An Active Regions approach for the segmentation of 3D biological tissue

Juan Cardelino

Advisors:

Dr. Marcelo Bertalmío
Departament de Tecnologia
Universitat Pompeu Fabra
Barcelona, Spain
marcelo.bertalmio@upf.edu

Dr. Gregory Randall
IIE, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
randall@fing.edu.uy

Jury:
Dr. Álvaro Pardo. IIE. Universidad de la República. Examiner.
Dr. Omar Gil. IMERL. Universidad de la República. Examiner.
Dr. Andrés Almansa. INCO. Universidad de la República. Examiner.
Dr. Vicent Caselles. Universitat Pompeu Fabra. Reviewer.

August 2005

Thesis submitted to the
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
to partially fulfill the requirements of the
Msc. Degree in Electrical Engineering

TypeSetting in LaTeX

Agradecimientos

En primer lugar quiero agradecer muy especialmente este trabajo a Gregory que ha sido el gestor de todo esto y sin el cual este trabajo no hubiera sido posible.

En segundo lugar, a Marcelo por "tirarse al agua" y aceptar ser mi tutor, y por su hospitalidad y paciencia.

A todos los bilogos con los cuales he trabajado estos aos que con paciencia han marcado manualmente muchos datos y han hecho posible este proyecto. En especial, a O. Trujillo, A. Bielli, G. Useta, C. Martinez, H. Katz. Les agradezco ademas las pacientes respuestas a mis preguntas.

A todos los compañeros del GTI, que crearon el ambiente necesario para poder desarrollar mi trabajo. Y a todos los compañeros del IIE en general.

Quisiera agradecer especialmente a toda las personas que me recibieron t me trataron muy bien durante mi estadia en la UPF. En particular a Josep Blat por haberme recibido en su departamento.

A Vicent Caselles por aceptar ser mi revisor y recibirme en su grupo en la UPF.

A los miembros del tribunal: Andrés Almansa, Omar Gil, Alvaro Pardo, por haber aceptado leer y corregir este trabajo.

Al Dr. Rachid Deriche por dictar su curso en Uruguay el cual sirvió de inspiración y dió lugar a este trabajo, además de haberme recibido cordialmente en el INRIA.

En penúltimo lugar, pero no menos importante a mi familia y amigos por que sin ellos nada de esto tendría sentido.

Finalmente a Leticia, por estar siempre ahi.

Contents

List of Tables

List of Figures

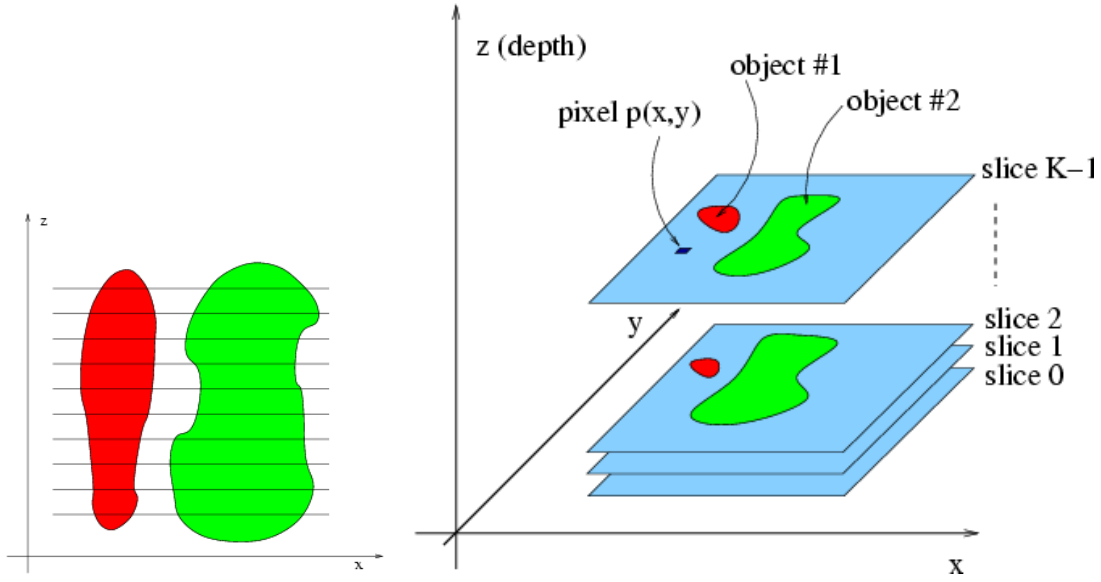
Chapter 1

Introduction

1.1 The problem

Reconstruction and visualization of 3D structures is a widely used and useful technique in medical and biological research. The range of application of this technique includes the study of neural interconnections, early detection of diseases based on the shape of certain types of cells, understanding the behavior of certain biological system based on the study of its morphology, and the tracking of cell development during an animal's growth, among others. In general, this technique is useful in the cases were knowledge of the 3D morphology and structure is relevant. The fields of application of this technique are varied and not only go beyond biology applications, there also are many applications in fields like medicine, crystallography, etc.

The 3D volumetric data is usually obtained from series of 2D images. These are obtained by slicing the 3D biological tissue into thin equidistant sections, making a biological preparation, staining the objects of interest and observing each slice in a microscope and finally taking pictures of each slice. This is only an example of an acquisition technique, in the general case not all the stages will be present in the acquisition process, depending on the type of tissue and microscope used. Other acquisition techniques include Nuclear Magnetic Resonance (NMR), Computed Tomography (CT), Confocal (CM), Optical (OM) or Electron (EM) Microscopy, etc. We will give details of some of these techniques in the following sections.



(a) Illustration of the slicing process

(b) Data structure

Figure 1.1: Data structure

However, series obtained by these techniques usually share the same structure, which is depicted in figure 1.1. They are composed by a collection of consecutive slices of a 3D volume. We will call z (depth) to the direction normal to these slices. We will call slice or frame to each of the resulting 2D images, and we will number them $0 \dots K-1$ where K is the number of slices. In this work we will call pixel to the 2D point in the position (x, y) in each frame. We will also call spatial resolution to pixel spacing in the

$x-y$ plane, and vertical (or depth) spacing to the distance between two consecutive slices.

As mentioned before these series come from different image acquisition techniques: electron (EM), optical (OM) or confocal microscopy (CM), and tomography (CT) among others. Generally, these types of acquisition processes produce sequences with tens to hundreds of slices and with resolutions which may vary from less than 800x800 to more than 3000x3000 pixels. In some cases the size of the structures present in these tissues can be as small as 10-20 pixels wide and 2 or 3 frames of depth. In other cases a single object occupies the whole image, and all the slices. In addition, the spacing between slices is generally orders of magnitude (10 or 100) greater than the $x-y$ spacing. This is often due to the size of the slicing blade and the tissue destroyed by the slicing process.

For these reasons images may present a wide range of variation along the different slices of the same sequence. In addition, the diversity in the nature of the biological tissue produces very dissimilar sequences. These differences reflect on the number, the shape, the scale and the organization of the objects appearing in the sequence. Another important difference between these sequences is the set of relevant features (edges, color, texture, etc.) that characterize the objects of interest.

Many image processing tasks arise in the analysis of these sequences: enhancement, registration, mosaicking, segmentation, surface reconstruction and visualization.

In the first place, the images could be acquired at different times, or under different conditions, like the illumination or the staining of the tissue. Even if the conditions are the same, within each frame the illumination could be non-uniform, or noise can be added. So the images must be *enhanced* in two ways: to ensure the correspondence between properties of different frames, and to improve the conditions in each frame.

Due to the different relative positions of the objects in the tissue, the camera must be rotated or displaced to correctly frame the objects of interest in each picture. Then, a matching process is needed to align objects from consecutive frames, which is called *registration*.

Sometimes a good spatial resolution is needed and simultaneously the objects of interest are big enough, so one image is not sufficient to represent the whole object. In these cases many images are taken and then composed in order to cover the whole object. This technique is called *mosaicking*.

After all these previous stages are completed we have a 3D volume with the *raw* data. This means that no interpretation of the data was performed. The next stage is *segmentation*, which consists in the detection of relevant structures in the sequence. This is usually achieved by locating the position of the boundaries of the objects. Once these boundaries are located, a 3D surface is constructed with them. This procedure is called *reconstruction*. The final step is the rendering of the scene which permits the visualization and further analysis of the 3D structure.

1.1.1 Acquisition techniques

There are many different acquisition techniques, but to illustrate the process we will describe an example of the traditional method used by neuroanatomists in the analysis of neural interconnection [1]. Then we will compare it with another technique (confocal and optical microscopy) in order to show which differences arise.

First, a biologist selects the neuron of interest from a sample of neural tissue. After the neuron is identified, it is marked via the injection of a color fluid. Then, a portion of the tissue is extracted from the animal, dehydrated and embedded into a hardening substance (like plastic epoxy). The hardened tissue is then sliced into very thin sections (800 to 1000 Å), then they are collected, ordered and placed in a microscope support. In this process usually the relative position and orientation between slices is lost. Then the slices are observed via a Transmission Electron Microscope (TEM) and captured with a photographic acquisition system. Before the introduction of computer aided systems, the procedure continued with the biologist marking the contours of the neuron on a transparent paper. Then this contour is transferred to an appropriate material (e.g. wood) and then sliced and superposed in order to build a 3D scale-model. Then the biologist performs his analysis over this model.

One disadvantage of this kind of acquisition sometimes is that some portion of the tissue is damaged by the slicing process, thus adding noise to the resulting slices.

Another example is the case of confocal microscopy (CM), where the procedure is analogous, but in this case no real slicing is performed. The structures of interest are injected with a substance which reacts with lights of certain bandwidths. Then the sample of tissue is placed in the microscope and illuminated with a laser that highlights the object of interest. Then the focal depth of the microscope is varied along the z axis, taking pictures at evenly spaced steps. From this procedure we get a stack of *virtual* slices. An advantage of this technique is that, because no real slicing is performed, the information about alignment between frames is not lost. In addition, this task is much less time consuming than the TEM or OM procedures.

Finally, another common technique used by the biologists, is optical microscopy, where the procedure is analogous to the TEM case, but other types of staining are used, and the images are taken with an optical microscope. The minimum resolution attained and the width of the slices also vary with respect to the TEM case. However, from the image processing point of view there are not many differences.

1.1.2 Previous work

For many years, 3D reconstruction of biological tissue was an entirely manual operation. In the past few years, some efforts have been done to introduce computer aided techniques to alleviate this time consuming task. Carlbom et al. [2] have developed an interactive system which allowed manual registration of slices and provided an active contour based algorithm for the semi-automated cell segmentation. Montgomery et al. [3] at the NASA-Ames Biocomputation Center have built a system called ROSS (Reconstruction Of Serial Sections) which allows manual segmentation, registration and reconstruction of slices.

A more recent work is SLICER [4], an open source software developed at MIT, which allows manual and semi-automatic segmentation, and registration of sequences. It also provides tools for 3D visualization and manipulation. This software has the additional advantage of providing a module system which permits the incorporation of newly developed algorithms. However, at the moment of this writing, the only segmentation algorithms incorporated were based on edge detection.

Finally, the propietary application Northen Eclipse [5] provides many interesting features: performs various geometric measures, enhancement, morphological operations,

deconvolution, visualization, etc.

In addition, there are some related works carried out by our group that will be presented in a following section.

1.2 Objective

The main objective is to devise a general segmentation algorithm, independent of the acquisition technique and the type of cellular tissue involved that is able to automatically detect structures of interest in 3D data in the most accurate way possible.

1.2.1 Previous work in our group

This work is part of an ongoing effort in our group started with [1] which aims to devise a computer system able to aid the specialist in the acquisition, segmentation, reconstruction and visualization of 3D biological tissue. It must be a general technique, independent of the acquisition technique and the type of cellular tissue involved.

As mentioned before, the first step towards this objective was the software Neuro3d developed in [1]. It was an interactive system which allowed the automatic enhancement and registration of the sequence. Then the segmentation was performed manually on the slices, and then it performed 3D visualization of the reconstructed structures.

In order to automate the segmentation procedure, the Morphing Active Contours (MAC) algorithm [6] was developed an tested for tracking over TEM acquired neural tissue.

Later a second version of Neuro3D, called Bio3d [7] was developed . This software allows manual mosaicking, registration, and segmentation. After these steps are performed, this tool provides 3D reconstruction and visualization, plus 3D editing of the resulting surface.

1.2.2 Scope of this work

As pointed out in the previous section this work is part of a bigger attempt to develop a complete 3D reconstruction system. This work is focused on the segmentation stage of this system. Our goal then, is to devise a general technique, independent of the acquisition technique and the type of cellular tissue involved that, once the biologist has selected in the first frame of the sequence the structures in which he/she is interested, is able to automatically follow these structures along the sequence, segmenting them as they evolve and also detecting and segmenting any new objects of the same kind that may appear. For this purpose we will review, evaluate, and eventually modify some well established segmentation algorithms, in order to successfully segment our biological sequences. In addition, we will test the performance of these algorithms on a broad range of real biological data.

1.3 State of the art

In the past few decades researchers have developed a great number of approaches to image segmentation from different perspectives. In this section we will try to present a review of some of these approaches and situate our algorithm in this context.

It is a hard task, while not impossible, to determine a unique criterion to classify all possible segmentation algorithms. For this reason, in the following we will present various possible classifications. We start with the classic algorithms present in the literature and classify them from the point of view of the domain used by the algorithm. Using this criterion we can roughly classify the algorithms into (see [8]):

- local filtering: these algorithms make use of local information and cannot guarantee continuous boundaries. Examples of this are Sobel, Prewitt, Hildredt-Marr operators, or canny and deriche edge detectors.
- boundary based methods: these methods make use of information along the boundary of the regions and requires a good initial condition. An example of this formulation is Kass et al. [9].
- region growing and merging: these methods use statistics inside regions, but they often generate irregular edges.
- global optimization approaches based on energy functionals, bayesian or Minimum Description Length (MDL) criteria. These approaches use a global cost which when minimized yields the segmented image. A drawback of these approaches is that it is often difficult to find their minima.

In the past years, there have been some efforts towards the unification of many of those approaches, starting with Haralick [10] which integrated region growing and edge detection. Kass et al. [9] combined the active contour approach with a global energy criterion. Perhaps one of the most complete in the sense of integration is the work of Zhu et al. [8] which unified most of these approaches. This work jointly with Ronfard's [11] settled the basis to the region based active contours algorithms later improved by many authors: Chan et al. [12], Paragios et al. [13], Jehan-Besson et al. [14]. The end of the road in this family are algorithms like [15]. This algorithm starts from a global maximum likelihood (ML) criterion, that yields a variational problem which consists in the minimization of an energy functional. This minimization is performed by computing its gradient descent. This approach includes all the discussed approaches in the following way:

- geometric formulation by the use of active contours
- regularization via global measures of the evolving curve (e.g. length or area).
- edge detection and local filtering by gradient computation.
- region merging and growing by the active region term
- global approach by settling the problem in a ML framework
- handling complex topologies by using the level set approach
- integration of multiple features by introducing them as joint densities in the ML approach.

The abovementioned algorithms can be considered of low level of abstraction, because the cues used are somehow crude, i.e. they are extracted directly from the image

with little or no processing. Examples of these low-level features are brightness, color, or texture.

On the other hand, there are some higher-level approaches that make use of mid or high-level knowledge about the input images. Although they are not directly related with our work, in the following we will mention some of the most remarkable of them

- edge/feature integration methods: these algorithms aim to combine local features, using a measure of meaningfulness and to find the groups of features that maximizes that criterion. These are high-level algorithms, as their goal is to group previously computed low-level features trying to maximize the meaningfulness of the set.
  - Tensor Voting (Medioni et al. [16]): this approach consists on the local computation of certain features at input sites of the image, and their encoding in a tensor. Then this information is propagated to the neighboring sites according to a predefined tensor field. After that, each site collects all the information cast at its location and encodes in a new tensor. In this way, the information is gathered in a parallel process that simultaneously discovers *relevant* features.
  - Structural Saliency: Sa'ashua et al. [17]. In this approach a saliency map is computed for each pixel in the image. This computation is carried out locally by an iterative optimization process. They measure the saliency of curves by taking into account its length and curvature. Then the interesting locations are defined as those with maximum saliency.
  - Generalized Grouping: This approach, proposed by Lindenbaum [18], consists on a generalized grouping algorithm independent of the domain. This algorithm has two components: the grouping cues and the grouping mechanism. With these cues they construct a graph where the vertices are the cues and the arcs contain the grouping information (estimated from the cues). Then the vertices are grouped by trying to maximize the likelihood of the observed data.
- Graph Cuts (Malik et al. [19]): in this approach a graph is constructed where each node is a point in the feature space and the edges connect every pair of nodes. The weight of each edge is related to the similarity between the connected nodes. Then the cost of a certain partition (or cut) is defined using the weights of the cutted edges. Similarly, they proposed a cost for an association, and then they proposed an algorithm that minimizes this combined cost.
- Level Lines methods (Cao et al. [20], Pardo [21]): these methods are based on an alternative representation on the image, that uses as inputs the level lines of the image instead of using the grey level. This representation is often called topographic map and it is a structured representation of the image that is contrast invariant. Segmentation is often achieved by retaining only those level lines that maximize a certain criterion (number of T-junctions, big contrast changes, etc).
- A Contrario models: this formulation, introduced by Morel [22], states that a set of features is perceptually relevant only if it cannot arise by chance in white noise. That leads to the computation of a universal variable, the number of false alarms, which is a threshold that tells us when a certain configuration is significative or not. Edge detectors using this principle are also usually implemented using the topographic map.

The kind of algorithm we will discuss and implement on this work falls on the category of low-level algorithms, and aims to integrate all those low-level approaches.

Usually, when facing a real segmentation problem, one needs to use as much information as possible. In addition, objects of interest generally are characterized by very different properties (color, texture), so multiple features of the input images must be integrated in the algorithms. Thus a segmentation/tracking algorithm must take into account several characteristics in order to success over varied real data sets:

- borders: many real-life objects are characterized by edges, so the algorithm must include an edge detection step.
- integration of different features: as real data present a wide range of variation, the features that distinguish the objects also vary a lot. Thus, a feature relevant in one case, can be completely useless in another. For this reason, an algorithm must take into account all the information available.
- global energy/cost criterion: There are two kind of arguments that can be given in favor of this approach. The first concerns the complexity of the algorithms. While it is possible to heuristically define an algorithm with many *good properties*, a great number of parameters appear. If we state the problem on a cost minimization framework this number can be drastically reduced.

  The second argument deals with comparison between algorithms. If we segment an image using two different algorithms, the cost associated with each result is a suitable measure of accuracy. For an extension on this arguments see [23].
- ability to handle topologically complex objects and changes in this topology during the segmentation.

Other features, while not essential are highly desirable:

- automatic selection of the relevant features over a set of possible features.
- a minimization process (for the cost function) that avoids falling into local minima.

In the last few years, many successful algorithms for the automated segmentation of images use PDE based active contour approaches, where a curve is evolved so as to maximize certain criterion involving many descriptors of its interior and exterior and the boundaries. However, these techniques require the manual selection of several parameters, which make impractical the work with long sequences or with a very dissimilar set of sequences. Unfortunately this is precisely the case with the real 3D biological sequences we are facing.

For this reason, the main goal of this work is to evaluate, and modify if needed, some of the well established PDE based algorithms in order to achieve an accurate segmentation over these biological sequences.

1.3.1 Related Work

Many algorithms have been proposed in the literature to address this type of problem, we will recall here only the works using active contour models, which are directly related to our technique.

Boundary functionals were first proposed by Kass et. al. [9] with the snake model, where a curve is deformed toward the edges of an image. Later, this model was improved by the geodesic active contours model introduced by Caselles et al. [24] were they propose an intrisically geometric variational approach.

One of the first variational approaches which make use of region information was introduced by Mumford and Shah. [25] where a piecewise constant model for each region was assumed. Region based active contours were first introduced by Ronfard [11] and Cohen [26]. The key idea was the introduction of global region information to drive the active contour model. This was performed by evolving a curve according to a velocity which depends on the competition between region descriptors. However it is not trivial to derive the evolution equation from a criterion including both boundary and region terms.

Later, Zhu [8] et al. proposed the use of region based statistical descriptors in an energy based variational approach. Then, an evolution equation was derived from the energy by transforming region terms to boundary terms. Many authors (Chan et al. [12], Paragios et al. [13], Jehan-Besson et al. [14]) proposed a similar approach, with the addition of the implementation using the level set paradigm, which provided an elegant solution to the curve evolution problem.

From the point of view of the integration of different features, one of the pioneer works was also Zhu et al. [8] where they introduce a global criterion capable to deal with multiple features. In that work they test their algorithm with color and texture.
Deriche et al. [15] extended this work to include an edge detection term, and they added motion channels to the feature set. In addition, they also implement the evolution in the level set paradigm. Chan et al. also adapted their framework [27] to vector valued images.

For dealing with textured images Deriche used the model [28] with the structure tensor from [29]. Aujol et al. [30] introduced a wavelet based algorithm in the same framework, and also Chan et. al [31] introduced texture in that framework using Gabor filters.

In this kind of model, shape information has been also integrated, in this direction, Rousson et al. [32] combined region based approaches with geometric shape priors.

Freedman [33] presents an alternative formulation for the energy functional, based on pdf metrics (Kullback-Leibler and Battacharyya) instead of Maximum Likelihood.

Finally, it is worth to mention the work of Jehan-Besson et al. [34] where a review and an unifying theoretical analysis of region based active contours is presented.

1.4 Outline of the algorithm

Since the works of Ronfard and Cohen, many authors have proposed region based active contour (RBAC) algorithms. There are many ways to state the problem, and derive the energy, but in the end, all approaches end up in the same kind of variational problem. Our starting point is the region-based active contour framework proposed by Rousson et

al. [15].

This framework allows the integration of different features in a variational framework. They model these features as a vector including color, texture and motion channels.

For a single gray-level (or color) image $I : \Omega \rightarrow R^3$, with $\Omega \in R^2$, we compute a corresponding *feature image* $U : \Omega \rightarrow R^N$ where N is the number of feature channels (e.g. color, texture, optical flow, etc).

Using these features, an energy functional is constructed including a data term and a regularity term. The data term is related to the likelihood of the probability density function of each region with respect to the observed features and an edge detection term, for which we follow [24]. The regularity term is the classical mean curvature flow.

To solve the segmentation problem, we seek for a curve Γ which gives a partition of the space Ω in two (non necessarily connected) regions Ω_1 and Ω_2: the object of interest and the background. This is accomplished by finding the curve that divides the feature image in such a way that it maximizes the likelihood of each region probability distribution with respect to the observed previous frame. This process is repeated for each frame, taking as ground truth the segmentation of the previous one. The algorithm takes as input the sequence of images and an user provided segmentation for the first image (a set of manually segmented objects of interest). Then for each frame it evolves the curve that defines the segmented regions towards a (local) minimum of the energy.

As we are dealing with multiple features, we are interested in assigning different weights to each channel. This is performed in an adaptive way: after the segmentation of one frame is complete, we measure the *distance* between the two regions and assign the weights according to this distance. As big distance value implies that this feature separates well the two regions, thus it gets a big weight value. On the other hand, if the distance is small, the discrimination power of this feature is low, so it gets a small weight.

Finally, as objects tend to appear and dissapear, we add a detection/correction algorithm in order to detect if the evolving curve has lost any part of the object, and if it did so, we correct this situation by launching a re-initialization algorithm that finds lost objects.

1.4.1 Our contribution

As we stated before, in this work we try to achieve the segmentation of several different biological sequences, using an algorithm that must be general enough to deal with all of them without changes, but also it must be adaptive enough in order to perform well in each particular sequence.

In order to do so, we improve on previous Region Based algorithms in two aspects: by introducing a way to compute and update the optimum weights for the different channels involved (color, texture, etc.) and by estimating if the moving curve has lost any object so as to launch a re-initialization step.

Finally, to validate our approach, several examples of biological sequences, quite long and different among themselves, are presented. Over these sequences, our method outperforms previous approaches.

1.5 Organization of this thesis

In previous sections we have presented the problem that we addressed in this work. Chapter 2 introduces the basic background on curve evolution and the general formulation family of algorithms used in this work. Pre-processing and feature extraction are covered in chapter 3. In chapter 4 we show our contribution to improve the existing algorithms. In chapter 5 we present a discussion of the experimental results and a comparison with previous approaches. Finally, we give some conclusions and possible extensions in chapter 6.

Chapter 2

Theoretical background

2.1 Introduction

The purpose of this chapter is to introduce the basic background on PDE based segmentation algorithms. This introduction is not intended to be exhaustive and we will refer to in-depth studies on several areas to obtain a more comprehensive treatment of the subject.

The use of partial differential equations (PDE's) for image processing has become a well established research topic in the past few years. The basic idea is to think of an image as a function in a continuous domain, rather than in a discrete grid. Thus, we define an image as the mapping $I : \mathbb{R}^n \to \mathbb{R}$ ($n = 2$ for classical two-dimensional image processing), where $I(x, y)$ represents the gray-level value at point (x, y). Then, an image processing algorithm can be defined as the evolution of an initial image $I_0(x, y)$, following a PDE of the form

$$I_t = \mathcal{F}(I, I_x, I_y, \ldots) \tag{2.1}$$

where $I(x, y, t) : \mathbb{R}^2 \times [0, \tau) \to \mathbb{R}$ is the evolving image and $\mathcal{F}(.)$ is a functional determined by the algorithm. When the evolution (2.1) finishes ($t = t_{final}$, usually $t_{final} = \infty$), the processed image $I(x, y, t_{final})$ is obtained. The challenge here is to find the *correct* $\mathcal{F}(.)$ such that $I(x, y, t_{final})$ matches the desired result (segmentation, noise removal, etc).

PDE based algorithms are used in image restoration and enhancement, shape analysis, shape from shading and stereo, segmentation, tracking, morphing, surface reconstruction, inpainting and mathematical morphology, among others. For a more extensive review of the theory and applications, we refer to [35, 23, 36].

A first advantage of this approach is that the accuracy of the algorithms relies on the precision of the discretization of the continuous model, and not on the formulation of the theory, as in classical image processing algorithms or Discrete Mathematical Morphology.

The consideration of the output of an algorithm as the solution of a PDE also provides a formal framework in which it is possible to guarantee interesting properties (like existence and uniqueness, or maximum principles among others) of these solutions. In addition, one counts with the aid of extensive research in numerical analysis, differential geometry and PDE theory.

2.1.1 chapter outline

Our starting point was the basics of PDE based algorithms and its motivation. In the following we will introduce the basics of curve evolution theory. We will continue with a review of some well known PDE based segmentation algorithms and give a closer look on the GAR framework, on which our algorithm is based. And finally, we will review some recent works that are related to our approach.

2.2 Fundamentals of curve evolution

The goal of this section is to review the theory of curve (or front in higher dimensions) evolution. For this topic we will follow [37, 35, 38, 6] and in some cases we will include excerpts of these works.

We will concentrate on tracking the evolution of a curve $\mathcal{C}$ over time when it is governed by a given velocity field $\vec{F}$. We will introduce first the planar case to gain in-

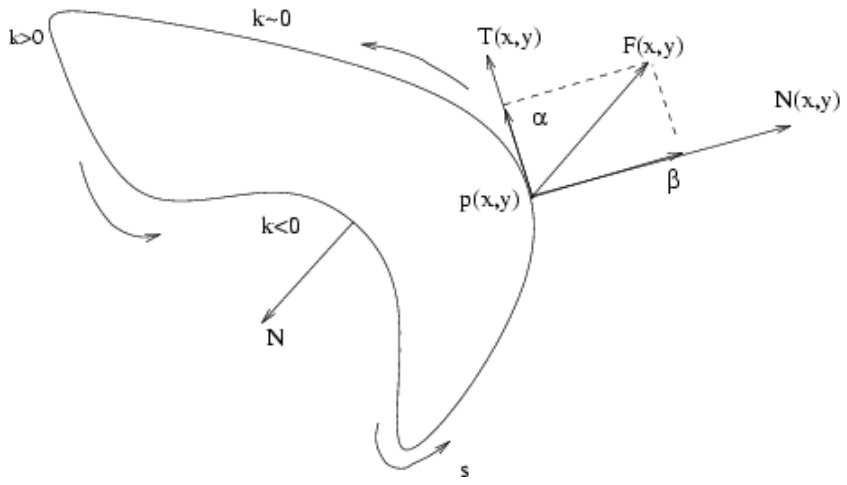


Figure 2.1: Time evolving curve

sight and after that we will see how to extend the presented concepts to higher dimensions.

We start by defining a curve $\mathcal{C}$ as the mapping $C(s) : S^1 \to \mathbb{R}^2$ which can be written in Cartesian coordinates as $C(s) = [x(s), y(s)]$ where $x, y : S^1 \to R$ are smooth enough to define all the relevant derivatives.

Then, if our front is evolving over time, we obtain a family of curves $C(s,t) : S^1 \times [0,\tau) \to \mathbb{R}^2$. For the following we assume that $\mathcal{C}$ is a family of simple and closed (embedded) curves, where t represents the time. Then, if the curve evolves according to $\vec{F}$ the curve must satisfy

$$\frac{\partial C}{\partial t} = \vec{F} \tag{2.2}$$

We consider $\vec{F} = (\alpha, \beta)$ as seen in figure 2.1, where and α and β are the tangent and normal projections of $\vec{F}$, respectively. Then, we can rewrite the evolution equation (2.2) as

$$\begin{cases} \frac{\partial C}{\partial t} = \alpha\vec{T} + \beta\vec{N} \\ C(s,0) = C_0(s) \end{cases} \tag{2.3}$$

where $\vec{N}$ is the normal (outward) unit vector and $\vec{T}$ is the unit tangent vector.

Since we are interested in the *geometric* evolution of the curve, we will follow the evolution of the geometric trace of the curve $\mathcal{C}$ in the plane. In this way, if we take another parametrization of the curve and if we look at its trace, we will see the same curve.

If we evolve $\mathcal{C}$ with a velocity $\vec{F}$ such that β only depends on intrinsic properties of the curve (i.e. is independent of the parametrization), then the resulting deformation of $\mathcal{C}$ is determined only by β. This means that the tangential velocity only induces a re-parametrization (i.e. the velocity with one travels over the curve) but not a geometric deformation. A proof of this can be found in [39]. Thus, the evolution (2.3) is equivalent to

$$\frac{\partial C}{\partial t} = \beta \vec{N} \tag{2.4}$$

In general, β will be an arbitrary function which can be described as

$$\beta = \beta(L, G, I) \tag{2.5}$$

where

- L: Local properties of the front, determined by its geometric features, like the curvature or normal direction in each point of the front.
- G: Global properties of the front. These depends on shape and position of the curve. For example integrals along the front of some quantity, like Hausdorff or Lebesgue measures of the front.
- I: Properties independent of the front, e.g. an external velocity which drives the evolution of the front.

2.2.1 Examples

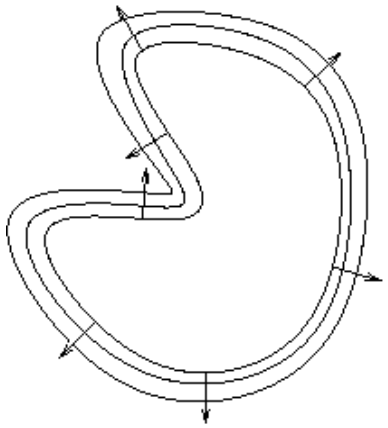


Figure 2.2: Constant velocity evolution ($\beta = 1$)

In this section we will present some examples of well known evolutions that are common in the literature. We choose this examples because they show the issues that arise in curve evolution and they also show the basic properties of the flows we will use for segmentation.

We start with the simplest evolution, which comes from taking $\beta = 1$ in (2.4)

$$\frac{\partial \mathcal{C}}{\partial t} = 1.\vec{N} \tag{2.6}$$

In this evolution all points move with the same velocity outwards and the curve constantly *inflates*. As seen in figure 2.2 the points in the zones of high negative curvature will come closer and collapse developing a sharp corner. In other words, the solution which started from a smooth initial condition, develop a singularity in finite time. Once this singularity appears, the normal vector is ambiguously defined and it is not clear how to

continue with the evolution. Thus, after the development of this singularity we cannot find a differentiable solution. The computation of suitable solutions for this kind of evolution is discussed in section 2.2.4.

In addition, it can be shown that this evolution is a continuous formulation for the dilation in mathematical morphology (see [35]).

