMS error data can be grasped in a compact form. Figure 4.2a and 4.2b do not differ much in the information they show; it is just another way of showing the results. There is one detail that can be extracted though, and that is the fact that some researchers have similar or different ways of annotating the landmarks. It is not easily discernible but there are definitely signs of low or high error for some researchers. For example, in both plots in Figure 4.2, the researcher Majki (illustrated with yellow circles) seems to obtain high errors when compared to these specific researcher-references. This does not mean that the annotations are wrong; it probably points to the idea that Majki uses a different annotating style. It is difficult to know if each researcher’s style is consistent across all the samples, and a Procrustes analysis might help to determine that. This analysis is not part of the project, although it could be a good addition to the future work list.

A target mean RMS error and standard deviation can be derived from these results. If the automatic algorithm performs with a mean RMS error in the vicinity of 9 pixels and a standard deviation close to 3, it could be said that the procedure operates as good as a researcher and therefore be able to replace
(a) Histogram for a researcher-reference (Zorana).

(b) Histogram for a researcher-reference (MaraT).

Figure 4.1. Histograms of the RMS error in pixels for manual annotations: two researcher-references.
(a) Stem plot for a researcher-reference (Zorana).

(b) Stem plot for a researcher-reference (MaraT).

Figure 4.2. Stem plots of the RMS error in pixels for manual annotations: two researcher-references.
The RMS error described earlier corresponds to a calculation for each sample. Similarly, a landmark RMS error was calculated to explore how does each landmark perform individually. The idea is to check whether some landmarks are more difficult to pin-point and if their location varies across researchers. Figure 4.3 shows the RMS error for each landmark across all samples and researchers (one plot for each reference).

The main observation to mention from Figure 4.3 is the fact that the landmarks that have a clear lower RMS error correspond to locations in the fly’s wing that are easier to detect and do not require any thorough description to be able to identify them. These locations are usually small vein intersections. Landmarks that correspond to joints or thicker vein interesections and endings, inevitably introduce some doubt on the researcher’s side and location variation is therefore higher.

A researcher might want to use this information obtained from the work on the project, to re-define the landmark locations in order to reduce the error. In addition, the computer algorithm developed for this purpose can handle a higher number of landmarks without increasing the processing time considerably. This means that this work can change the way in which the research is carried out, since the conditions and time limits change.

4.2 Classic Image Analysis

4.2.1 Algorithm’s Error

The error for the classic algorithm’s case was measured in almost the same way as it was done for the manual error. The algorithm’s result is treated as an extra researcher’s work, with a couple of details that need to be mentioned.

Since the reference models include manually annotated landmarks which belong to a specific researcher, the reference landmarks used to calculate the error are taken from the same researcher. This is done to keep the researcher’s style/preference and in hopes of minimizing the error. In addition, only the algorithm results are compared with the corresponding reference; in contrast with the previous comparisons that involved seven researchers compared to one researcher-reference.

With the intention of comparing performance measure with respect to two different researcher-references, two histograms are shown in Figure 4.4, one for each researcher-reference, and both illustrating the RMS error sample.
Figure 4.3. Landmark RMS error in pixels, for manual annotations, for two researcher-references.

(a) Landmark error for a researcher-reference (Zorana).

(b) Landmark error for a researcher-reference (MaraT).
counts and some statistics obtained from the results. A direct comparison with Figure 4.1 needs to be done carefully. The previous histograms consider a higher number of landmark-sets being compared, so it can be assumed that the statistics shown have a higher confidence. Nevertheless, the classic algorithm has been run many times with very similar results; indicating that the statistics are a good approximation of the algorithm’s real performance. With these cautions in consideration, it is clear that in terms of the calculated statistics, the algorithm performs almost as good as the researchers.

It is very important to mention that the classic algorithm was initially optimized by choosing the reference models, which were five. After that, parameters were optimized based on the choice of reference models and their results. This was the way in which progress accumulated slowly over the course of each experiment, and therefore a dependency might exist between model choices and parameter values. It is desirable that the dependency is weak, and that seems to be the case. The chosen parameters perform well when conditions change, a fact that will be supported by evidence in Section 4.2.2.

As with the manual error case, stem plots were generated to express the results in a different fashion. Figure 4.5 presents the data in a way that specific sample RMS errors can be examined and comparisons can be made across two different references. For example, it can be observed that sample 71 has a high error in both cases, pointing at the probable fact that the sample might be the problem. Other samples obtain very low errors, which in many cases means that the specific sample is used as a reference model. This is the case in Figure 4.5b, were the five lowest errors coincide with the five reference models being used. Using reference models in evaluation indeed introduces some amount of bias, which should be taken into account when analyzing the results.

