Image Processing to Detect Worms

Javier Fernández
Abstract

Image Processing to Detect Worms

Javier Fernández

The nematode C. elegans is a widely used model organism. It has many cells with human equivalents, making it possible to study pathways conserved in humans and related conditions. Being small and transparent, it also lends itself well to a variety of high-throughput screening techniques. Worm identification should be as automated as possible since it is too labor-intense and time-consuming to do it manually.

Here we present an image processing methodology to detect C. elegans in high-throughput microscope images. The provided semi-automatic solution makes it possible to effectively identify individual worms in worm clusters. In general terms, the process is as follows: A given image is segmented, thus separating groups of worms from the background. Individual worms are detected automatically, following a worm-shape matching process. For worm clusters, the matching process is based on finding feasible worm shapes by minimizing the distance between the cluster and generic worm shapes, which are deformed to fit it. Wrong and missing conformations can be quickly fixed manually.

The provided methodology is a novel approach to successfully detect individual C. elegans worms in high-throughput microscope images. Results show that this semi-automatic solution makes it possible to fit the shape of 100% of worms in the image, unlike previous automated methods that reach, at most, less than 90% in average, for similar test sets. The detection process is usually achieved in less than half a minute for difficult images. For easier images, the total match can often be calculated in a fully automatic way. Time cost and matching accuracy are considerably improved with respect to manual identification.

The solution was implemented in Java and adjusted to Endrov, which is an open source plug-in architecture for image analysis, and is to be used at the Department of Bioscience and Nutrition, Karolinska Institute, Sweden.
## Contents

1 **Introduction** 1  
1.1 Motivation and Purpose 1  
1.2 Earlier Work 3  
1.3 Structure of Thesis 5  

2 **Theoretical Framework** 7  
2.1 Endrov 7  
2.2 Thresholding 8  
2.3 Distance Transform 10  
2.4 Skeletonization 12  
2.5 Shape Matching 14  
2.6 Shape Descriptor 17  
2.7 Splines 19  

3 **Methodology** 21  
3.1 Solution Methodology Design 21  
3.1.1 Previous Reasoning 21  
3.1.2 Methodology Description 23  
3.1.3 Thresholding 27  
3.1.4 Distance Transform 28  
3.1.5 Worm Skeletonization 31  
3.1.6 Worm Segmentation 32
3.1.7 Worm Shape Descriptor ........................................ 40
3.1.8 Shape Rasterization .......................................... 44
3.1.9 Profile-Driven Shape Fitting ................................. 45
3.1.10 Manual Adjustment ......................................... 51

4 Experiments and Results ........................................... 53
  4.1 Experiments ..................................................... 53
  4.2 Results ........................................................ 55
    4.2.1 Initial Processing ...................................... 55
    4.2.2 Shape Matching ........................................ 57
    4.2.3 Matching Energy ........................................ 62

5 Conclusions and Future Work .................................... 69
  5.1 Conclusions ................................................... 69
  5.2 Future Work .................................................. 72
List of Figures

2.1 Gray-scale image before and after a thresholding effect is applied 10
2.2 The distances from a point for the six distance transforms . . 11
2.3 Horse binary shape and skeleton . . . . . . . . . . . . . . . . . 13

3.1 Graphical description of the solution methodology for the shape
detection of C. elegans worms in digital images . . . . . . . . 24
3.2 Worms in liquid media. Original image and binary image ob-
tained through percentile thresholding with a percentile value
of 0.074 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
3.3 Binary Image and three distance transform metrics from a
single worm image . . . . . . . . . . . . . . . . . . . . . . . . 30
3.4 Skeleton obtained through iterative thinning over a worm bi-
nary image . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
3.5 Three different directional neighborhoods . . . . . . . . . . 36
3.6 Construction of a worm shape based on shape descriptor . . 42

4.1 Best automatic shape matching on test image 1 . . . . . . . 60
4.2 Best automatic shape matching and manual match fixing on
test image 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
4.3 Best automatic shape matching and manual match fixing on
test image 3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
4.4 Energy value for the best three conformations by endpoint ID
on test images . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
List of Tables

4.1 Characteristics of the test set images . . . . . . . . . . . . . . . 54
4.2 Best percentile values for percentile thresholding of test images 56
4.3 Worm endpoints detection and fixing for the test set . . . . . . 57
4.4 Results of automatic worm shape matching on test image 1 . . 59
4.5 Results of automatic worm shape matching on test image 2,
   with and without missing endpoints . . . . . . . . . . . . . 61
4.6 Results of automatic worm shape matching on test image 3,
   with and without missing endpoints . . . . . . . . . . . . . 61
Chapter 1

Introduction

1.1 Motivation and Purpose

C. elegans is a widely used model organism. It has the advantage of all worms being exactly the same down to the cellular level, short life cycles and rapid genetics. Thus, small deviations from wild-type can be detected and experiments are cheap compared to higher organisms. Nematodes have many cells with human equivalents making it possible to study many pathways conserved in humans, and related conditions e.g. drug addiction, aging, dysfunction of certain proteins etc. Being small and transparent it also lends itself well to a variety of microscopy-based high-throughput screening techniques.

Before quantification, the worms have to be identified; this should be automatic since there are too many worms to feasibly do this manually. Despite the utility of C. elegans models for genetic manipulations, the deployment in high-throughput screens has been limited by labor-intensive manual assays used to score phenotypes. This urges the need for more rapid and more consistent methods. Hereby, a computer program that identifies C. elegans individuals in digital images would provide an automated solution for the
recognition task, thus vastly improving the time-cost and accuracy with respect to manual identification and converting the images into manageable data.

The focus of this work is to design and implement an image processing methodology to detect C. elegans worms shapes in microscope images. It is easy for a human to find the worm conformation, rotation and direction. In this project we will only try to find the conformation. We will investigate if is possible to detect and fit the conformations of worms in an automated way and if this can be achieved faster than manual identification.

In this project we consider larval worms in microtiter plates. The larvae are grown in liquid culture, which causes the microscope image background to be very clear. However, worms overlap frequently. Ordinary bright-field images are used for the morphology. The implementation is adjusted to Endrov, an open-source plug-in architecture aimed for image analysis and data processing, developed and used at the Department of Bioscience and Nutrition, Karolinska Institute, where this project is carried out.

- **General Objective**
  - To design and implement an image processing methodology to detect and fit the shape of worms in digital images.

- **Specific Objectives**
  - To design an algorithm based on image processing techniques that receives images of worms in liquid culture as input, and outputs fitted shapes of these worms.
    - * To review the background on image segmentation techniques.
    - * To design a shape descriptor and a rasterization method to represent worms in numerical terms.
* To review the background on shape matching and object recognition and propose a matching approach.

– To implement the designed algorithm as a plug-in for Endrov

1.2 Earlier Work

C. elegans has been deployed quickly in genetic screens and chemical screens, as mentioned in [4]. In the previously mentioned paper, the strategies for automated analysis of C. elegans locomotion are divided in three groups according to their methodological approach: Tracking overall behavior, detection and measurement of distinct behaviors and measuring the complete behavioral repertoire using large parameters sets. All of these strategies are based on an initial processing step that performs the detection of the worm shapes in the images. This usually involves the extraction of the worms from the background (segmentation), reduction to a skeleton and parameterization of worm outlines.

Reduction to skeleton and subsequent parameterization have become a standard method. However, since the image properties such as lighting, noise and clutters (e.g. worm tracks and eggs), can vary strongly from one image to another and the segmentation depends directly on the visual context, the parameters for this process are highly variable. The segmentation methods that are usually used on worm images are thresholding, morphological closing, hole filling, and their combinations. It is stated in [4] that among those programs that track multiple worms, few attempt to resolve the problem that arise when worms interact or when individual worms coil up on themselves, which may severely affect the individual worm identification. In [8] is indicated that programs and algorithms are being developed to address this problem.

The numerical description of a worm shape or worm parameterization de-
termines the range of possible shapes a worm may adopt. Diverse representations have been used from one approach to another. The most common is the reproduction of an abstract shape, normalized for position, orientation, scale and parameterization of the worm skeleton.

Very recent studies present new approaches for detecting individual worms in cluttered clusters. Riklin Raviv et al. [17] present an approach for extracting cluttered objects based on their morphological properties. This study addresses the problem of untangling C. elegans clusters in high-throughput screening experiments. The method is based on concepts from machine learning and graph theory. The worm skeleton is used as shape descriptor. The clustered worm segments are represented as graph vertices and then a search for more likely worm paths in the graph is carried out. The detection of the most likely worm descriptors within the graph search is guided by a probability distribution, defined by a low dimensional feature space that captures the worms’ variability. This probabilistic shape model and a similar worm detection approach was first presented by Wählby et al. in [28].

There are many studies in computer vision dealing with the automated analysis of C. elegans and nematodes in general. Most studies are based on worm locomotion, so the process of identification and tracking is performed by the simultaneous analysis of a set of images, rather than just one. There is a standard general strategy followed for an initial worm identification step consisting in segmentation, skeleton reduction and shape parameterization, while the matching strategies depend on the approach and normally involves image sets and not individual images, as explained. Although some automated worm detection approaches are able to identify individual worms and group of worms, few of them attempt to overcome the problem of worm interaction and none solves it successfully.
1.3 Structure of Thesis

This document is divided the following way:

• **Chapter 1: Introduction**
  The motivation and purpose of the thesis is discussed. Then, earlier work on worms detection is presented, pointing out different approaches, achievements and current problems.

• **Chapter 2: Theoretical Framework**
  The theory related to the problem and the solution is explained. Each topic is described, and the different studied approaches are addressed.

• **Chapter 3: Methodology**
  The proposed solution is discussed and presented as a methodology. Then, each step involving the methodology is explained thoroughly, comprising a description of the approach and some implementation details.

• **Chapter 4: Experiments and Results**
  A series of experiments to test the performance of the solution are presented. The purpose and characteristics of the experiments are described. Then, the results are shown and discussed.

• **Chapter 5: Conclusions and Future Work**
  The conclusions obtained from the discussion of the results are presented. Suggestions for future work are given.
Chapter 2

Theoretical Framework

2.1 Endrov

Endrov is an open source plugin architecture aimed for image analysis and data processing. Being based on Java, it is portable and can both be run locally and as an applet, as mentioned in [31]. It grew out of the need for advanced open source software that can cope complex spatio-temporal image data, mainly obtained from microscopes in biological research. Endrov aims to improve the features of the standard open source image analysis program, ImageJ, by providing a more modern design.