The next evolution we consider is the well know *Euclidean shortening flow*.

$$\frac{\partial \mathcal{C}}{\partial t} = -\kappa \vec{N} \tag{2.7}$$

In this case, the front evolves according to the values of the curvature. This evolution can be seen intuitively in figure 2.1, where the points with $\kappa > 0$ will move outwards and those with $\kappa < 0$ will move inwards, thus the curve reduces its oscillations.
Formally, this is equivalent to say that the curve monotonically reduces its total variation (see [37]). Thus, F acts as a regularizing force. It can be proven (see [40]) that the front first evolves to a convex curve and then collapses smoothly to a point and disappears.
In addition, this evolution defines a geometric, euclidean invariant scale space as shown in [41].

It is very interesting to see the effect of combining the two previously mentioned flows. This leads us to the following equation

$$\frac{\partial \mathcal{C}}{\partial t} = (1 - \epsilon\kappa)\vec{N} \tag{2.8}$$

here we assume that ϵ is a small positive constant. Although similar with the flow (2.6), it has a very different behavior. In this evolution the front does not develop singularities, due to the regularization provided by the curvature term.

The last example presented is the *Affine Shortening Flow*

$$\frac{\partial \mathcal{C}}{\partial t} = \kappa^{\frac{1}{3}} \vec{N} \tag{2.9}$$

which has the same properties of the euclidean flow, with the addition that the generated scale space is affine-invariant. It is also numerically more stable than the euclidean one, and allows a higher degree of smoothing while preserving the same amount of structure (see [42]).

2.2.2 Lagrangian Formulation

We recall from previous section that $C(s) = [x(s), y(s)]$, then we can write the equations of motion in terms of x and y. But before we need to compute some geometric quantities from x and y. For instance the normal unit vector, and the curvature can be written as follows

$$\begin{cases} \vec{N} = \frac{(y_s, -x_s)}{\sqrt{x_s^2 + y_s^2}} \\ \kappa = \frac{y_{ss}x_s - x_{ss}y_s}{\sqrt{x_s^2 + y_s^2}} \end{cases} \tag{2.10}$$

Then, if we evolve $\mathcal{C}$ following (2.4), each point of the curve moves according to the

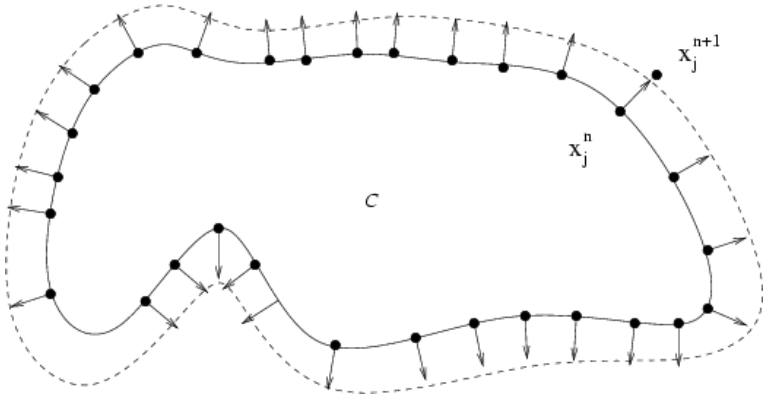


Figure 2.3: Curve evolution in Lagrangian formulation

following PDE

$$\begin{cases} x_t = \beta \frac{x_s}{\sqrt{x_s^2+y_s^2}} \\ y_t = \beta \frac{y_s}{\sqrt{x_s^2+y_s^2}} \end{cases} \tag{2.11}$$

where we have substituted the expressions for $\vec{N}$ and κ from eq. (2.10).

In order to implement this evolution, a discretization of the curve must be introduced. This is usually performed by taking N evenly spaced points over the curve (see fig. 2.3), and solving (2.11) for all these particles.

Although simple, this approach has many drawbacks:

- As the front evolves, the marked points do not remain evenly spaced, so we must recover the curve by interpolation, and then re-sample with an uniform distribution. This changes the problem that we are solving in a non-obvious way: as the density of particles will be low in some regions of the front and high in another regions. In regions with lower density, this interpolation will give inaccurate results for the recovered curve.

- The computation of geometrical quantities (normal, curvature) depends on this marker points, and their accuracy is strongly dependent on this sampling. Moreover, the velocity of particle propagation depends on this geometrical quantities, and the new position of these particles depends on the velocity. This leads us to an unstable feedback loop, which can make the evolution unstable.

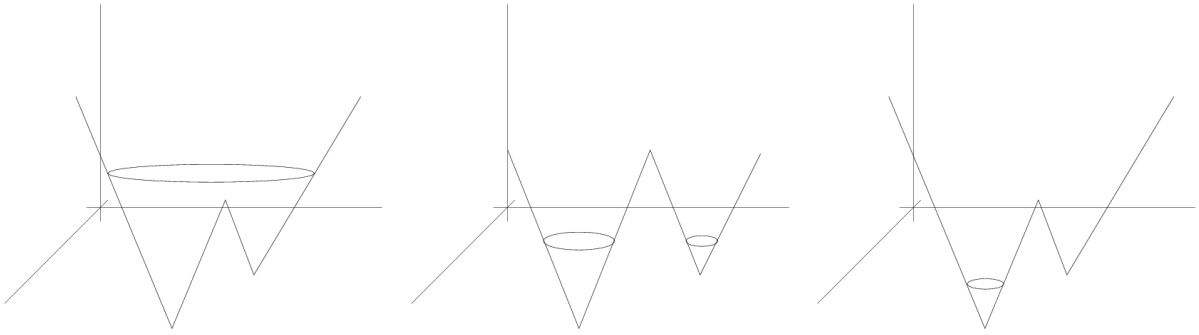
- Even with the simplest evolution, topological changes can occur, e.g. a curve that splits in two parts or various curves that merge into one. Using this type of representation it is hard (when not impossible) to handle this kind of change.

- As shown in section 2.2.1, even the simplest of the evolutions also develop singularities. For example this leads to particles that collapse. In this situation we must select a correct solution in some sense[1], which is a difficult task because we must choose the particle that gives this solution.

[1] we will present this solution in a following section

For all the abovementioned problems, there are various *workarounds* in the literature, but all have the same problems: they alter the evolution in non-obvious ways leading to solutions with different properties, or they rely in heuristics that work well in particular cases and are not extensible to other dimensions.
For these reasons we will introduce in section 2.2.3 the Eulerian formulation proposed by Osher and Sethian to circumvent those problems.

2.2.3 Level Set Formulation



(a) Embedded curve (b) Splitting (c) Disappearance

Figure 2.4: Handling of topological changes via the embedding function

As pointed out in section 2.2.2 the Lagrangian approach has many drawbacks which can be avoided using the level set formulation. This approach was first introduced by Osher and Sethian in 1988 (see [43]). The central idea, is to avoid the problems associated with the topological changes by embedding the curve in a high dimensional surface. Thus, what is a topological change in a 2D planar curve, does not have a special appearance for a 3D surface, and is handled naturally. This is depicted in figure 2.4, where the initial curve is a unique connected component that divides itself in two parts, and then one of them disappears. During the whole process, the embedding surface has not suffered any special change.

In this formulation we consider that the evolving curve $C(t)$ is embedded in a function $\phi : \Omega \times [0, \tau) \to \mathbb{R}$ such that

$$\begin{cases} \phi(\mathbf{x}, t) < 0 \; in \; \Omega_{int} \\ \phi(\mathbf{x}, t) = 0 \; in \; \mathcal{C} \\ \phi(\mathbf{x}, t) > 0 \; in \; \Omega_{ext} \end{cases} \tag{2.12}$$

where Ω_{int} and Ω_{ext} are the interior and the exterior of the curve $\mathcal{C}$. Then $C(t)$ is the zero level-set of $\phi(\mathbf{x}, t)$, i.e. $C(t) = \{x \in \Omega \; : \; \phi(\mathbf{x}, t) = 0\}$.

If we differentiate the equation $\phi(C(t), t) \equiv 0$ with respect to t we get the following PDE for $\phi(\mathbf{x}, t)$

$$\phi_t + \frac{\partial C(t)}{\partial t} \nabla \phi = 0 \tag{2.13}$$

then using equation (2.4)

$$\phi_t + \beta|\nabla\phi| = 0 \tag{2.14}$$

where β is the normal velocity as defined in (2.4).

In spite of following the evolution of the curve $C(t)$, the PDE (2.14) is solved in a rectangular grid. When $\frac{d\phi}{dt} \approx 0$, the resulting curve $\mathcal{C}$ can be obtained computing the zero level-set of ϕ. This can be achieved by using some well-known contour finding algorithms, like marching squares (2D) or marching cubes (3D) [44].

An important point to validate this formulation has to deal with the choice of the embbeding function. As we are embbeding our evolving curve in a higher dimension function, and the only requisite is that its zero level-set is our curve, the choice of this function is not unique. Thus we must be sure that for all possible embedding functions the obtained curve when the evolution finishes is the same. Fortunately, for a large family of these functions, this can be proven, and thus the resulting curve is independent of the embedding.

This approach has many advantages:

- Data structures are simple: we solve a PDE in a fixed rectangular grid, which is naturally associated with the notion of pixel in a digital image.
- As the grid points are fixed the numerical unstability associated to Lagrangian approximations are avoided.
- Topological changes are naturally handled by the embedding function.
- The equations that arise in curve evolution generally lead to non-smooth solutions that must be specially treated, as we will see in the following chapters. An advantage of this framework is that it is easier to impose the necessary conditions for computing these kind of solutions.
- Geometrical quantities are easy to compute from the embedding function: normals, curvatures, area, volume, etc. As an example, we can compute the the normal unit vector and the mean curvature:

$$\begin{cases} \vec{N} = \frac{\nabla\phi}{|\nabla\phi|} \\ \kappa = \nabla.\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \end{cases} \tag{2.15}$$

- The algorithms derived in this formulation are easy extended to higher dimensions, because this change reflects only on the number of variables of the embedding function, but the equation to solve remains the same, provided we can compute β. However this could by tricky if, for example, the evolution depends on the curvature. In 2D there is only one curvature, but in 3D there are two main curvatures.

The addition of the extra dimension to the problem increases the computational effort of the algorithm. However, there are some techniques to reduce this cost.

Perhaps the most popular of this approaches, presented in [45], is the computation of the solution of (2.13) only in a narrow band around the contour. Thus the real dimension of the problem remains the same as the original.

2.2.3.1 Re-distancing Algorithm

In order to ensure numerical stability, is necessary that

$$0 < k < |\nabla\phi| < K \tag{2.16}$$

or in other words, that the level-set function is not very flat nor very steep. One common choice for ϕ is the signed distance function to the front $\mathcal{C}$, because $|\nabla\phi| = 1$ and thus the condition (2.16) holds.

As the front evolves following the PDE (2.13), ϕ not necessarily remains a distance function. In order to ensure the condition (2.16), a re-initialization algorithm is needed. This is performed periodically after a few iterations of the main algorithm in order to ensure that ϕ be a signed distance again. This evolution is shown in [46] and its performed using the following equation

$$\phi_t = S(\phi_0)\,(|\nabla\phi| - 1) \tag{2.17}$$

where S is a smoothed version of the sign function such as

$$S(\phi_t) = \frac{\phi_0}{\sqrt{\phi_0^2 + \epsilon^2}} \tag{2.18}$$

where ϵ is a small positive constant.

One important property that a re-distancing algorithm must hold is that the zero-level set of ϕ remains unchanged. This ensures that the evolving curve is not modified by this algorithm.

2.2.4 Weak solutions

As mentioned before, even if we move a curve with the simplest of the evolutions the smooth initial condition develops a singularity in finite time. Once this singularity appears, it is not clear how to continue with the evolution. In the following, we will show how to compute a suitable solution for this kind of problems.
We will start considering the case when the PDE can be written in the following form:

$$\frac{\partial\phi}{\partial t} + \nabla.F(\phi) = 0 \tag{2.19}$$

where F is a function $F : \mathbb{R} \to \mathbb{R}^N$. In order to compute a suitable solution for this problem, we must extend the domain in which the solution is searched for. This leads us to the notion of *weak* solution. The main idea is to compute a solution of an integral version of the problem, in which the solution is not necessarily differentiable. In this framework the obtained solution is not unique, so we must provide a criterion to select the solution among all possible ones.

In the case of the equation (2.19) this integral formulation comes from its physical interpretation. It can be shown that it represents an hyperbolic conservation law of the scalar quantity ϕ with flux F. Thus it can be written as

$$\frac{\partial}{\partial t}\int_V \phi dv = -\int_{\partial V} F(\phi).\vec{N}da \tag{2.20}$$

The interpretation of (2.20) is that the change of the amount of ϕ in the volume V is equal to the flux F crossing the boundaries of the volume. This formulation of the problem suggests the existence of a single physically meaningful solution.

In this kind of equations two different types of singularities appear: *shocks* and *rarefactions*, due to converging or diverging characteristics respectively.

In the first case, we have a finite time for which two characteristics collide. Each one carries its own value, so after the collision it is not clear which value is the correct one. Along the line that marks the collision, known as *shock*, the solution discontinuously switches its value according to the values carried by the collided characteristics. The physically correct solution is chosen by enforcing conservation along the shock, the so-called *jump* condition.

In the second case, we have two diverging characteristics that transport the solution from a certain point. In this case, there will be a gap between these characteristics in which the solution cannot be determined. This zone is called *rarefaction zone*. In this case there are many ways to fill the gap. This is usually performed by imposing the so-called *Entropy* condition, which states that the characteristics must flow into the discontinuity. This condition can be illustrated with the following example, proposed by Sethian for the propagation of flame fronts. If we consider the front as a burning flame, the entropy condition says: "once a particle is burnt, it remains burnt".

From the geometrical point of view, we are interested in tracking the evolution of the curve, considered as an advancing front that divides the domain in two regions. As the front advances it intersects itself developing loops, this condition rules out those solutions with self-intersections.

Lax proved in [47] that a generalized solution of (2.19) which has only singularities satisfying both the jump and the entropy conditions, exist and is unique.

Summarizing, a smooth initial condition evolves according to eq. (2.19) and remains smooth up to the point where a singularity forms in the solution. After this point, the PDE in its classical form is not longer valid, but the integral conservation law of eq. (2.20) remains valid. Thus we can look for a not necessarily differentiable *weak* solution, imposing certain conditions to pick only one among all possible solutions. We can also be sure that, with these conditions, the chosen solution is unique; and if a classical solution exist, it is identical to the *weak* one.

There is another way to derive a weak solution of (2.19) by considering a modified version of the PDE of the form

$$\frac{\partial \phi}{\partial t} + \nabla . F(\phi) = \epsilon \Delta \phi \tag{2.21}$$

where the added term $\Delta\phi$ is a diffusive term that acts as a regularizing force. It can be proved that the solution ϕ_ϵ of (2.21) stays smooth for all time. If we take the limit of ϕ_ϵ as $\epsilon \to 0$, the resulting function $\hat{\phi}$ is a solution of (2.19). This solution is often called *viscosity* solution. As pointed above by Lax result, the solutions obtained by imposing the entropy condition are unique, thus $\hat{\phi}$ is identical to the entropy solution.

2.2.4.1 Viscosity solutions

All the above considerations were presented in the case of an equation of the form (2.19) with a smooth initial condition. In the following we will discuss what happens if we remove those hypotheses. In the first place, we will consider a more general PDE of the form

$$\frac{\partial \phi}{\partial t} + H(D\phi, x) = 0 \tag{2.22}$$

this is called a Hamilton-Jacobi equation, for which (2.19) is a particular case. Here $D\phi$ represents all possible partial derivatives of ϕ. In the case of our evolution equation (2.14) $H = \beta|\nabla\phi|$. For the following we assume that the Hamiltonean H is a smooth function of its arguments and that the initial condition it is not necessarily smooth. In this case we must resort to the theory of viscosity solutions introduced by Crandall et al. [48]. The first reasonable approach, according to the previous discussion, is to add a viscosity term, that is,

$$\frac{\partial \phi}{\partial t} + H(D\phi, x) = \epsilon\Delta\phi \tag{2.23}$$

then, given a solution ϕ_ϵ of the above equation, one must show that such a solution is smooth, and that its limit as ϵ vanishes gives the appropriate weak solution.

Rather than defining the weak solution as the limit of smooth solutions, Crandall et al. [48] devised the theory of viscosity solutions, in which they present an alternate definition of these solutions. Informally, this theory deals with non-smooth curves by enclosing them between two smooth curves, one inside and one outside, taking them to the limit, and using the causality principle in the evolution. This principle states that enclosed curves will not cross each other when evolving, so this ensures that the non-smooth solution remains enclosed between the two smooth solutions. For a formal presentation of this theory we refer to [49].

In order to check the validity of this approach and to make a link with the previous approaches, several results must be taken into account (see [37]):

- If ϕ is a smooth solution of (2.22), then it is a viscosity solution
- If a viscosity solution ϕ is differentiable at some point, then it satisfies the Hamilton-Jacobi equation there.
- The above viscosity solution is unique, given appropriate initial conditions
- The solution produced by taking the limit of the smooth solutions ϕ_ϵ as $\epsilon \to 0$, is a viscosity solution. By uniqueness, this solution must be the given above.

These statements ensure us that, the viscosity solution, the solution obtained by the entropy condition and by the limit of the smooth solutions are the same.

2.2.5 Numerical implementation

The goal of a numerical implementation for a PDE is to compute an approximate solution $\hat{u}$ that corresponds with the analytical solution u. Usually this solutions are achieved by discretizing the spatial domain in a rectangular grid (with size Δx) and the

temporal domain in a grid with size Δt. Then the discrete equation obtained is solved for each grid point.

To check if such a solution is suitable, several concepts must be discussed:
Convergence: The difference between the numerical solution and the real solution must vanish as the discretization becomes finer ($\Delta x \to 0$).
Consistency: A discretization is said to be consistent if the discrete equation converges to the PDE as $\Delta x \to 0$
Stability: As we are solving the discrete equation in a computer, we are tied to numerical errors due to the representation in the computer, and we do not want these errors to grow unbound. To avoid this we must check the stability of the scheme: that is given an initial condition, a solution is said to be stable if the solution remain bounded for all times. This is usually checked using the Von Neumann method (see [50]).

All these concepts are closely related, and in the linear case Lax's theorem ([47]) states that for a consistent scheme with initial data piecewise continuous, stability and convergence are equivalent.

Another important consideration is to respect the direction of the flow of information. For a point x in the domain, the solution in this point depends on a certain region of the domain. This region is called "domain of dependence". On the other hand, the numerical solution at x depend on other grid points (given by the type of discretization used), that we can call "numerical domain of dependence". Thus for computing an accurate solution, the numerical domain of dependence must contain the domain of dependence. These methods are usually called *upwind* because they respect the direction on which information flows.

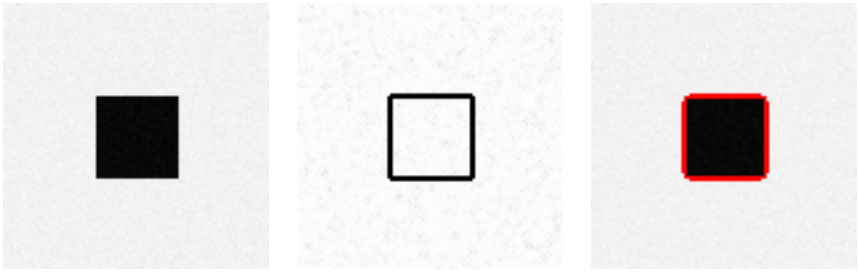
In the case of non-linear PDEs the picture is not as good as in the linear case. As we showed before, even with a smooth initial condition the solutions develop singularities, and a different notion of solution must be considered. From the numerical point of view, this reflects on the failure of classical approximations for solving this kind of equations. An example of this can be seen in [37] where a very simple Eulerian evolution is discretized using central differences in space, and the solution is intrinsically unstable. The reason for this is that, as explained in section 2.2.4, the entropy condition must be satisfied in order to pick the physically meaningful solution. Thus, the numerical scheme must also satisfy the entropy condition in order to pick the correct weak solution. In [43] Osher and Sethian proposed such a scheme: first, the scheme must approximate the hyperbolic conservation law, and then they impose monotonicity on it in order to guarantee the correct choice of the solution. In addition, they use an upwind scheme.

2.3 Active Contours for segmentation

In the following we will review the basic theory and some well known variational approaches to segmentation. We will derive the formulation in the case of scalar gray-level images and we will extend the results for color or vector-valued images when appropriate. We will also derive all results for the two-region case, and after that we will extend these results to the M-regions case.

In the active contour model, the input image I is considered as a function $I : \Omega \subset$

$\mathbb{R}^2 \rightarrow R$. To solve the segmentation problem, we seek for a curve Γ which gives a partition of the space Ω in two (non necessarily connected) regions Ω_0 and Ω_1: the background and the object of interest. This is usually performed by evolving an initial (arbitrary) curve following certain velocity field β towards the desired curve. The goal is to design this velocity in such a way that the final curve matches the object of interest. This can be accomplished by to ways: by designing directly β in the evolution equation or by deriving it indirectly from a variational approach.



(a) Original image (b) Stopping function g (c) Final front

Figure 2.5: Segmentation using active contours .

To illustrate the first approach, suppose that we are trying to segment the image depicted in figure 2.5. As we are interested on locating the borders of the object, we consider a function $g(\mathbf{x})$ such that $g(\mathbf{x}) = 0$ on the borders and 1 otherwise. Details of how such a function can be constructed are given in section 2.3.1.

We initially place a front outside the objects of interest, and then we evolve it according to eq. (2.4) taking $\beta = -g(\mathbf{x})$. Initially, the contour is far from the borders so $g = 1$ and the front will deflate and move inwards (as show in section 2.2.1), and when a part of the front reaches the border ($g = 0$) the velocity is zero and thus the curve stops its movement.

Variational methods consist in finding a curve Γ such that minimizes (maximizes) some cost functional. The main goal, then, is to design this cost (or energy) in such a way that the minimum of this energy corresponds with the desired partition of the image. Once the energy functional is designed, its gradient descent is computed, yielding a velocity β which drives the evolution of the front.

Most of the active contour based algorithms present in the literature could be classified in two categories: Boundary or Region based. The former uses boundary information, like maximum contrast of the borders of the object (computed as boundary integrals) and the latter uses region information, like the mean of the gray-level on each region (computed as region integrals).

2.3.1 Geodesic Active Contours

Since the original work of Kass et al. [9] extensive research was performed on active contour models for boundary detection. The classical approach was to deform an initial curve Γ_0 towards the borders of the objects of interest. This deformation was obtained

as a (local) minimum of an energy functional defined in such a way that this minimum is reached at the object boundaries

$$E(\Gamma) = \gamma \int |\Gamma'(s)|^2 ds + \mu \int |\Gamma''(s)|^2 ds - \lambda \int |\nabla I(\Gamma(s))|^2 ds \tag{2.24}$$

where μ, γ and λ are positive constants. The first two terms control the smoothness of the contours while the third term is responsible for attracting the contour to the boundaries of the object. As this term encourages high gradient values, boundaries in this model are characterized by big changes of intensity.

This formulation has two major drawbacks:

- It can not directly deal with topology changes during the evolution. The topology of the final curve remains the same of the initial one, or heuristics must be added in order to allow this changes.
- It is not intrinsically geometric: the energy functional, and thus the evolution, depends on the parametrization of the curve.

Caselles et al. [51] and Malladi et al. [52] improved on this model proposing an alternative geometric flow. In this model, they propose an evolution law (2.25) composed by two terms: a regularity term and a data term. The latter is responsible for driving the contour towards the boundary of the object by means of the g function.

$$\frac{\partial \phi}{\partial t} = g(|\nabla I|)(1 + \epsilon\kappa)|\nabla \phi| \tag{2.25}$$

This model solves the abovementioned problems by means of using the level-set paradigm presented in section 2.2.3. Here ϕ is the embedding function of Γ.

In this model g is a monotonically decreasing function such as $g(0) = 1$ and $g(r \to \infty) \to 0$. Driven by (2.25) the contour Γ evolves outwards with constant velocity (first term), plus a regularization force (second term) that smooths the curve. When the front reaches a high gradient zone, g goes to zero and thus the front stops. There are many possible choices for the stopping function which are discussed in the following sections.

However, this was a curve evolution approach and not an energy based one. In a posterior work Casselles et al. [24] improved this model by the introduction of the Geodesic Active Contour formulation. The idea here, is to pose the evolution (2.25) as an energy minimization problem. They proposed the following energy

$$E(\Gamma) = \int |\Gamma'(s)|^2 ds + \gamma \int g(|\nabla I(\Gamma(s)|) ds \tag{2.26}$$

and they proved that this energy is equivalent to minimizing

$$E(\Gamma) = \int g(|\nabla I(\Gamma(s))|)|\Gamma'(s)| ds \tag{2.27}$$

This formulation holds the best properties of previous models, plus some advantages:

- Includes regularization and border attraction terms like in the previous model
- With the formulation (2.27), the problem of boundary detection is stated as the problem of finding a curve of minimal (weighted) length. This is a geodesic in a

Riemannian space with a metric derived from the image. In this way, the solution is intrinsic to the geometry, and does not depend on a particular parametrization of the initial front.

In order to minimize (2.27) we compute its gradient descent

$$\frac{\partial \Gamma}{\partial t} = \left(\hat{g}(\mathbf{x})\kappa + \nabla g(\mathbf{x}).\vec{N}\right)\vec{N} \tag{2.28}$$

where we have defined $\hat{g}$ such that $\hat{g}(\mathbf{x}) = g(|\nabla I(\Gamma(\mathbf{x}))|)$.
And using the level-set formulation the gradient descent yields,

$$\frac{\partial \phi}{\partial t} = \hat{g}(\mathbf{x})\kappa|\nabla\phi| + \nabla\hat{g}(\mathbf{x})\nabla\phi \tag{2.29}$$

This approach has two main drawbacks:

- The initial curve must be close to the object of interest: if we are far from the edges g and ∇g vanish and the velocity is approximately zero.
- Objects of interest are characterized by strong contrast variations: in real situations the boundaries can have weak changes of contrast, or have no change at all, as occurs in textured images.

2.3.1.1 Choice of the edge detection function

There are many possibles choices for the stopping function $g(x)$:

- Lorentz:
$$g(x) = \frac{1}{1 + \frac{x^2}{2\sigma^2}}$$
- Leclerc:
$$g(x) = e^{-\left(\frac{x}{\sigma}\right)^2}$$
- Tukey:
$$g(x) = \begin{cases} \left(1 - \left(\frac{x}{\sigma}\right)^2\right)^2 \; if \; |x| \leq \sigma \\ 0 \; if \; |x| > \sigma \end{cases} \tag{2.30}$$

In all cases, σ is a noise parameter that determines the minimum value of the gradient that is considered a significative edge. Values smaller than σ are considered noise. This parameter can be imposed a priori, or automatically estimated from the input image.

2.3.2 Region Based Active Contours

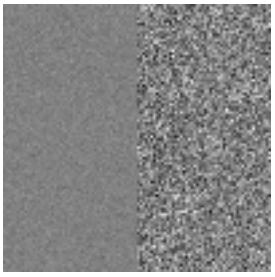


Figure 2.6: Sample image where region information seems more useful than boundary information

The basic idea of region based active contours (RBAC) is to introduce global descriptors of the different regions to drive the evolution of the front. Examples of these descriptors are the mean, or the variance of the gray-level in each region.

This is specially useful when dealing with textured images, in which objects of interest are determined by the homogeneity of a certain property (e.g. a periodically repeated pattern), rather than by strong edges. As an example of such an image we consider figure 2.6 where an image with two regions is presented. Each region is composed with pixels drawn from Gaussian distributions with same mean an different variance. In this case, an edge driven algorithm seems completely useless.

There are many ways to derive the energy and the evolution equations in RBAC, but at the end all formulations end up with a term of the form

$$E_{RBAC}(\Omega_0, \ldots, \Omega_M) = \sum_{i=1}^{M} \int_{\Omega_i} k_i(x, y, \Omega_i) d\mathbf{x} \tag{2.31}$$

where Ω_i are the M regions present in the image, and $k_i()$ are the global region descriptors attached to each of these regions.

In this section we will derive this energy following [13] and in some parts we will include some excerpts from this work. In section 2.4 we will discuss other possible derivations and their advantages. We refer to [34] for a review of different RBAC approaches and a more extensive and formal approach.

We start by defining the terms used in this section (see figure 2.7):

- Ω_A, Ω_B: we model the input image as composed by two regions with different statistical properties.
- Ω_0, Ω_1: partition of the image given by Γ.
- M: the number of regions defined by the algorithm.
- Z: the number of regions present in the image.

We try to find a partition of the domain consistent with the observed data, the hypothesis assumed and their expected properties. Ideally this must lead to $\Omega_0 \approx \Omega_B$ and

Figure 2.7: Region based Active Contours: curve Γ divides the domain in two regions (Ω_0,Ω_1), which will try to match the corresponding regions (Ω_A,Ω_B) in the image.

$\Omega_1 \approx \Omega_A$. This problem can be stated as an optimization problem with respect to the *a posteriori* partition probability, given the observed data.

Let us define that all possible partitions Γ of the image I obey the *a posteriori* partition probability density $p(\Gamma|I)$. Using the Bayes rule, this density can be written as

$$p(\Gamma|I) = \frac{p(I|\Gamma)}{p(I)} p(\Gamma) \tag{2.32}$$

where

- $p(I|\Gamma)$ is the class conditional probability density function for the image I, given the partition Γ.
- $P(\Gamma)$ is the *prior* probability of the partition Γ among the space of all posible partitions.
- $P(I)$ is the probability of having the input image I among all possible images.

Under the hypothesis that both regions are *a-priori* equiprobable ($p(\Gamma) = \frac{1}{M}$), $P(\Gamma)$ is constant (0.5). In addition, $P(I)$ is merely a scale factor, thus the posterior density can be rewritten as

$$p(\Gamma|I) = C.p(I|\Gamma) = C.p(I|\Omega_0, \Omega_1) \tag{2.33}$$

where C is some constant.

In addition, since there is no correlation between region labeling, and the region probabilities depend only on their observation set (pixels within the region), we obtain the following expresion

$$p(\Gamma|I) = p(I|\Omega_0) \cap p(I|\Omega_1) = p(I|\Omega_0)p(I|\Omega_1) \tag{2.34}$$

where $p(I|\Omega_i)$ is the conditional probability density function for the image intensities given that the pixel belongs to region Ω_i. Assuming that the pixels within each region are independent, we can factorize the region probability by the joint probability of its pixels:

$$p(I|\Omega_i) = \prod_{\mathbf{x}\in\Omega_i} p(I(\mathbf{x}))$$

Taking all this into account, the *a posteriori* probabability for a partition Γ is determined by

$$p(\Gamma|I) = \prod_{\mathbf{x}\in\Omega_0} p_0(I(\mathbf{x})) \prod_{\mathbf{x}\in\Omega_1} p_1(I(\mathbf{x}))$$

In order to obtain the correct classification of the pixels, we need to maximize this *a posteriori* probabability. This is equivalent to minimize $-\log\{p(\Gamma|I)\}$. Thus we have to minimize the following functional

$$E_{data}(\Gamma) = -\int_{\Omega_0} \log[p_0(\mathbf{x})]d\mathbf{x} - \int_{\Omega_1} \log[p_1(\mathbf{x})]d\mathbf{x} \tag{2.35}$$

where we have defined $p_i(\mathbf{x}) = p(I(\mathbf{x})|\Omega_i)$, the probability density of the gray level $I(\mathbf{x})$ given that the pixel $\mathbf{x}$ belongs to region Ω_i.

In addition to the data term, we add a regularization term that penalizes the curve length. Thus the resulting energy yields

$$E(\Gamma) = -\alpha \sum_{i=1}^{2} \int_{\Omega_i} \log p_i(\mathbf{x})d\mathbf{x} + \gamma \int_0^1 |\dot{\Gamma(s)}|ds \tag{2.36}$$

where α and γ are positive constants that regulate the balance between regularization and data fidelity.