One of the samples of the dataset was very difficult to process due to its special characteristics. Some images contain small objects that do not belong to the wing being imaged, and usually the classic algorithm can get past them. This is not the case for sample number 89; reason for the plots in Figure 4.5 to show only 92 RMS errors of the 93 that were expected. The algorithm has a stochastic component in the skeleton sampling procedures, so the results can vary a bit among executions. In some cases and using specific parameters, sample 89 can be processed with relative success, but the purpose of the algorithm is to achieve generalization power. Therefore, it was decided that sample 89 would be treated as an outlier. The original image of sample number 89 and its skeleton are presented in Figure 4.6. It is easy to see why the protruding object distorts the skeleton, making it difficult for the alignment step to find the correct transformation. This problem also shows that there is room for
Figure 4.4. Histograms of the RMS error in pixels, for the classic algorithm, for two researcher-references.

(a) Histogram for a researcher-reference (Zorana).

(b) Histogram for a researcher-reference (MaraT).
Figure 4.5. Stem plots of the RMS error in pixels, for the classic algorithm, for both annotation references.

(a) Stem plot for a researcher-reference (Zorana).

(b) Stem plot for a researcher-reference (MaraT).
improvement in the skeletonization process.

Focusing on the results for each landmark individually gives a slightly different view than the one observed in the manual error case. In the classic algorithm’s case, landmark errors are more similar to each other in comparison to the manual case. The tendency noted in Figure 4.3 is also present in Figure 4.7 but in a milder flavor. Both plots look quite similar, showing some evidence for the independence between the algorithm’s procedures and the choice of researcher-reference. Landmarks with higher RMS errors might indicate that they are harder to align in the non-rigid transformation step. The opposite can be said for low-error landmarks such as landmark number 4.

Using the classic algorithm involves many different choices that have to be made intuitively and by trial and error. There is an enormous amount of combinations of models and parameters that can be tested. Even small refinement routines can be experimented with at every step, introducing lots of variation and possibilities for the algorithm. This means that the configuration used cannot be declared the optimal setting, and whether the result obtained is good enough, there is a lot of space for exploration and improvement.

While optimizing the classic algorithm, a base configuration is used as a starting point. It was mentioned earlier that the first step usually involves choosing acceptable reference models that are able to align with the majority of the samples. From this point on, parameters are varied and small fixes are added so that the results improve. This kind of process always includes a fear of overfitting the algorithm to the dataset; losing the desired generalization power. It would be highly desirable if the algorithm could also be used for other types of wings coming from other fly species. These ideas served as inspiration for the experiment presented in Section 4.2.2.
Figure 4.7. Landmark RMS error in pixels, for the classic algorithm, for two researcher-references.

(a) Landmark error for a researcher-reference (Zorana).

(b) Landmark error for a researcher-reference (MaraT).
4.2.2 An Extra Dataset

In order to test the usability of the classic algorithm under sample variation and compare datasets, it was decided to experiment with other samples. These were taken from the dataset provided in the software package Wing 4\(^1\) [19]. All the 100 samples on the dataset are 8-bit gray images that have been stored in the TIF format. The size of the images is 640x480 pixels, which has the advantage of avoiding a resizing step. The smaller size produces a trade off between ease of processing and precision potential.

A sample image of the dataset was already shown in Figure 2.3. Section 2.2.2 explained one of the approaches used to solve the problem using samples from this specific dataset. It is worth to mention again that the software provided might be very effective under certain conditions, but it requires expertise and tedious tuning of parameters. This is one of the recurrent weaknesses of this kind of softwares, one that has been taken into consideration extensively during the development of this project.

The objective of the experiment was to use the classic algorithm on the new dataset with as few parameter changes as possible. This would allow to test the algorithm’s ability to generalize over different datasets. First of all, the samples needed to be manually annotated; task that was carried out carefully so as to achieve the best results possible. Moreover, it is evident that the reference models needed to be changed, and this proved to be the most challenging part of the process. Five models were chosen randomly from the dataset and then through trial and error the ones that performed worst were changed by new ones. This was done until a good enough initial affine transformation (first alignment step) could be found for all the samples of the dataset. This required the change of one parameter; the only one that changed value. The parameter corresponded to the affine transformation error threshold used to decide whether an initial affine transformation is good enough, which had to be increased considerably to tolerate worse matches. Apparently the models and samples produce a high amount of outliers in the skeletonization process, reason for the need of a higher tolerance. Here lies another argument to put extra work into the skeletonization process in future developments.