The main issues with ImageJ are that it does not support metadata, there is no real support of 5D, the plugin architecture is messy, views cannot easily be extended and the batch process is difficult, as pointed out in [7]. Other problems that inspired the creation of Endrov were: the lack of a standardize image format and the difficulty to store complex data in the existing open formats. The development group created the OST format in order to handle the large image sets. OST is now a tree based object file format and it can store any type of data, but is optimized for images. Reading and writing single image panes is fast and there is no need to read everything into memory, [7].
Endrov is both a library and an imaging program. The design has made strong emphasis on separating GUI code from data types, filters and other data processing plug-ins. The idea is that the program can be used for most daily use or prototyping, and for bigger batch processing or integration, [31].

Endrov was developed at the TBU Group, Karolinska Institute and was officially released on 17 June 2009, under BSD license.

2.2 Thresholding

Thresholding is a process of image segmentation that can be used to create binary images from gray-scale images. A binary image is a type of discrete image in which every pixel has assigned one of two possible values (typically 1 or 0) depending on whether the pixel belongs to the foreground or to the background of the original image.

As stated on [33], during the thresholding process individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled ”object” if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside [20].

In image processing applications where the study is focused on particular objects contained in an image, thresholding becomes a simple tool to separate these objects from the background, but not always accurate. Commonly, the gray levels belonging to the object are substantially different from the gray levels of the background pixels. In [19, p.146] many thresholding applications in image processing are mentioned such as: document image analysis, where the goal is to extract printed characters, logos, graphical content, or musical
scores; map processing where lines, legends and characters are to be found; scene processing, where a target is to be detected; and quality inspection of materials, where defective parts must be delineated, among many others.

The key parameter in the thresholding process is the thresholding value (or values for threshold inside approach). The value can be automatically computed, what is called automatic thresholding, as well as set or tuned through user input.

According to the information they are exploiting, the different thresholding methods can be categorized. In [19, p.147], Sezgin and Sankur categorize the thresholding methods in six groups:

- Histogram shape-based methods: the peaks, valleys and curvatures of the smoothed histogram are analyzed.

- Clustering-based methods: gray-level samples are clustered in two parts as background and foreground, or modeled as a mixture of two Gaussians.

- Entropy-based methods: algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and the binary image, etc.

- Spatial methods: use higher-order probability distribution and/or correlation between pixels.

- Local methods: adapt the threshold value on each pixel to the local image characteristics.

In fig. 2.1, two images are shown that correspond to a gray-scale image and binary image obtained by thresholding.
2.3 Distance Transform

A distance transform or distance map is a representation of a digital image in which each pixel has a value corresponding to the distance to the nearest non-object pixel. It is calculated from a digital binary image, consisting in object and non-object pixels. The object pixels can be considered as foreground and the non-object pixels as background. The obtained image is then a sort of gray-scale representation of the foreground pixels in the binary image.

The pixel mapping depends mainly on the distance metric, which is the measurement method of distance between image pixels. Different metrics have been studied to find distance maps such as City Block or Manhattan, Chessboard, Euclidean, Chamfer 3-4, Octagonal, among others.[3, p.363]. There exist great amounts of distance metrics of other kinds that are useful for different purposes. These are commonly derived from the previous. Fig. 2.2 shows the images obtained by applying different distance metrics.

As stated in [6] distance transforms play a central role in the comparison of
Figure 2.2: The distances from a point for the six distance transforms. The lighter the color the larger the distance [3, p.365]
binary images, particularly for images resulting from local feature detection techniques such as edge or corner detection. For example, both the Chamfer and Hausdorff matching approaches make use of distance transforms in comparing binary images. Distance maps can also be interpreted as landscapes of islands where the label of every pixel indicates the height of the region. This allows the detection of ridges and peaks which is a straightforward way to find the skeleton of an object.[1, 237]. The nature of distance transforms in which the objects are represented as contour layers of different depth makes them also a useful tool for edge analysis and to improve efficiency of morphology algorithms such as thinning and thickening.

2.4 Skeletonization

A skeleton is a compact and simple representation of an object that consists of a thin version of it that is equidistant to its boundaries and preserves many of the topological and geometrical characteristics of the original image, as explained in [30, 10, 23]. Usually the skeleton is defined as the centers of maximal discs contained in the original image, [10, 23]. Regardless of the definition that is adopted, if the skeleton points are attributed with their distances to the original boundary of the object, the skeleton can be used to exactly reconstruct the original shape. Figure 2.3 shows a skeleton calculated from a horse shape and the original binary image.

Depending on the way they are produced, skeletons can be categorized in different types. Telea et al describe three types in [23], such as: morphological thinning, geometric methods and distance transform. The morphological thinning methods iteratively peel off (or reduce) the boundary, layer by layer, identifying points whose removal does not affect the object’s topology. These are usually straightforward and normally require intricate heuristics to ensure
the skeletal connectivity, as mentioned in [23]. Two fast parallel approaches that ensure connectivity using the \textit{morphological thinning} method are described in [5] and [35].

The \textit{geometric methods} compute the Voronoi diagram of a discrete polyline-like sampling of the boundary. A Voronoi diagram is the boundary’s medial axis. “Such methods produce an accurate connected skeleton, but are fairly complex to implement, require a robust boundary discretization, and are computationally expensive” [23, p.251].

Another method computes \textit{the distance transform} (see Sec. 2.3) of the objects boundary. The common approach consists in finding the ridge points and connecting them [22, 2, 1]. Usually they can ensure the accurate localization of skeleton points but neither connectivity nor completeness.

Skeletons are important for object representation and recognition in different areas, such as: computer vision, image analysis, and digital image processing, including optical character recognition, fingerprint recognition, visual inspection, pattern recognition, binary image compression, and protein folding [18].
2.5 Shape Matching

Shape matching is a central problem in visual information systems, computer vision, pattern recognition and robotics [27]. It consists of identifying the area or contour of a specific shape or class of shapes in an image, and plays a fundamental role in content extraction from images and content-based image retrieval. In [26], Veltkamp explains that shape matching deals with transforming a shape and measuring the resemblance with another one, using some similarity measure, that normally correspond to the notion of distance between shapes.

The concept of shape is abstract, but most approaches in shape matching represent a shape as a geometrical object. This can be both a set of points, curves, surfaces, solids etc. and a geometrical pattern modulo some transformation group, in particular similarity transformations (translation, rotation and scaling), as is stated in [26]. Usually a geometrical pattern of a shape called shape descriptor is used to represent the class of the matching object. There are different types of shape descriptors depending on the information they supply and the nature of the problem (see Sec.2.6)

There are different studied approaches to the shape matching problem. The emphasis on this section will be on the computational geometry based approaches for shape matching, since is the most related with the approach followed in this thesis work. Computational geometry studies algorithms that can be declared in terms of geometry.

In [27] Veltkamp and Hagedoorn mention different approaches of shape matching such as: tree pruning, the generalized Hough transform or pose clustering, geometric hashing, the alignment method, statistics, deformable templates, relaxation labeling, Fourier descriptors, wavelet transform, curvature scale space and neural networks. They also categorize the matching techniques in two main groups: global image transforms and global objects.
methods. The global image transform group refers to the techniques that “transform the image from color information in the spatial domain to color variation in the frequency domain”. These approaches do not represent the shape explicitly for matching, instead they represent color or intensity transitions in the image. This makes it impossible to measure the difference of two images in terms of shape as well as to match a shape with a specific part of an image.

On the other hand the global object methods work with a complete object area or contour and can analyze specific areas in the image instead of requiring processing the whole image as in the global image transforms. In order to perform a proper matching, the objects in the image have to be completely and clearly segmented. Some of these methods are: moments, where an object is described as a set of moments, modal matching, where the boundary is used instead of the area and is described with Fourier descriptors and curvature scale space, where a scale space and parameterized representation of the contour of the objects is used.

Veltkamp describes in [26] various forms in which shape matching is studied, given two shape patterns and a dissimilarity measure. These are:

- **Computation Problem:** Compute the dissimilarity between the two patterns

- **Decision Problem:**
  - For a given threshold, decide whether the dissimilarity is smaller than the threshold.
  - For a given threshold, decide whether there exists a transformation such that the dissimilarity between the transformed pattern and the other pattern is smaller than the threshold

- **Optimization Problem:** Find the transformation that minimizes the dissimilarity between the transformed pattern and the other pattern.
A well studied optimization approach for shape matching is Active Contour Models (Snakes), which inspired much of the shape fitting approach of this work (see Sec 3.1.9). In [11] a snake is defined as an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. The snakes are said to be active contour models because they lock onto nearby edges, localizing them accurately.

The snakes model is defined as a controlled continuous spline that is bound by internal and external image forces, called energies. The external energy models how well the deformed model matches the data. The internal energy models the objects resistance to be pushed by the external force into directions not coherent with the prior knowledge [25]. In this case, the internal energy imposes a "piecewise smoothness constraint" [11]. This means that a contour is pushed to an image feature by the external force while the contour itself exhibits resistance to be deformed into a non-smooth curve. As explained in [25] the image forces push the snake toward salient image features like line, edges and subjective contours, while the external constraint forces are responsible for putting the snake near the desired local minimum.

Given these definitions, let $M$ be the model and $D$ a data set, the total energy $E$ can be defined as:

$$E(M) = E_{ext}(M, D) + E_{int}(M)$$

where $E_{ext}$ is the external energy function and $E_{int}$ the internal energy function. Having this, the optimization algorithm consists of minimizing the objective function until the best solution is found.
2.6 Shape Descriptor

A shape descriptor is a structured abstraction of a class of shapes that describes them in geometrical terms. Shape descriptors can have either fixed or variable geometrical shapes. Variable descriptors depend on the different values assigned to its parameters, different shapes are generated but still belong to the same type or class of shapes. Shape models have been used widely to achieve robust interpretation of complex images [24]. They allow image evidence to be organized into plausible interpretations which can then be verified.