In order to use the level set formulation introduced in section 2.2.3, we will define an embedding function ϕ whose zero level-set is Γ as

$$\begin{cases} \phi(\mathbf{x},t) < 0 \; in \; \Omega_1 \\ \phi(\mathbf{x},t) = 0 \; in \; \Gamma \\ \phi(\mathbf{x},t) > 0 \; in \; \Omega_0 \end{cases} \tag{2.37}$$

Thus, the energy (2.36) can be re-written as

$$E(\phi) = \int_\Omega \left(\gamma\delta(\phi)|\nabla\phi| - \alpha \sum_{i=1}^{2} \chi_i \log p_i(\mathbf{x}) \right) d\mathbf{x} \tag{2.38}$$

where

$$\chi_i = \begin{cases} 1 \; if \; \mathbf{x} \in \Omega_i \\ 0 \; if \; \mathbf{x} \notin \Omega_i \end{cases} \tag{2.39}$$

and δ is the Dirac function. χ_i can be computed from the embedding function as

$$\begin{cases} \chi_0 = H(\phi) \\ \chi_1 = 1 - H(\phi) \end{cases} \tag{2.40}$$

where H is the Heaviside function. In addition χ_i must satisfy $\sum_0^{M-1} \chi_i = 1$.
For the numerical implementation we use the continuous approximation for δ and H

presented in [53].

2.3.2.1 Extension to vector valued images

In order to integrate multiple cues in this framework, we consider a vector-valued image where a vector $U(\mathbf{x})$ is asociated to each pixel $\mathbf{x}$. Now we have a feature image $U : \Omega \rightarrow R^N$, with $p_i(\mathbf{x})$ the multivariate distribution associated to pixel $\mathbf{x}$ and N the number of feature channels (like color, texture, optical flow, etc). Under the assumption of statistical independence between the feature channels, we can factorize the joint PDF as

$$p_i(\mathbf{x}) = \prod_{j=1}^{N} p_i^j(\mathbf{x}) \tag{2.41}$$

where p_i^j are the PDF for each feature channel.

As we are dealing with multiple features, we are interested in assigning different weights w_j to each of them. Assuming statistical independence and using the factorization (2.41), we can re-write the energy (2.38) as

$$E(\phi) = -\int_{\Omega} \left(\gamma.\delta(\phi)|\nabla\phi| + \alpha.\sum_{i=1}^{2} \chi_i \sum_{j=1}^{N} w_j \log p_i^j(\mathbf{x}) \right) d\mathbf{x} \tag{2.42}$$

This energy depends on the curve ϕ, and the parameters θ_i of the PDF of each region.

Thus, to find a minimum of this energy we must differentiate with respect both to the curve and the parameters and equate to zero. In the case of the optimum curve, is hard to solve $\frac{\partial E(\phi)}{\partial \phi} = 0$, instead one computes its gradient descent, in order to iteratively reach the minimum. In this case the gradient descent yields

$$\frac{d\phi}{dt} = \left(\gamma\kappa + \alpha\delta(\phi) \sum_{j=1}^{N} w_j \log \frac{p_1^j(\mathbf{x})}{p_0^j(\mathbf{x})} \right) |\nabla\phi| \tag{2.43}$$

As an example, we will derive the expression for the optimum parameters assuming a gaussian model for each region. Thus the parameters to estimate are $\theta_i = \{\mu_i, \sigma_i\}$. Differentiating (2.42) and equating to zero yields

$$\begin{cases} \mu_i = \frac{\int_\Omega \chi_i \vec{U}(\mathbf{x}) d\mathbf{x}}{\int_\Omega \chi_i d\mathbf{x}} \\ \sigma_i^2 = \frac{\int_\Omega \chi_i (\vec{U}(\mathbf{x}) - \mu_i)^2 d\mathbf{x}}{\int_\Omega \chi_i d\mathbf{x}} \end{cases} \tag{2.44}$$

2.3.2.2 Extension to M regions

There are many ways to extend the region based active contour formulation to the case of more than two regions. For the sake of simplicity, we will present the subject for the case of four regions, and then we will extend the results to the general case. The simplest approach is to attach one curve to each of the objects, thus we need M curves. In this approach usually appear some regions which are not covered by any curve (vacuum) and other regions covered by many curves (overlap). To avoid this, one usually introduces a decoupling force that forbids these behaviours.
In order to reduce the number of curves, and to avoid the introduction of artificial coupling,

some authors propose a multi-phase approach, where only $\log M$ curves are needed. Then, to represent four regions using two curves, we re-define the indicator functions χ as

$$\begin{cases} \chi_0 = H(\phi_0)H(\phi_1) \\ \chi_1 = (1 - H(\phi_0))H(\phi_1) \\ \chi_2 = (1 - H(\phi_0))(1 - H(\phi_1)) \\ \chi_3 = H(\phi_0)(1 - H(\phi_1)) \end{cases} \tag{2.45}$$

where ϕ_i with $i \in 0, 1$ are the embbeding functions for each curve Γ_i.
With this notation, the energy is redefined as

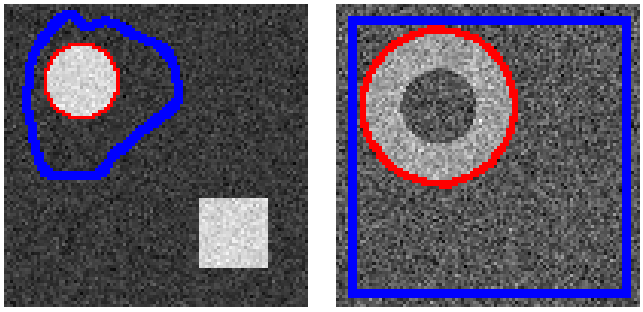
$$E_{RBAC}(\phi_0, \phi_1) = \int_\Omega \sum_{i=1}^{4} \chi_i e_i(\mathbf{x}) d\mathbf{x} \tag{2.46}$$

where we have defined $e_i(\mathbf{x}) = -\log p_1(\mathbf{x})$.
After that, the extension to M regions is straightforward

$$E_{RBAC}(\phi_0, \phi_1) = \int_\Omega \sum_{i=1}^{M} \chi_i e_i(\mathbf{x}) d\mathbf{x} \tag{2.47}$$

and the gradient descent yields the system of equations.
For more details on this extension we recommend the reading of [53, 27].

2.3.2.3 Concluding remarks



(a) inicial (b) inicial

Figure 2.8: Initialization problems

This approach has some drawbacks:

- This algorithm relies on a gradient descent to find minima of the energy 2.42, so it suffers the problem of local minima. This reflects in many ways, being the most important the strong dependence on initialization. Altough the algorithm uses global descriptors, the evolution is performed locally. That is, a pixel is moved according to its value, or its neibourghood values, but does not take into account far away regions. This reflects in two ways: problems with the curve, with correct parameters, and problems with the parameter estimation. To illustrate the first issue, we show two

examples. In these examples we supposse that the parameters of the distributions are perfectly estimated and fixed for all the evolution.

The first case is shown in figure 2.8(a), suppose that in the inner region the mean value corresponds to the white value, and in the outer region to the dark gray value. In this case, the curve will evolve towards the circle, and stop in its border. But it is unable to split and to enclose the square, even with the level-set formulation.

In the second case, shown in figure 2.8(b), when a point of the front arrives to the border of the ring, it will stop. But the front it is unable to split and *jump* across the light gray zone to reach the inner circle of the ring.

- As the algorithm codifies the information in a global manner (by the global descriptors), it does not allow to deal with images with smooth variations within each region (see figure 2.9). This can be avoided by considering some kind of neighborhood of each point in the curve and performing region competition in this neighborhood.
- The number of regions is fixed, this can lead to unexpected results when facing an image with a different number of regions. This specially reflects on bad PDF estimations in each region. A possible solution for this problem, is to estimate a Mixture Model before segmenting, in order to discover the number of regions, and then using the algorithm with this number.
- The choice of the weights w_j is fixed: to set the relative importance *prior* knowledge must be used. In addition, the relevant features may vary from sequence to sequence, and also between frames of the sequence, so this approach seems inadecuate to reflect this fact.

Figure 2.9: Sample image where global descriptors fail

2.3.3 Segmentation

In order to segment an image using the RBAC model, we need to make some choices. In the first place, we need a model for the PDF in each region, which can be gaussian, exponential, or a non-parametric estimate, among others. Then we need to choose the relative importance of each term in the energy, i.e. the amount of regularization, the

1. Prior choices:
   - $\mathcal{M}, \theta_i$: PDF model and its parameters.
   - N: number of features.
   - M: number of regions.
2. Parameters:
   - α, γ, λ: weights of the different energy terms (region, boundary, smoothness, etc).
   - w: feature weighting vector.
   - $d\mathbf{x}$: size of the spatial discretization.
   - dt: time step for the evolution.
3. Initialization:
   (a) compute feature vector $U(x)$
   (b) perform anisotropic diffusion over the feature vector
   (c) Start with an arbitrary curve Γ_0 (usually provided by the user)
4. Curve evolution:
   (a) evolve the curve according to (2.43)
   (b) for each region i compute θ_i (μ and σ in the gaussian case)
5. Stop condition:
   goto 4 until one of the stop conditions is reached:
   (a) the curve reaches steady state ($\frac{\partial \Gamma}{\partial t} \approx 0$)
   (b) the energy reaches a minimum
   (c) the iteration count reaches a maximum

Figure 2.10: Summary of a region based segmentation algorithm.

weights for the region and borders terms, etc. As we are dealing with vector valued images, we need to choose the weights w_j for each feature channel, according to the prior information given.

Finally we need to specify an initial curve, that will evolve towards the minimum. Once this intial condition is set, the algorithm works as follows. For each step, given the curve, the PDF parameters are estimated according to (2.44). Then the curve is updated following (2.43), and this continues until the evolution reaches steady state.

2.3.4 Extension to tracking

In the case of tracking we have a sequence composed of K frames. In the first frame, an user provided segmentation is used. As this segmentation is the ground truth, we can estimate the PDF parameters (θ_i) for each region. We assume a *slow* variation of θ_i and the position and shape of the curve. Thus the curve in frame k is a good initial guess for the curve in frame $k+1$. In addition, as we supposed that the object does not change its properties (color, texture, etc), we can use the previously estimated parameters (θ_i)

to drive the evolution.

In each frame we use as initialization the final front from previous frame and we assume that the p_i are fixed and evolve the curve towards a minimum of the energy. Once this minimum is reached, we estimate the p_i to be used in the next frame. This estimation is detailed in section 3.4.2.

In the particular case of video sequences, it is useful to add motion information to the segmentation with the goal of grouping parts of the image with coherent movement. For this purpose we adopted an approach similar to [15]. In this work, motion information is introduced by means of the optical flow. As we will see in section 3.5 this magnitude represents the velocity (between two frames) of each pixel of the image [2].

This will be useful to us in two different ways. The first has to deal with the initial condition used. That is, if this velocity is known, then, given a point x_N in frame N we can estimate where this point will be in frame $N+1$. Thus, if we have the curve Γ_N which segments the object in frame N, and we know the velocity, then we can estimate the curve for the next frame Γ_{N+1}. This estimation will be far from perfect, but would serve as a better initial guess than Γ_N. On the other hand, as we are interested on the tracking of moving objects, we will use the computed optical flow as two more features (v_x and v_y). Thus to segment the frame $N+1$ we will use the color channels, the texture channels, and the motion channels.

2.3.5 Stability of the algorithm

In this section we will show the necessary conditions for the stability of the algorithm, following [38]. In order to do so, we recall from previous sections the evolution equation 2.43

$$\frac{d\phi}{dt} = \left(\alpha\kappa + \delta(\phi) \sum_{j=1}^{N} w_j \log \frac{p_1^j(\mathbf{x})}{p_0^j(\mathbf{x})} \right) |\nabla\phi| \tag{2.48}$$

This evolution is composed by two terms of the form:

$$\gamma.g(x).\kappa.|\nabla\phi| \tag{2.49}$$
$$\alpha.f(x).|\nabla\phi| \tag{2.50}$$

where $g(x) = 1$ and $f(x) = \log \frac{p_1(x)}{p_0(x)}$. The first term (2.49) is the curvature evolution, which can be spatially discretized using standard central differences, as shown in [43]. If we apply the Von Neumann method to this term, the stability bound for this term yields

$$\Delta t \leq \frac{\Delta x}{4(D-1).\alpha.||g(x)||_\infty} \tag{2.51}$$

where D is the dimension of the domain.

In the case of the convection term (2.50) the numerical scheme used is the one presented in [43]. In this case, the stability bound is

[2] we shall call v_N the velocity computed from frames N and $N+1$, that is $v_N = (x_{N+1} - x_N)/\Delta t$

1. Prior choices:
   - $\mathcal{M}, \theta_i$: PDF model and its parameters
   - N: number of features
   - K: number of frames
   - M: number of regions.
2. Parameters:
   - α, γ, λ: weights of the different energy terms.
   - w: feature weighting vector.
   - $d\mathbf{x}$: size of the spatial discretization.
   - dt: time step for the evolution.
3. Initialization:
   (a) for each frame k of the sequence ($k \in \{1..K\}$):
      i. compute feature vector $U_k(x)$
      ii. perform anisotropic diffusion over the feature vector
   (b) Start with an arbitrary curve Γ_0 (manually drawn or a result of an unsupervised segmentation)
4. Curve evolution:
   for each frame $k \in \{2..K\}$:
   (a) for each region i compute θ_i^{k-1} from previous frame.
   (b) use the curve from previous frame Γ_{k-1} as initial condition
   (c) evolve the curve according to (2.43) using θ_i^{k-1} fixed during the frame.
   (d) Stop condition:
   goto 4c until one of the stop conditions is reached:
      i. the curve reaches steady state ($\frac{\partial \Gamma}{\partial t} \approx 0$)
      ii. the energy reaches a minimum
      iii. the iteration count reaches a maximum
   (e) go to next frame.

Figure 2.11: Summary of a region based tracking algorithm.

$$\Delta t \leq \frac{\Delta x}{\sqrt{D}.||f(x)||_\infty} \tag{2.52}$$

For the time, we use an explicit first order time discretization, with adaptive time step. This step is computed from the previously computed stability bounds as

$$\Delta t = \frac{1}{2} \min \left\{ \frac{\Delta x^2}{4(D-1).\alpha.||g(x)||_\infty}, \frac{\Delta x}{\sqrt{D}.||f(x)||_\infty} \right\} \tag{2.53}$$

2.4 Related work

We will now extend the review presented in section 1.3.1, but focusing on the approaches closely related to our work. In this analysis (which is based on [34]) we will consider several important points:

- It is an energy based approach?
- How is the evolution derived from the criteron?
- What are the different descriptors used?
- Which features are used for segmentation? How are they integrated into the algorithm?

As we pointed out before, there are many ways to construct an energy for the RBAC model, but in the end all formulations are a particular case of the general energy:

$$E_{RBAC}(\Omega_0, \ldots, \Omega_M) = \sum_{i=1}^{M} \int_{\Omega_i} k_i(x, y, \Omega_i) d\mathbf{x} + \nu E_{coupling} \tag{2.54}$$

In the following, we discuss each of the related works, and how they compute their region descriptors:

- Deriche et al. [54, 55, 13, 15]: They use as descriptors the probability p_i of a value $U(\mathbf{x})$ given that the pixel $\mathbf{x}$ belongs to region Ω_i:

$$\begin{cases} k_i(\mathbf{x}, \Omega_i) = p(U(\mathbf{x})|\Omega_i) \\ k_b = \mu + \alpha g(|\nabla U(\mathbf{x})|) \end{cases} \tag{2.55}$$

  Here the multi-valued case is handled by the distributions $p(U(\mathbf{x})|\Omega_i)$ that turn to be multivalued.

  With respect to the features used: they use the RGB channels for color description, and the structure tensor for texture description. Of the presented approaches this is the only one that introduces optic flow in the feature set in order to track moving objects.

  Drawbacks: The result of the algorithm is strongly dependent on the amount of diffusion applied to the feature image U. The structure tensor is a very simple texture (orientation) descriptor, that does not take into account the scale. In addition, in images with little or no texture (cartoon), borders appear due to the derivatives that hinder the segmentation. Finally, the algorithm is unable to distinguish multi-scale images. To solve this and also to speed up the algorithm, they use a multiresolution approach for the implementation.

- Chan et al. [31, 12, 27]: This is a region based variational approach, which uses multiple channels and where the energy is constructed using only the mean c_i value of each channel as descriptor of each region.

$$\begin{cases} k_i(\mathbf{x}, \Omega_i) = \lambda_i ||U(\mathbf{x}) - c_i||^2 \\ k_b = \mu \end{cases} \tag{2.56}$$

The multi-channel framework is simply achieved by changing the absolute value of the difference $|I(x) - c_i|^2$ with the norm of the difference of feature image and the multi-valued mean $||U(\mathbf{x}) - c_i||^2$.

In their last works, they show an interesting link between its model and the Mumford-Shah functional, in the particular case of piecewise constant images. They also present a multiphase extension to their flow, up to four regions.

For the case of textured images they use a set of Gabor filters as texture descriptors in their model. As they do not know in advance which texture channels are relevant, they use an automatic selection algorithm to choose which feature channel is relevant. This is performed for channel j by computing $s^j = |c_0^j - c_1^j|$ and selecting the n channels with higher s.

- Aubert et al. [14, 30, 34, 56]:

  One of the major contribution of their work is [34], where they propose a general framework for region-based active contours. One common fact about the rest of the approaches is that they ignore, in the minimization of the functional, a term due to the variation of the descriptors over the evolution. They address this issue posing the energy in a dynamical scheme such that $E = E(\Omega_0, \ldots, \Omega_M, \tau)$. This means that the time τ is included in the energy. This functional is then minimized using shape optimization tools.

  From the point of view of the features, in [30] they introduce an evolution using wavelet coefficients as texture descriptors. In this work, their region descriptors are the same as in Deriche, but without an edge term:

$$\begin{cases} k_i(\mathbf{x}, \Omega_i) = p(U(\mathbf{x})|\Omega_i) \\ k_b = \mu \end{cases} \tag{2.57}$$

  This approach essentially shares the same drawbacks as the approach of Deriche et al. but they address the multiscale issue by the use of the wavelet transform. However, they use fixed weights for the channels, and they do not show how to select the number and the wavelet subbands to use.

- Freedman et al. [33]: This approach is sligthly different from the others because its main goal is to *chase* an objective distribution. This is achieved by minimizing the Kullback-Leibler divergence between the region distribution and the objective distribution. In their model it is straightforward to integrate multiple features. One of its major drawbacks is that their approach is supervised (the objective distribution is given and fixed for all frames) and uses only two regions, and they do not show how to extend to the unsupervised case, nor to a multiphase flow. In addition the velocity of the evolution vanishes when far from the objects of interest, so this algorithm requires a good intial guess.

Chapter 3

Feature Extraction

3.1 Introduction

In this chapter we will explain the computation of the features used by the algorithm. It is clear that an algorithm in order to be robust enough to deal with a broad range of images, needs to use as much information as possible. As we explained in previous chapters the model used for segmentation allows this integration of multiple features. In this work we will consider how to deal with color, texture and motion. However it is possible to integrate whichever feature desired, as long as it could be expressed as a PDF defined over a region.

We will start by presenting the preprocessing performed on the sequences in order to obtain suitable images for segmentation. Then we will show how to model color and texture. Finally, we will show the optical flow computation, useful when dealing with video sequences.

3.2 Preprocessing

The first step in our algorithm starts after data is acquired, converted to a sequence of slices and registered. As we have shown in section 1.1 there are many acquisition techniques, and the biological tissues are very different. However, in all cases we end up with the same data structure: series of consecutive digital images. The main problems that appear in these sequences are:

1. noise: due to the acquisition techniques, biological preparations, microscopy noise, slicing process, etc.

2. non-uniform illumination along the sequence: different frames have different brightness levels.

3. non-uniform illumination within a frame: different parts of the image get different amounts of light.

4. non-uniform properties of the images: in some kind of sequences, like *spider*, the biologist need to stain the structures of interest. In some cases this staining varies between different parts of the image.

5. non-uniform sampling in the different axes: as explained in the introduction, resolution between the x-y and z axes may present great differences. For this reason, in this work we will treat the data as a sequence of images rather than a 3D volume. This applies to the segmentation process, but also to the preprocessing stages.

In order to deal with these problems, the sequences must be preprocessed. The third problem does not appear to be significant in our sequences so it has not been considered. In the case of staining (fourth item), this is controlled by the process made by the biologist and therefore it is out of our control (i.e if a non-interesting structure is stained in the same way as the interesting ones). However, in the case of chromaticity variations between objects of interest, it can be addressed using color correction techniques. This kind of treatment is beyond the scope of this work, but it could be an interesting extension.

For the reasons presented above, we deal only with the two first problems. To reduce noise in the images anisotropic diffusion filtering is performed independently in each frame. Finally, to correct the non-uniform illumination, we apply an histogram matching algorithm to the sequence. We will explain these algorithms in the following sections of this chapter.

3.2.1 Histogram Matching

This algorithm matches the histogram of two images (see ItkHistogramMatching-Filter in [57]). In order to explain how to use this algorithm we recall from section 2.3.4 how our algorithm works. As we are interested in the use of the statistical descriptors estimated in frame N to segment frame $N+1$, we need to be sure that these properties present a slow variation between two consecutive frames. For this reason, we run the matching algorithm on pairs of consecutive images.

Thus the algorithm works as follows: evolve the curve for frame N, estimate the PDF parameters θ^n, run the matching algorithm with frames I_n and I_{n+1} modifying the histogram of frame I_{n+1} yielding an image $\hat{I}_{n+1}$ that matches the histogram of I_n. Then, evolve the algorithm over $\hat{I}_{n+1}$ with parameters θ^n. When this evolution finishes estimating the new parameters θ^{n+1} with the *original* frame I_{n+1}.

3.2.2 Anisotropic diffusion

As we mentioned before, we have noisy sequences and an algorithm that divides the image in regions with homogeneous properties, so we need an algorithm that achieves these results. Explicitly, this algorithm must reduce noise in the images while respecting the borders between different regions. One common way to do so, is to perform anisotropic diffusion. This algorithm reduces the noise, flattens regions (yields cartoon-like images) and respects the borders between objects.

The basic idea behind anisotropic diffusion, first proposed by Perona and Malik [41], is to perform a selective smoothing that reduces noise while preserving strong edges. It was proposed as an alternative to the traditional gaussian (isotropic) smoothing, which blurs edges as well.

As edges are usually associated with high gradient zones, the diffusion is performed in the direction ξ perpendicular to the gradient ∇I with the following equation

$$I_t = I_{\xi\xi} \tag{3.1}$$

where $\xi = \nabla I^T$. Taking this into account $I_{\xi\xi}$ is $|\nabla I|\nabla.\left(\frac{\nabla I}{|\nabla I|}\right)$. Then, recalling the definition of mean curvature from section 2.15 the diffusion equation yields

$$I_t = \kappa|\nabla I| \tag{3.2}$$

It is important to note that the evolution 3.2 is such that the level lines of I move according to the Euclidean Shortening Flow presented in section 2.2.1. From this point of view, it is also possible to perform affine invariant smoothing using the Affine Shortening Flow presented in the same section.

In the case of this work, we are performing the segmentation over a vector valued feature image $U(\mathbf{x}) : R^2 \rightarrow R^N$, so we are interested in the reduction of noise of this vector image. We will briefly review this topic following [58]. We recommend the reading of [58, 59] for further details.

As pointed out in that works there are many applications in which vector-valued regularization could be useful:

- Color images: in this case we have a three dimensional vector field containing the RGB components of the image. Here it is interesting to regularize in a way that

avoids color blending, so usually one adds some constraints on the regularization process.

- Optic Flow: in this case the vector field is composed by the velocity vector computed from two images. It will be of two or three components depending on the dimension. As one deals with direction fields, it is necessary to constrain the diffusion in such a way that the regularized vectors remain unit vectors.
- Fields of diffusion tensors: DT-MRI images are examples of matrix-valued images encountered in medical imaging. These can be constrained or unconstrained depending on the application.
- Vector Probability Diffusion: this is another example of vector valued regularization, which is constrained to ensure that the field vectors remain in a PDF space.

The simplest extension of anisotropic smoothing to vector images is to independently smooth each channel. As one quickly realizes (see [59]) this type of smoothing leads to incoherences between the different channels as each one evolves with different smoothing geometries. In addition, this can lead to vector components blending. This is very clear in the case of color images, where non-obvious chromaticity distortions can appear (e.g. colors originally not present that appear after the diffusion process).

For this reason it is necessary to define a coherent notion of vector geometry. This definition has to include how to define a vector gradient norm, and two corresponding variation directions (of max and min variation). This magnitude must also reduce to the corresponding magnitudes in the scalar case. As an example of this magnitudes we will consider the approach proposed by Di Zenzo [60], where the image is considered as the mapping $I : R^2 \rightarrow R^N$ vector field, and a *structure tensor* G is considered as

$$G = \sum_{i=1}^{N} \nabla I_i \nabla I_i^T \tag{3.3}$$

under this formulation the variations are computed from the eigenvectors $\theta_{\pm}$ and the eigenvalues $\lambda_{\pm}$ of G. These represent the directions and the magnitudes of maximum and minimum variation respectively.
One of the possible ways to diffuse using these directions is to use a smoothed version $G_\sigma = g_\sigma * G$ of the *structure tensor*[3] in a divergence form (see [59]) yielding

$$\frac{\partial I_i}{\partial t} = \nabla . \left(G_\sigma^{-1} |\nabla I_i| \right) \tag{3.4}$$

where I_i are components of the vector image .

Summarizing, we apply the anisotropic diffusion given by (3.4) before the histogram matching algorithm.

3.3 Color descriptors

An important topic when processing color images is how we represent them. Usually this images are given by the (R, G, B) tri-stimulus values corresponding to the Red, Green,

[3] g_σ is a gaussian kernel with variance σ^2

Blue frequency bands of the visible spectrum. Color is then represented as a point in a 3D space with the value in each axis representing the amount of each color present in the image.

However, is usual to transform the original (R, G, B) values to another color space. There is a large variety of color transforms (linear and non-linear) and color spaces in the literature [61]. There is no agreement about which color space is the *best*, and also the performance of an image processing algorithm will heavily depend on this choice.

Although they can be perceptually more accurate, one of the major drawbacks of non-linear transforms is the introduction of instability or singularities that may increase the numerical errors (see [62]). For this reason, in this work we will consider only linear transforms. In addition, as we will see in the following sections, linear transforms will present interesting properties regarding texture information.

Then, the following question arises: which is the *best* transform among the space of all possible linear transforms? And also, we must define in which sense we will search for the best transform.

Vision has poor response to spatial detail in colored areas of the same luminance, compared to its response to luminance spatial detail ([61]). So it makes sense to use a representation that separates color information from intensity. Examples of this kind of representations are the (Y, Cb, Cr) (linear), and (H, S, V) (non-linear) among others. In the first case, color information is obtained by subtracting the intensity channel from the color ones. Explicitly this transform is obtained by the following linear transform

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = M. \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.5}$$

where M is

$$M = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \tag{3.6}$$

The choice of this kind of model is supported by the affirmation made by Caselles et al. [63], which states that the essential geometric information of an image is contained on its level lines. The level lines are the set on which the gray level of the image is constant. So it makes sense to apply a transformation that isolates geometry from color information.

In addition, this model allows us to correct the non-uniform illumination problem, described in the previous section, in a coherent way. Explicitly, as the three (R, G, B) components have brightness information, we need to equalize the three channels, which could result in non-obvious alterations of color information. If we isolate the brightness information, and apply an illumination correction algorithm, color information will not be altered.

With the arguments presented there is no real evidence of which color space (among those which separate color from intensity) is best suited for the segmentation tasks, so we must postpone the decision until we discuss texture modeling. Indeed, as we will see in the following sections, we will choose the space which is best suited for texture description. In addition, we will see that linear models present interesting properties with respect to

texture.

3.4 Texture descriptors

In this section we will discuss some concepts about texture description and modeling. This discussion will be very succinct and is based on [64, 65], in some cases we will include an excerpt from these works, in which a more profound treatment on the subject is presented.

Texture is a difficult term to define, in our visual world there are many different things that we refer to as texture, so a question arises: what do all this thing have in common that make us classify them as texture? Many people have attempted to describe texture (see [64]) and to compute texture descriptors, but until now there is no general consent on this term. However, there are some intuitive properties of texture that are generally assumed to be true:

- regionality: texture is a property of areas, if one looks at a single pixel the notion of texture is meaningless. So texture analysis must consider a spatial neighborhood (which depends upon the texture type) around the pixel.
- scale: Texture can be perceived at different levels of resolution, and a texture present in certain level could not be present at a different level.
- miscellaneous properties: Some authors identify some properties relevant in describing texture. These are uniformity, density, coarseness, roughness, regularity, linearity, directionality and spatial frequency among others.

As we have mentioned before, there is no agreement on what a texture is and over the past years many different approaches to texture analysis have been proposed, however none of these approaches works for all types of textures and for all applications. The texture analysis present in the literature can be roughly classified in the following categories:

- Statistical approaches: This methods try to capture the spatial distribution of gray level values in the image. The simplest description that fall in this category are the first order statistics derived from the image histogram, like mean, variance or entropy. The major drawback of this method are that it assumes statistical independence between pixels. A more accurate model are second order statistics which take into account interactions between pixels. Examples of these kind of statistics are co-occurrence matrices or auto-correlation features.
- Model-based approaches: These methods aim to construct an image model which can be used to describe texture, but also to synthesize it. Examples of these are Random Fields and Fractals.
- Geometrical methods: This category comprises the methods that define texture as being composed of texture elements or primitives. This elements are characterized by their geometrical properties. Once this elements are identified there are two major approaches to analyze the texture. One computes statistical properties of these elements and uses these properties as texture features. The other, attempts to extract the placement rule that describe the location of the texture elements. Examples of these are Voronoi tessellations and structural methods, respectively.

- Signal processing methods: These methods are based on evidence given by psychophysical research which states that the brain performs frequency analysis. Most of these methods work on spatial or frequency domains. As examples of the former, we have Roberts or Laplacian operators. As an example of the latter we consider the Fourier transform, which gives us a frequency and orientation analysis. Finally, we must mention Gabor and Wavelet models which combines frequency analysis with the spatial localization.

Of all these methods we will concentrate on the last category. Within this framework, wavelets have emerged as a particularly useful tool for texture analysis. This is due to the ability to analyze the image at various scales and orientations in an efficient manner and with a sound mathematical background. In the following we will review how to model textures using the wavelet transform.

3.4.1 Wavelet analysis

The essential idea behind the Wavelet Transform (DWT) is best explained in the 1-dimensional case, and is easily extended to the 2-D case (images). The idea is to decompose a signal f into its high and low frequency components through convolutions with two conjugate mirror filters h and g respectively. In the standard wavelet transform, the outputs of these filters are decimated. The low-pass component is usually called approximation $A1$ and the high-pass component is called detail $D1$. This two signals (of half the original size) together form a complete representation of the signal, i.e. it can be reconstructed from them.

The procedure is then to repeat this decomposition, but now using the first level approximation $A1$ as input signal, so $A1$ is filtered again with h and g and decimated yielding a new pair of approximation and detail signals $A2$ and $D2$ respectively. This procedure is repeated until a chosen level J is reached or the obtained signal is a single pixel. The criterion used in the choice of the level J depends on each application. In figure 3.1 we show an example of this decomposition.

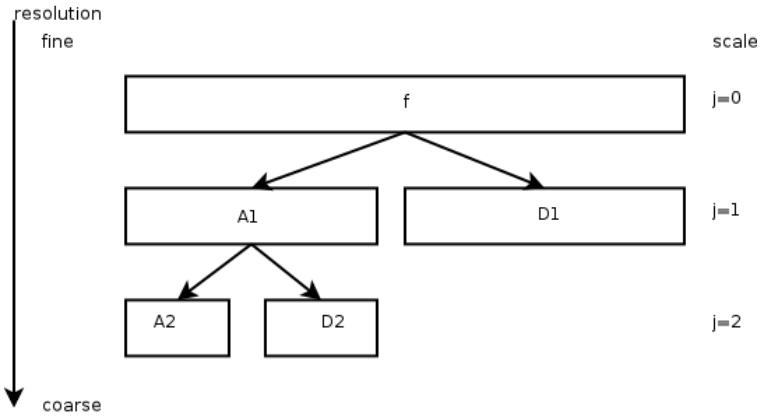


Figure 3.1: Levels of decomposition of the 1-D standard wavelet transform

For the case of a 2-D signal (an image), the analysis is performed usually in a separable manner using 1-D filters first applied to the rows and then to the columns of the image[4]. After each convolution, a down-sampling step is performed. In this way we

[4]the filtering order is interchangeable and does not affect the final result

obtain four sub-images, the approximation $A1$ and three detail images D_1^1,D_1^2 and D_1^3. The details are obtained by alternating low and high pass filtering applied to rows and columns. The order of application is LL(low on rows, low on cols), LH,HL,HH respectively (see figure 3.2). Each of this sub-bands represents the vertical, horizontal, and diagonal details. In this way 2-D wavelet analysis perform an analysis at different scales and also at different orientations.