After several runs of the classic algorithm, the results seemed stable and the best outcome was chosen to be displayed in Figure 4.8. At a first glance, the results look a lot better, but the scale needs to be considered to carry out a fair comparison. What can be said is that the algorithm can definitely be used for other datasets that share some basic essential characteristics; especially ones that translate into a healthy skeletonization process, such as vein contrast and

\(^1\)Latest version of the software mentioned in the cited paper. The software and dataset are available at: http://www.bio.fsu.edu/~dhoule/wings.html.
In order to compare results with the original dataset, a normalization of both results is carried out by scaling coordinate values. In detail, what is done is to divide column coordinates by the width of the sample images, and row coordinates by the corresponding height. Then the values are multiplied by 100 so as to have a projection of the coordinates onto a 100x100 square, in which the new values express percentage of the total width or height in the original scale. The RMS error is then calculated as usual, obtaining values that are comparable across datasets with different scales. Histograms for the RMS error on the original dataset and the new one are presented in Figure 4.9.

The mean RMS error is now much easier to compare and the result is still better in the new dataset. Even though the choice of models was harder to make for the new dataset, and the algorithm’s stability with them seemed less reliable, this result is strong evidence that the algorithm has generalization power. The algorithm can confidently replace manual landmark-localization if appropriate precautions are taken into account and special attention is given to the reference model choices.

4.3 Machine Learning
4.3.1 Algorithm’s Error

This algorithm’s error was measured exactly in the same way as for the previous classic algorithms. Using the same dataset and researcher references, the errors can easily be compared to each other.

As was done previously, two histograms were generated for the machine learning algorithm. These show the RMS error sample counts and some statistic information about the results. Figure 4.10 shows both these histograms which are directly comparable to the results of the previous algorithm presented in Figure 4.4. The mean error is lower but the standard deviation is higher due to the high errors obtained for some problematic samples. It is not clear why some samples are difficult to process. Maybe the high errors come from samples that are not well represented in the training set. It was expected to see that using MaraT as reference would result in a higher error, since Zorana’s landmarks were used to train the network.

It is important to mention that the comparison between algorithms is not completely fair. Both algorithms use some of the dataset’s information to process the samples. The classic algorithm uses the reference models and the
Figure 4.8. Histogram of the RMS error in pixels and stem plot, for the classic method, on the new dataset.
(a) Histogram of the normalized RMS error for the original dataset (MaraT reference).

(b) Histogram of the normalized RMS error for the new dataset (Sebastian reference).

Figure 4.9. Histograms of the normalized RMS error, for the classic method, on the original dataset and the new dataset.
Figure 4.10. Histograms of the RMS error in pixels, for the machine learning based method, for two researcher-references.
machine learning algorithm is based on a training set which contains around 80% of the samples. Even though the latter algorithm performs better, it seems that it uses more of the original information in the dataset. A fair test set can be generated by taking the first 19 samples (non-training samples) and discard the ones that are used as reference models in the classic algorithm. The problem with this idea is that the result would not be statistically significant. The amount of samples would be too low. If the histogram were to be plotted using only the first 19 samples, the mean error would climb up to 11.00 and the standard deviation would increase to 7.98. Even though the increase is considerable, it still performs within the acceptable range, taking into account that it is not an accurate measurement.

Stem plots were also generated for this case. It can be observed in Figure 4.11 that clearly the standard deviation of the error is higher. The majority of the samples have a low error across the whole dataset, but some of them have a very high error. The differences among samples for this algorithm are a lot higher, and, therefore, detecting wrongly-processed samples would be even more beneficial for this algorithm.

There are two samples within the test set that have a considerable error; samples 7 and 15. These were analyzed thoroughly to try and understand why their process fails. The reason for this was not found due to the nature of the machine learning algorithm. The neural network operates like a black box that is not trivial to unfold. It is easier to change parameters and adjust the dataset to find a better solution. Figure 4.12 presents the results for these two samples. There are four sets of landmarks that correspond to each level of the cascaded network. Level 1 is red, level 2 is yellow, level 3 is blue, and level 4 is green (final result). Usually, the landmark’s location prediction starts far away and closes in progressively. This is not the case for landmarks 4 and 5 in sample 7 (Figure 4.12a). Similarly, the process for sample 15 (Figure 4.12b) also fails to lock into some landmarks. In this case, level 3’s prediction looks like it is in the right direction, but then level 4’s output fails for landmarks 2, 9, and 10.

Compared to the manual algorithm, the classic algorithm does not show a very clear strength or weakness when dealing with specific landmarks. Similarly, the machine learning algorithm’s landmark errors are close to each other, but the best and worst landmarks are different. Figure 4.13 shows the landmark errors for the machine learning algorithm using two researcher-references. The fact that there is no strong tendency might mean that either the result is not statistically significant, or the algorithms just behave this way for the specific dataset. The only conclusion that can be stated is that landmark 13 is always difficult to locate.
Figure 4.11. Stem plots of the RMS error in pixels, for the machine learning based approach, for both annotation references.
Figure 4.12. Four sets of landmarks that correspond to each level of the cascaded network (Level 1: red, Level 2: yellow, Level 3: blue, and Level 4: green). Convergence analysis for samples 7 and 15.
(a) Landmark error for a researcher-reference (Zorana).