Latecki et al [9], divide shape descriptors into three main categories:

- **Contour based descriptors**: The contour of a given object is mapped to some representation from which a shape descriptor is derived

- **Image based descriptors**: The computation of a shape descriptor is based on summing up pixel values in a digital image containing the silhouette of a given object; the shape descriptor is a vector of a certain number of parameters derived this way

- **Skeleton based descriptors**: After a skeleton is computed, it is mapped to a tree structure that forms the shape descriptor; the shape similarity is computed by some tree-matching algorithm

Considering that basically shape descriptors are “attempts to quantify shape in ways that agree with human intuition” [14, p.1], any kind of geometrical interpretation that covers the contents or properties that want to be described on a shape, can be used as a shape descriptor. In [14] region-based shape descriptors. These are such descriptors that attempt to describe a shape based on the geometrical and numerical properties of the region of the shape. Some simple descriptors are mentioned such as: area, perimeter,
(non-)compactness or (non-)circularity, eccentricity, elongation, rectangularity and orientation. Any combination of these properties of a shape are useful to describe them in a basic and general way. Other more complex properties are mentioned to improve the accuracy of the descriptor: convex hull, extremal points, profiles, moments and profile moments. Convex hull or bays describes the shape by measuring the number or size of concavities in the shape. Extremal points is based on finding the points that are at the extreme of the shape. This can be a simple representation as the bounding box or a more powerful one as it is finding the eight extremal points defined by: top left, top right, left top, left bottom, bottom right, bottom left, right top and right bottom. The profiles shape descriptor is based on the number of pixels that the shape has in a given direction: either vertical, horizontal or diagonal. Moments refer to the calculation of moment statistical properties and profile moments, or a combination of the last two.

In [29], shape descriptors are classified by their invariance with respect to the transformations allowed in the associated shape definition. The main classes are descriptors invariant with respect to congruence and descriptors invariant with respect to isometry. The congruence class comprehends identical shape descriptors for congruent shapes (shapes obtained from translation, rotation or mirroring). The intrinsic shape descriptor refer to those that do not change with different isometric embedding of the shape, and thus can be applied accurately to deformable objects.

Depending on the properties that are controlled and measured, the descriptor may or may not allow reconstructing a shape of the class. In [24], a trainable method of shape representation is described which can automatically capture the invariant properties of a class of shapes and provide a compact parametric description of variability. The method was applied on worms, obtaining a shape descriptor that reconstruct different bending worm shapes by modifying the values of the parameters.
2.7 Splines

The term spline, as it is used in this work, refers to a piecewise polynomial curve. Splines are widely used in computer science subfields because of the simplicity of their constructions, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design, as mentioned in [32]. The continuous signal representation is particularly apposite for problems such as: edge detection, surface fitting and multi-resolution techniques. It is useful for many other problems in computer vision such as: optical flow, surface reconstruction, the recovery of lightness and color, shape from shading and stereo matching, [12, 821].

Special types of splines receive different names, depending on different conditions.

A commonly used type of spline in object recognition is the Hermite spline. This is a third-degree spline, expressed using Hermite polynomials to represent each of the individual polynomial pieces. Several methods have been invented to fit such splines to given data points such as cardinal Splines, Catmull-Rom splines, Kochanek-Bartels splines. They allow constructing smooth curves that go through every point in a given data set. Thus, e.g. given a series of points belonging to the contour of an object, a smooth shape can be calculated that models the shape of the defined object. Such as B-splines, Hermit splines have a number of advantages for image processing, as mentioned in [12]. First, they are usually smooth and well behaved, therefore they do not tend to oscillate as higher order polynomials do. Second, the juxtaposition of local polynomial approximations may produce strong discontinuities in the connecting regions. B-spline surfaces, by contrast, are continuous everywhere. Finally, they can be evaluated efficiently.
Chapter 3

Methodology

3.1 Solution Methodology Design

This section presents the solution methodology which consists in the different steps in image processing that must be performed in order to successfully fit the shape of C. elegans worms present in digital images. First, a general description of the solution is presented, where the shape fitting approach is explained, indicating the different processes involved in the solution design. Then, for each process a reasoning of its requirement and usefulness is given, as well as the corresponding implementation approach.

3.1.1 Previous Reasoning

As covered in [16, 25, 26, 27], shape matching is usually accomplished adopting a shape descriptor and then placing a constructed shape sufficiently close to the image shape and adjusting the values of the parameters of the descriptor until a match is found. A shape descriptor is a representation of a specific class of objects that is defined in geometrical terms. It is comprised of a number of parameters, where different values for each parameter give different shapes of a given class of objects. This approach is appropriate when the
objects that are to be matched can be categorized in a certain class and thus can be represented or described in terms of geometry, i.e. a shape descriptor.

The problem of study aims the detection of worms, particularly those belonging to the C. elegans species. Given the vermiform (worm shape) property of these individuals, the objects to detect can be defined as part of a *worm* class, that would refer geometrically to long, thin and cylindrical shapes, in general terms. Following this idea, a shape descriptor could be designed that comprehends a cylindrical, long and connected shape, thus making possible to generate worm-like shapes. This could be represented as two endpoints (the ends of a worm) and a set of thickness values along a medial axis that connects the endpoints. Then the problem would be reduced to find every pair of endpoints belonging to each worm in the image, place an approximated shape (built through the shape descriptor) near the matching worm, and adjust the values of the parameters of the shape descriptor until an acceptable match is found.

To design a methodology for the solution of the problem the following points must be taken into consideration: the nature of the input images, the positional identification of worms in the global image, the gathering and loss of information and the efficacy and efficiency of each of the involved processes (and their respective algorithms).

For this study the input images consist of a number of worms that are put together in liquid media. The image can contain some noise such as shadows, water bubbles or little remains that do not belong to the worms, so these last, as objects of study, must be separated from the rest of the information in the image. The position of each individual worm in the image is variable and can be distinguished into two groups: *worm clusters* and *isolated worms*. A *worm cluster* corresponds to a group of worms in which each worm is connected to any other directly or indirectly through overlapping. It can also be described
as a group of worms in which a path can be traced from every worm to another without passing over background pixels. On the other hand, an isolated worm is such that it is surrounded by background pixels and does not overlap with any other worm. The image can be segmented by separating the different worm clusters and isolated worms, so each segment can be processed individually. The contour of isolated worms can be traced automatically following the pixels that are closest to background pixels. These can also be used to generate a profile that will set the general values for the shape descriptor that best represents the shape of the worms in the image. The worms that are clustered can be matched through an energy minimization process, based on the manipulation of the shape descriptor and its distance to the matching image, in order to obtain the best possible match.

3.1.2 Methodology Description

Following the previous reasoning, a methodology was designed taking into account the main components of the matching process, as reasoned before, such as: determination of the objects of study, objects segmentation, worm shape descriptor and shape matching based on energy minimization. Below the solution methodology is described, pointing out the sequential steps that are followed to match and fit the shape of every worm in the input image. In Fig. 3.1 a graphical description of the solution methodology is presented.

Given the input image, the first step is to separate the pixels belonging to the object of study from the rest of the image pixels. A thresholding algorithm is then used to obtain a binary image that separates worm pixels from background pixels. Usually some noise is obtained after the thresholding process, but it is ignored in further processing. This corresponds to an initial segmentation of the input. Once one have a binary image, a distance map can be obtained in which each pixel represents the distance to the nearest background pixel, as explained in [3]. The distance transform makes
Figure 3.1: Graphical description of the solution methodology for the shape detection of C. elegans worms in digital images.
possible to identify the contour pixels in the binary image, which makes it a fundamental tool for the automatic generation of a shape descriptor, contour tracing on isolated worms and optimization of the skeletonization algorithm, among others. Having the image separated in object pixels and background pixels, the image is segmented to separate the worms.

The image worms have been distinguished into two groups: \textit{isolated worms} and \textit{worm clusters} (see Sec.3.1.1). A way to differentiate the image worms is to count the number of endpoints of every object (as defined after the binary transform). An object with exactly two endpoints would correspond to an \textit{isolated worm} while more than two points indicate the presence of overlapping worms, thus a \textit{worm cluster}. At the same time a path from one endpoint of a worm to another is needed to match the shape, due to the need of placing the matcher shape near to the matching shape, which is a usual approach for shape matching, as covered in Sec. 2.5 and Sec. 3.1.1. Having said this, the topological skeleton of the image would provide a simple way to recognize many endpoints as well as an approximated connected medial path between the endpoints of the image objects. Therefore, the skeleton is calculated. Then, the image is segmented into different subareas corresponding to either clusters or \textit{isolated worms}. Each type of worm group is processed in a different way.

\textbf{Isolated Worms}

The \textit{isolated worms} shape contour can be traced easily by selecting a border pixel (indicated in the distance map) and following the neighboring contour pixels until the initial pixel is reached back again. Then the whole shape can be rasterized by triangulating it through the \textit{ear clipping triangulation method} and then rasterizing each triangle separately. This provides all the pixels belonging to the shape, \textit{i.e.} a match. The nearly perfect match that can be obtained from the \textit{isolated worms} makes possible to calculate a worm
profile from the currently analyzed worms in order to generate an accurate shape descriptor (this is explained thoroughly in Sec. 3.1.7).

**Worm Clusters**

To match the shape of the worms present in a *worm cluster*, first the number of worms in the cluster is determined (follows from the number of endpoints), then the best match between pairs of endpoints is found. Given a pair of endpoints, the path between them is calculated. Then a matcher worm shape is generated from the shape descriptor, selecting a given number of control points in the path. Afterwards an energy minimization process is performed that varies the angles between the straight lines that connect the control points (generating different shape representations), until the best match is found. Finally the contour of the shape descriptor is slightly modified by finding the closest contour segments (or a lower value in the distance map) to the contour points, in order to adapt the generic shape silhouette to the matching object.

This process is repeated for every feasible worm path that can be found starting from every endpoint, thus obtaining all the different worm conformations in the image. Then an assignation algorithm would select the best set of conformations that maximizes the number of covered endpoints and minimizes the total energy value. Since the total energy is the sum of energies of each worm, speed can be improved further by branch and bound. A path guessing algorithm can be used to find the most likely path starting from a given endpoint. The conformation resulting from optimizing this path could be favored over the others in order to make it more probable to be selected.

In the following sections, each of the sub-processes involved in the solution methodology are explained, covering their need and usefulness as well as the followed implementation approach.
3.1.3 Thresholding

Since the main purpose of this study is to fit the shape of *C. elegans* worms on digital images, it is useful to differentiate these from the rest of the image in order to perform a more accurate analysis. The shape of the worms can be characterized as objects and the rest of the image as background. More precisely the image pixels can be separated into two groups: object pixels, which are all of those that belong to a worm shape and background pixels, which are all the remaining ones.

Given this theoretical characterization, a thresholding filter would come to be a useful tool to locate the objects of study in the digital representation and to discard unnecessary information, obtaining a binary image from the original one. A binary image would then provide an initial segmentation of the processed image, being as well a key element to obtain a distance transform, as explained in Sec. 3.1.4.