For a complete presentation on the wavelet transform we recommend the reading of [66].

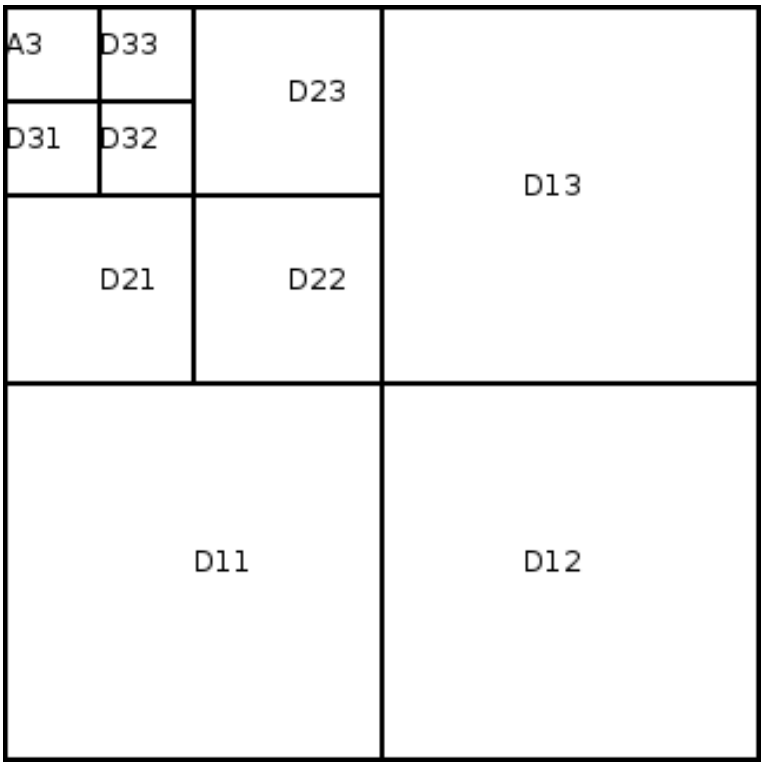


Figure 3.2: Levels of decomposition of the standard 2-D wavelet transform

3.4.2 Wavelet based texture features

As the approximation sub-band A_j is computed by an iterative low-pass filtering it is common to assume that it lacks of any relevant texture information, yielding only intensity information. For this reason, the approximation channel is often ignored and only the detail sub-bands are used for texture modeling.

One of the most frequently used measures for wavelet texture analysis is the mean energy (measured as the L1 or L2 norm), which in the case of L2 norm yields

$$E_j^{(2)} = \frac{1}{N_j} \sum_{i=0}^{N_j} w_{i,j}^2 \tag{3.7}$$

where j denotes the level of decomposition, $w_{i,j}$ is the $i-th$ coefficient , and N_j is the number of coefficients in this sub-band.

As the detail coefficients are derived by high-pass filtering, it is generally true that the mean of these coefficients is zero. Thus, the use of mean energy features can be com-

pared with modeling the histogram of the wavelet coefficients as a gaussian distribution with zero mean.

Instead of using the energies of each sub-band, it is more accurate to model the marginal densities of the coefficients in each sub-band (see [67, 68]). This modeling by the coefficients are usually called Wavelet signatures.

Then we have to assume a model for the density and estimate its parameters. As we shown before, the L2 energy modeling corresponds to the assumption of gaussian distribution, so we can choose this distribution. However, Mallat ([66]) conducted experiments over natural images which verified experimentally that the modulus of the DWT follows a generalized gaussian distribution (GGD) (3.8) on each sub-band. So it is more appropriate to use this model in each sub-band. The PDF in each sub-band is then

$$p(x) = \frac{\beta}{2\alpha\Gamma(\frac{1}{\beta})} \exp\left\{-\left(\frac{|x|}{\alpha}\right)^{\beta}\right\} \tag{3.8}$$

were α and β are the parameters of the distribution and $\Gamma(x) = \int_0^\infty e^{-t}t^{x-1}dt$. The parameters α and β can be computed from samples of the texture, looking for a maximum likelihood estimator and solving a non-linear equation. In a first approximation this can be computed from the first and second moments of the sample. This can be refined by a few Newton-Raphson iterations. For details on this computation we refer to [68].

As a final remark, we must argue that the statistical modeling of the texture not only is more accurate than the use of the energy value, but also gives us a coherent and justifiable way to measure similarity between two textures.

In most applications, to ensure computational tractability, it is common to work under the assumption of statistical independence between the feature channels. Although this will generally be false, we will need to use a decomposition that is approximately independent between sub-bands, so the independent model is pretty accurate.

3.4.3 Other wavelet decompositions

One of the major drawbacks of texture modeling using wavelet signatures is the fact that the DWT is not translation invariant. This is due to the down-sampling performed at the output of the filters. This is particularly problematic in segmentation of images with multiple textures present, as the same texture can appear anywhere in the image. To avoid this, one common choice is to use an undecimated version of the DWT, known as wavelet frames (DWFT). In [67] it is shown that a classification based on the DWFT outperforms the DWT approach.

However this representation is over-complete (highly redundant) and hence a significant amount of information is duplicated. This is specially delicate in applications where computational cost is an issue. In addition, as we will work under the assumption of statistical independence between sub-bands, this redundancy between channels poses a problem.

Another drawback of the DWT is that only the approximation channel is further decomposed. This is only appropriate when the relevant information of the signal is concentrated in the low frequency portion of the spectrum. This is not the general case, so it would be interesting to decompose all frequency components. This lead us to the

Wavelet Packet Transform (DWPT), which performs the decomposition of the approximation channels, but also of the detail sub-bands. Due to this property, the DWPT is best suited for texture analysis (see [65]). An example of this decomposition is shown in figure 3.3.

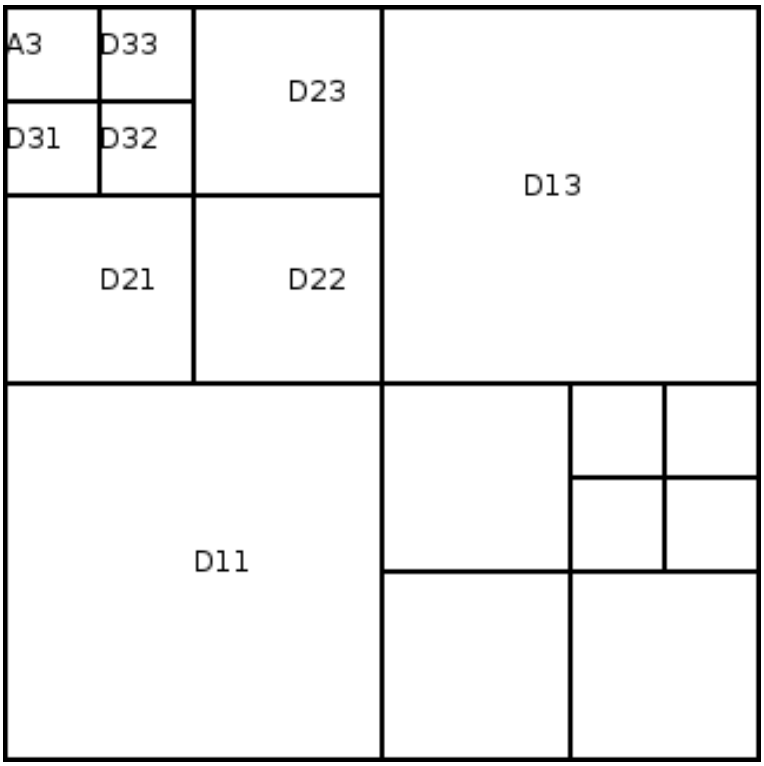


Figure 3.3: Levels of decomposition of the 2-D wavelet packet transform

All wavelet transforms share the problem of the decomposition of sub-bands that have little or no information. For instance imagine an image with only high frequency components, in this case the further decomposition of the approximation channel is completely useless. In the case of the DWPT this problem increases as the number of sub-bands is bigger. For this reason there are some adaptive algorithms that decide, based on certain information or energy measures, which channels are further decomposed.

3.4.4 Color texture

In order to complete the analysis, it remains to be presented how to deal with texture in color images. There are many approaches to work with color texture. The simplest of all them is to independently compute 3 wavelet signatures from the R, G, B tri-stimulus values. This presents the disadvantage that the RGB channels are highly correlated, so the independence assumption is violated. This forces us to model the correlation between channels. There are few works in the literature that model this correlation, see for example [69]. In that work, the authors analyze the behavior of wavelet signatures taking into account the correlation between channels and comparing between different (linear) color spaces. They conclude that the best basis to describe color texture is the one given by

$$M = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0.5 & 0 & -0.5 \\ -0.5 & 1 & -0.5 \end{bmatrix} \tag{3.9}$$

they claim that this matrix approximates the Karhunen-Loève matrix computed for a large set of natural color images. For this reason this color space is called $(K1, K2, K3)$. Under this hypothesis, this basis gives us a space in which the axis are statistically uncorrelated. In addition, they show that a linear transform results in a complete different feature set (as the energy computation is non-linear), hence the quality of the results is heavily dependent on this transformation. This transformation allows us to model this channels as statistically independent, and guarantees that the largest amount of information is concentrated in the first dimension, and the remaining information decreases as the dimension increases (i.e. the last axis gets the smallest amount of information).

For this reason, if our goal is to build a simpler model, we can decompose only the intensity channel and reduce in this way the amount of wavelet signatures. It is important to note that this model shares the same properties of the $YCbCr$ space, i.e. separability between color and brightness [5]. Thus the use of this color space is coherent with the discussion presented in section 3.3.

3.5 Motion estimation

As discussed in section 2.3.4 we need to estimate the optic flow in the sequence. Optic flow computation is a research field on its own, and no attempt will be done here to discuss and compare all the algorithms present in the literature. We will rather mention some remarkable (classic) algorithms present in the literature and give references to further reading. According to [70], the methods present in the literature for optic flow computation can be roughly classified into these categories:

- differential: these methods compute velocity from spatio-temporal derivatives of the image. First and second order derivatives are often used, and also local and global constraints are used. Examples of this technique are [71] and [72].
- region-based matching: under some circumstances, numerical differentiation may be impractical, because a small number of frames is available or because of aliasing in the acquisition process. In these cases region based matching could be more suitable. This technique attempts to estimate the shift between two images by matching points or regions in these images by maximizing some similarity measure. This measure is usually computed over a weighted neighborhood of the pixel.
- energy based methods: These are based on the output energy of velocity tuned filters. They are also called frequency methods, because the design of these filters is usually performed on the Fourier domain. They usually attempt to find peaks of amplitude on the Fourier domain.
- phase-based: Here velocity is computed from the phase behavior of band-pass filters outputs. The use of phase is motivated by the claim that the phase component of band-pass filter outputs is more stable than the amplitude component when small deviations from image translation are considered.

[5] In addition these transforms are somehow similar, in the sense of the meaning of each channel (brightness and color contrasts).

- grouping methods: In these methods, an initial field of velocity is computed locally and then propagated and reinforced by some integrating process. An example of this technique is the tensor voting approach of Medioni et al. [73].

In this work we will concentrate on differential methods. We start by defining the problem in a more precise way, for this purpose, let $I_1 \dots I_K$ be an image sequence. We will consider a dynamic model where the image $I(\mathbf{x}, t)$ is a function of the pixel x and the time t.

Many models work under the assumption that the gray levels of an object do not alter its value during their motion:

$$I(\mathbf{x}, t) = I(\mathbf{x} + \vec{v}, t + \Delta t) \tag{3.10}$$

where $\vec{v}(\mathbf{x}) = (v_x, v_y)$ is the velocity of point $\mathbf{x}$ and Δt the time elapsed between two consecutive frames. In order to simplify this model many authors [71] take a first order approximation of (3.10) yielding the so-called *gradient constraint*

$$\nabla I(\mathbf{x}, t)\vec{v} + I_t(\mathbf{x}, t) = 0 \tag{3.11}$$

This is one linear equation on the two components of v, so we need an additional equation in order to obtain an unique solution. This is often called the *aperture problem*, and is tackled in many ways in the literature. One possibility is to add some global regularity constraint over the flow (see [72]). Another commonly used constraint is to assume that the velocity is constant within a neighborhood $B_\rho(\mathbf{x}_0)$ of size ρ (see [71, 29]). In this way the problem of solving (3.11) is turned into the following minimization problem

$$\min_{\vec{v}} \left\{ E(\vec{v}) = \frac{1}{2} \int_{B_\rho(\mathbf{x}_0)} \left(\nabla I(\mathbf{x}, t)\vec{v} + I_t(\mathbf{x}, t)\right)^2 d\mathbf{x} \right\} \tag{3.12}$$

Differentiating with respect to $\vec{v}$ and equating to zero yields the following 2x2 system of equations

$$\int_{B_\rho} \nabla I \nabla I^T . \vec{v} = \nabla I . I_t \tag{3.13}$$

if we recall the definition of the structure tensor G from section 3.2.2 this equation yields

$$\int_{B_\rho} G . \vec{v} = \nabla I . I_t \tag{3.14}$$

thus decomposing G into its components we arrive to

$$\begin{bmatrix} \int_{B_\rho} I_x^2 d\mathbf{x} & \int_{B_\rho} I_x I_y d\mathbf{x} \\ \int_{B_\rho} I_x I_y d\mathbf{x} & \int_{B_\rho} I_y^2 d\mathbf{x} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -\int_{B_\rho} I_x I_z d\mathbf{x} \\ -\int_{B_\rho} I_y I_z d\mathbf{x} \end{bmatrix} \tag{3.15}$$

This integration over the neighborhood B_ρ can be seen as the convolution of a *sharp* window $b_\rho = \mathcal{I}(B_\rho)$[6]. It is usual to replace the sharp window b_ρ with a Gaussian Kernel g_ρ yielding

[6] $\mathcal{I}(\Omega)$ is the indicator function of Ω, that is, a function whose value is 1 if $x \in \Omega$ and 0 otherwise

$$G_\rho.\vec{v} = (g_\rho * \nabla I)\,.I_t \tag{3.16}$$

where $G_\rho = g_\rho * G$.
Recently, some authors [74] proposed the use of anisotropic diffusion instead of the convolution with the gaussian kernel, so the structure tensor is diffused according to (3.4).

Once the structure tensor is diffused, we have to solve the following 2x2 system of equation in each pixel

$$D(G).\vec{v} = D\,(\nabla I)\,.I_t \tag{3.17}$$

where $D(.)$ is the anisotropic diffusion operator applied.

In the regions of the image where the gradient is flat, this system is ill-posed and the solution will be inaccurate. In order to avoid this, many authors propose the usage of a quality measure given by the eigenvalues of the structure tensor. This is achieved by considering the solution valid only on those points where both eigenvalues (or equivalently the minimum) are greater than a threshold.

3.6 Concluding remarks

Summarizing, for the segmentation task, we will use the following features. For color description, we will use the three $(K1, K2, K3)$ channels. For texture description, we will use the three wavelet sub-bands (hd, vd, dd) resulting from using only the first level of decomposition. Finally, motion information is introduced using both velocity components (v_x, v_y).

After these features are computed, the resulting vector is diffused using eq. (3.4). Then, the correction of histogram is applied to all frames in the sequence.

There are several ways in which the feature extraction could be improved. In the first place, a more complete texture description could be performed, taking into account color information with wavelet color signatures.

From the point of view of preprocessing, not only brightness but also color correction along the sequences could be performed.

Optic flow computation can also be improved by using more recent approaches which combine local and global computations, like the work of Weickert et al. [75].

Chapter 4

Proposed approach

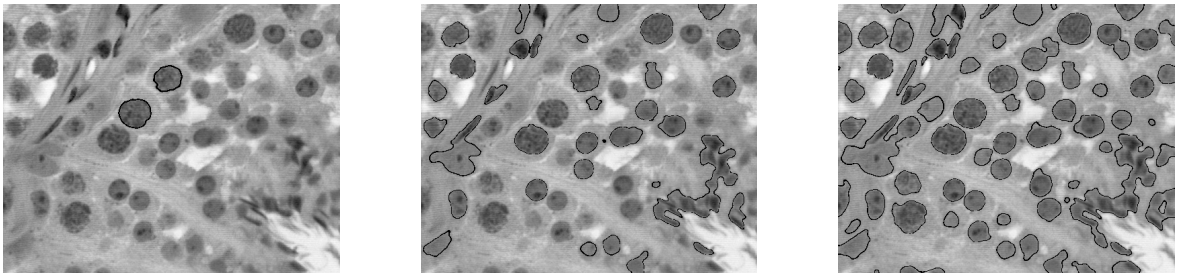


Figure 4.1: Objects detected in a sample image of our data sets. Left: without re-initialization (using the previous frame). Center: using evenly spaced spheres in each frame. Right: using the re-initialization algorithm.

4.1 Introduction

All the algorithms presented in section 2.3 present two main problems when faced with real data. The first comes from the broad range of variation between different biological tissues. For this reason, it is hard to find a feature set that fits well with all sequences. The second problem is due to the great variability between slices belonging to the same sequences. This causes the loss of tracked objects from one frame to another, and the fall of the algorithm in poor local minima. For this reasons we modify the classical GAR framework by adding two novel mechanisms: automatic weighting and re-initialization. In the following two sections we will explain these algorithms.

4.2 Reinitialization

As in many tracking algorithms, we take as initial estimation for the curve Γ in the $(n+1)$-th frame the final result for Γ in the n-th frame. Assuming that the difference between two consecutive frames is not too big, this choice of initial condition speeds up the convergence of equation 2.43. This also minimizes the risk of falling into poor local minima and not adequately segment our object of interest, something that would happen often if our starting condition were an arbitrary curve (a set of circles covering the image, for instance).

The downside of this choice of initial condition is that, if a new object of interest appears at frame $n+1$ that was not there at frame n, it will most probably not be detected. See for instance figure 4.1, where the 9th frame of one of our sequences is shown. In this image the objects of interest are the small dark round shapes, which generally keep their position (so they are easy to track) but appear and disappear along the sequence. For this sequence, choosing as initial condition the curve(s) found in the previous frame would imply that only the objects present in the first frame of the sequence would be tracked, they would gradually disappear and after just a few frames the segmentation would yield an empty set, although there are always many objects of interest in each one of the frames of this sequence.

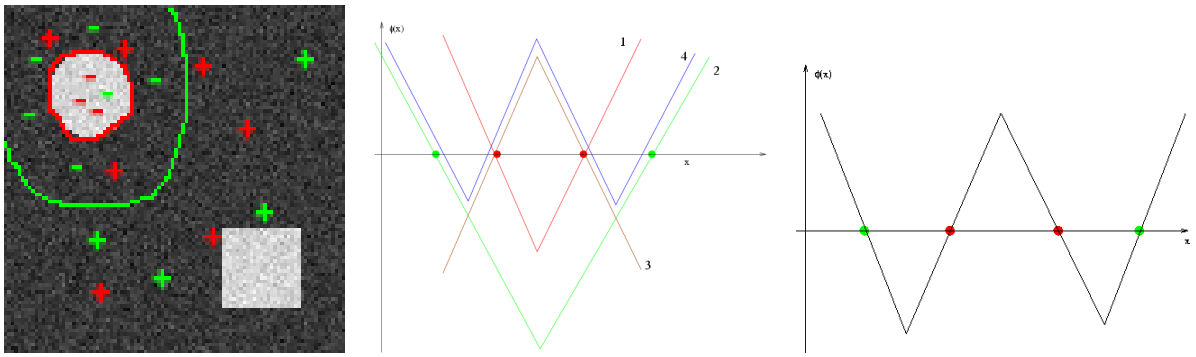
To solve this problem, we propose a method to detect if the curve Γ is missing any object and, if it does, we launch a re-initialization step. This mechanism is composed by two stages: decision and re-initialization.

The decision mechanism works as follows. As the evolution reaches a minimum, we assume that we have a considerable part of the object detected. Thus we can obtain a roughly accurate estimation of the object PDF in the inner region (Ω_1). In the outer

region, we have the background and the missing part of the object. So we can model the distribution of Ω_0 as a mixture model (MM) of M components. Thus the PDF in Ω_0 can be expressed as $p_0(x) = P_0^0.p(x|\theta_0^0) + P_0^1.p(x|\theta_0^1)$ where P_0^m are the *prior* probabilities of the m-th distribution and θ_0^m its parameters. We use the Expectation Maximization (EM) algorithm to learn P_0^m and θ_0^m. After the mixture is trained, we search for the component that is closer to $p_1(x)$ looking for the minimum distance between $p_1(x)$ and $p(x|\theta_0^m)\ \forall m \in \{1 \dots M\}$. If this minimum is lower than a threshold (d_{max}), we say that they are the same object. Details about the computation of this distance are given in section 4.4.

The decision mechanism presented above allows the detection of part of the object of interest which was assigned to the background. There is a reciprocal case in which the curve encloses the object and part of the background. To detect this case, we follow the same procedure, but inverting the estimation. That is, we estimate a MM for the object and then compare each component $p(x|\theta_0^i)$ with the background model $p_0(x)$. We search for the minimum distance and compare it with the threshold d_{max}. This allows not only to detect if there are lost objects but also to detect in which side of the front they are located.

Once a match is found, we check if the area of the found object is bigger than a threshold ($P_i > a_{min}$), and if so, we start the re-initialization process. Note that there are two reinitialization evolutions, one for the object and one for the background.



(a) sign (b) example of construction (c) final front

Figure 4.2: Construction of the re-initialized front

For the re-initialization step, we modify the front in such a way that we get two curves: the current curve attached to the object, and another that will evolve outwards to find the lost objects. An example of the construction of this front is shown in figure 4.2. The red curve is the result of the segmentation, as we can see the front lost the square on the lower-right corner. Then to construct the second part (see[7] 4.2(b)) of the front we simply displace the final curve Γ (no. 1) outwards a few pixels (no. 2) as in 4.2(b), and then we intersect it [8] with the original curve inverted (no. 3) obtaining the curve (no. 4) shown in figure 4.2(c). In the reciprocal case (front moving inwards) an analogous construction can be performed.

If reinitialization is needed for both sides we construct only one front combining the two processes (see fig 4.3) which is able to evolve simultaneously in both directions.

[7] in the color version numbers 1,2,3 and 4 are equivalent to colors red, green, brown and blue

[8] an intersection of curves in the level set formulation is equivalent to taking the maximum between both embbeding funcitions

Altough it appear to be three curves, there is only one embbeding function, thus there is only one curve.

Once we have this new front, it is evolved according to the same law (2.43), with the difference that the parameters of the PDFs remain fixed.

Summarizing, each time the evolution finishes, we perform two decisions: one that detects if we need to go outwards and another which detects if we need to go inwards. If both decisions are `true` we reinitialize for both sides, and if not, we evolve to the side which was true. In the case that the two algorithms return `false` no reinitialization is performed.

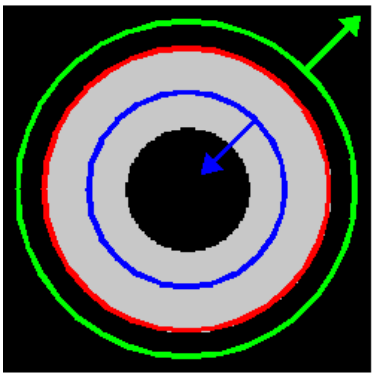
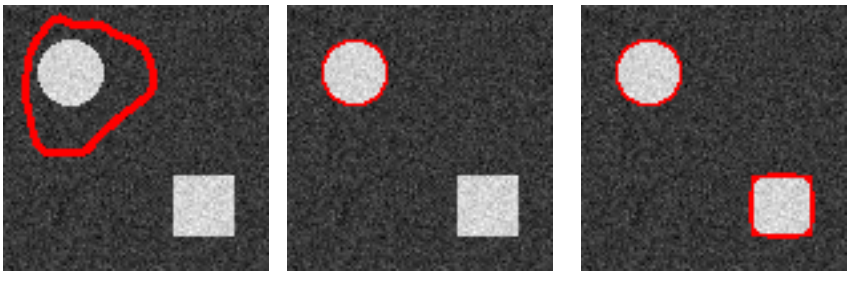


Figure 4.3: Construction of the re-initialized front. Red: original front, Blue: added curve which evolves inwards, Green: added curve which evolves outwards.

As an example we show in figure 4.4 the re-initialization algorithm applied to a simple test case. Initially we start with a front 4.4(a) that evolves towards the circle. After segmentation finishes we obtain a perfect segmentation of the circle, but the square is lost (see 4.4(b)). Finally, after the reinitialization is performed, we are able to detect the square (see 4.4(b)). More details on how this algorithm works will be shown in chapter 5.



(a) (b) (c)

Figure 4.4: Example of the reinitialization algorithm: (a) initial front, (b) after normal segmentation, (b) after the reinitialization algorithm

4.3 Automatic weighting

To deal with inter-frame variability, we propose a method for the automatic choice and updating of the weights w_j. For the first frame the user supplies a segmentation which is taken as ground truth. For each feature channel j we compute the PDF in Ω_0, call it p_0^j, and the PDF in Ω_1, call it p_1^j. Then we measure the *distance* d_j between p_0^j and p_1^j, as we will explain in section 4.4. A big value for d_j means that the channel j provides good discrimination, so w_j should be big as well. Thus, we choose:

$$w_j = \frac{d_j}{\sum_{i=1}^{N} d_i} \tag{4.1}$$

For the second and following frames, the procedure is the same, but using the PDF's computed from the segmentation obtained in the previous frame.

4.4 Metrics in the PDF space

As we mentioned in previous sections, we need to define a notion of *distance* between two PDFs. There are many possible metrics in the space of probability distributions with different properties, but in this work we will only consider those beloging to the Ali-Silvey class of information-theoretic distance measures. This family of measures offers many interesting properties, that make them particularly well suited for classification tasks. These properties relate these distances between two PDFs with the performance of the optimal classification between them.

Note that the term *distance* should not be taken rigorously, none of this measures satisfy all the axioms that distances must satisfy. We refer to [76] for a review of the properties of such metrics.

One common choice in the literature (see ([33],[68]) is the Kullback-Leibler divergence (KLD). Let $p_1(x)$ and $p_2(x)$ be two PDF, then the KLD is defined as

$$\mathcal{D}(p_1, p_2) = \int_{\mathcal{X}} p_1 \log(\frac{p_1}{p_2}) dx \tag{4.2}$$

From the practical point of view, this distance has proven ([68]) to outperform traditional measures like normalized Euclidean or Mahalanobis distances. This is specially relevant when the distributions are not gaussian.

This metric has many interesting properties, such that it is additive if the PDF components are independent. In addition, it satisfies the triangular inequality, but it has the drawback that it is not symmetric. To deal with this problem a symmetrized version of the KLD can be used. There are many ways to derive this symmetric *distance*, as discussed in [76]. Following this work we use the resistor-KLD, which is merely the harmonic mean of the two possible KL-distances. In that work, they show many theoretical properties and geometric interpretations of the KL divergence. In particular, they show that the resistor-KL is one of the tigthest bounds on the optimal classifier performance.

Another useful metric is the Bhattacharyya measure, this one is a similarity measure

rather than a distance, and it is defined as

$$\mathcal{S}(p_1, p_2) = \int_{\mathcal{X}} \sqrt{p_1 p_2} dx \tag{4.3}$$

This is a symmetric measure, additive, but does not satisfy the triangular inequality. An important drawback of the Bhattacharyya measure is that it is a bound on classifier performance, and not an equality like in the case of the KLD.

4.5 Implementation

In the following we will summarize the characteristics of the developed algorithm and give some details on the implementation.
Our algorithm is based on the region-based active contour framework presented in section 2.3.2 and allows the combination of different features in the segmentation process. The automatic weighting mechanism presented in previous section is responsible for the relative importance of each channel in the segmentation of a given sequence.
In the experiments we use the (Y, Cb, Cr) components for color images or the intensity channel in the case of gray level images.
In order to produce texture channels, we process the intensity channel with the wavelet packet transform (assuming that all texture information is contained in the Y channel).
In the case of video sequences, we add two channels correspondig to the optic flow components.
In the most general case we use 8 feature channels: intensity Y, Cb and Cr for the color images, the horizontal, vertical and diagonal detail for the first level of wavelet decomposition, and the two velocity components.

For color and movement channels we assume a gaussian model, and for texture we use a GGD model. In all cases we work under the hypothesis of independence between different channels. We currently estimate the MM only in the case of gaussian distributions.

The algorithm initialization is manually provided by the user in the first frame. This also can be done in an un-supervised way starting with an arbitrary contour, and performing a segmentation of the initial frame like in 2.3.3. With this result we can start the tracking algorithm.

From the numerical point of view, we implemented the evolution 2.43 with a time explicit first order scheme, and a spatial discretization as discussed in section 2.3.5. The time step is adaptive and is updated in each step according to 2.53.

As we seek for a steady state solution of (2.43), we must define a way to determine if we reached this state. This is achieved by measuring the amount of change of the curve between consecutive frames, relative to its length. When this change is lower than a threshold (η) the algoritm stops.

The values chosen for the parameters needed by the algorithm are fixed and the same for all the experiments: the weight of the regularization term α, the weight of the data term γ, the minimum size of the detected object a_{min}, the maximum distance between the PDFs d_{max}, and the stopping condition η for the curve evolution.

All the code was programmed in C++ on Linux. For the PDE solution and curve evolution we used and extended the Flujos Toolbox [77] which provides a dimension-independent framework which makes very easy to solve PDE problems. For the numerical analysis we have also used Blitz++ [78]. For visualization and I/O we used ImageMagick library [79] in the 2D case and the Visualization Toolkit (VTK) [80] in 3D. For 3D reconstruction we have used BioVis3D [7]. In addition, several other library were used: GSL [81],[57]

4.5.1 Outline of the algorithm

With the details presented above the algorithm work as follows. In the first frame, a user provided segmentation (or unsupervised segmentation) is used . This segmentation is taken as the ground truth, so we can estimate the initial PDF parameters (θ_i) for each region. We assume a *smooth* variation of θ_i and the position and shape of the curve. Thus the curve in frame k is a good initial guess for the curve in frame $k+1$. In addition, we suppose that the object smoothly changes its properties (color, texture, etc) between frames, so we use the previously estimated parameters (θ_i) to drive the evolution.

In each frame we use as initialization the final front from previous frame and we assume that the p_i are fixed, then we evolve the curve towards a minimum of the energy. Once this minimum is reached, we estimate the p_i and the MMs to decide if reinitialization is required. In the case of reinitialization, we evolve the front, with the PDFs fixed, until the new steady state is reached. After the evolution finishes (with or without reinitialization) we estimate the PDFs to be used in the next frame. With this estimation, we compute the weights w_j for each channel and repeat all the process again.

A summary of this algorithm is shown in figure 4.5.

4.6 Concluding remarks

4.6.1 Re-initialization

In the first place we must note the importance of this mechanism, as shown in figure 4.1, the number of lost objects is reduced dramatically.

An important property of the re-initialization algorithm is that it is devised (by the construction of the front) in such a way that the segmentation previously achieved cannot be hindered and the detected objects cannot be lost. However this step has two major drawbacks: evolving outwards we can add spurious objects to the previously achieved segmentation and it heavly increases the (time consuming) iterations for each frame treatment. For these reasons it is important to use the decision mechanism in order to re-initialize only when it is needed.

An important property of the decision algorithm is the choice of the side to reinitialize. In the case of figure 2.8(b) if we reinitialize both sides, we are wasting a great number of iterations in the outer region[9]. This number could be even higher than the iterations needed to segment the original torus.

[9] If the (mean) velocity of the front is approximately constant, the number of iterations needed by the curve to travel through this region is approximately proportional to the largest dimension of the region.