(b) Landmark error for a researcher-reference (MaraT).

Figure 4.13. Landmark RMS error in pixels, for the machine learning based approach, for two researcher-references.
4.4 Algorithm Comparison

At the implementation stage, the classic algorithm clearly takes more time to develop. Incorporating knowledge about the problem into the algorithm is a really difficult endeavor that the machine learning approach tries to avoid. The development is carried out by processing a small sample of the dataset, trying to be as general as possible so as to avoid overfitting. The machine learning algorithm’s design only needs a well prepared input and a desired output to train. It is a lot simpler to obtain a functioning implementation.

A good idea to have a fair comparison of the algorithms is to compare the main outputs with all the manual references. This is essentially what was done for the manual error’s analysis. So, in this case the errors are calculated by comparing each algorithm’s results with each of the 8 human annotations. That is to say, there are 8 calculations for each of the 93 samples (744 error values).

For both algorithms, the main results are the ones that use Zorana’s annotations as ground truth. The histogram obtained for the errors is shown in Figure 4.14. The algorithms perform almost the same under this comparison, but it is important to mention that the classic algorithm does not include sample 89, since it fails at processing it. With this in mind, it can be concluded that the main difference in performance at this level is the standard deviation of the errors. The classic algorithm has a higher precision and the machine learning algorithm has a better accuracy.

Stem plots were also generated for this comparison and they are presented in Figure 4.15.

In addition, sample 4 was chosen to show the output of both algorithms and compare them visually. It was chosen because it belongs to the test set, it is not used as a model reference, and the resulting errors represent the mean for each algorithm. The error for the classic algorithm is 10.41 and the error for the machine learning algorithm is 8.03. Figure 4.16 presents the wing with the landmark locations predicted by each algorithm. Main differences can observed for landmarks 7, 8, and 12.

In terms of execution time, running on an Intel Core i7 processor (2.2 GHz) with 16 GB of RAM memory, the classic algorithm takes around 114 seconds per sample, and the machine learning algorithm takes 52 seconds per sample. The classic algorithm was run using 5 model references, which usually take around the same time each. So the time can be reduced significantly if the required accuracy is achieved with less models. The machine learning algorithm uses 30 neural networks that take similar times to process, therefore,
(a) Classic algorithm’s error histogram for all researcher-references.

(b) Machine learning algorithm’s error histogram for all researcher-references.

Figure 4.14. Comparison of the algorithms: Histograms of the RMS error in pixels for all researcher-references.
(a) Classic algorithm’s stem plot for all researcher-references.

(b) Machine learning algorithm’s stem plot for all researcher-references.

Figure 4.15. Comparison of the algorithms: Stem plots of the RMS error in pixels for all researcher-references.
Figure 4.16. Comparison of the algorithms: Output of the two approaches, sample 4.
a reduction in time would probably need the removal of some networks and redesign the level progression. It seems like a better base architecture could allow reducing the amount of levels. In addition, execution speed was never the main priority throughout the project so these execution times can be easily improved by working on basic optimizations. For example, parallelization could be used in both algorithms to hugely improve the speed of the processes.
5. Conclusions

The main objective of this project was to try and answer the following question:

**Is manual annotation replaceable by automatic landmark-localization?**

After developing two different paths to automatize the process, the answer is yes. A computer can replace a human in this task and perform in the same precision and accuracy range. Errors will always be present though, and these are different for humans and computers. Landmark-localization problems need to be prepared and adapted for the particular methods used. Both algorithms developed are very sensitive to the input and require that it remains similar to the training dataset. A computer cannot easily adapt to a new situation or unique input that it has not been designed to process.

The fact that the dataset is quite small and it has been used in different ways to train the computer methods, makes it difficult to have statistically significant performance measurements. This would require a separate dataset with thousands of labelled wing samples, which are not available at this point but would be interesting to process in a future project. Nevertheless, the performance of both algorithms are very close to a human’s performance.

There was considerable effort in trying to automatize the error detection process. Some redundant process can be used to check if the algorithms performed correctly. This proved to be very challenging since the samples that lead to problems were usually outliers; samples that were not very well represented in the dataset. This brings us back to the preparation of the input; it’s important to describe and understand it so that the algorithm can be designed to process all variations. In the case of the machine learning approach, data augmentation tries to deal with this issue. Sample variations can be very subtle and difficult to mimic though, and here is were the main errors arise.

Future work should be focused on error detection. This is the main advantage a human has over the proposed algorithms. A human can check for errors easily by going through the samples a second time. If the computer is able to check for errors effectively, it would be a lot better than a human, given the input restrictions stated beforehand. This is definitely possible with some
more work.

It is certainly a good idea to combine both approaches and have some redundancy in the process. This way it would be possible to take advantage of each approach’s strengths and create a more robust solution.
References