3.1.3.1 Implementation

There are four thresholding filters for 2D images implemented on *Endrov*, these are: *Fukunaga*, *max entropy*, *Otsu* and *percentile*, which cover the histogram and entropy-based thresholding methods categories as defined in Sec.2.2. Considering that the implemented methods are sufficiently different and given the transparency of *C. elegans* worms, it is hard to determine theoretically which would be the most appropriate thresholding method to obtain an accurate binary image from the study data-set. In order to select a thresholding method, a series of experiments where performed that consisted on tweaking the parameters for the different mentioned methods. The selected method was *percentile threshold 2D*, as proved to be the easiest to tune manually and the one that calculated the best binary counterpart, for the images in the data-set.

Figure 3.2 shows a binary image obtained after applying the *percentile*
threshold 2D method with a percentile value of 0.074.

Figure 3.2: Worms in liquid media. Original image and binary image obtained through percentile thresholding with a percentile value of 0.074.

3.1.4 Distance Transform

In this shape fitting approach, the distance transform of the input image is used thoroughly for contour detection and different kinds of image segmentation procedures. Specifically the distance map allows detecting and following the exact contour of isolated worms (Sec. 3.1.9.1), is useful in the shape profile generation (Sec. 3.1.7.1), and essential in the heuristic guessing of the more likely worm-paths in worm clusters (Sec. 3.1.6). It also improves the performance of the iterative thinning algorithm designed by Zhang and Suen [35] as it is described on Sec. 3.1.5.

3.1.4.1 Implementation

As stated in [34, p.196] the algorithms of distance transform can be categorized into two classes: one is the iterative method which is efficient in
a cellular array computer since all the pixels at each iteration can be pro-
cessed in parallel, and the other is sequential (or recursive) method which is
suited for a conventional computer by avoiding iterations to be independent
of object size. Using the machines that most people working in digital image
processing have access to, sequential algorithms are often much more efficient
than iterative ones. For this reason a sequential approach was chosen to cal-
culate the distance transform of the input images. Particularly the two-scan
transformation using 3x3 neighborhoods [34], which is both efficient and easy
to implement.

In the mentioned paper a distance map calculation algorithm is described
which consist of only two scans of the image bitmap, one left to right - top
to bottom, and another right to left - bottom to top, with one operation per
pixel. This makes the complexity of the algorithm $O(N)$ where $N$ is the size
of the image array. In [34, p.197] a pseudo-code for chessboard and Manhattan
or city-block distances is given, while in [34, p.198] the definition is extended
to improve the efficiency of the calculations needed to generate a distance
map using Euclidean distances. The two-scans algorithm was implemented
using the three different distance metrics mentioned before. This allows a
wider analysis on the behavior and accuracy of the shape fitting process from
one metric to another, “The city block or chessboard distance measures are
sensitive to the rotations of an object, but the Euclidean distance measure
is rotation invariant. However, its square root operation is costly...” [21,
p.332]. Given the straight-like shape of worms and the different levels of
accuracy of the distance metrics it is hard to tell at first sight which would
be the most adequate to use, so it had to be determined experimentally. The
figure 3.3 shows the binary image and three distance maps obtained from a
single worm image.
Figure 3.3: Binary Image and three distance transform metrics from a single worm image
3.1.5 Worm Skeletonization

The skeletonization of the image corresponds to the process of obtaining a connected and thin (1-pixel width) medial axis that represents the worms in the image. This is a key process on the shape matching approach followed in this work, as first mentioned on Sec. 3.1.2. It makes possible to identify the amount of worms in the input image, to separate them distinguishing between isolated worms and worm clusters, and to obtain paths between endpoints of worms (that tend to the medial axis), which plays a fundamental role in the shape matching process (see Sec.3.1.6). The skeleton image would then be fundamental to determine the area of the image in which the worms are located and give estimated paths along which the different worms would be disposed.

3.1.5.1 Implementation

For the purpose of this work the skeletonization algorithm to be selected must ensure the connectivity of the skeleton points, i.e. every skeleton point must be connected to at least another skeleton point belonging to the same skeleton. Also, the skeleton must be as thin as possible to simplify the path processing and analysis.

Different methods that consist of finding ridge points on the distance maps and then connecting them, have been covered in [22, 2, 1]. The approach in [22] was followed as a first attempt to calculate a thin skeleton with a low time cost. This algorithm defines the ridge points as such pixels that have the greatest numerical value among its 3x3 neighborhood (bitmap image) in the distance map. After finding the ridge points an up-hill reconnection is performed, followed by a down-hill connection for missing points. The study covered in [22] states that this approach makes possible to find successfully a connected 1-pixel-thin skeleton, which was actually the case for
isolated worms or those which do not overlap with other worms. Nevertheless for worm clusters the obtained skeletons were usually disconnected, thicker than 1-pixel and not accurate. Although the approach seemed appropriate in theory, the costly reconnecting operations and the inaccurate skeletons obtained for worm clusters gave rise to the need for a different approach.

Given the long, thin and cylindrical shape of the worms in general, a thinning algorithm approach that reduces the different layers by removing pixels that should not belong to the skeleton was then taken into account. In [35] an iterative and parallel thinning algorithm is presented, which consist in two sub-iterations per main iteration aimed at deleting the south-east boundary points and north-west boundary points respectively. The study is aimed at parallel computers so the different operations in each pixel can be performed at the same time, improving the performance. In order to avoid the requirement of using a parallel computer without losing the time performance improvement, the distance map was used to discard unnecessary pixel checking (those belonging to inner layers). Thus in each iteration only the pixels belonging to the currently selected shape layer are taken into account. The layers are defined by the distance map value of its pixels. The first layer corresponds to a distance value of one (1), the next to a distance value of two (2) and so on. This is presented in algorithm 3.1.1.

The algorithm deals well with overlapping worms by constructing a path that is very close from the shapes medial axis, and results in a totally connected 1-pixel width skeleton. In fig. 3.4 the skeleton of a sample image is shown. The worms in the image are successfully skeletonized.

3.1.6 Worm Segmentation

Since the goal is to match the shape of individual worms, it is necessary to locate them in the image and then separate them as much as possible,
Algorithm 3.1.1 Calculate shape skeleton

\[\begin{align*}
\text{shapePts} & \leftarrow \text{getBinaryObjectPixels}() \\
\text{dtImage} & \leftarrow \text{getImageDistanceMap}() \\
\text{contourIndex} & \leftarrow 1 \\
\text{makeThinner} & = \text{True}
\end{align*}\]

while \text{makeThinner} do

\{remove south-east boundary points and the north-west corner point\}

for \text{pixel} in \text{shapePts} do

if \text{dtImage}(\text{pixel}) > \text{contourIndex} then

\{skip iteration\}

else

\text{pixelRemove} \leftarrow \text{southEastCondition}(\text{pixel})

if \text{pixelRemove} then

\text{shapePts.remove}(\text{pixel})

\text{makeThinner} \leftarrow \text{True}

end if

end if

end for

\{remove the north-west boundary points and the south-east corner points\}

for \text{pixel} in \text{shapePts} do

if \text{dtImage}(\text{pixel}) > \text{contourIndex} then

\{skip iteration\}

else

\text{pixelRemove} \leftarrow \text{northWestCondition}(\text{pixel})

if \text{pixelRemove} then

\text{shapePts.remove}(\text{pixel})

\text{makeThinner} \leftarrow \text{True}

end if

end if

end for

end while

return \text{shapePts}
Figure 3.4: Skeleton obtained through iterative thinning over a worm binary image
*i.e.* to segment the image. This allows improving the efficiency and accuracy of the shape fitting process, while reducing the matching area and thus the amount of different combinations that must be taken into account. After the process of *skeletonization*, a set of paths are obtained between endpoints of worms (the objects of study), in some cases overlapping. By identifying the worm endpoints and tracing the paths that connect them together, the different groups of paths can be separated, thus segmenting the image. As explained in Sec. 3.1.5 and Sec. 3.1.1, the different groups of paths can be distinguished, in correspondence to the objects they represent, by *isolated worms* and *worm clusters*. Then the shape fitting process of the whole image would consist of matching and fitting the worms present in each of the obtained sub-images separately.

Another process of segmentation that is performed is the identification of single worms paths, in both *isolated* and *cluster* worms. For *isolated worms*, the path that determines its medial axis is used for two different processes. First to find the surrounding contour to fit its shape (see Sec. 3.1.9.1). And second, to generate a profile that would define a general representation of the worms in the image through a shape descriptor, as explained in Sec. 3.1.7.1. On the other hand, for *worm clusters*, feasible worm paths must be found between endpoints. If a path exist between a pair of endpoints then a valid worm conformation between them will be generated through the optimization process. These paths are chosen among all the technically possible paths between endpoints, and by a more sophisticated path guessing algorithm that will be described below in the implementation section.

**Worm Endpoints**

The skeleton calculation returns an image containing groups of curves or paths. The pixels that are connected with two or more other pixels (neighbors) are *body-pixels*, which belong to the path and are not endpoints. On
the other hand those that are connected with only one other pixel are endpoints, thus each one could belong to the extreme of a worm. It is important to consider that since the thinning algorithm used obtaining the skeleton (covered in Sec. 3.1.5.1) consists in removing the shape layers until finding pixels that are not surrounded, the endpoints found in the skeleton will not necessarily correspond to worm endpoints.

In order to find the pixels that are actually worm endpoints a skeleton expanding process is performed that aims to stretch the skeleton path up to a contour point that would come to be the worm endpoint. The skeleton expanding algorithm uses the definition of directional neighborhood stated in [22, p.334], where given a pixel $P$ in the bitmap, a directional neighborhood $D$ of $P$ is composed by those pixels that belong to the 8-neighborhood of $P$ and that are located within $\pm 45^\circ$ slope changes from the current medial axis orientation of $P$. In fig. 3.5 three examples of directional neighbors are presented. The algorithm consist in following the best directional path starting in every endpoint and expanding the skeleton until a contour pixel is found.

![Directional Neighbor and Skeleton Path](image)

Figure 3.5: Three different directional neighborhoods

The expanding algorithm can be summarized in the following steps:

- Select an endpoint
• Find the previous skeleton point and calculate the directional neighborhood.

• Select the directional neighbor with the lowest distance map pixel, and mark it as skeleton pixel.

• If the neighbor is not a contour point repeat the process.

A filtering process is done removing incorrect object pixels, which consists in removing skeletons that have a size (number of pixels) lower than a small threshold. This allows to remove some slightly noisy regions and incorrect endpoints. Once the skeleton is successfully expanded, the endpoints of the skeleton are considered worm endpoints. It must also be considered that some worm endpoints cannot be detected following this process, particularly in crowded images where there is a big chance that the overlaps “hide” the endpoints. Nonetheless, a manual process can be carried out to add the missing endpoints, as explained in Sec. 3.1.10.