1. Prior choices:
   - $\mathcal{M}, \theta_i$: PDF model and its parameters
   - N: number of features
   - K: number of frames
   - M: number of regions.
2. Parameters:
   - $\lambda, \mu, \gamma, \zeta$: weights of the different energy terms.
   - w: feature weighting vector.
   - $d\mathbf{x}$: size of the spatial discretization.
   - dt: time step for the evolution.
3. Initialization:
   (a) for each frame k of the sequence ($k \in \{1..K\}$):
      i. compute feature vector $U_k(x)$
      ii. perform anisotropic diffusion over the feature vector
   (b) Start with an arbitrary curve Γ_0 (manually drawn or a result of an unsupervised segmentation)
4. Curve evolution:
   for each frame $k \in \{2..K\}$:
   (a) for each region i compute θ_i^{k-1} from previous frame.
   (b) for each feature channel j compute w_j from previous frame.
   (c) use the curve from previous frame Γ_{k-1} as initial condition
   (d) evolve the curve according to (2.43) using θ_i^{k-1} and w_j fixed during the frame.
   (e) Stop condition:
   goto 4d until one of the stop conditions is reached:
      i. the curve reaches steady state ($\frac{\partial \Gamma}{\partial t} \approx 0$)
      ii. the energy reaches a minimum
      iii. the iteration count reaches a maximum
   (f) Decision:
      i. estimate a MM $p_m m \in \{a, b\}$ for region Ω_0
      ii. compare with p_1
      iii. if $d(p_1, p_m) < d_{max}$ then $r_0 = true$
      iv. estimate a MM $p_m m \in \{a, b\}$ for region Ω_1
      v. compare with p_0
      vi. if $d(p_0, p_m) < d_{max}$ then $r_1 = true$
   (g) Reinitialization:
      i. if $r_1 = true$ construct the outwards front
      ii. if $r_0 = true$ construct the inwards front
      iii. go to step 4d
   (h) go to next frame.

Figure 4.5: Summary of our tracking algorithm

Finally, in some cases the decision algorithm can fail, and no re-initialization will be performed, so we could lose some objects. If one wants to be sure that no object is lost, re-initialization can be forced in all frames.

4.6.2 Weighting

This weighting algorithm, altough very simple, proved to be a great improvement over the original formulation, since it eliminates the manual choice of the feature weights. In order to be an useful algorithm it must hold two basic properties: in the first place, when the choice of w is (manually) evident, this algorithm must agree with the manual choice. And in the second place, the performance achieved must be higher than when we do not use prior knowledge, i.e. choosing equally weighted features. In addition, as we will see in the results chapter, this weighting algorithm will help to *discover* the optimal weights when this choice is not obvious.

Although embbeded in a geometric formulation, the segmentation algorithm presented relies on the *comparison* of two PDF. This is carried out by the term $\log\left(\frac{p_1}{p_2}\right)$, which decides to which side the curve moves based on the relative magnitude of the two PDFs. Thus this reduces to a bayesian classifier. As we are measuring here the discrimination power of a feature channel, it is important that the separability measure agree with the classification procedure used.

As an example of this, we present the following example. Suppose that our PDF model is gaussian, but our similarity measure consists in the comparison of the mean gray level of each region. It is clear that if we face an image with two gaussian regions with different means and same variance, the separability measure will tell us the right thing. On the other hand, if we had an image with same means and different variances, this measure will be tricky, it will tell us that the regions are not separable, but in fact they are.

This feature weighting algorithm is related to the feature selection mechanism presented by Sandberg et. al [31]. In that work, they do not know in advance which texture channels are relevant, so they use an automatic selection algorithm to choose which feature channel is relevant. This is performed for channel j by computing a similarity measure $s^j = |\mu_0^j - mu_1^j|$ where μ_i^j are the mean values for each feature channel. Then the the n channels with higher s are selected. The differences with the proposed approach are:

- They use this measure to order and select the relevant channels (as a binary choice), but once selected, the resulting feature vector is equally weighted. In our approach, each channel is weighted by this measure.
- They re-select the features every 50-100 iterations, in our approach the weights are selected in the previous frame and then are fixed during the evolution in the current frame.
- Althoug their measure (difference of means) is coherent with their energy measure (fitting a piecewise constant model), this measure is rather simple. In our case we measure distances between PDFs, so our approach is more general. It can be shown, that if we use a gaussian model with constant variance and mean c, our measure (the KLD) reduces to s^j.

Chapter 5

Experimental Results

chapter outline

This chapter is devoted to the testing of the proposed algorithms and to the discussion of the experimental results obtained over real data sets.

In section 5.1 a detailed description about the data sets used for the tests is presented. The experimental results over these data sets are divided in four categories: segmentation of static images, segmentation of biological sequences, 3D reconstruction and tracking of video sequences. We will start in section 5.2 with some simple examples to illustrate the proposed algorithm in detail. These results will be shown over still images. Segmentation of (biological) sequences is discussed in section 5.3 and the corresponding 3D reconstructions are shown in section 5.4. Finally, results of tracking over video sequences are presented in section 5.5.
All the results shown in this chapter are presented only for a few selected frames of each sequence and in some cases over a cropped version of the images. The complete results are available in appendix A.

5.1 Data sets

All sequences come from real biological tissue used by biological researchers in Uruguay, and they were acquired by biologists of the Instituto de Investigaciones Biológicas Clemente Estable (IIBCE); the Sección Bioquímica, Facultad de Ciencias, Universidad de la República (SB-FC) and Departamento de Morfología y Desarrollo, Facultad de Veterinaria (DMD-FV). A summary of the sequences and their properties is shown in table 5.1, and some examples of each sequence are depicted in fig. 5.1. The complete sequences with the results of our algorithms are shown in appendix A. In the following we will present a detailed description about each data set.

Sequence	Acq.	Image Size	Nb. of slices	Mosaick
Spider	OM	1360x1024	53	-
Crypt	OM	2200x1000	32	8
Ram	OM	1000x800	19	8
Echinococcus	CM	512x768	50	-

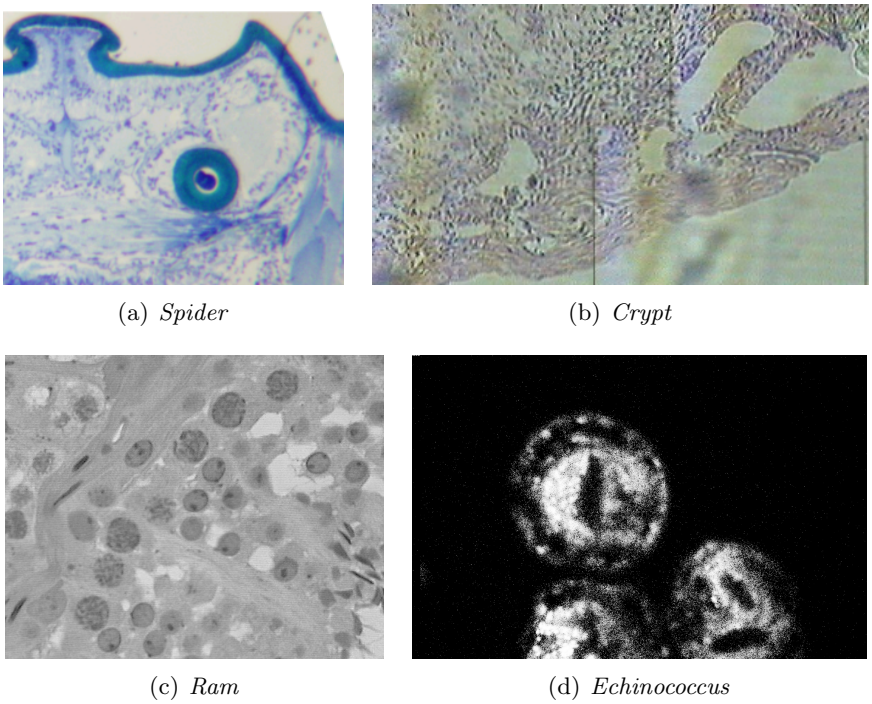
Table 5.1: Data sets used for the experiments

The *spider* series is a 1360x1024x53 RGB sequence of genital tissue from a spider (*Schizocosa Malitiosa*). The tissue is sliced with a diamond blade, stained in blue and finally the pictures are taken by optical microscopy (OM) with a magnification factor of 100X. The z spacing is approximately $1.5\mu m$. For all these sequences we do not have any available information about the $x - y$ resolution. This sequence was acquired at the Laboratorio de Etología, Ecología y Evolución, of the IIBCE[10].

The purpose of the biological study (see [82]) is to understand the structure of the sperm containers and the dynamics of the sperm in the genital system of the female spider. This allows the biologist to understand the reproductive behavior of these spiders. The objects of interest in this case are the membranes (border) that delimit the tissue and the torus-shaped object in the middle of the image.

The difficult part of this sequence is the variation of color and intensity due to the non-uniform blue dying of the tissue. There are also non-interesting objects stained by

[10] in collaboration with the Zoological Research Institute and Museum, Leibniz Institute for Terrestrial Biodiversity Research



(a) *Spider* (b) *Crypt*

(c) *Ram* (d) *Echinococcus*

Figure 5.1: Examples of the data sets

mistake, like the round-shaped object within the torus. In addition, the structures of interest suffer of important topological changes along the sequence.
In this sequence the most important features are the color channels, on the other hand texture seems to be completely useless.

The *Ram* series is a 1000x800x20 RGB sequence obtained from optical microscopy (OM), at the DMD-FV. The width of the slicing is approximately $1\mu m$. The biological material corresponds to slices of seminiferal cells of rams. In this case, the goal of the biologists is to correlate the morphology of certain cells (*sertoli* cells) in several stages of the development of the animal, with the growth and differentiation of other cells and with sperm production (see [83]).

This sequence is composed by many small dark gray objects (nuclei) over a light gray background, plus a few white zones. The objects of interest in this sequence are the nuclei, which tend to appear and disappear, remaining only few frames. Another difficulty about this sequences is the strong contrast variation that occur between different slices. The regions of interest are easily classified by its mean gray-level. Shape is also an interesting feature to use here. On the other hand, texture seems to be completely useless.

Crypt is a 2200x1000x32 RGB sequence where each slice is a collage of 6 to 8 OM images. Each image is acquired with a magnification factor of 100X and a resolution of 640x480. The mosaick was created manually and after that the resulting image was sub-sampled by a factor of 2. The z spacing is approximately $6\mu m$.
The tissue comes from ovaries of female sea wolves (Arctocephalus australis) from the DMD-FV. The problem here is to find the *crypts* (the cavities in the tissue) and to discover the interconnections (if they exist) and the structure of this *crypt* system (see [84]).

The main difficulties presented by this sequence are the artifacts introduced by the mosaicking algorithms, the strong intensity and color variation between consecutive frames, the a-periodicity and non-uniformity of the texture of the tissue (in the same frame). In addition the small dark structures that compose the textured tissue are particularly difficult for an edge detector. The relevant features here are the texture, the intensity and the color.

The *Echinococcus* sequence is a 512x768x50 series of tissue extracted from an *Echinococcus Granulosus* using confocal microscopy (CM) by C. Martinez at the Instituto de Biotecnología,UNAM, Cuernavaca, Morelos, México. The objects of interest are the round-shaped structures composed by zones of high density point clouds. This sequence is very different from the other and is particularly difficult due to the absence of strong edges and the noise level in some of its frames. However, the uniform black background facilitates the task.

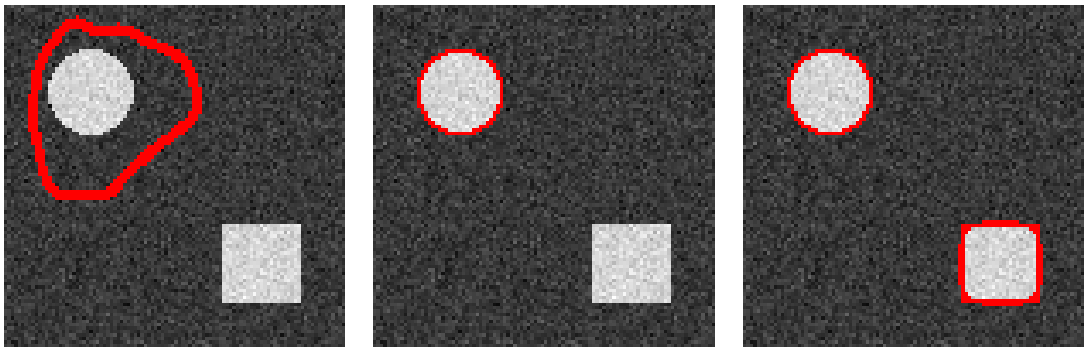
The *Echinococcus Granulosus* is the causative agent of the hydatidosis. The disease is caused by the pressure exerted on viscera by hydatid cysts that are formed upon ingestion of E. granulosus eggs excreted by canine. The goal of the biologists (see [85]) is to describe the cellular organization and the appearance of differentiated structures both in nascent buds and developed larvas.
The relevant features here are the density of the white points. This is easily classified in two regions, by using the mean and the variance of the intensity. Indeed, an algorithm using only the mean should perform well on this sequence.

5.2 Segmentation

5.2.1 Synthetic Test 1

In order to validate the reinitialization algorithm, we designed a toy example, in which we are in the exact hypothesis of the model. We have exactly two regions, drawn from gaussian distributions with different means and same variance. The object of interest is composed by two disjoint components with high mean, and the background has a lower mean. This is depicted in figure 5.2.

As this figure shows, we initialize the algorithm with a curve that encloses only one of the objects, in order to ensure that it loses the other.



(a) (b) (c)

Figure 5.2: Comparison of segmentation with and without reinitialization: (a) initial front, (b) after classic segmentation, (c) after reinitialization is performed

In the following we will show the output of the algorithm (bold typeface) and the estimated values of the parameters. The real values of the parameters are $\mu_0 = 70, \mu_1 = 220$ and $\sigma_1 = \sigma_2 = 15$.

In the first stage of the segmentation 5.2(a) the algorithm is learning both distributions, the interior region (region 1 with $\phi < 0$) attaches itself to the round object (see fig. 5.2(b)) and the exterior (regin 0 with $\phi > 0$) to the mixture of the background and the square. Thus we have a good estimation of p_1 and a poor one for p_0. The resulting parameters of this evolution are shown in table 5.2.

Region	0	1
area	9511	489
μ	92.91	217.45
σ	33.18	16.00

Table 5.2: Estimated parameters.

With this parameters we initialize the mixture model (MM) in the outer region, setting the prior probabilities equal ($P_a = P_b = 0.5$). After the MM is trained we get the parameters shown in table 5.3.

Region	0		1	
Mix. Comp.	0	1	0	1
Prob.	0.944	0.056	0.999	0.001
μ	85.33	219.19	220.11	217.64
σ	17.23	16.95	16.29	15.97

Table 5.3: Estimated MM parameters.

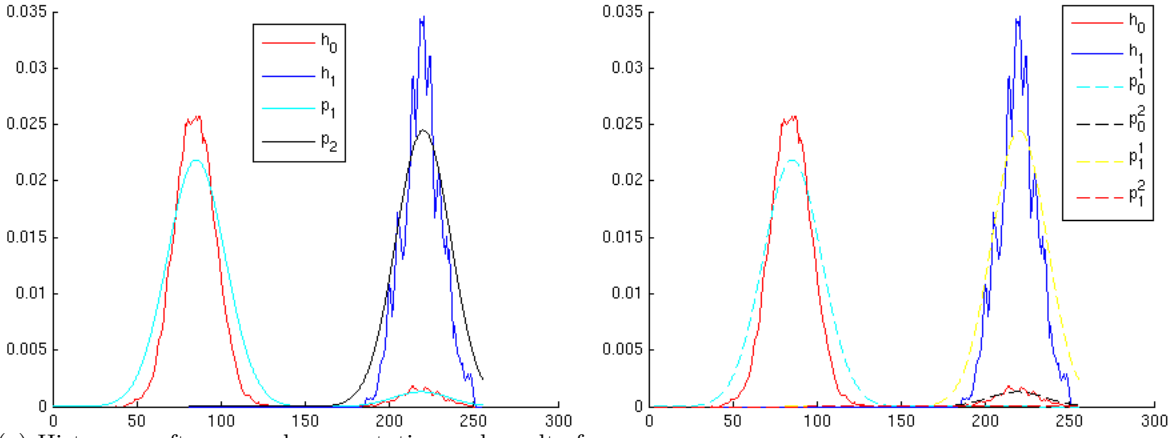
These results are depicted in figure 5.3. In the inner region (reg 0) the prior probabilities of each mixture components show that, in fact, we have only one distribution ($P_b \approx 0$). This corresponds with the fact that region 1 encloses almost only pixels of the object.

On the other hand, in region 0 (exterior), we found a Gaussian with $\mu = 83$ (dark) and another with $\mu = 219$ (light) which tells us that we have the background and part of the object present. Then we compare the distributions and seek for the most similar pair:

```
Most similar distributions found:
(219.186,16.9509)
(220.109,16.288)
Probability of detected object: 0.0556198
Similarity: 0.983751
Reinitialization needed!
```

As the similarity measure is greater than 0.7, we state that the matched objects are the same. In addition, as the probability P of that object is greater than 5%, we decide that is worth to reinitialize. In general, it is recommendable to choose the parameters in such a way that the algorithm tends to reinitialize. This ensures that we will find all the objects present.

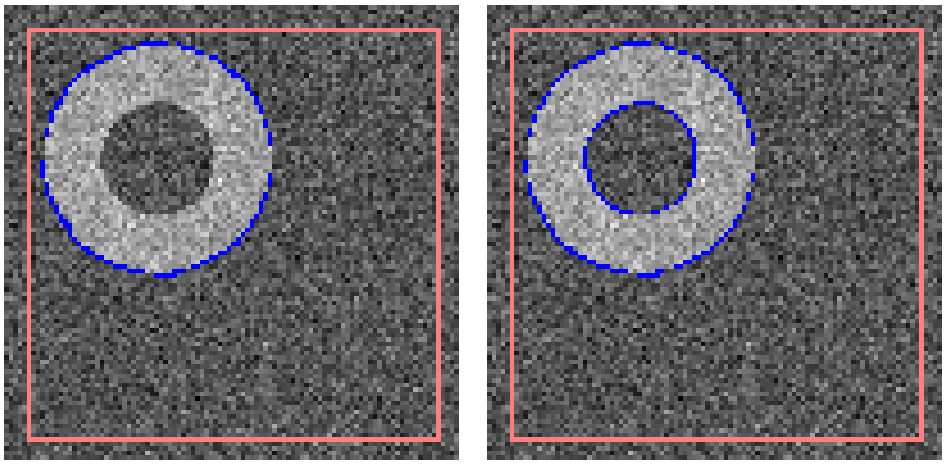
After modifying the front according to section 4.2 and evolving this front we get the result show in figure 5.2(c).



(a) Histogram after normal segmentation and result of GMM estimation

(b) GMM estimation: all mixture components

Figure 5.3: Results of the GMM estimation



(a) Result of classic segmentation

(b) Result after reinitialization

Figure 5.4: Results using reinitialization and decision. Red: initial curve, Blue: final curve

5.2.2 Synthetic Test 2

As we explained before, the reinitialization algorithm is a time consuming operation, so we need to reinitialize only if it is really necessary. In addition, sometimes the lost objects are only in one side of the front, so it is useful to decide if we must reinitialize and to which side we will move towards. In this example we will show how this mechanism works.

In this case, when the evolution finishes the front attaches itself to the external edge of the ring, and it is unable to split and thus it loses the center hole.

In table 5.4 the estimated parameters after the evolution are shown. The corresponding segmentation result is shown in figure 5.4(a). In the inner region (reg. 0) the mean is lower than in the outer region. As expected, the variance in reg. 0 is near its real value (50) but in the inner region the estimation is corrupted by the dark pixels in the center hole. So the variance in reg. 1 gets higher. Thus, in this example, we expect that the decision mechanism tell us that we need to go inwards.

region	0	1
area	8025	1975
μ	70.66	181.52
σ	49.9	81.41

Table 5.4: Estimated parameters

After the evolution finishes, we estimate the MM and get the parameter shown in table 5.5. In this case in the interior it found a distribution p_1^0 with parameters $(129.75, 36.8)$ which has a distance with p_0 of 0.01. As this value is lower than the threshold $d_{max} = 0.2$, the decision mechanism decides to go inwards.

In the outer region the algorithm the most similar distribution found has a distance of 0.21 with respect to p_1. Thus, as this is greater than the threshold, it decided not to reinitialize.

Region	1		0	
Mix. Comp.	0	1	0	1
Prob.	0.79	0.21	0.89	0.11
μ	63.05	129.75	214.10	57.62
σ	45.92	36.80	51.63	44.21
Distance	0.63	0.01	0.38	0.21

Table 5.5: Estimated MM parameters.

Finally, the final results of the algorithm with the reinitialization is shown in figure 5.4(b).

5.2.3 Follicle

Figure 5.5 shows the results of segmentation over the static image *Follicle*. The tissue shown in this image is similar to the one present in the *Crypt* sequence. The goal here is, again, to detect the holes in the tissue. The algorithm was initialized with evenly spaced spheres that are shown in step 1. The next images are intermediate results (every 10 steps) in the evolution of the front. The final results is almost correct, but it shows a spurious detection and it loses the narrow gap in the left side of the object. The first phenomenon has to deal with the quality of the features; if in that zone of the tissue they are not very uniform, the velocity can be small and that part of the front does not disappear. On the other hand, as the lost part of the object is very narrow, and we are performing sub-sampling (in the DWT), this gap is even thinner in resulting feature vector. Thus, the front is unable to detect the gap.

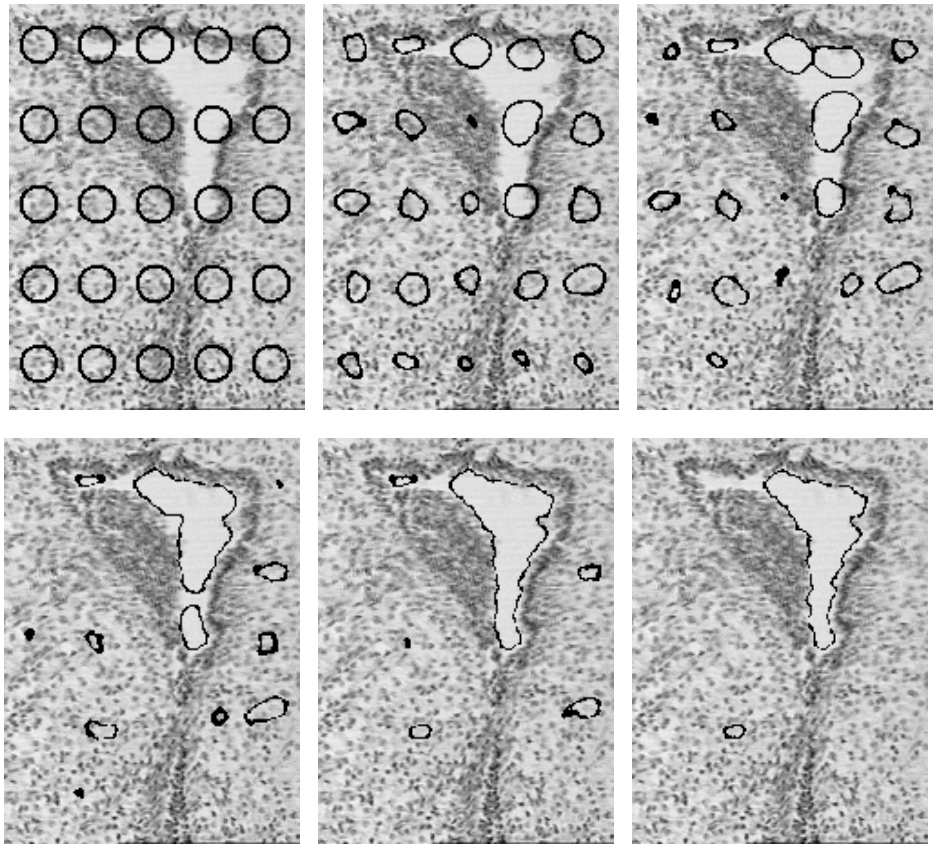


Figure 5.5: *Follicule*: Results of segmentation (steps 1,2,10,20,30,39)

5.3 Segmentation of sequences

In this section we will show the results obtained by the tracking version of the developed algorithm over biological sequences. A discussion of these results is given for each sequence.

5.3.1 Ram

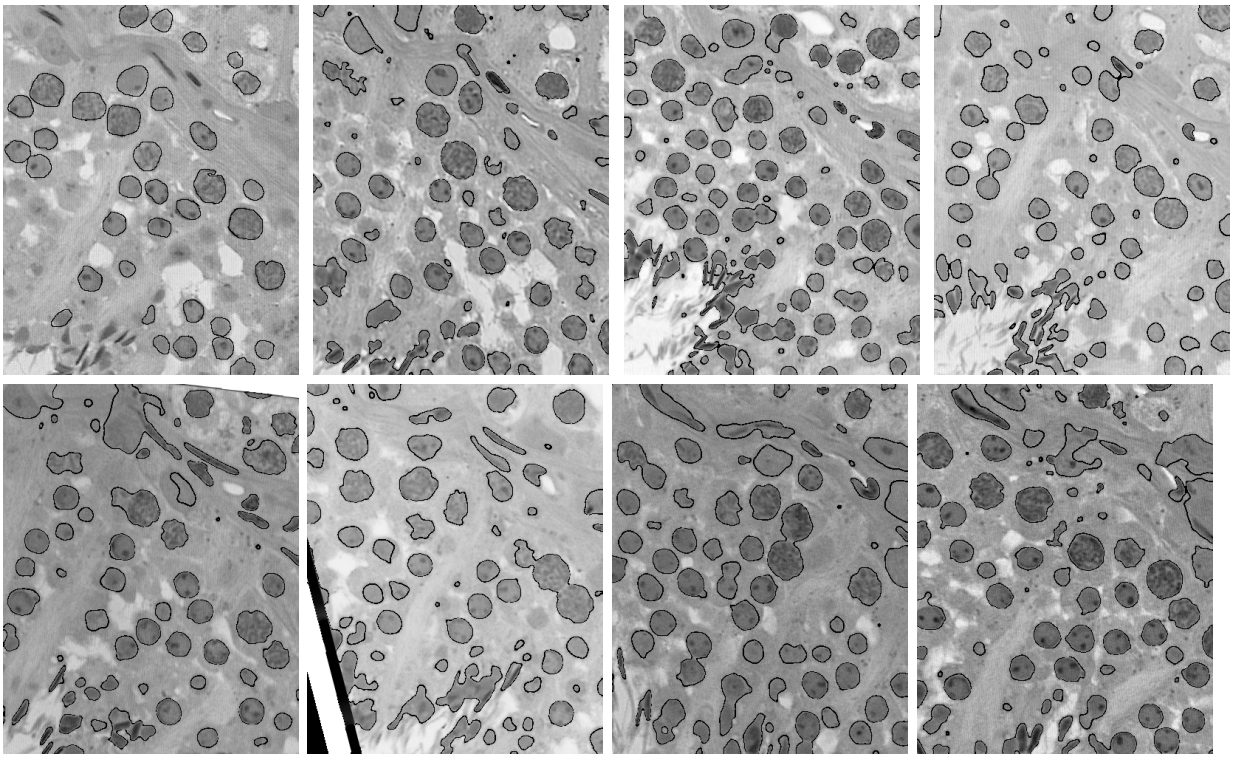


Figure 5.6: ***Ram*: Results (with reinit and weighting) over frames 0,1,5,8,11,15,19,20**

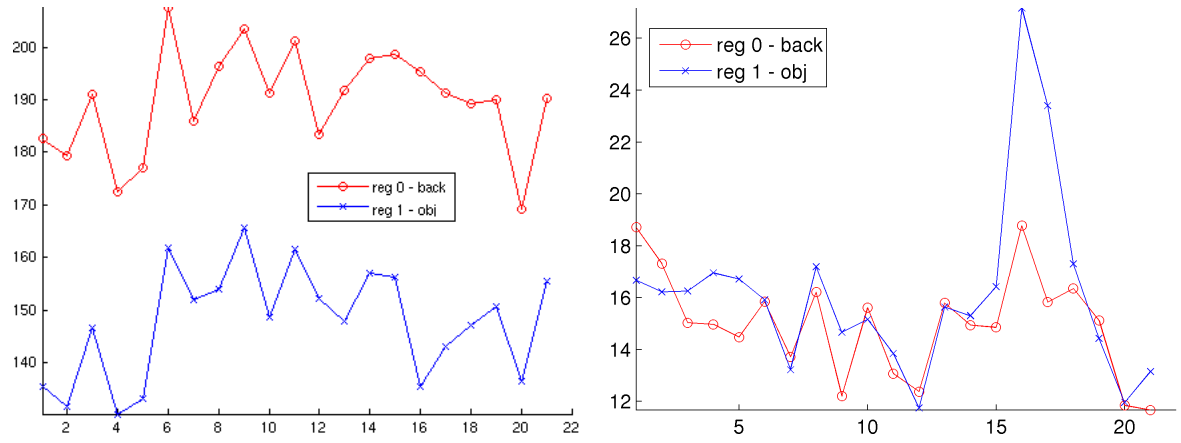
The results of our algorithm over this sequence are shown in figure 5.6, where the first frame is the user-provided initialization. This algorithm uses weighting and reinitialization, and the latter is forced in each frame.

As this figure show, the results are quite good. The only errors are that some disjoint objects are enclosed together by the same curve. These are due to the numerical implementation that does not allow to split the front when the gap between the objects is small (a few pixels) and also the gray level in this gap is not very different from the one inside of the objects.

All over the sequence, p_1 corresponds to the dark gray and p_0 to the joint distribution of light gray and white zones. As figure 5.7 shows, the parameters evolve according to these distributions: the mean value of the background is higher than the one of the foreground. And the deviations remain both relatively small, which gives a correct partition of the pixels.

However, in the 15-th and 16-th frames σ_1 gets a higher value. This bad estimation comes from the appearance of artifacts due to the mosaicking and the registration of the sequence. This is evidenced by the diagonal strip in the bottom left corner. This strip

adds a significant number of pixels with 0 and 255 values which widens the original σ_1.



(a) μ (b) σ

Figure 5.7: *Ram*: Evolution of the parameters

If σ_1 gets a value higher enough it can produce a big overlapping between the support of the two PDFs, and thus lead to a high missclasification rate. Currently we use a mask to avoid the downgrade on the estimation caused by these artifacts. This mask is computed from the original images taking as invalid those pixels with an exact 0(or 255) value. This can be easily modified to use an user provided mask in the cases when the invalid pixels values are not as evident as exact 0 (or 255).

In order to validate the proposed improvements we tested our two algorithms separately. To test the weighting algorithm, we compared the automatic weighting with two *natural* choices of fixed parameters. The first one is based on the assumption that we don't have any prior information, so we choose the same weights for all channels ($w_1 = \frac{1}{N}[1 \dots 1]$). The second choice, is to assign higher weights to the channels that appear to be (visually) more significative. In this case, the texture channels seem completely useless, so we assign $w_2 = [1, 0, 0, 0]$.

Figures 5.9 and 5.13 shows the results of the algorithms without the weighting algorithm, for the first and second case, respectively. As these figures shows, the results of the segmentation for the w_2 weights and the automatic algorithm are identical. This confirms the fact that, in this sequence, the only relevant feature is the gray level. As figure 5.8 shows, almost all the weight is assigned to the gray level channel. This results tell us that the weighting mechanism, chooses the correct weights for the features, and in the worst case, does not hinder the performance with respect to the fixed case (w_2).

In the case of the reinitialization algorithm we compared this mechanism with the algorithm without it. Results for this version are shown in figure 5.10. Without reinitialization, the algorithms rapidly loses objects and finally loses all structures in the 11-th frame. In the rest of the sequence no objects are detected. A first solution could be to forget the curve from the previous frame and to segment the image using an arbitrary initialization, using the statistical parameters from previous frame. As an example of this initialization, we tested the algorithm using as initialization evenly spaced spheres covering the image. The result of this algorithm is presented in figure 5.12. Although this

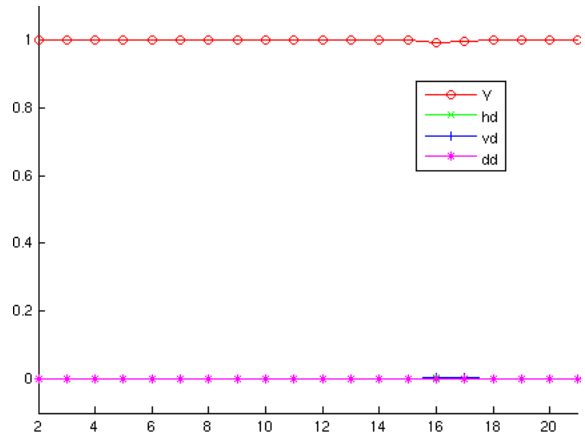


Figure 5.8: *Ram*: evolution of the optimal weights

approach improves the performance, it still loses an important number of objects.

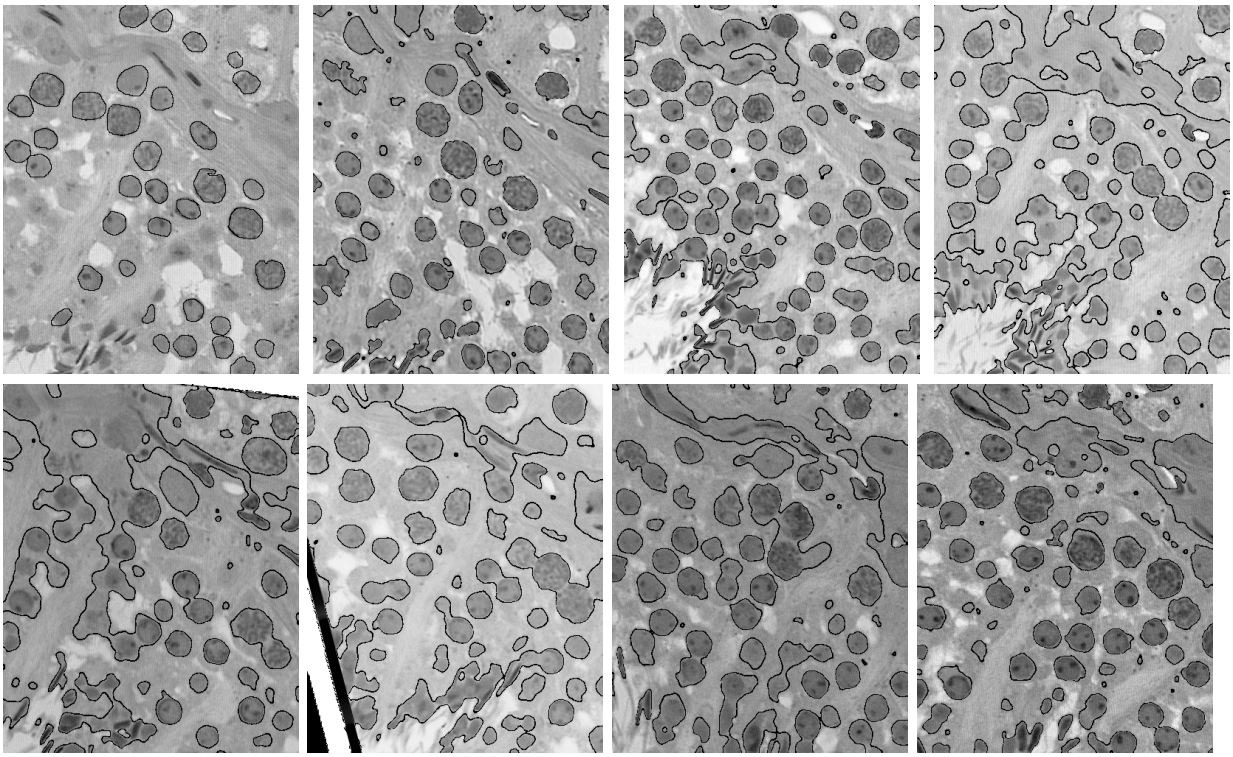


Figure 5.9: *Ram* (w/ reinit. and fixed weights w_1): frames 0,1,5,8,11,15,19,20

Finally, in figure 5.14 the results of the reinitialization using the decision algorithm is shown. Sometimes this mechanism fails, and no re-initialization is performed, so we lose some objects. However, the overall number of lost objects reduces dramatically using this mechanism.

A comparison of the abovementioned results, in terms of the number of objects detected, is summarized in table 5.6 and figure 5.15. As these results show, our reinitial-

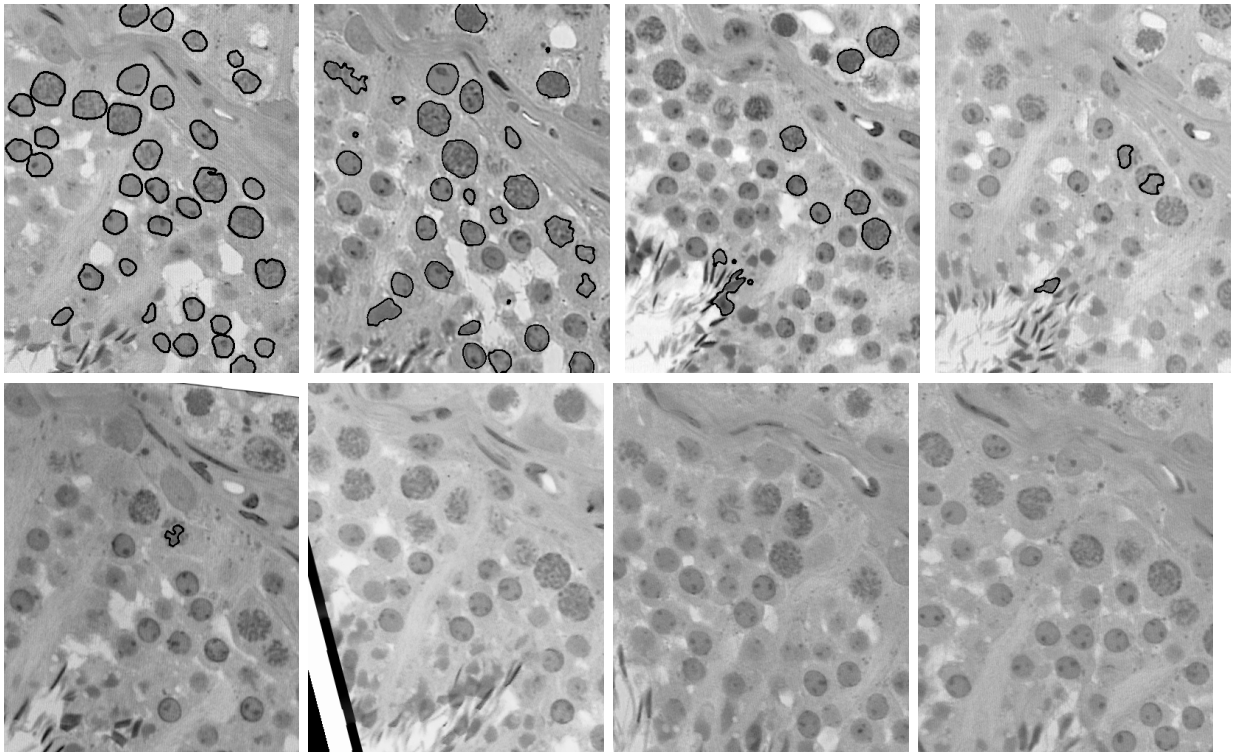


Figure 5.10: Results over the *Ram* sequence (using previous frame): frames 0,1,5,8,11,15,19,20

ization algorithm greatly improves the results over this sequence.

As we mentioned before, reinitialization is a time consumig task. To quantify this affirmation, we compared the iterations consumed by various versions of the algorithms. This is shown in figure 5.11 for 3 versiones of the algorithm: with forced reinitialization an fixed weights, with forced reinitialization and automatic weights, and with decision-reinitialization and automatic weighting.

As can be seen, the usage of automatic weighting does not affect the iterations performed. This happens beacause the automatic choice and the fixed weights are very similar. In an example where the weighting present bigger differences, the automatic weighting generally uses less iterations. This occurs beacase automatic features provides more separability and thus the force toward the minimum is bigger.

On the other hand, the use of the decision mechanism greatly reduces the iterations when it decides that it is not necessary to reinitialize (see low peaks in the figure). But, after a frame in which we do not perform reinitialization, the initial condition is worse than in the forced reinit, so the algorithm demands more iterations to reach the minimum (see high peaks in figure).

We are interested in locating all structures as a unique object, so no correspondence information between the objects is obtained. To achieve this, me must evolve a front per object (or at least log2(#objects) as in [86]), which heavily increases the complexity of the model.

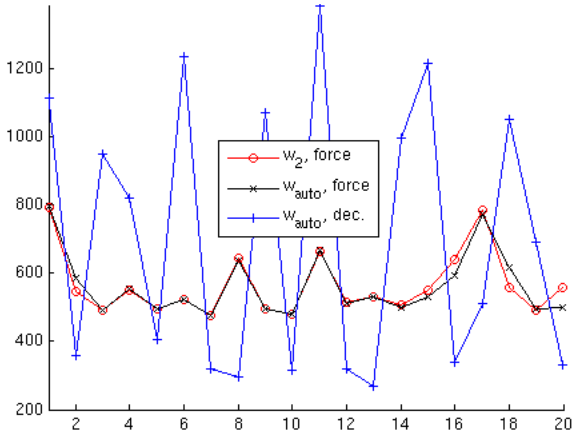


Figure 5.11: *Ram*: Iterations performed by each algorithm

Seq Frame	0	1	5	8	11	15	19	20
classic GAR	36	30	11	3	1	0	0	0
GAR w/spheres	36	30	49	29	31	30	41	47
GAR w/forced reinit.	36	81	101	73	71	68	71	80
GAR w/decision reinit.	36	96	77	56	75	50	70	59

Table 5.6: Number of objects detected

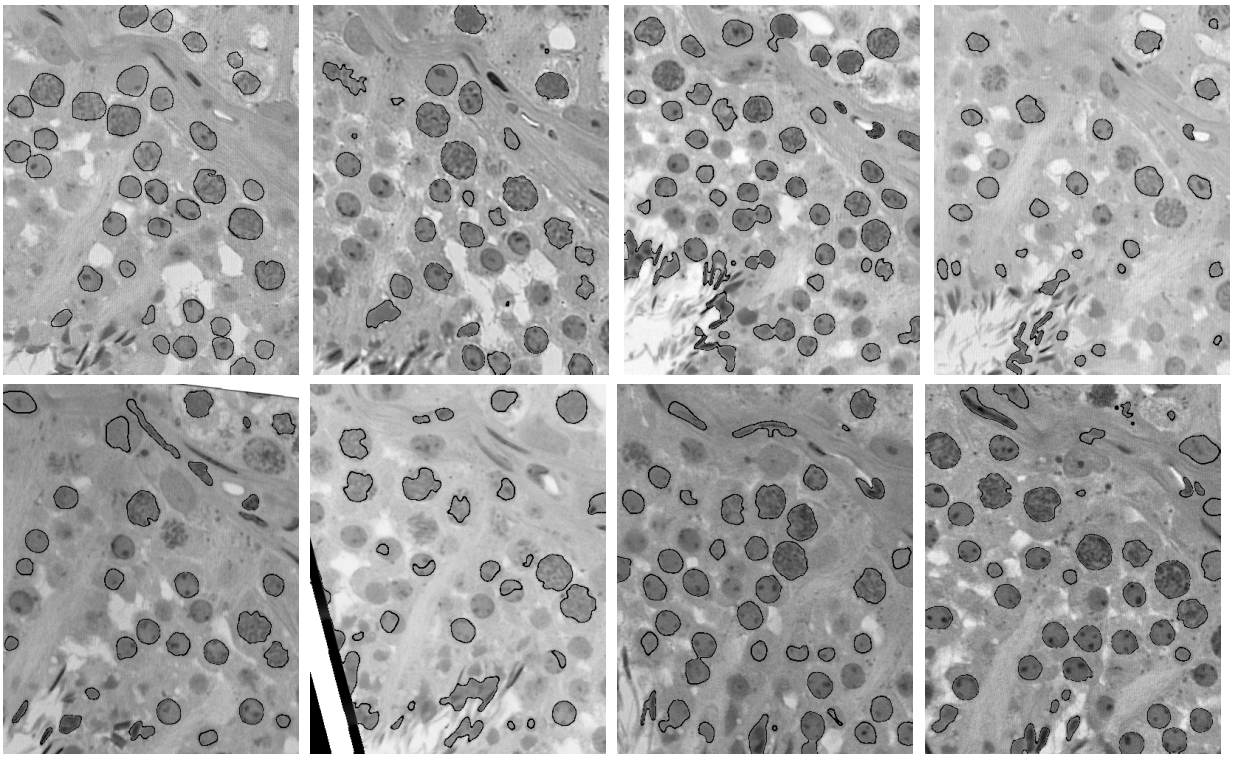


Figure 5.12: Results over the *Ram* sequence: frames 0,1,5,8,11,15,19,20. In each frame the initial condition are evenly *spheres*, no weighting or reinitialization is performed.

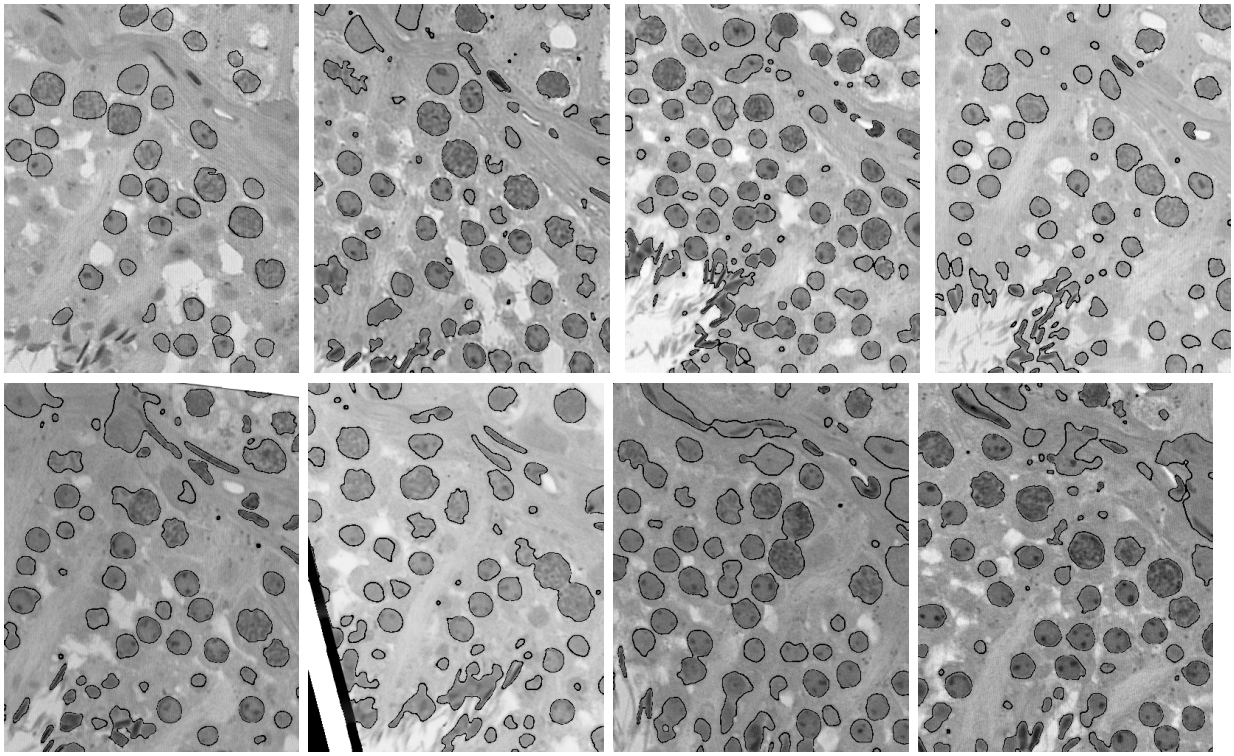


Figure 5.13: Results over the *Ram* sequence (w/ reinit and fixed weights w_2): frames 0,1,5,8,11,15,19,20

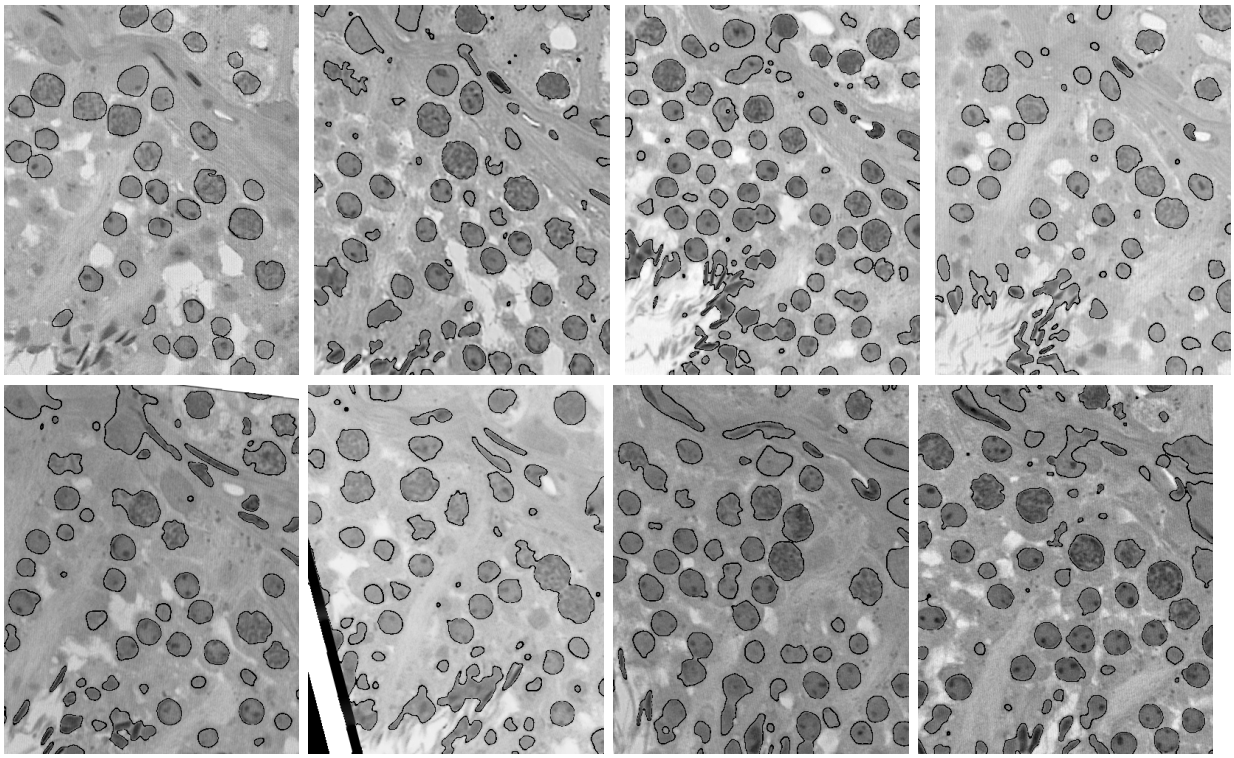


Figure 5.14: *Ram* (w/ reinit., decision and weighting): frames 0,1,5,8,11,15,19,20

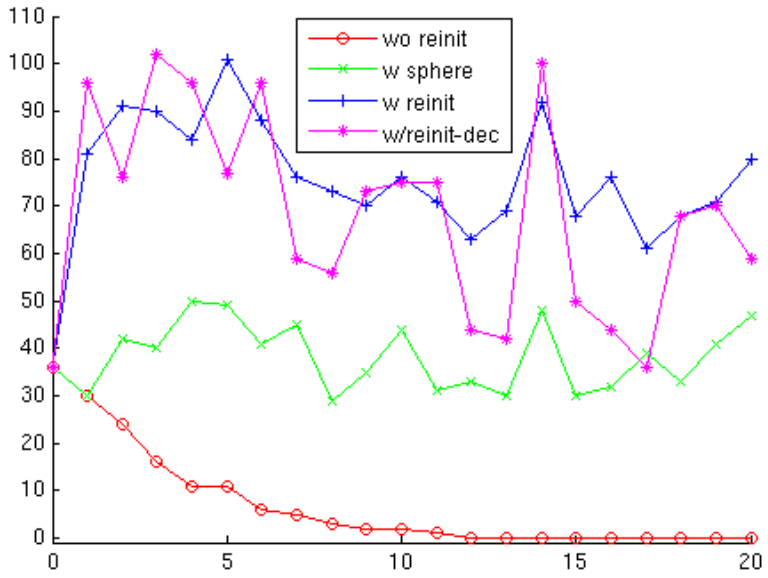


Figure 5.15: *Ram*: number of objects detected

5.3.2 Echinococcus

Figure 5.16 shows the results on the *Ehinococcus* sequence. The notation "5:10:45" used for frame numbering means: "frames from 5 to 45 in steps of 10 frames", namely frames 5,15,25,35,45.
The structures marked by the biologist in the first frame are well tracked along the sequence. After the 40th frame, the point density becomes too low and the algorithm quickly loses the structure of interest. Also, in 35-th frame, as the background noise level increases, some spurious objects tend to appear (upper-left corner).

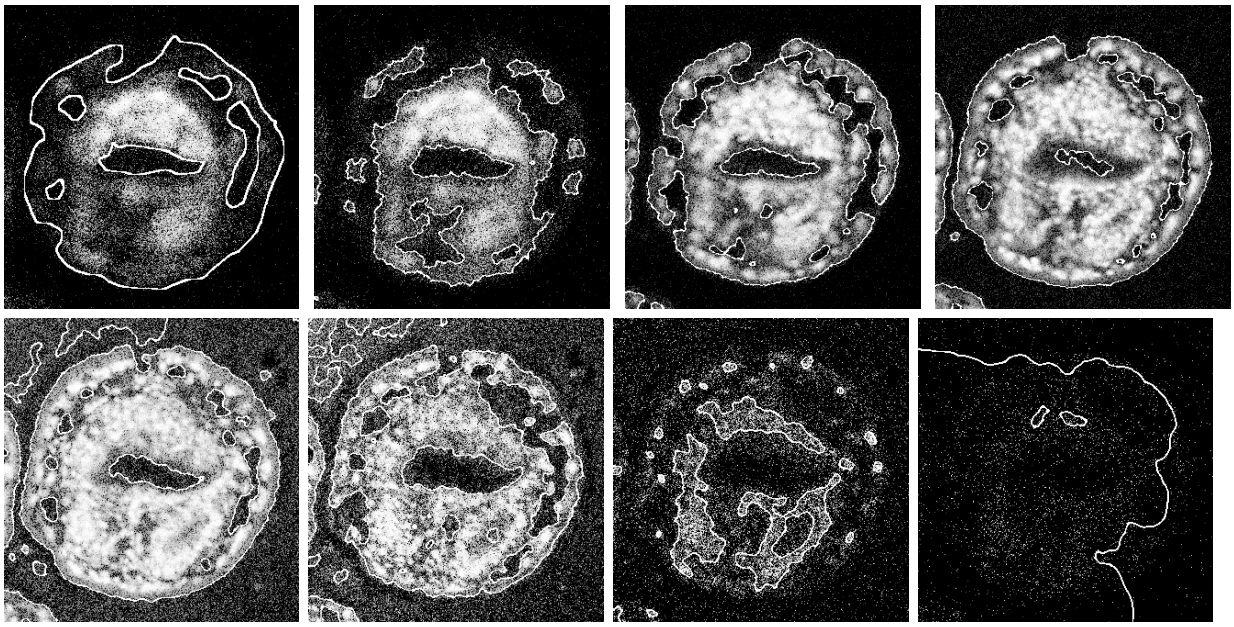


Figure 5.16: *Echinococcus* (w/ reinit and weighting): frames 14,15:5:45

In order to evaluate the reinitialization algorithm, in this example, we ran the algorithm without reinitialization. Results are shown in figure 5.17. In frames 25, 30 and 35, the front losses the small holes near the border of the object, and the central hole is significantly diminished. In addition, in the 35-th frame, the front almost completely loses the object.

Of all tested sequences, this is the only one were the weighting algorithms downgrades the performance compared with the fixed weights w_2 (see previous section). This can be verified comparing figures 5.16 and 5.18. In frame 25 the front is unable to completely detect the center hole. In addition, in frame 30 some spurious objects appear in the upper right corner. In the context of the whole sequence, this is an insignificant error. If one takes into account the robustness gained by the auto-selection of the weights, this downgrade is completely affordable. In addition, this comparison is especially disadvantageous because it is easy to manually select the relevant channels. In other sequences, like *Spider* or *Crypt*, this choice is not that evident, so this approach proves to be more useful. The evolution of the computed weights is shown in figure 5.19

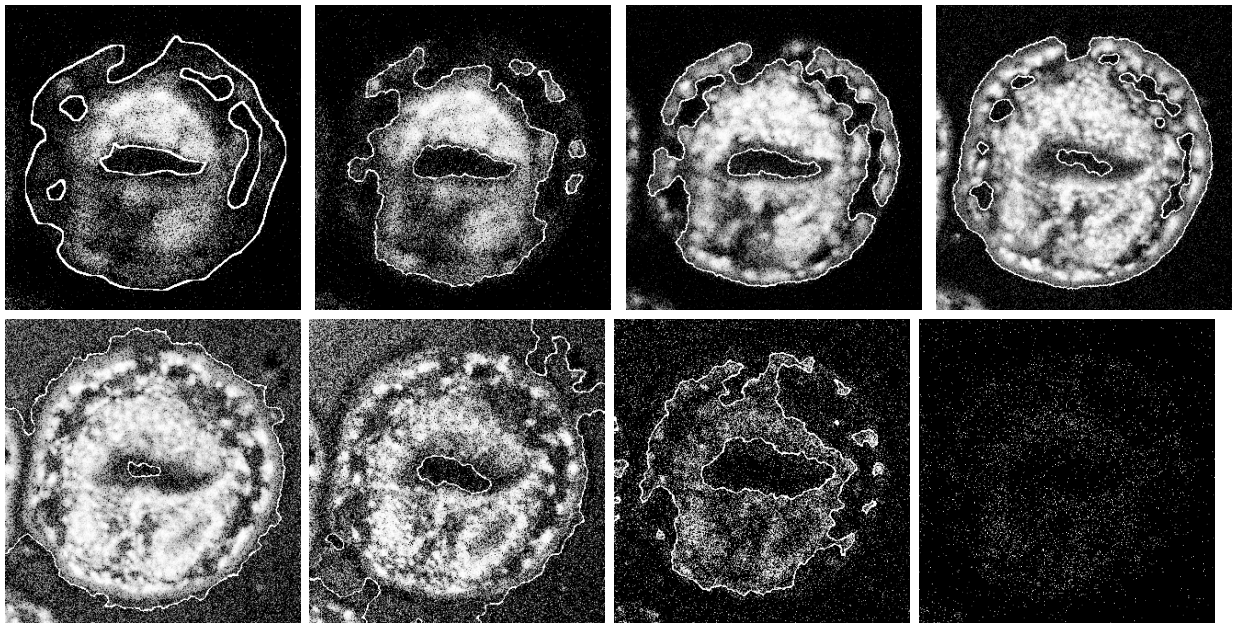


Figure 5.17: *Echinococcus* (classic): frames 14,15:5:45

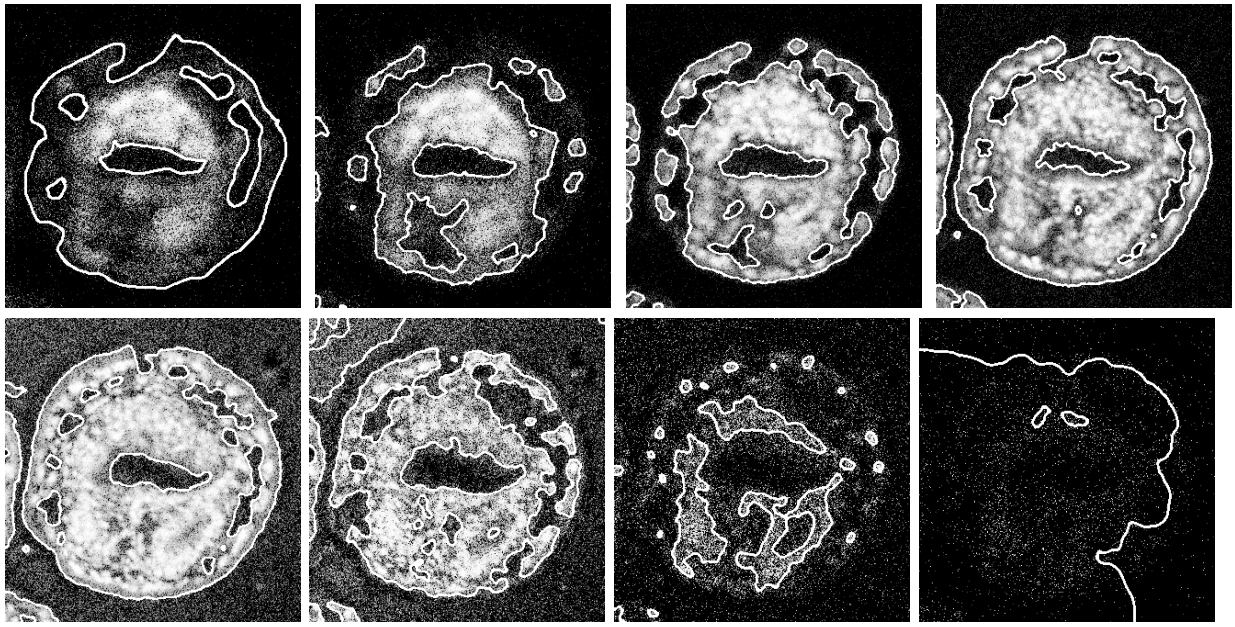


Figure 5.18: *Echinococcus* (w/ reinit and fixed weights w_2): frames 14,15:5:45

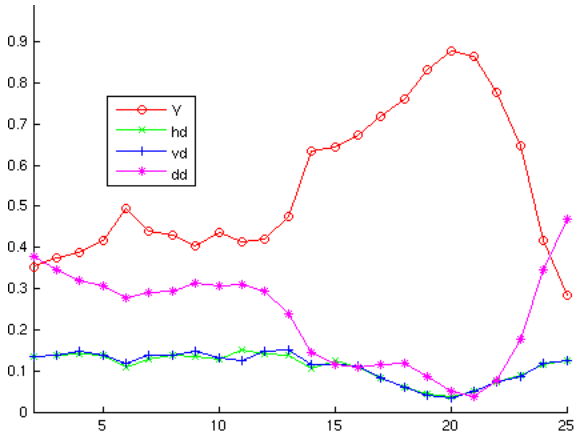


Figure 5.19: *Echinococcus*: evolution of the optimal weights

5.3.3 Spider

Figure 5.20 shows the results on the *spider* sequence. Once initially marked by the biologist, the objects are successfully tracked, despite the important topology changes taking place over the 50 frames of the sequence.

However if a similar color structure appears close to the boundary, the algorithm detects it as part of the object (e.g. frames 30 and 50).

Figure 5.21 illustrates the evolution of the computed parameters w_j for this sequence. Some features become more relevant (for the segmentation process) in different frames, compare for instance the weights at frames 10 and 30. In all the tested sequences the weighting mechanism outperforms the fixed-weight approach, even if we choose manually the channels that contain relevant information. In addition, the computed weights are close to the manual choice. However, their variability over the sequence shows the importance of using an adaptive strategy rather than a fixed approach.

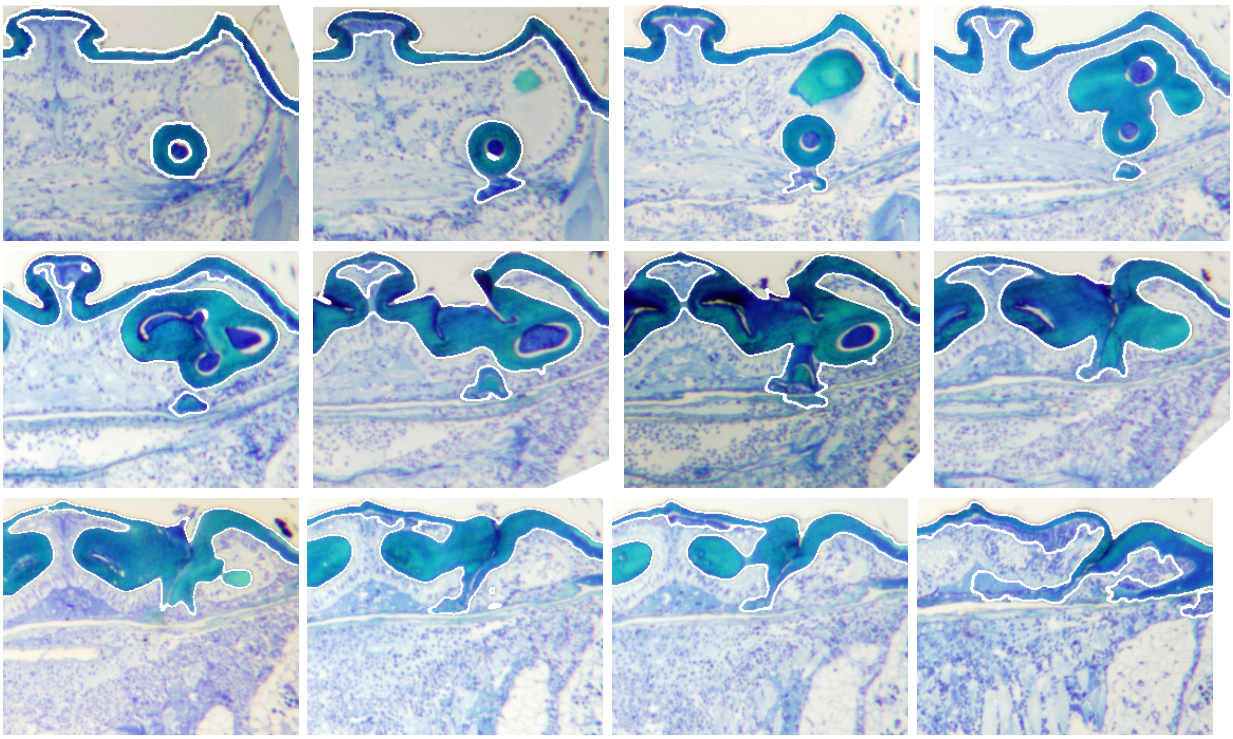


Figure 5.20: *Spider*: Results of segmentation with automatic weighting and (Y, Cb, Cr) colorspace (frames 1,2,6:5:46,52)

In order to compare the effects of different color spaces, we run using the (R, G, B) colorspace. The results obtained in this case are depicted in figure 5.22. These results are quite similar to the (Y, Cb, Cr) case. However, there are many frames where the front get trapped in a worst local minimum that the (Y, Cb, Cr). This can be seen in frames 16 and 21 when the front is unable to detect the cavity enclosed in the membrame in the upper right corner. In frame 36 it almost lost the small object in the right hand side of the image.