**Group segmentation**

Having detected every worm endpoint, the image can be segmented by determining the endpoints that are bound together through a skeleton path, *i.e.* finding the different *isolated worms* and *worm clusters*. As previously explained, the overlapping worms in the image are bound together as one object in the binary image, so then, in the skeletonization process, the endpoints belonging to *worm clusters* are bound together as well through a skeleton path. Based on the previous reasoning an algorithm was designed that detects the endpoints that are bound together through the path that links them, separating the linked paths into different groups. Those paths that link together exactly two endpoints represent *isolated worms* while a different number of endpoints correspond to *worm clusters*. This method is described in algorithm 3.1.2. The algorithm basically consists of following
every possible path starting from an endpoint until all the endpoints of a group have been reached.

**Algorithm 3.1.2 Calculate shape skeleton**

```plaintext
endPtList ← list of endpoints
clusterIndex ← 0
for endpoint in endPtList do
    if endpoint.wasVisited() then
        {skip iteration}
    else
        clusterIndex+ = 1
        followPath(endpoint, clusterIndex)
    end if
end for
```

**Path guessing**

The *worm clusters* found through segmentation are defined by a set of endpoints that are all connected through *skeleton* paths. However, the pair of endpoints belonging to each worm in the image and an accurate skeleton path that connects them is still unknown. The optimization algorithm, covered in 3.1.9, performs a shape manipulation process to match the worms in the image given two endpoints and the path that connects them. To calculate the most accurate match for a given endpoint, the algorithm would have to try every possible path starting from that endpoint. This could be costly in time, depending on the size of the worm clusters. In order to improve the time efficiency of the shape fitting algorithm as well as the accuracy of the matches (particularly avoiding incorrect endpoints bounding), a path guessing algorithm was designed that performs a heuristic guessing of the most-likely worm paths.

The algorithm is based on the idea of avoiding paths that tend to describe unnatural worm conformations. So, given S last steps in the skeleton tracing,
Algorithm 3.1.3 Follow Path algorithm (followPath(currentPoint, clusterCount) )

Require: currentPoint
Require: clusterCounter

if not currentPoint.isSkeletonPoint() then
    return
else
    addToCluster(endpoint, clusterIndex)
end if

{continue tracing path}

if currentPoint.isEndPoint() then
    markEndPointAsVisited(currentPoint)
end if

neighbors ← getNeighborhood()
for n in neighbors do
    followPath(endPoint, clusterCounter)
end for

the next step $S + 1$ will tend to follow the direction that was being followed lately (previous $S$ steps), thus avoiding unnatural bendings or abrupt changes in the shape. An important issue when following the most common direction in the last $S$ steps is that in some cases the path tracing will tend to avoid path bifurcations. A path bifurcation occurs when there is more than one neighbor pixel that can be followed next, thus dividing the path in two or more different paths.

In order to make the path tracing tend to reach path bifurcations and then decide the best path to take, a heuristic function was designed. This function basically consists of the value of the neighbor pixel in the distance transform multiplied by a variable factor. Then, the selection of the next pixel to follow is based on two main values: the amount of times that the direction of neighbor pixel has been taken in the last $S$ steps and the value of the heuristic function for that pixel. This can be expressed as below:
\[ \text{Next}(p) = \max_{n \in \text{neighbors}(p)} (\text{dirValue}(\text{direction}(p, n), S) + dt(n) \times h\text{factor}) \]

where \( p \) is the current path pixel, \( \text{dirValue} \) returns the amount of times that the direction of the neighbor pixel \( p \) have been taken in the last \( S \) steps, \( dt \) is the distance map and \( h\text{factor} \) is a heuristic factor that controls the influence of the distance map.

In order to cover the issue mentioned in Sec.3.1.6, that some endpoints could be hidden because of the overlapping and thus they are not detected, an extra endpoint is created after \( \text{wormLength} \) steps. The \( \text{wormLength} \) is an estimated value that is calculated as the mean of the length of the \textit{isolated worms} in the image. This process attempts to identify worms that otherwise would be discarded because of the absence of endpoints. Algorithm 3.1.4 presents a pseudo-code for the path guessing approach.

### 3.1.7 Worm Shape Descriptor

As first mentioned on Sec.3.1.2 the chosen methodology approach to match \( \text{C. elegans} \) worms is based on the manipulation of worm shapes generated from a shape descriptor. Worm shapes can be described in geometrical terms as long, thin and cylindrical objects. Given that the process of skeletonization and image segmentation make possible to obtain \textit{thin} paths between pairs of worm endpoints, a shape descriptor would allow to construct a worm shape around this medial axis, which would work as input for the optimization algorithm.

A shape descriptor was then designed based on the idea of generating a representative worm shape around the medial axis. The descriptor consists of two elements: a set of control points and a shape profile (worm profile). The set of control points is comprised of \( N \) equidistant points along the worm
Algorithm 3.1.4 Pseudo-code algorithm for path guessing between endpoints

- $\text{endPtList} \leftarrow$ list of endpoints
- $\text{wc} \leftarrow$ paths and endpoint in worm cluster
- $\text{length} \leftarrow$ worm estimated length multiplied by a scaling factor

for $\text{endPoint} \in \text{endPtList}$ do
  if $\text{alreadyReached}(\text{endPoint})$ then
    {skip iteration}
  end if
  $\text{markAsReached}(\text{endPoint})$
  $\text{path} \leftarrow$ empty list
  $\text{reachedEndPoint} \leftarrow \text{False}$
  $\text{currentPixel} \leftarrow \text{endPoint}$
  while not($\text{reachedPoint}$) and $\text{size(path)} < \text{length}$ do
    $\text{currentPixel} \leftarrow \text{getBestNeighbor}(\text{currentPixel})$
    $\text{updateDirectionsArray}(\text{direction(currentPixel)})$
    $\text{path}.\text{add}(\text{currentPixel})$
    if $\text{isEndPoint}(\text{currentPixel})$ then
      $\text{reachedEndPoint} \leftarrow \text{True}$
    end if
  end while
  if not($\text{reachedEndPoint}$) then
    if number Of Reachable Endpoints = 0 then
      $\text{createEndPoint}(\text{currentPixel})$
      $\text{path}.\text{add}(\text{currentPixel})$
    else
      {Select a path to reachable endpoint}
    end if
  end if
end for
medial axis, both endpoints included. Each control point has associated a \textit{thickness} value that represents the radius of the circumference that has the control point as center. Then by selecting two points in opposite directions of the thickness circumference for every control point, and joining these points together through a smooth curve, a contour line is obtained that traces the silhouette of a worm shape, as shown in fig. 3.6. The set of \textit{thickness} values that are associated to each control point is called a shape profile (worm profile for this study).

![Figure 3.6: Construction of a worm shape based on shape descriptor](image)

In order to obtain a contour that accurately represents the worm shape along the medial axis, the choice of the opposite points in the thickness circumferences must take into account the \textit{skeleton} bendings. Since the shape is constructed following the \textit{thickness} of control points, the bendings in the generated shape occur at each of these ones, and can then be calculated as the angle between the straight lines that connect the control points. The bisection of any angle provides a straight line that divides it into two equal portions, measuring the bending variation at each control point. Then, selecting the two points at respective \textit{thickness} distance from the corresponding
control point, and following the bisection line in opposite directions, provides a set of contour points. Joining together these points with a smooth curve, will provide a worm shape contour.

Generating a smooth curve around the control points improves the accuracy of the shape description, compared to tracing straight lines between the contour points. It provides a good representation of the shape with considerably fewest control points. The smooth curve is obtained by calculating the cardinal spline (covered in Sec. 2.7) given the contour points. A cardinal spline is a function that describes a smooth curve which passes over all the points of a set, given a starting and ending point. In this case the starting and ending points are the same, so the described contour is closed.

The worm profile for a set of control points can be both manually set or automatically calculated from the isolated worms as explained in the section below.

3.1.7.1 Automatic Profile generation

The shape of isolated worms can be accurately matched by following the contour points in their respective distance map, as explained in Sec. 3.1.9.1. Given a set of matched shapes for isolated worms, and the initially detected skeleton, a worm profile (as described previously) can be generated by measuring the thickness of every control point and finding the mean among these.

In order to measure the thickness for every control point, a set of $N$ equidistant points is generated that cover the skeleton of the isolated worm. Then, as described in the previous section, the bisectors of the angles between the straight lines connecting the control points are calculated. Starting from every control point, the bisector line is “walked” until the pixel with the lowest distance map value is found, a contour point in most of the cases. The oper-
ation is repeated for every control point bisector in both opposite directions. The Euclidean distance from a control point to each found pixel is calculated, and its average is recorded. Repeating this process with every isolated worm generates a set of thickness profiles, one for each isolated worm. Then a general worm profile is calculated by finding the mean between the thickness values for every control point in each thickness profile. In order to generate an accurate profile, that avoids oversized (or downsized) worms and wrongly detected isolated worms, the 20% higher and lower values are discarded when calculating the average. The thickness value for the endpoints, i.e. the first and last point in the set of length $N$, is zero, so only one contour point is generated in the ends instead of two. Having done this, a thickness profile is obtained that represents the average radius distance for every control point to its closest contour pixel, making possible to generate a generic worm shape around any skeleton.

3.1.8 Shape Rasterization

The proposed matching approach is focused on minimizing the difference between generated shapes and the shapes in the image. In order to do so, the area covered by the matching shape must be calculated. Given a shape descriptor, a worm shape can be constructed following a worm profile consisting of a worm silhouette defined by a B-spline curve. Given the contour of the worm shape, the area can be calculated and then rasterized in order to express it in image data terms.

The approach followed to achieve this consists of triangulating the space defined by the closed contour of the shape and then rasterizing every triangle by separate. The decomposition of a polygon into simple triangles whose vertices are only those of the simple polygon is a classic problem in computer graphics. So, many solutions has been proposed such as: Delaunay triangulation, minimum-weight triangulation and the ear clipping method,
among others. The ear clipping method was chosen because of its capacity to triangulate concave polygons and its ease of implementation. In order to convert the worm outline into a polygon, the spline curve that defines it is transformed into a discrete set of points. Each successive point is connected by a straight line, defining the closed polygon. Then, this polygon-like worm shape is triangulated.

Every triangle defining the area is rasterized following the scan conversion algorithm explained in [13]. The algorithm consists in tracing horizontal lines between the edges of the triangle until the area is totally covered.

This process calculates a set of pixels that constitute the area of the shape. Thus, the area and contour of the shape can be stored as manageable and displayable data.

3.1.9 Profile-Driven Shape Fitting

Once the initial processing of segmentation and worm skeletonization is performed, where information about the worms in the image is gathered, a shape matching process has to be carried out in order to detect the shape that corresponds to each worm. The matching process is different for isolated worms and worm clusters due to their nature. In this sections each process is explained and some implementations details are given.

3.1.9.1 Isolated Worms shape fitting

At this point, the information gathered about each isolated worm consists of a 1-pixel skeleton path between exactly two endpoints. Since this process is carried out after each endpoint is identified correctly, the area conformed by the object pixels surrounding the skeleton will correspond to the exact shape of the isolated worm.
In order to optimize the calculation of the worm area, the contour of the shape is first traced and then the internal area is triangulated and rasterized following the process explained in 3.1.9. The contour is traced by finding the closest border-pixel to any endpoint and then following the neighbor that is also a border-pixel. A border-pixel is such that has a value of one (1) in the image distance map. This makes possible to obtain the contour and area of the worm shape corresponding to each isolated worm accurately.

3.1.9.2 Worm Cluster shape fitting

The worm clusters represent a more complicated shape matching scenario due to their variable number of worms and the many different possible conformations that can be detected on them. The overlaps in worm clusters makes difficult to differentiate the group of pixels that belong to the area of one worm or another. A matching process must then be carried out that calculates the most likely worm conformation starting from every endpoint in order to obtain the set of conformations that best fit the whole worm cluster.

A shape optimization matching approach was followed that consists of calculating all the worm conformations starting from any endpoint. A worm conformation is obtained by generating a generic worm shape from a skeleton path and deforming that shape until the dissimilarity between the generated shape and the binary image is minimized. Below, the main steps and features of the algorithm are described:

- For every endpoint the set of feasible skeleton paths starting from that pixel are calculated.

- Given a skeleton path, a generic worm shape is constructed by generating a shape descriptor following the worm profile of the image.
• An optimization process deforms the generic worm shape until the dis-similarity between the shape and the corresponding binary image is minimized. The optimized shape corresponds to a feasible worm conformation.

• Once all the possible conformations from every endpoint are calculated, a set of conformations is selected that maximizes the number of endpoints covered and minimizes the sum of the energy values. The selected conformations are the best automatic matches.

Shape deformation

The calculated skeleton paths, starting from a given endpoint, are guesses of possible worm skeletons. Since the deformed model is constructed over a skeleton path (that is close to the medial axis of the worm cluster) the initially generated shape will always be close to a worm shape. Given this, just slight perturbations of the generated shape would deform the model sufficiently to correct the deviation of the skeleton from the actual medial axis, obtaining thus the most likely worm shape over the given path.

The deformations are made over a shape descriptor, which is defined by a series of control points and is constructed following a worm profile. In order to provide feasible shapes and to limit the number of possible deformations, a deformation will consist in the repositioning of a given control point. The number of different positions that a given control point can take is fixed and set along the bisection of the angle between the lines that connect the previous, current and next control point. Following this, the set of deformations that can be performed from an initial shape can be calculated quickly and provides a great number of possible different conformations.
Energy Formulation

The distance function must provide a measure of how far the deforming shape is from a worm in the image, *i.e.* how well the deforming shape fits. To measure the dissimilarity between a given worm shape and the matching image, an energy formulation was followed, based on Active Contour Models, [11], that consists of an external energy and an internal energy.

The external energy models how well the deformed model matches the data. Since the more background pixels are covered the farther the model is from a worm in the *worm cluster*, a good measure for the external energy would be the proportion of background pixels in the area covered by the deformed model. Then, the more object pixels are covered the lower the external energy is. So, given a model $M$ and the functions $bg$ and $fg$ that measure the amount of background and foreground pixels respectively, the external energy can be defined as:

$$E_{ext}(M) = \frac{bg(M)}{bg(M) + fg(M)}$$

The measure is not considered just as the sum of background pixels because the energy value will be vary too much, given the variability of the amount of pixels covered by a worm shape.

The internal energy is the one that models the objects resistance to be deformed into a shape that do not corresponds to the modeling class, *e.g.* the worm class. This means that the internal energy resists to obtain not feasible worm shapes after deformation. The internal energy for snakes, as explained in [11], works as a smoothness constraint and is explicitly formulated (usually in terms of first and second derivatives). As stated in [15], many problems arising from this formulation have been recognized in literature such as: slow convergence speed, difficulty in determining the weights associated with the
smoothness constraint and possible inaccuracy in noisy environment of high order derivatives on discrete curves.

However, in the approach proposed in this work, every shape is generated from a worm profile following a shape descriptor, so every deformed shape will still belong to the worm class. Thus it is not necessary to include the internal energy in the formulation.

Matching Optimization

The optimization process consists in obtaining the most likely worm shape that can be obtained by deforming a shape constructed over a skeleton path, known as a worm conformation. Given how quickly a wide range of deformations can be computed, local-search is chosen as a metaheuristic for the optimization. The process consists in obtaining the best individual in the neighborhood that improves the energy function, until is minimized. The neighborhood is calculated in the following way: for a given shape, four (4) different deformations are calculated for control point. The possible deformations or new positions are the next and second next position in north and south direction, i.e. the two opposites directions of the bisection of the control point. There are $N \times 4$ different possible deformations (neighbors) for a given shape, where $N$ is the number of control points.

Then a neighborhood consists of mild and slightly stronger deformations of a given worm shape for any control point.

Since the focus is to obtain the most accurate worm shape for a given skeleton, the first and last control points, which correspond to the worm endpoints, are fixed.

Once the best shape is obtained, a contour fixing process is carried out. This process consists in the expansion or contraction of areas of the shape contour, in order to adapt to the actual worm in the image. This is required since the optimized shape is built following a worm profile, which just rep-
resents the shape of the worms in the image generically. In this process, the extremes constructed from the control points are pushed to contour points following the distance map, either expanding or contracting. While expanding or contracting, every considered new possible position must have either the same or lower distance map value. A better point must be found after a fixed number of steps, otherwise is discarded. This is done to avoid too big and strange deformations.

Conformations Assignment

After the optimization process is performed, a subset of all the possible calculated conformations for worm clusters must be selected, thus obtaining the image worms match. The selected subset must maximize the number of endpoints covered and minimize the sum of energies of the conformations. Any given endpoint may belong to at most one conformation. So, to any endpoint, there corresponds one and only one other endpoint. Therefore, the optimal subset of conformations would be the minimal assignment between the sets of endpoints. This can be obtained by solving the non-bipartite graph assignation problem. However, given the implementation complexity (and possible performance overhead) of this algorithm, a different algorithm was designed. The implemented algorithm consists in a iterative greedy solution. The algorithm is applied for the sub clusters of worm clusters. These sub clusters comprehend just the endpoints that are connected by feasible paths. The endpoints belonging to a given subcluster will be called conflicting endpoints.

The algorithm consists in the following: Given $N$ conflicting endpoints a $N \times N$ table is constructed indicating the cost of the best conformation that connects an endpoint with another. Then, for a given starting row endpoint, the lowest value is selected and an assignment is made between the row endpoint and the column endpoint (row and column are always different).
All the conformations starting from the row and column point are deleted. Then, among the remaining rows, the one with the lowest value among the possible matches is selected. The process is repeated after all the possible endpoints are assigned, thus finding a solution. Once a solution is found, it is stored, the table is reconstructed and the process is repeated starting from a different row, until all the rows have been taken as starters. From all the gathered solutions, the one with the higher number of endpoints covered and minimum total energy value is selected as the subcluster assignment.

3.1.10 Manual Adjustment

A process of manual adjustment can be performed by the user to improve the shape fitting accuracy, such as adding missing endpoints or fixing wrong conformations. Below, both manual processes are explained.

3.1.10.1 Endpoint operations

Worm endpoints are detected by first identifying extremes in the worm skeleton. However, when the extreme of a given worm in the image overlaps with another worm shape, the shapes become continuous, so the shape skeleton will be generated as a continuous path. Thus, the extreme point will be missing and the endpoint will not be detected. Since the solution methodology is centered on finding paths between endpoints and then determining the most likely worm paths, having missing endpoints could affect the matching accuracy.

Worm endpoints are simple to detect for a human being, once had the skeleton of the a worm cluster, therefore the missing endpoints could be added quickly by following a manual process. This process consists in looking at the extreme of worms and adding an extreme point when missing. Since by definition an endpoint is the extreme of a worm path, it must have one
and only one neighbor belonging to the skeleton. When an added extreme point has more than one skeleton neighbor, the user would have to select the extra neighbor pixel for deletion. In summary the user can add the missing endpoints by selecting the missing pixel and perhaps selecting some neighbors as well if it is necessary to disconnect.

3.1.10.2 Match Fixing

After the optimization process is completed for the whole image, a set of worm conformations is given that corresponds to the best match assignment. For some images there are worms that are not matched correctly, so conformations are given between incorrect pair of endpoints. Since in the optimization process all the possible conformations are calculated between pair of endpoints, the wrong assignation could be fixed manually. A user can recognize easily which conformations are incorrect, as well as quickly identify the endpoint that corresponds to any other endpoint. Then, by selecting the correct pair of endpoints, the corresponding conformation will be modified, thus obtaining a correct fitting.
Chapter 4

Experiments and Results

In this chapter, the different experiments carried out to test the performance of the shape matching solution are presented, explaining the reason for their choice and the test objectives. Then, the results are presented and discussed, pointing out the advantages and drawbacks of the suggested shape matching model that are inferred from the first ones.

4.1 Experiments

In order to test the implemented solution, three digital images of worms in liquid media were provided by Johan Henriksson at the Department of Bioscience and Nutrition, Karolinska Institutet. Each image corresponds to a different difficulty level, following the criteria of number of worms and degree of overlap. Thus, the greater the number of worms in the image and the greater the number of worms that are connected by overlapping, the greater the difficulty level. The overlapping degree is determined by the number of worms that belong to the different worm clusters (first defined in Sec. 3.1.1) and the number of worm clusters in the image. The selected images are named test image 1, test image 2 and test image 3 and are displayed in an increasing difficulty level order, from the easiest to the hardest. The
characteristics of the test set are presented in Table 4.1.