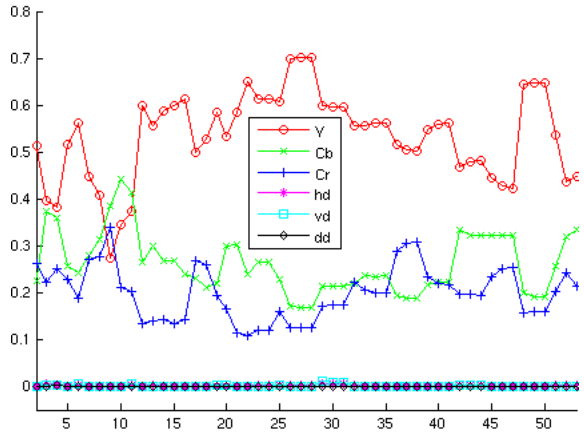


Figure 5.21: *Spider*: evolution of the optimal weights

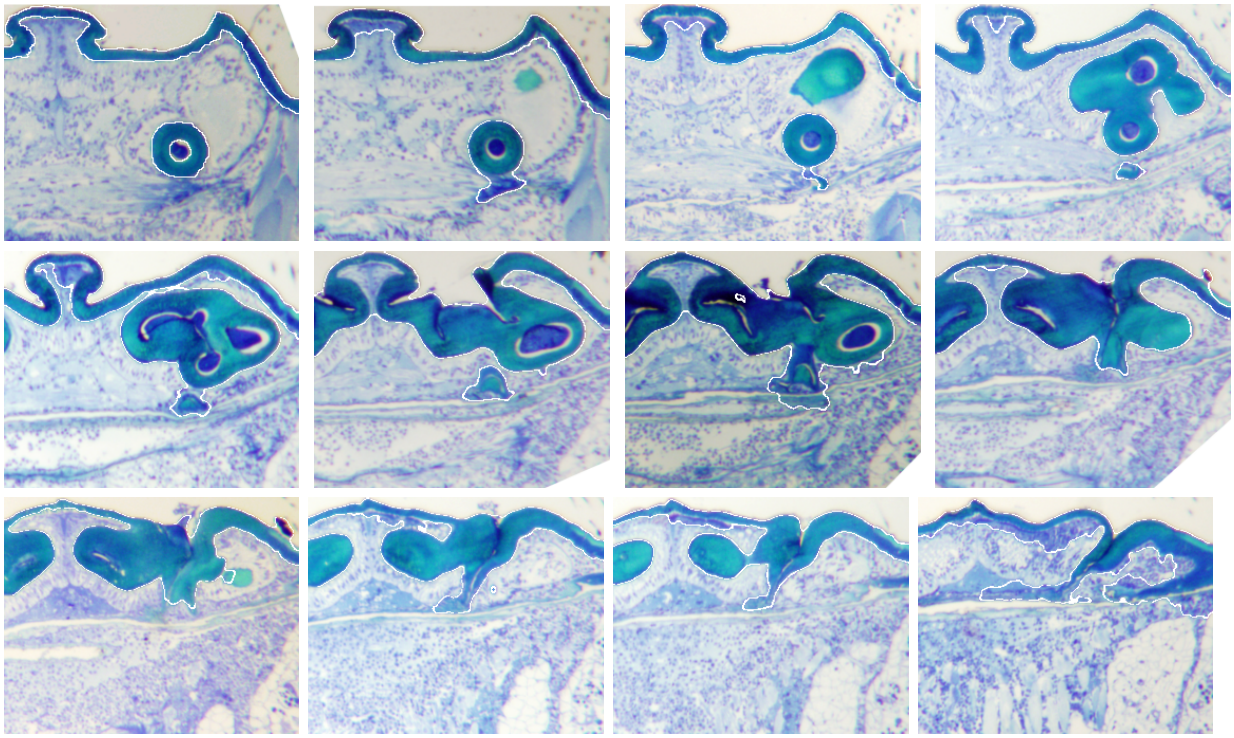


Figure 5.22: *Spider*: Results of segmentation with automatic weighting and (R, G, B) colorspace (frames 1,2,6:5:46,52)

5.3.4 Crypt

Figure 5.24 shows the results on the *crypt* sequence. Here, the interesting structure is the homogeneous region which forms a cavity that must be reconstructed in order to analyze its shape. The images are built with a collage of several OM images and some mosaic artifacts can be seen. In addition there also are strong color and contrast changes, which makes this sequence a very difficult one. However, the obtained results are quite good. The structures are well segmented in general but in some frames a couple of regions are lost. In addition, in some frames the curve encloses some of the artifacts (see frame 6).

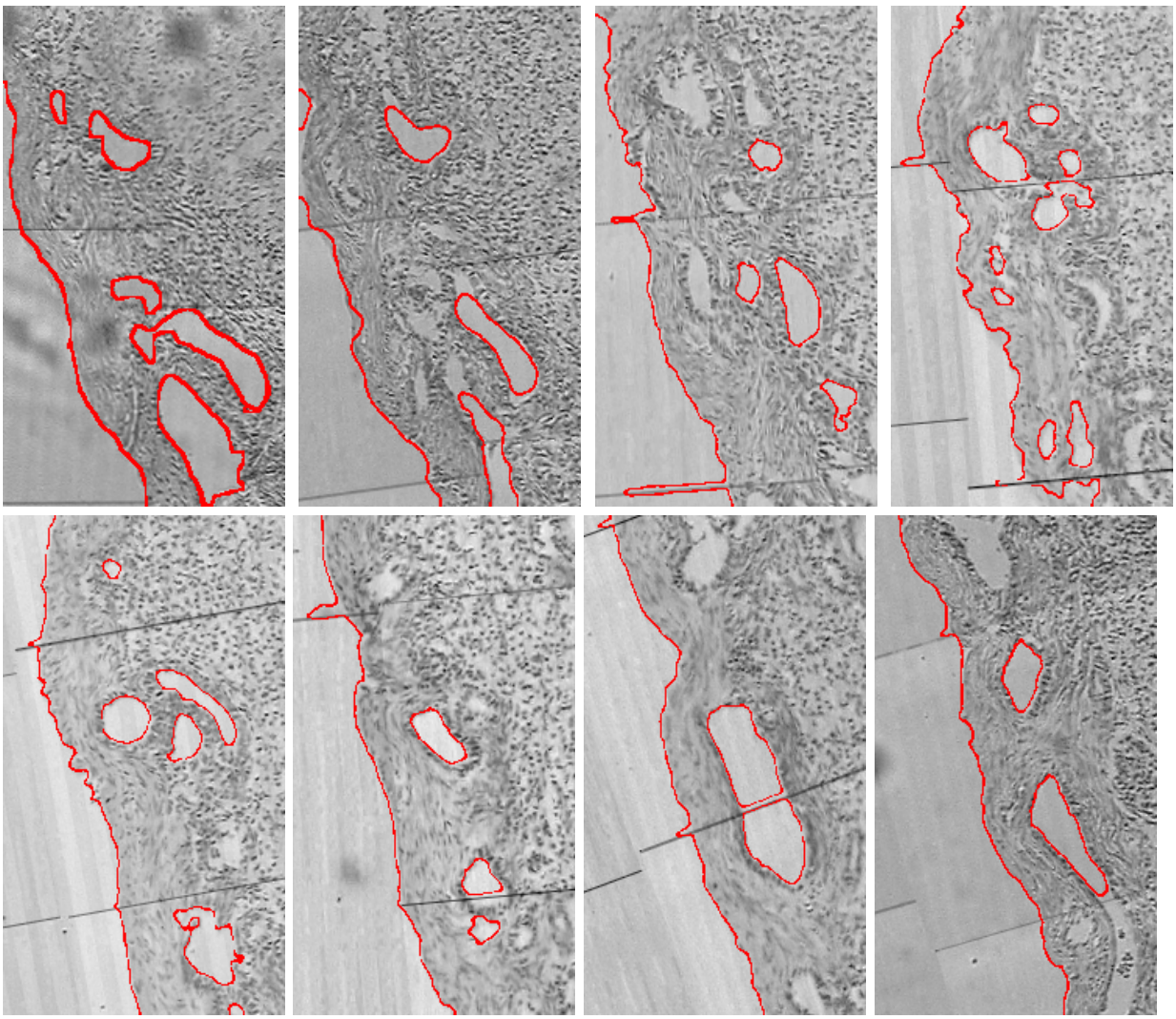


Figure 5.23: *Crypt*: Segmentation results over frames 1,2,6:5:46,52

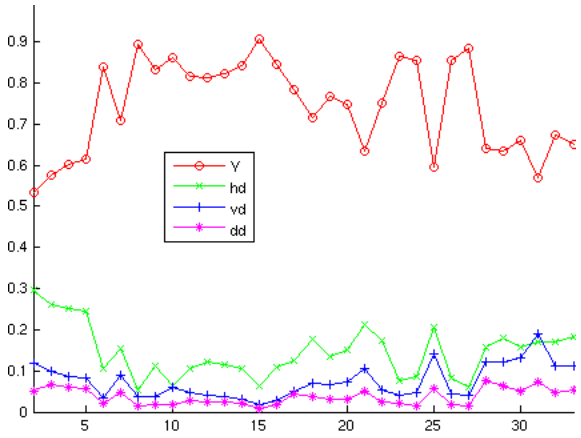


Figure 5.24: *Crypt*: evolution of the optimal weights

5.4 3D Reconstruction

In this section we present the 3D reconstruction of each of the biological sequences presented. All this reconstructions were performed with the software BioVis3D[11] [7]. In some cases some selected slices of the sequence are superimposed with the reconstruction.

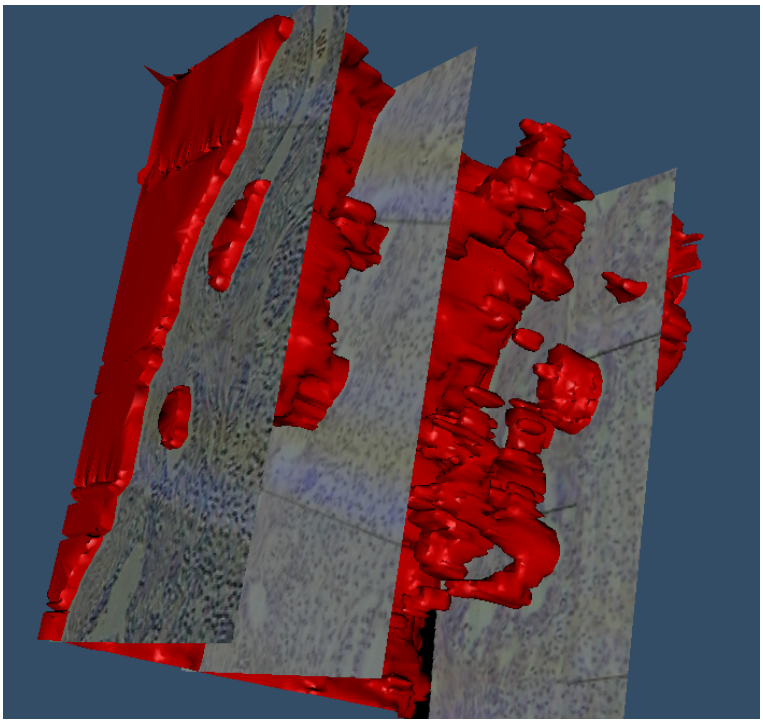


Figure 5.25: *Crypt*: 3D reconstruction

[11] This software is experimental, and in some cases in introduced some artifacts in the reconstructed volume, as seen in figure 5.26 where many horizontal stripes appear

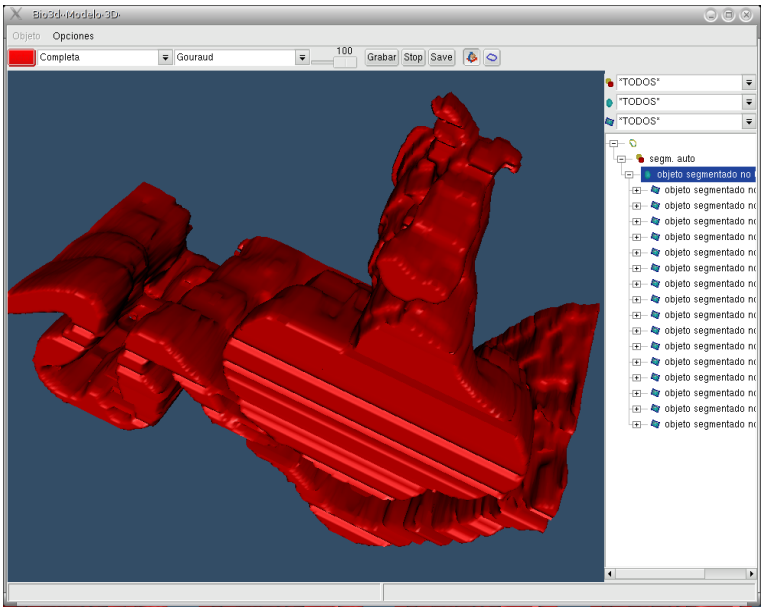


Figure 5.26: *Spider*: 3D reconstruction

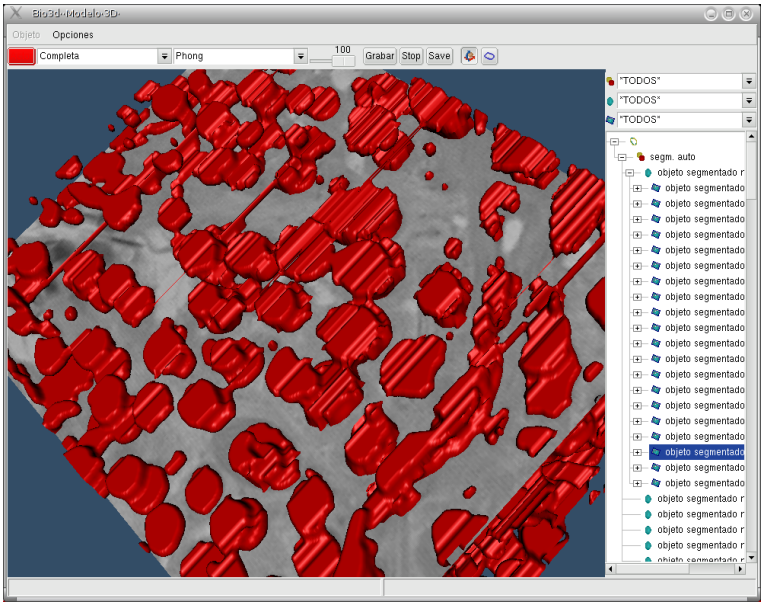


Figure 5.27: *Ram*: 3D reconstruction

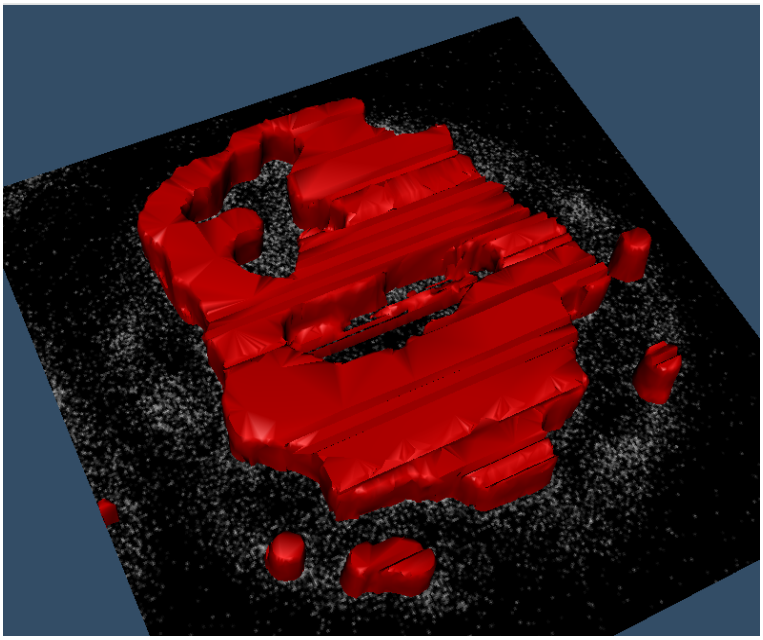
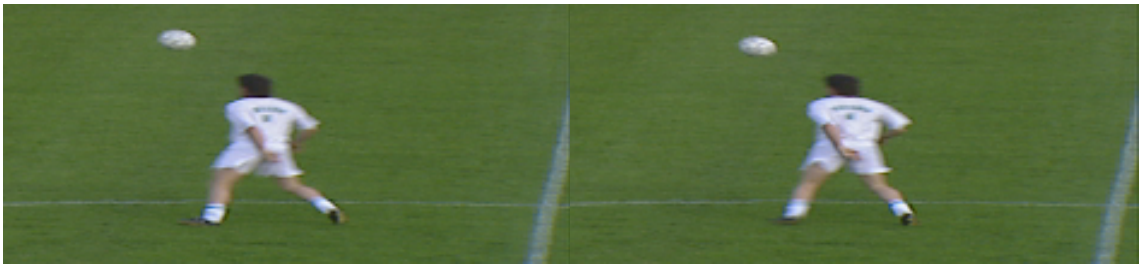


Figure 5.28: *Echinococcus*: 3D reconstruction

5.5 Tracking of video sequences

5.5.0.1 Data sets

The *Foot* sequence is 236x108x70 RGB series[12] where the goal is to follow the football player. The average displacement between consecutive frames is approximately 5 pixels. Here the relevant features are color and movement, texture does not seem to be relevant.



(a) frame 02 (b) frame 03

Figure 5.29: Two consecutive frames of the *Foot* sequence

The *Bad* sequence is 512x512x70 RGB series[13] where the goal is to track the movement of the bus. The average displacement between consecutive frames is approximately 1 pixel. Here the relevant features are grey-level and movement, texture does not seem to be relevant.

(a) frame 02 (b) frame 03

Figure 5.30: Two consecutive frames of the *Bad* sequence

The *Taxi* sequence is 256x190x20 gray-level series[14] where the goal is to track the movement of the white taxi. The average displacement between consecutive frames is

[12]This images are courtesy of Dr. R. Deriche at INRIA, France.

[13]This images were obtained from the Institut fuer Algorithmen und Kognitive Systeme at the University of Karlsruhe, Germany and can be downloaded from http://i21www.ira.uka.de/image_sequences

[14]This sequence was created at the University of Hamburg and can be downloaded from ftp://csd.uwo.ca/pub/vision

approximately 1 pixel. Here the relevant features are color and movement, texture does not seem to be relevant.

(a) frame 02 (b) frame 03

Figure 5.31: Two consecutive frames of the *Taxi* sequence

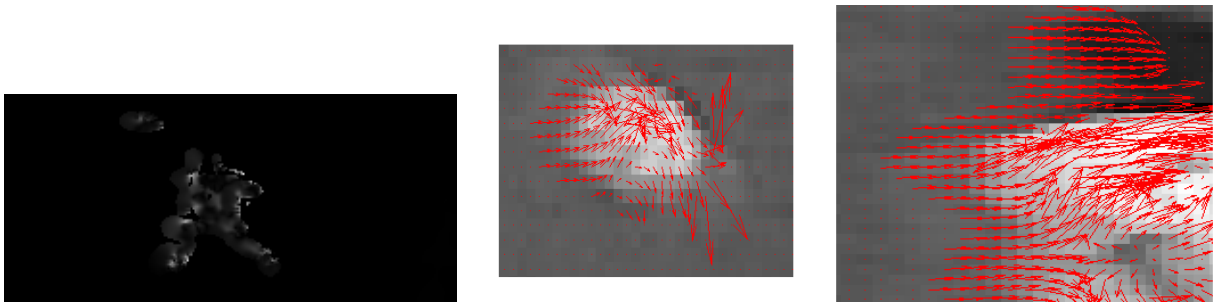
5.5.1 Optic Flow computation

In this section we will show the results of the optical flow computation. We will show these results over three sequences: *Foot*, *Bad* and *Taxi*. For the implementation, we have used standard fourth order central differences for all the spatial derivatives and first order forward derivatives for the spatial discretization. Before the optic flow computation, the sequences are convolved with a gaussian kernel of $\sigma = 0.5$. After the Structure Tensor is computed, it is anisotropically diffused.

For all the experiments we have used the same parameters: a threshold for the confidence ($\lambda_{min} > 0.2$) and a threshold for the maximum velocity ($v_{max} = 10$) in order to detect outliers.

5.5.1.1 Foot

The results over this sequences is shown in figure 5.32. As this figure shows, the results obtained are very crude, but enough for our purpose (i.e detecting which objects are moving). In general the directions computed are pretty accurated, but in the right side of the ball there are some outliers. The pure black zones in the modulus figure correspond to the points discarded using the confidence measure. In addition, the velocity values are spilled across object boundaries, this is due to combination of the diffusion process used for the computation and and the low value used for the eigenvalue threshold. If this value is raised we will get a more accurate estimation, but the flow will be sparser. This last issue is very important from the segmentation point of view, because we are interested in precisely locating the borders of the objects.

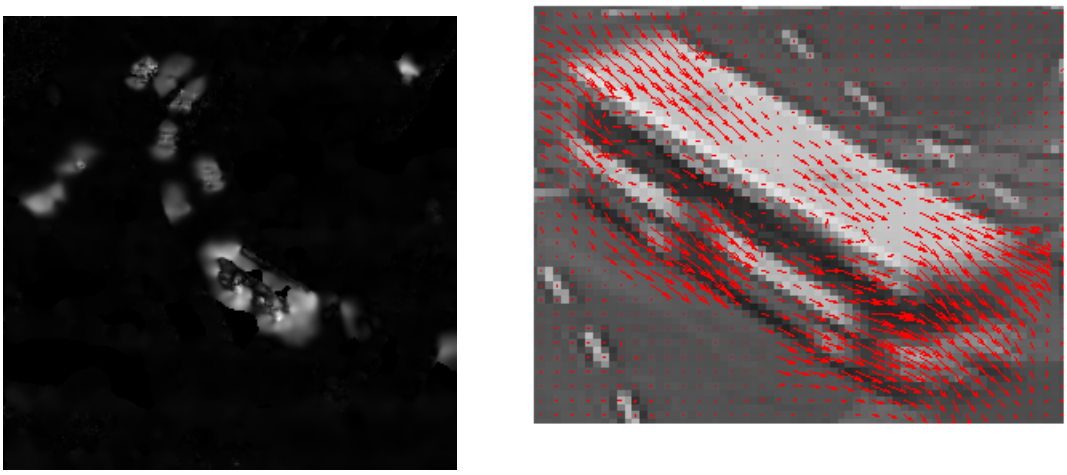


(a) (b) (c)

Figure 5.32: *Foot*: Results of the optic flow computation. (a) Modulus of the velocity, (b) detail of the computation near the ball, (c) detaul of the computation near player's shoulder

5.5.1.2 Bad

In this sequence, the results present the same behaviour as in the case of the football player. However, in this case there are less outliers points and the directions seem more accurate. As it happens in all these sequences, in the flat zones ($\nabla I \approx 0$) the resulting estimation has a small confidence value and thus is discarded. See for instance the zero values on the roof of the bus in the detailed figure.

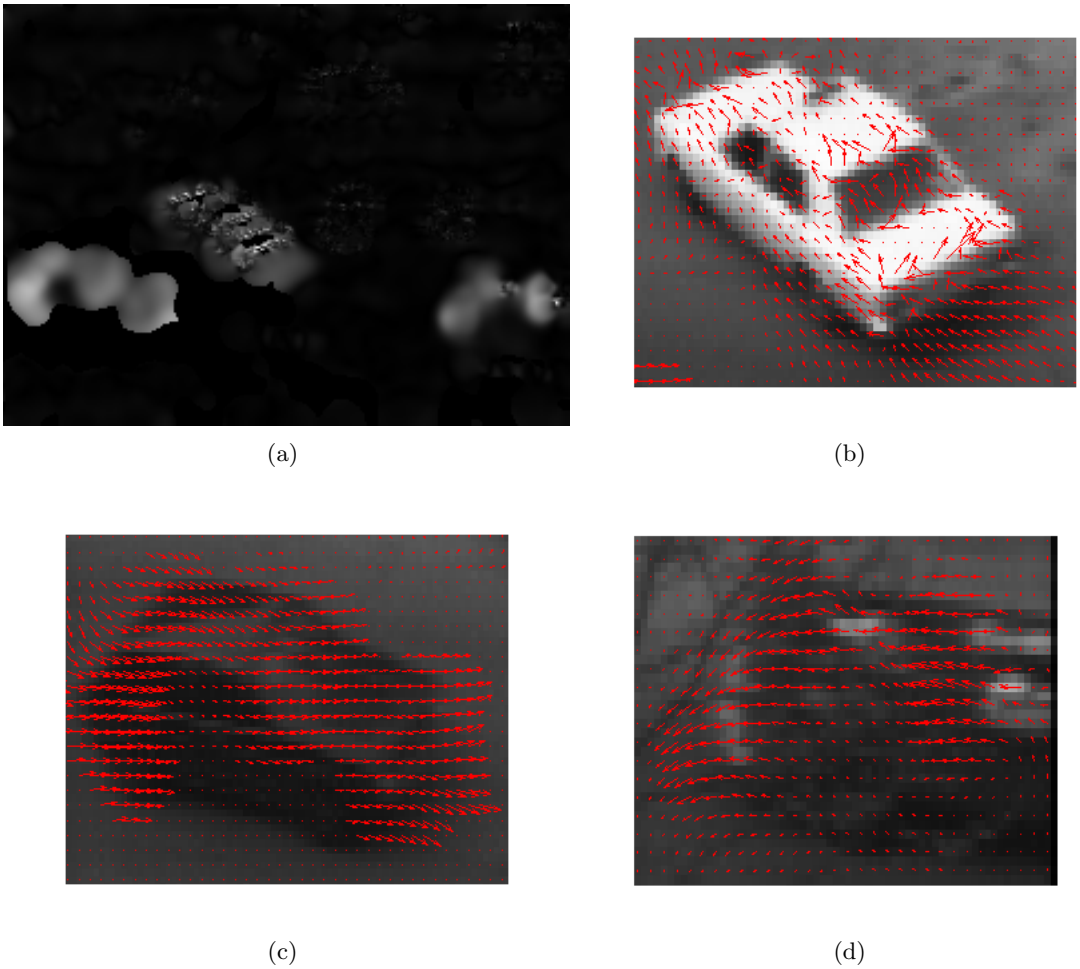


(a) (b)

Figure 5.33: ***Bad*: Results of the optic flow computation. (a) Modulus of the velocity, (b) detail of the computation over the bus**

5.5.1.3 Taxi

All the discussion presented for the previous sequences applies to this sequence. In addition, in this case, the spill of the velocity field is more evident than in the other sequences.



(a) (b)

(c) (d)

Figure 5.34: *Taxi*: Results of the optic flow computation. (a)

5.5.2 Foot

In figure 5.35 the results of tracking over the *foot* sequence are shown. The player is initially selected in the first frame, and then is succesfully tracked along the sequence. However, after frame 20, the algorithm includes the white line in the segmentation. This is due to the similarity of color between the line and the player.

One possible solution to this problem, is to assign more weigth to the motion channels because the line has no movement at all. Results using this approach are show in figure 5.36. As it can be seen, in most frames the front includes a smaller part of the line.

Finally, if we look at the computed weights (figure 5.37), we can see that all channels get approximately the same weight. Thus the velocity channel gets the third part of the total weight, in comparison with the color channels. This supports the previously given arguments about the inclusion of the line in the segmentation.

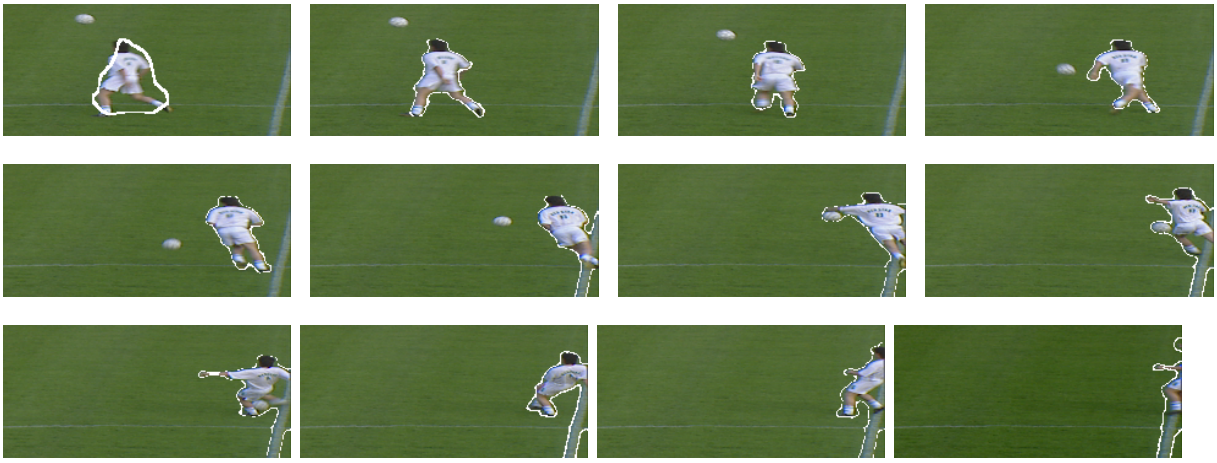


Figure 5.35: *Foot*: Results of tracking a football player with automatic weighting (frames 0,1,5:5:50).

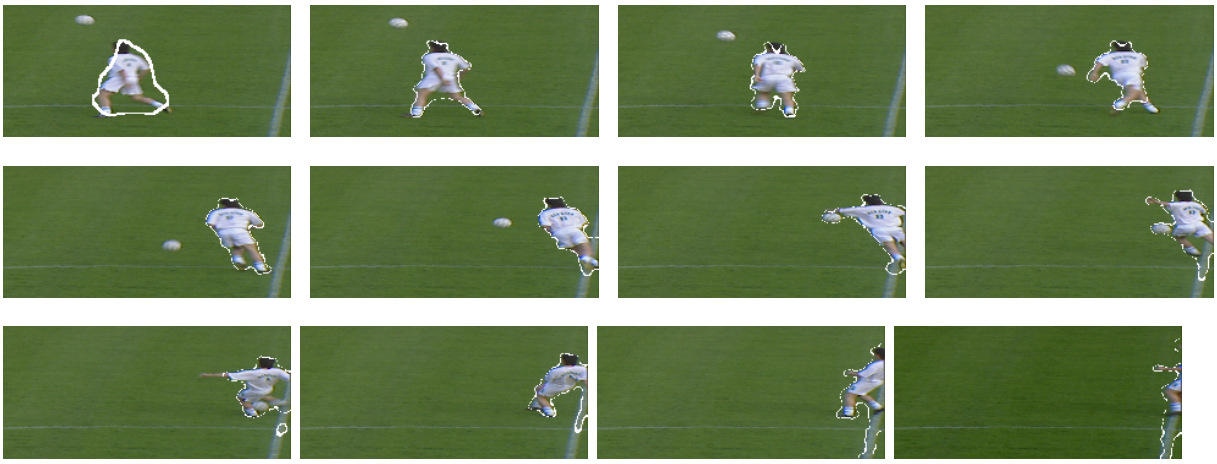


Figure 5.36: *Foot*: Results of tracking a football player with fixed weights and with more emphasis on the motion channels (frames 0,1,5:5:50).