Table 4.1: Characteristics of the test set images

<table>
<thead>
<tr>
<th>Test Image</th>
<th>Number of Isolated Worms</th>
<th>Number of Worm Clusters</th>
<th>Number of Worms in Clusters</th>
<th>Total Number of Worms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>11/19 (57.8%)</td>
<td>3</td>
<td>8/19 (42.1%)</td>
<td>19</td>
</tr>
<tr>
<td>Test 2</td>
<td>8/33 (24.2%)</td>
<td>3</td>
<td>25/33 (75.7%)</td>
<td>33</td>
</tr>
<tr>
<td>Test 3</td>
<td>13/38 (34.2%)</td>
<td>5</td>
<td>25/38 (65.7%)</td>
<td>38</td>
</tr>
</tbody>
</table>

For each test image a series of experiments were carried out to test the performance of the different processes that are involved in the solution methodology. The entire shape fitting process was divided into three stages that represent the main fitting steps. These are: initial Processing, automatic shape matching and manual adjustment.

The initial processing involves all the image processing steps that are performed before the shape matching optimization process, as explained in Sec. 3.1, such as: thresholding, distance transformation, skeletonization, clustering, endpoint detection and worm profiling. Among these, the distance transformation and skeletonization follow already implemented and tested algorithms (covered in Sec. 3.1.4 and Sec. 3.1.5) and produce straight-forward results, so there is no need to analyze them further. The clustering process follows the skeletonization process and is straight-forward as well. On the other hand the thresholding, endpoint detection and worm profiling processes vary from image to image, so different experiments are carried out in order to test these sub processes for the test set.

The automatic shape matching stage consists of experiments dealing with the automatic optimization process to match worm clusters and contour-
following technique to match *isolated worms* that produce the initial fitting of the worms. These experiments attempt to measure the efficacy and time efficiency of different variations of the matching algorithm, in order to make a conclusion about properties of the algorithm and the feasibility of the automatic solution.

The third stage, *manual adjustment*, attempts to measure the type and number of manual adjustments that must be performed by the user in order to correct the wrongly assigned matches. The experiment is also aimed at determining how much the efficacy of the automatic solution can be improved, in order to get a better matching. Once the best possible matching is found through manual adjustment, the stability of the best match is studied by comparing the energy to the second and third best matches.

The experiments were executed on a personal computer with a 2.00 GHz AMD Turion Dual-Core Mobile Processor, 1 MB Microprocessor cache and 3Gb of RAM Memory, on Linux Operating System, Ubuntu Distribution.

### 4.2 Results

In this section the results obtained for the test set are presented and discussed. The results are distributed in the three main stages: *initial processing*, *automatic shape matching* and *manual adjustment*. In the section *initial Processing*, the results for the three different images in the test set are presented. On the other hand the section *shape fitting* presents the different results for the stages *automatic shape matching* and *manual adjustment* for every test image individually.

#### 4.2.1 Initial Processing

This section presents the results for the experiments carried out over the test set for the processes: thresholding and endpoint finding. Then a brief
discussion about the result for the worm profiling process is presented.

**Thresholding**

As explained in Sec. 3.1.3.1, *Endrov* has implementations for the thresholding filters *Fukunaga, max entropy, Otsu* and *percentile*. In order to select an appropriate thresholding filter, the four previously mentioned were tried on every test image, tweaking the input parameters until the best possible binary image was obtained for every one. From visual inspection, the *percentile* filter generated the best binary images for the three tests. The best percentile values for every test image are shown in Table 4.2. The second closest method was *Fukunaga* that produced acceptable solutions after the combination of binary images generated from different number of classes, however the results were not as good as the ones obtained with *percentile* and also required more processing.

Table 4.2: Best percentile values for percentile thresholding of test images

<table>
<thead>
<tr>
<th>Test Percentile Value</th>
<th>Test Image 1</th>
<th>Test Image 2</th>
<th>Test Image 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.074</td>
<td>0.1</td>
<td>0.11</td>
<td></td>
</tr>
</tbody>
</table>

The resultant best percentile values oscillate between 0.074 and 0.11, and are simple to determine using *Endrov*.

**Endpoint detection**

This section presents the results for the detection of worm endpoints. Table 4.3 shows the number of worm endpoints that are automatically detected and those that have to be manually added, in order to cover all the endpoints for the worms in the image.

Table 4.3 presents the number of endpoints identified automatically for every test image and the number of endpoints that must be added manually.
Table 4.3: Worm endpoints detection and fixing for the test set

<table>
<thead>
<tr>
<th>Test Image</th>
<th>Total Endpoints</th>
<th>Detected Endpoints</th>
<th>Manually Added Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td>38</td>
<td>38 (100%)</td>
<td>0</td>
</tr>
<tr>
<td>Image 2</td>
<td>66</td>
<td>53 (80%)</td>
<td>13</td>
</tr>
<tr>
<td>Image 3</td>
<td>76</td>
<td>57 (75%)</td>
<td>19</td>
</tr>
</tbody>
</table>

It shows that a large number of worms belonging to *worm clusters* increases the probability to have endpoints overlapping, as does a high density of worms. Considering the large number of worms that belong to worm clusters for images two and three (as show in Table 4.1) and the low number of clusters for each one (which increases the overlapping), the amount of missing endpoints can be considered fairly low and it is feasible to add them manually.

**Worm Profiling**

In order to generate an accurate worm profile of the worms present in the image it is necessary to have *isolated worms* in the processed image. The percentage of *isolated worms* for every image were 57.8%, 24.2% and 34.2% respectively, oscillating between 8 and 13 worms, as shown in Table 4.1. For all images, the generated worm profiles were sufficiently accurate to conduct the optimization shape matching process, whose results are presented in the subsections named *automatic shape matching* for every test image result.

### 4.2.2 Shape Matching

In this section, the results for the second and third stage in the solution methodology are presented, as described above in Sec. 4.1, which consist in *automatic shape matching* and *manual adjustment*. For the *automatic shape matching* subsection the results for a series of four variations of the
matching algorithm are presented, focusing on the matching accuracy and
time performance. The two main variants are: path guessing and every path. 
Path guessing is the version of the algorithm in which every guessed path is
favored by an improvement in its shape optimization value, by reducing it to
the half. So when a conformation is to be chosen for an endpoint the value of
the guessed path is more likely to be selected. On the other hand, the every
path variant considers every possible conformation for every endpoint. Each
variant is executed with and without manually adding the missing endpoints.

The manual adjustment presents the results of modifying the erroneous
matches obtained by the best automatic matching algorithm, showing the
resulting best possible matching image and indicating the number of worm
shapes that had to be re-arranged. This process is done manually: The user
detects the wrong match visually and selects two endpoints to obtain the
best optimized shape that connects them, as explained in Sec. 3.1.10.2. An
operation (as used in the tables for this section) is considered to be the pro-
cess of selecting two endpoints and generate a new shape.

Then a subsection matching energy is presented, in which the distribution
of the energy values for the different conformations is discussed. This sec-
tion presents a graph showing the different energy values (objective function)
for the correct conformation and the next two best conformations for every
endpoint (of those having three or more conformations). The correct confor-
mation corresponds to the one that is constructed between the two endpoints
that belong to the worm in the image. The other two are conformations that
start in the selected endpoint and end in a wrong one.

Automatic Shape Matching (Test Image 1)

Since for the test image 1 all the endpoints are found automatically, as
shown in Table 4.4, the results presented here show the matching efficiency
and runtime just for the variations *every path* and *path guessing*, which include all the worm endpoints. Table 4.4 presents the obtained results.

<table>
<thead>
<tr>
<th>Path Finding</th>
<th>Isolated Worms Matching</th>
<th>Cluster Worms Matching</th>
<th>Total Matching</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Path</td>
<td>11/11 (100%)</td>
<td>6/8 (75%)</td>
<td>17/19 (89.5%)</td>
<td>6.47</td>
</tr>
<tr>
<td>P. Guessing</td>
<td>11/11 (100%)</td>
<td>8/8 (100%)</td>
<td>19/19 (100%)</td>
<td>7.53</td>
</tr>
</tbody>
</table>

It can be observed that in both variations of the algorithm the *isolated worms* were all matched. For the *every path* variation three quarters of the worms in clusters were matched correctly. On the other hand, the *path guessing* variation managed to match automatically all the worms in the image. The execution time is slightly higher for the *path guessing* Variation, as expected because of the extra calculations that must be performed to calculate the best paths departing from each endpoint.

For this image, the *path guessing* variant represented an improvement in the solution, by inducing the shape produced from guessed paths to be more likely to be selected. On the other hand, at first glance, the isolated worms seem to be matched regardless of the algorithm variant, as expected. The best matching for test image 1, corresponding to the *path guessing* variation, is shown in Fig. 4.1.

**Automatic Shape Matching (Test Image 2)**

The results for automatic matching on test image 2 are shown in table 4.2.2.

It can be observed that for every variation the *isolated worms* were matched totally. For the two variations that have missing endpoints, just around the
half of the total worms could be matched. Although the execution time is slower, which is expected given the decrease of feasible paths, not having an endpoint of a worm makes impossible to find a shape that matches it correctly. For the variations that include endpoints the results are considerably better. It can be observed that the path guessing variation increases the matching accuracy, in both missing and not missing endpoints set of variations. The execution time also increases when all endpoints are added, as expected. For the path guessing variation the matching percentage increases considerably, although is slightly slower than every path. In the best case the automatic matching solution manages to fit the shape for all the isolated worms and a high percentage of the clustered worms in less than a minute.

Manual Adjustment (Test Image 2)

Path guessing with no missing endpoints gave the best result, with just 4 worms wrongly matched among 33. Fig. 4.2 shows the matches.

Manual adjustment required selecting two endpoints that correspond to an actual worm shape. After selecting the endpoints the assigned shapes
Path Finding

<table>
<thead>
<tr>
<th>Path Finding</th>
<th>Isolated Worms Matching</th>
<th>Cluster Worms Matching</th>
<th>Total Matching</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Path - me</td>
<td>8/8 (100%)</td>
<td>7/25 (28%)</td>
<td>15/33 (45.4%)</td>
<td>21.8</td>
</tr>
<tr>
<td>P. Guessing - me</td>
<td>8/8 (100%)</td>
<td>10/25 (40%)</td>
<td>18/33 (54.5%)</td>
<td>23.7</td>
</tr>
<tr>
<td>Every Path + me</td>
<td>8/8 (100%)</td>
<td>15/23 (65.2%)</td>
<td>23/33 (69.7%)</td>
<td>42.3</td>
</tr>
<tr>
<td>P. Guessing + me</td>
<td>8/8 (100%)</td>
<td>21/25 (84%)</td>
<td>29/33 (87.8%)</td>
<td>45</td>
</tr>
</tbody>
</table>

Table 4.5: Results of automatic worm shape matching on test image 2, with and without missing endpoints (me)

starting at this endpoints are removed and the best shape that connects the two selected endpoints is added. Finally all of the worms in the image could be fitted.

**Automatic Shape Matching (Test Image 3)**

The results for automatic matching on test image 3 are presented in table 4.2.2. The four variations for the algorithm are shown.

<table>
<thead>
<tr>
<th>Path Finding</th>
<th>Isolated Worms Matching</th>
<th>Cluster Worms Matching</th>
<th>Total Matching</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Path - me</td>
<td>13/13 (100%)</td>
<td>5/25 (20%)</td>
<td>18/38 (47.3%)</td>
<td>26.4</td>
</tr>
<tr>
<td>P. Guessing - me</td>
<td>13/13 (100%)</td>
<td>7/25 (28%)</td>
<td>20/38 (52.6%)</td>
<td>28.7</td>
</tr>
<tr>
<td>Every Path + me</td>
<td>13/13 (100%)</td>
<td>13/25 (52%)</td>
<td>26/38 (68.4%)</td>
<td>36.2</td>
</tr>
<tr>
<td>P. Guessing + me</td>
<td>13/13 (100%)</td>
<td>16/25 (64%)</td>
<td>29/38 (76.3%)</td>
<td>39.8</td>
</tr>
</tbody>
</table>

Table 4.6: Results of automatic worm shape matching on test image 3, with and without missing endpoints (me)

The isolated worms were always matched successfully. *every path* and *path guessing* with missing points, the total matching is around the half of the
total. With missing endpoints adjusted, the level of accuracy increases considerably. Calculation is slower with more endpoints. The path guessing variations is always better than every path, and is slightly slower. The best variation was the path guessing with no missing endpoints, managing to match all the isolated worms and a total of three quarters of the whole image.

**Manual Adjustment (Test Image 3)**

The path guessing without missing endpoints turned out to be the best one, with 9 worms wrongly matched among 38. Fig. 4.3 presents four images for this algorithm.

Nine operations were required to manually adjust the incorrectly matched worms. All the worms in the image could then be matched and fitted.

### 4.2.3 Matching Energy

The energy for the best three conformations in each endpoint is shown in Fig. 4.4. The first is correct and matches the worm, the other two match wrong endpoints.

Recall that the energy function, covered in Sec.3.1.9.2, evaluates the distance between a matching shape and a matched shape as the percentage of background pixels contained in the area covered by the matching shape, so all the possible energy values are contained in the interval [0, 1] and the optimized shapes tend to take values from one to three decimals close to 0.

**Test Image 1**

For most endpoints the correct conformation has the lowest energy value. In two cases a wrong conformation had a lower energy value than the correct one. The third best conformation always has higher energy than the correct
It can be observed that the second best conformation (in green) is in all the cases worst that the best conformation, normally from two to four times in terms of the energy function value, thus the correct conformation is either the best or the second best conformation, from all the possible. In this case only four endpoints among twenty nine show a better energy function value for the best next conformation (in red) over the correct conformation (in blue). This coincides directly with the results shown in Table 4.2.2 where, for the best automatic solution (path guessing), the amount of worms successfully matched was 29/33, which is only four worms away from the optimal solution, the same number of endpoints in which the correct conformation is not the best in value. In fact, the difference of values between the correct and best next conformation for the these four endpoints is close enough to think that a more sensitive objective function could retrieve the correct conformation for all the cases for the automatic algorithm.

Among the 22 endpoints, for 9 of them the second best conformation resulted to have a better energy value than the correct conformation. This is consistent with the results presented in Table 4.2.2 where for the best variation (path guessing), the number of wrongly detected worms is also nine. This means that an incorrect path is considered to be more likely to be a worm starting from this endpoints. For every endpoint, with the exception of one, the second best solution is worse than the correct solution. So the energy value of the correct solution is either the best or the second best.
Given that for this image the amount of worms that belong to clusters is high (25), the number of possible paths starting from every endpoint is also high and so is the number of wrong conformations. Since so many correct conformations did not have the lowest energy value, the energy formulation must not be sensitive enough. However the differences between the correct and selected conformations for these cases are close (just as for test image 2), so a more sophisticated objective function could lead to better results.
Figure 4.2: Best automatic shape matching and manual match fixing on test image 2. Colored shapes and outlines in images 4.2a and 4.2c indicate incorrectly detected worms.
Figure 4.3: Best automatic shape matching and manual match fixing on test image 3. Colored shapes and outlines in images 4.3a and 4.3c indicate incorrectly detected worms.
Figure 4.4: Energy value for the best three conformations by endpoint ID on test images. The correct conformation is red. The selected endpoints correspond to worms in worm clusters that have more than two possible conformations.
Chapter 5

Conclusions and Future Work

In this chapter, the conclusions obtained from the experiment results of the worm shape fitting methodology developed in this work are addressed. Then some future work suggestions are presented, pointing out the modifications that can be performed to improve the solution methodology.

5.1 Conclusions

Solution Methodology

The proposed methodology provides a feasible semi-automatic solution to identify and fit worm shapes in bright-field digital images. This allows to turn the microscope worm images into manageable computer data and improve considerably the time cost and matching accuracy with respect to manual identification.

The methodology design and implementation are efficient enough to provide a complete identification of worms in short time using a personal computer. The solution is said to be semi-automatic because of the need of manual tweaking in two steps of the process: endpoint identification and final matching. Once the endpoints are completely identified, the automatic solution
provides a high matching percentage (more than three quarters of the total, in the worst case), that can be even be optimal in easy images. The final matching process allows the user to re-arrange the automatic solution, providing an optimal match for the image.

The initial processing step is quickly executed, less than 1% of the total execution time. On the other hand The shape matching step is the most time consuming process, consuming the 99% of the total time excluding the time taken for manual adjustments (normally fast).

**Isolated Worms and Worm Cluster matching**

The isolated worms are fully identified in every image following an automatic process, without requiring manual addition of endpoints or match fixing. An accurate image worm profile can be successfully calculated from the isolated worms.

The identification of worms in worm clusters represented the most challenging process of the worm fitting methodology. A high density of worms leads to multiple overlaps and clustering, where endpoints can fail to be detected. Worms will not be identified correctly if endpoints are missing, so manual adjustments are required.

The shape of isolated worms can be traced accurately in all the cases. For worms in clusters the shape is initially matched through the perturbation of a descriptor that is built from a worm profile. Once the optimal shape is found the contours of the generated shape are expanded or contracted to fit the individual shapes. This process is performed successfully, and in all the cases the best shape between endpoints is either exact or very close to the worm shape in the image.
Path Guessing

The path guessing heuristic improves considerably the matching accuracy of the automatic process. Since worm clusters provide a large amount of candidate paths between pairs of endpoints, the path guessing becomes a useful tool to determine the more likely worm paths departing from every endpoint. Nonetheless, given the highly deformable nature of worm shapes, the path guessing heuristic fails in occasions to determine the correct path for endpoints.

Optimization and Energy Function

The optimization process manages to reduce considerably the difference between a descriptor based shape and the matched worm shape, by the perturbation process performed over the angles between control points. The best-in-neighborhood local search approach for the optimization algorithm is then an effective and fast way to obtain an accurate matching shape by deformation. The efficacy of local search for this approach resides in the fact that the original shape is constructed over a sub-path of the skeleton image. Since the skeleton is an approximated medial axis path, the initial shape tends to be near (in shape distance terms) to the shape in the image.

The energy formulation is sensitive enough to position the optimal solution among the two best possible conformations for every worm. For the vast majority of the cases the optimal solution is considered to have the lowest energy value, hence leading to a correct match. However, the top two conformations tend to be very close to each other, which leads to matching errors, thus making the objective function not sensitive enough to provide an automatic perfect match in difficult images. A more sophisticated energy formulation, which takes more advantage of the information in the image, could differentiate better between conformations and possibly lead to perfect
automatic matches for difficult images.

5.2 Future Work

Below, a set of suggestions are given to improve the presented solution methodology and to solve new problems based in this formulation.

Energy Formulation

A more sophisticated energy formulation, defining the distance between shapes, would allow to have bigger differences between optimal conformations, thus leading to a better matching. In this work, the external energy was defined as the percentage of background pixels covered by the matching shape, over the total area, while the internal energy was supposed to be stable, provided the worm profile based shape construction. A better energy formulation would push the shape up to the contours of worms in the image, and avoid to stabilize in worm intersections.

A possible better formulation could make use of the previously calculated distance map, where there is a value assigned to every pixel indicating the distance to the nearest background point. Then, in order to reduce the energy, the shape will tend to reach border pixels. Since the distance map assigned distance value 0 to background pixels, the presence of these would have to be penalized in some way.

Endpoint Detection

Since the detection of endpoints play a fundamental role in the matching process, a more sophisticated detection technique would improve the efficiency of the automatic solution, and reduce the time spent in manual endpoint addition. A suggested way of finding missing endpoints is to use the path guessing algorithm to trace the more likely worm paths from end-
points, after a number of pixels have been covered (e.g. the estimated worm length calculated for profiling). Once this point is reached a neighborhood analysis can be performed to look for pre-existing worm endpoints and then place a new one at this point in case no endpoint is found. This approach has the possible drawback that the path guessing algorithm not always follows a correct path so incorrect endpoints would be added. A way of reducing the misplacing of endpoints would be to execute the matching algorithm with the missing endpoints, and then execute the endpoint finding algorithm covering skeleton path portions that were not matched previously.

**Non-bipartite assignment**

In this work an improved greedy algorithm is used to find the best possible set of matching shapes assignments between endpoints in order to provide one and only one shape for endpoint and to cover the greatest amount of endpoints as possible, thus maximizing the match. This algorithm does not selected the best possible set of assignments in all the cases. An implementation of a non-bipartite graph assignment algorithm, where the departure and arrival set are conformed by all the endpoints of a worm cluster, would allow to obtain the best possible assignation, leading probably to a higher matching percentage for the automatic algorithm.

It is however not clear if a different algorithm will improve precision. Solutions for other problems not found by greedy algorithms tend to be non-intuitive and physically infeasible.

**Worm Movement Tracking**

Once the worms in an image are completely matched, information about their position and size is obtained, and other extra information like rotation and head-tail positions could be calculated. This kind of information from a
total match could be valuable for *worm trackers* and other approaches based on large image datasets (such as those reviewed in [4]) to untangle worm clusters and improve the matching accuracy.
Bibliography