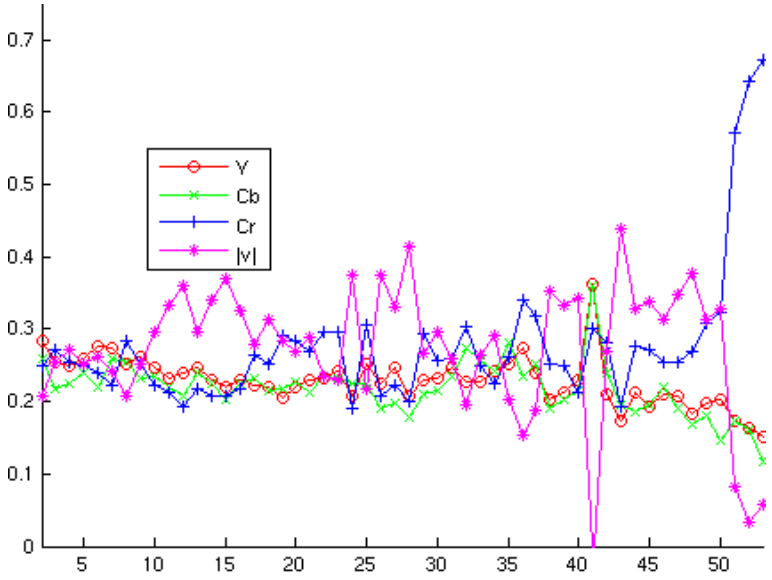


Figure 5.37: *Foot*: evolution of the optimal weights.

5.5.3 Bad

In figure 5.38 the results over the *Bad* sequence are shown. In this case, we are using only color (gray-level) channels to drive the segmentation. In addition, the model used for the PDF is gaussian. As the gray level in the bus is not uniform, the algorithm attachs only to the roof of the bus. Also, in frame 15 when the bus meets the column, the algorithm starts losing the roof.

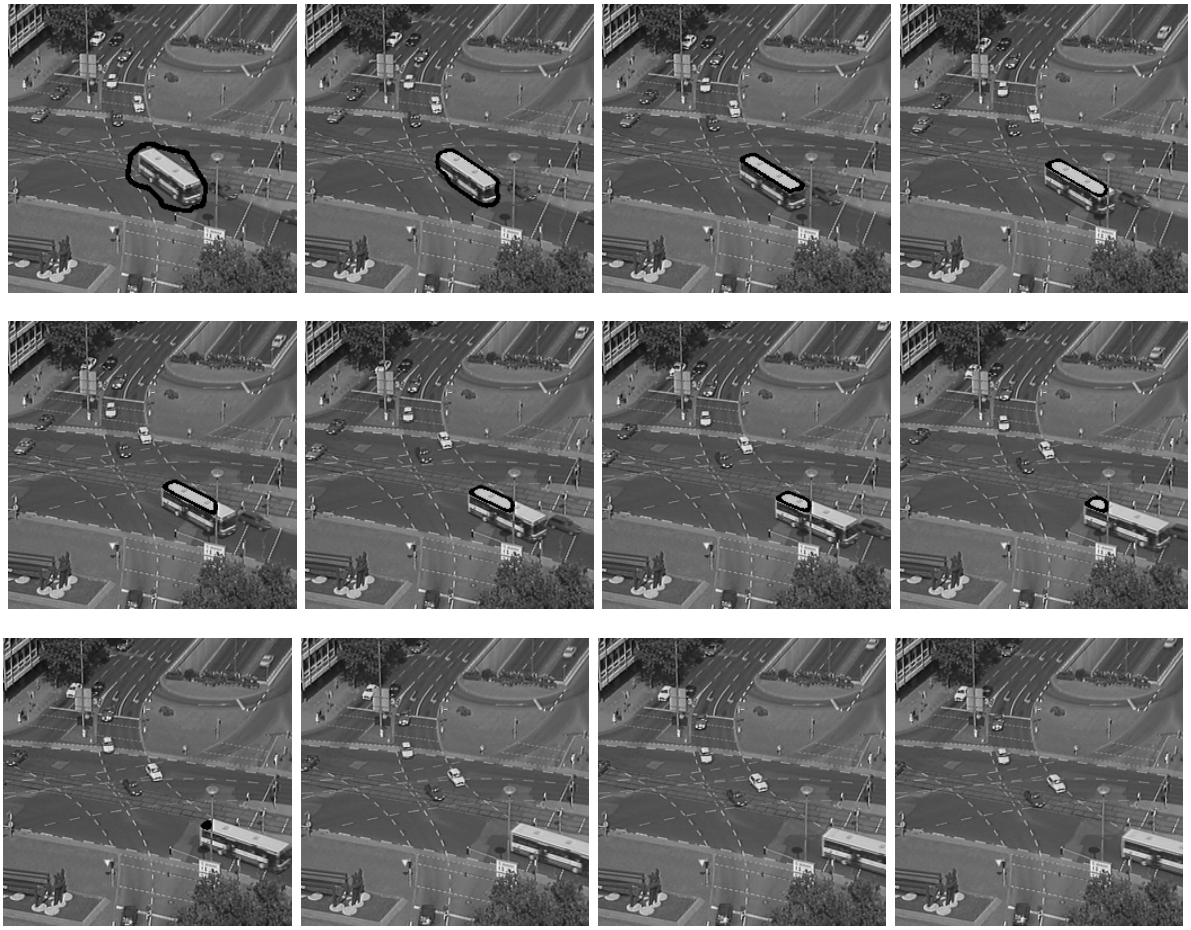


Figure 5.38: *Bad*: Results of tracking a bus with fixed weighting (frames 0,1,5:5:45,46) and using only gray level channel.

If we add the motion channels, the results are much better (see figure 5.39), and the front is able to track the bus all over the sequence. However, due to the inaccuracy of the optic flow computation (recall the *spill* phenomenon from previous sections) and the blurring introduced by the anisotropic diffusion, the front also includes the car at the right of the bus. This can be corrected reducing the amount of diffusion in both stages: optic flow computation and feature smoothing.

Figure 5.39: *Bad*: Results of tracking a bus with equally fixed weighting (frames 0,1,5:5:45,46) using gray level and modulus of the optic flow.

Chapter 6

Conclusions

In this work we studied the problem of segmentation of biological sequences. We reproduced and tested a state of the art family of segmentation algorithms over real biological sequences. We have identified the main problems of these algorithms and proposed solutions for some of them. We also studied what features are suitable for this kind of sequences and how to select them. Finally, we tested the resulting algorithm obtaining remarkable results over these sequences. Our approach includes all the benefits from the original formulation of Region Based Active Contours: It is a variational formulation, which aims to minimize an energy depending on boundary and region information. This region information allows the integration of multiple cues into the segmentation. A (local) minimum of this energy is found by a gradient descent technique, which is implemented using the level-set paradigm.

In addition, we improved the selection of relevant features by introducing an adaptive algorithm which automatically assigns weights to each channel. We also added a detection-reinitialization algorithm which allows the recovery of lost objects, and thus avoiding some often encountered local minima. And finally, the third contribution of this work is the exhaustive testing of the algorithms over a wide range of real sequences.

Although the presented approach proved to be successful over these sequences, it still has many drawbacks in addition to those presented in chapter 2:

- As it is an iterative technique, the processing of an image under this formulation is a time consuming task. However, in the last years, many improvements have been proposed which greatly reduce the computational cost.
- Although the results are greatly improved by the introduction of anisotropic diffusion filtering, these results are strongly dependent on the amount of diffusion performed.

The abovementioned drawbacks depend on the original formulation, but there are also aspects in which we can improve our particular implementation:

- The analysis of the results was manually performed comparing the output given by the biologist with the results of the algorithm. It would be interesting to evaluate the algorithm over a broader database of sequences, and also to perform a quantitative evaluation of the results. We have performed this quantitative analysis only in the particular case of the *Ram* sequence.
- We tested our proposal using two region segmentation, this is clearly a strong restriction and it is obvious that in many real cases the two-region hypothesis does not hold.
- Although we improved the results avoiding local minima corresponding to a particular case (front unable to split across a region), there are many situations in which other local minima appear which are not depending on that case.
- Once the features to be used are selected, the weighting algorithm automatically estimates the weights for each channel. However, the features to be used are manually selected. This is not a problem in the case of color and motion, but this choice is not that clear when dealing with texture. For each kind of texture we don't know in advance which level of decomposition is needed and which sub-band must be further

decomposed. Thus the manual selection of wavelet channels is not a simple task and requires further analysis.

- The weight assignment is memoryless: as we use the segmentation from the previous frame as the ground truth to estimate the weights, we take the risk of being confused by an *outlier* frame.

- In our implementation, we were focused on the results of the segmentation, and we payed less attention to the computational effort. This makes our implementation pretty slow.

  This issue can be addressed in various ways: in the first place, higher order, efficient numerical methods can be applied. In the second place, we can implement our evolution using the narrow band approach, which removes the extra dimension introduced by the level-set formulation.

6.1 Improvements & Future work

There are many directions in which the proposed technique could be improved

- In order to tackle the two region restriction, a multiphase approach must be introduced, as shown in section 2.3.2.2. This will introduce additional problems that will be interesting to study. These problems are associated with the computation of the optimal weights and the reinitialization algorithm.

  In the case of re-initialization it is not obvious how to construct the front that will evolve outwards. Indeed, it is not clear what *outwards* means.

  With respect to the weights, we need to compute the separability between M regions. This can be achieved by computing a distance matrix with the distances between pairs of regions, and compute a measure of separability between the different clusters of data.

- When the minimization of the energy functional is performed, some terms are neglected. In [34] the authors remark the importance of these terms and the impact on their results. Thus, it would be interesting to introduce these terms in the evolution.

- As mentioned before this approach lacks a rigorous quantitative analysis, and it would be also interesting to introduce a way to compare two different segmentations. At the moment the only objective comparison criterion is the energy associated to the evolving curve. This is an useful measure, but there are some cases when it is not representative and two very different curves get the same energy value. So another way to compare curves is required. This can be achieved by introducing a notion of distance (in some suitable space) between curves.

- A great improvement on this technique would be the introduction of shape priors into the functional. This could be achieved by using an approach like the one presented in [32].

- With respect to the weighting algorithm, it could be improved by adding some kind of inertia in the update, e.g. by computing the weights as an average between many previous frames.

Appendix A

Complete results

A.1 Ram

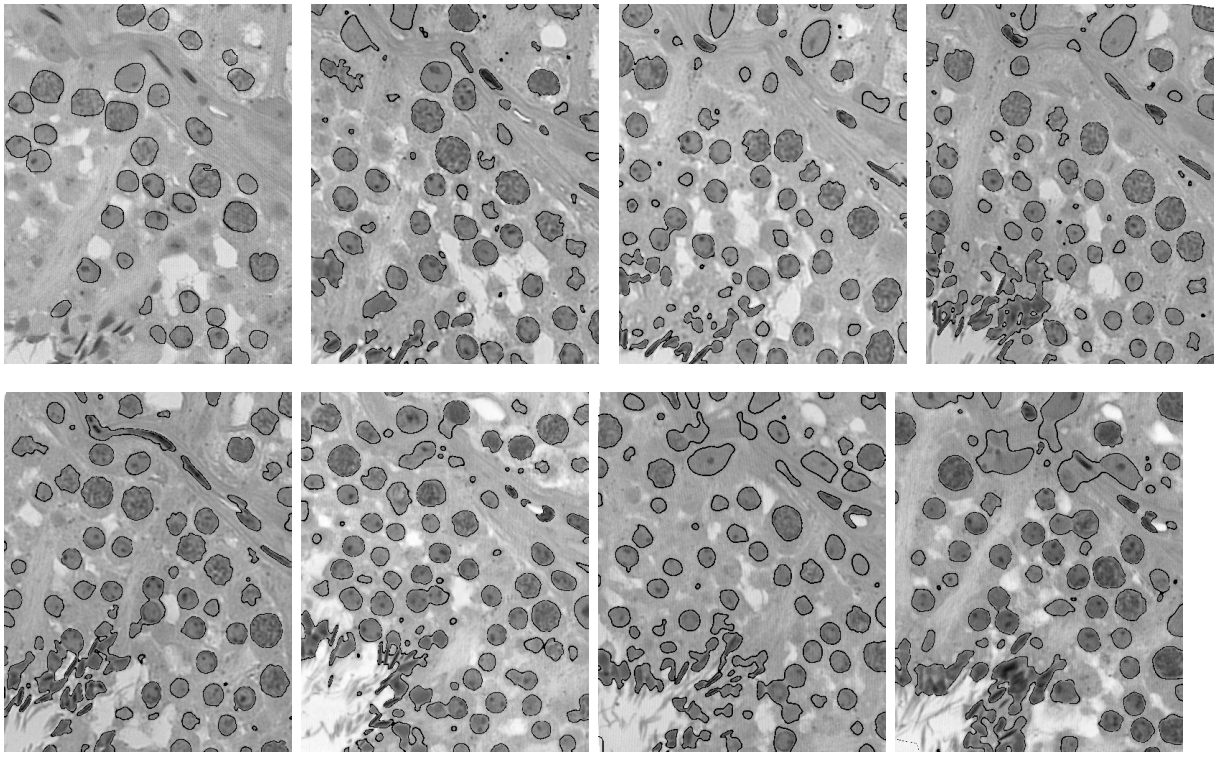


Figure A.1: Results over the *Ram* sequence: frames 0-7

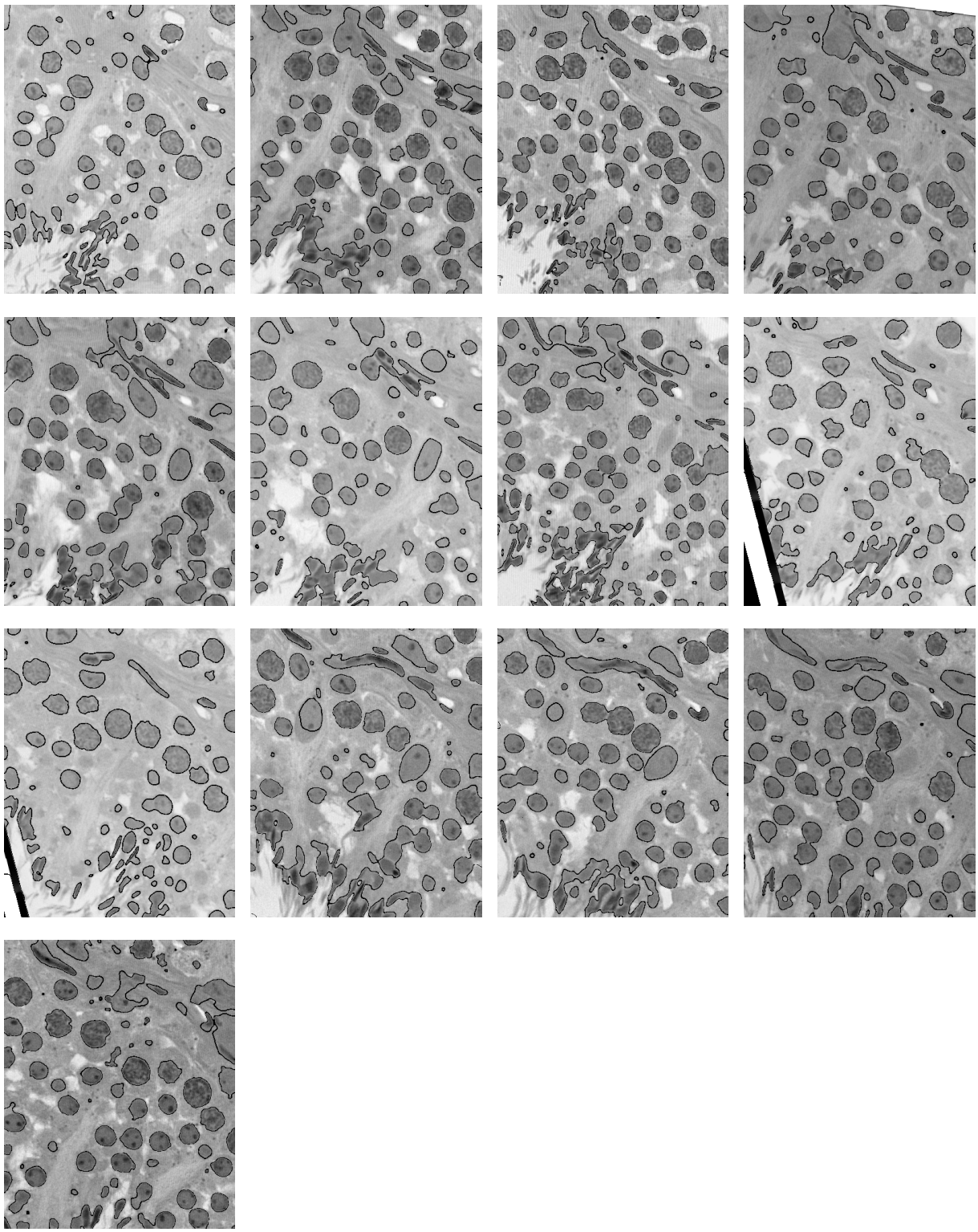


Figure A.2: Results over the *Ram* sequence: frames 8-20

A.2 Crypt

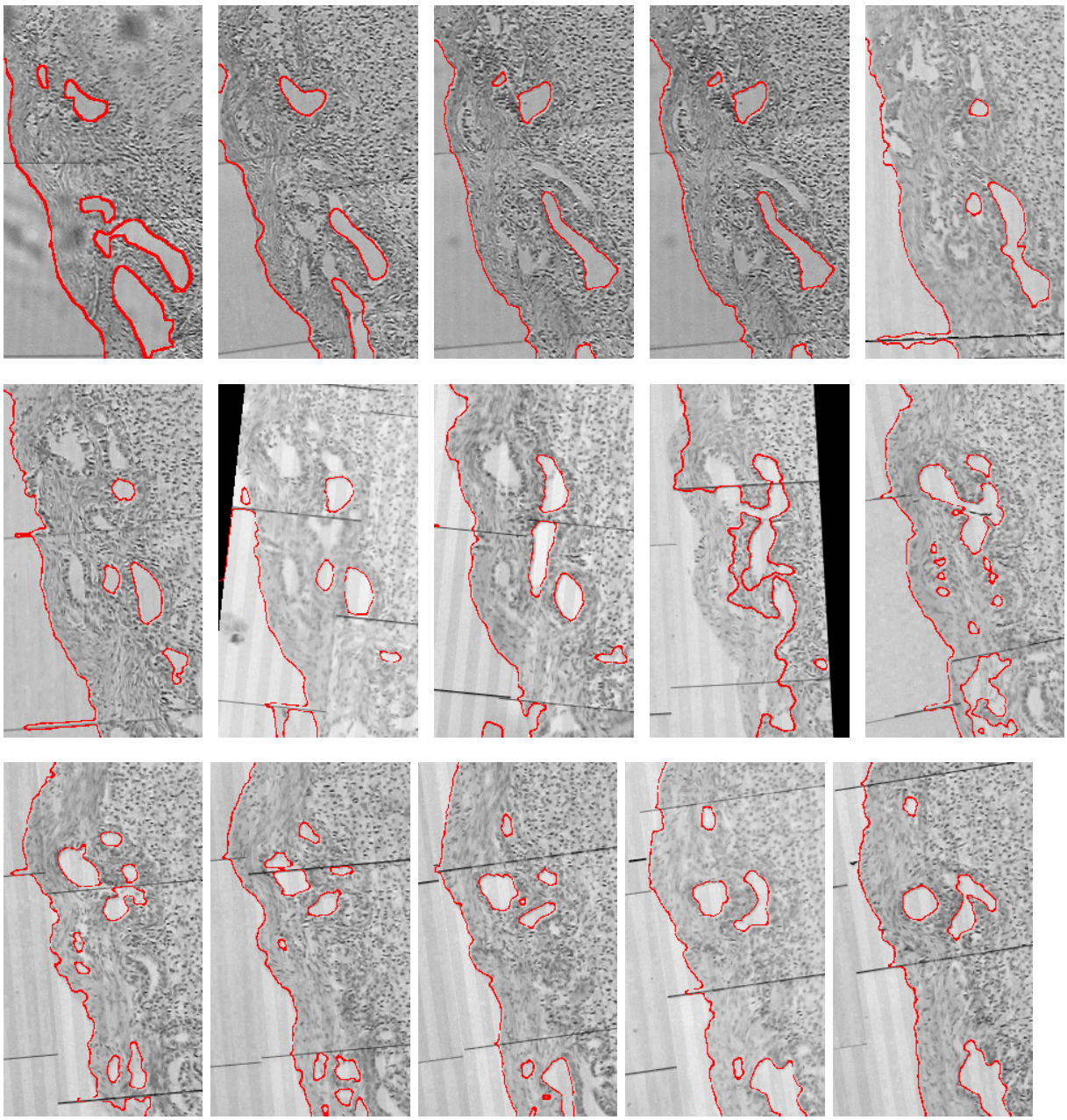


Figure A.3: *Crypt*: frames 0-14

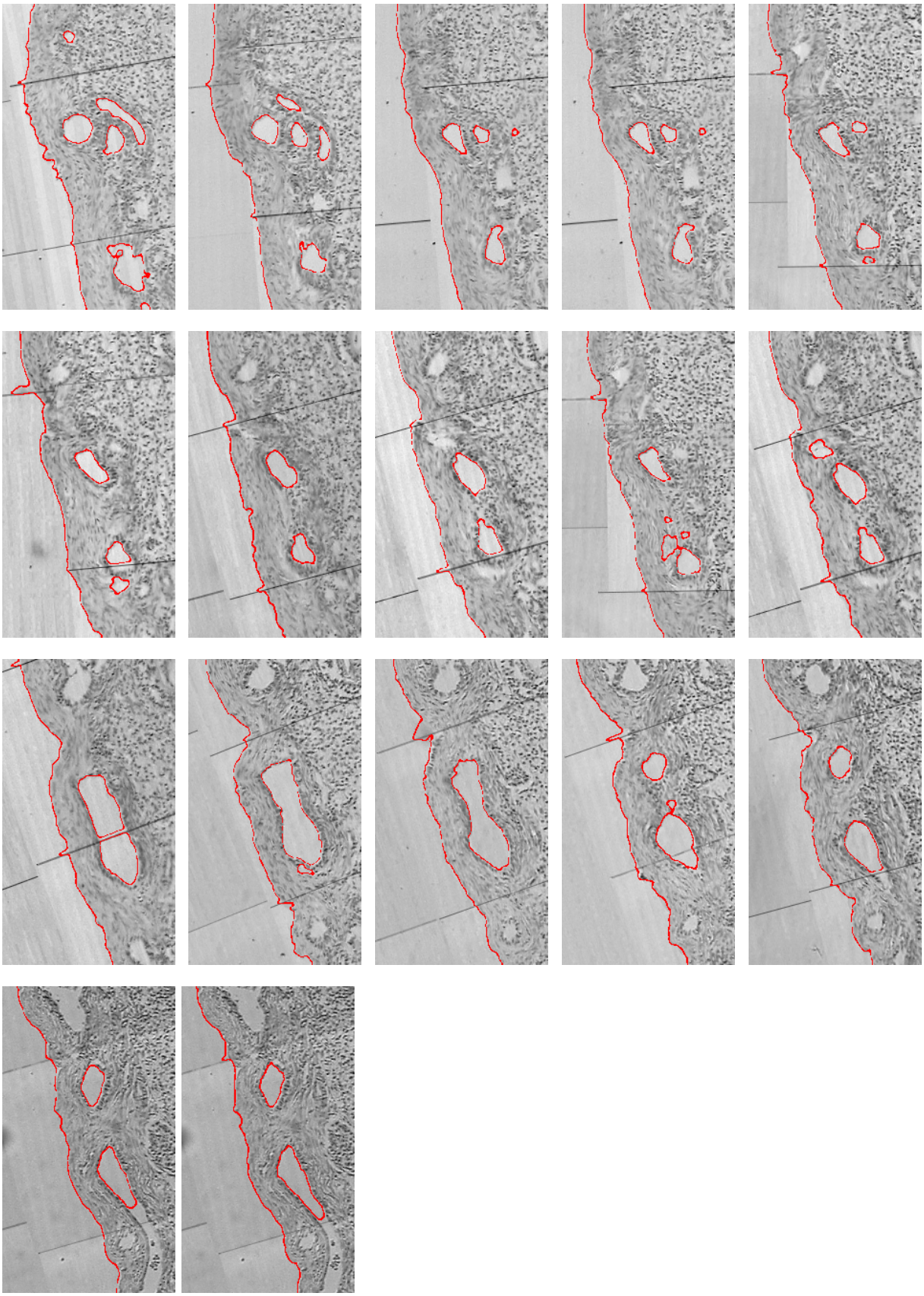


Figure A.4: *Crypt*: frames 15-31

A.3 Echinococcus

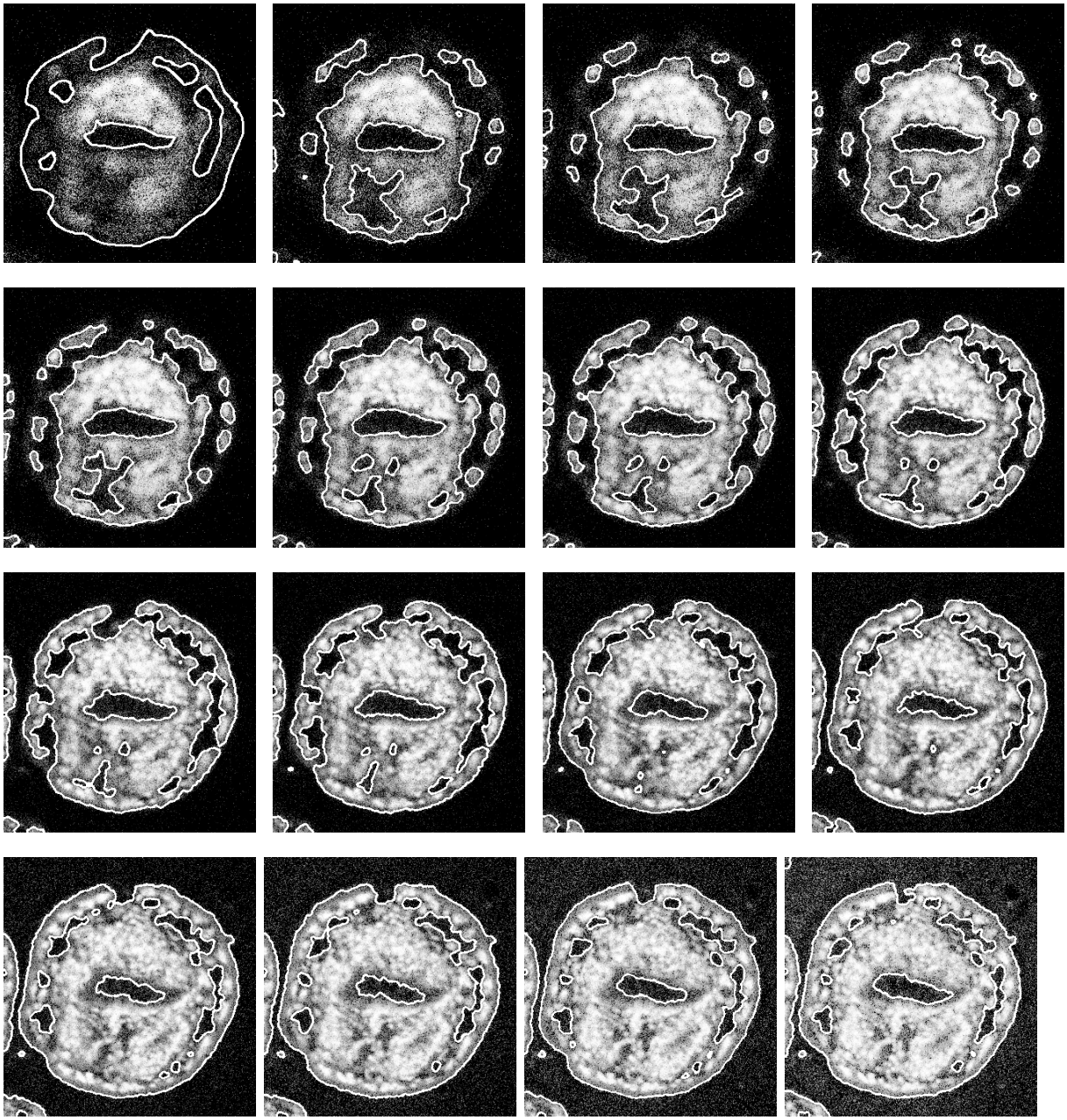


Figure A.5: *Echinococcus*: frames 14-29

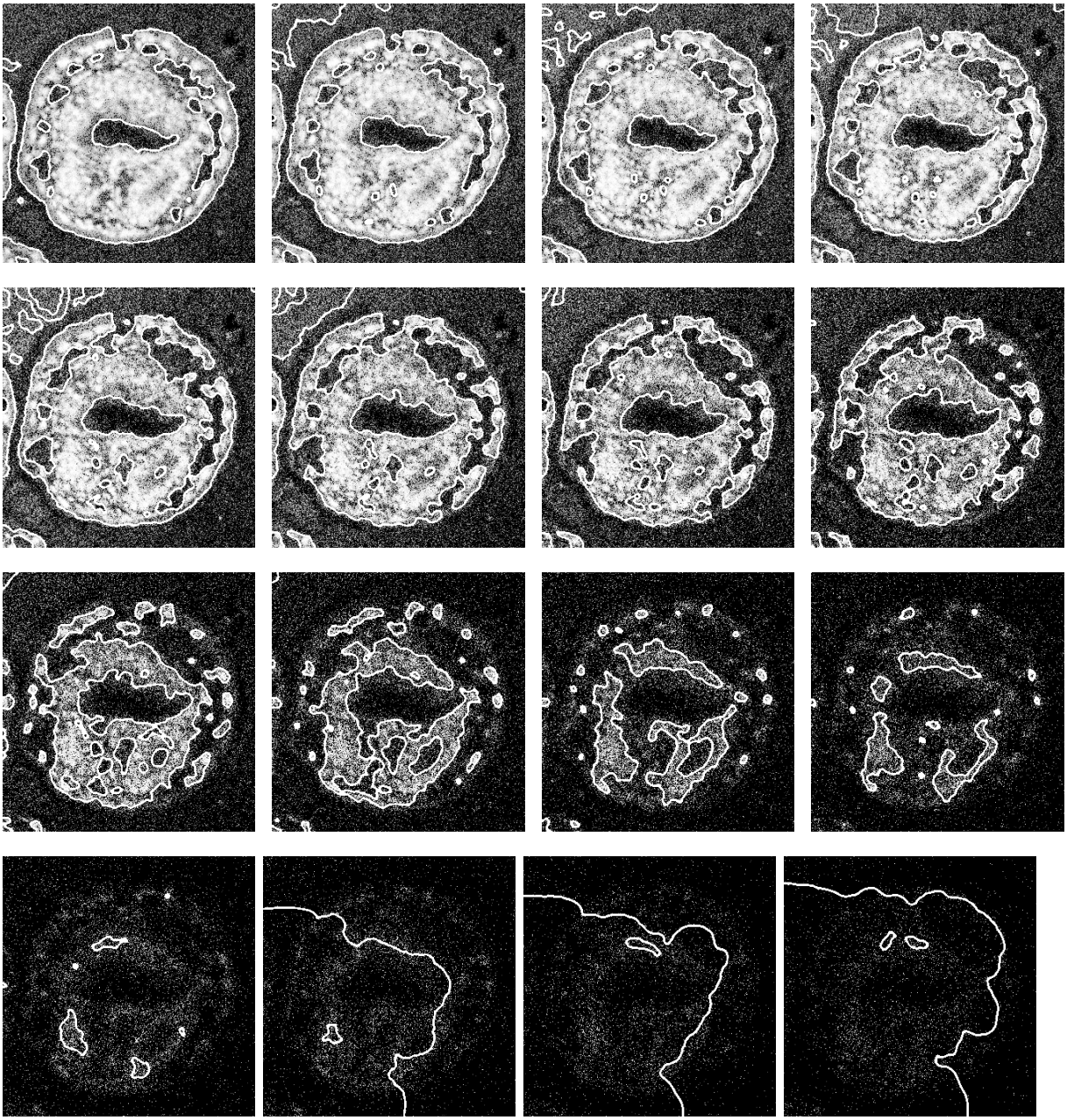


Figure A.6: *Echinococcus*: frames 30-45

A.4 Spider

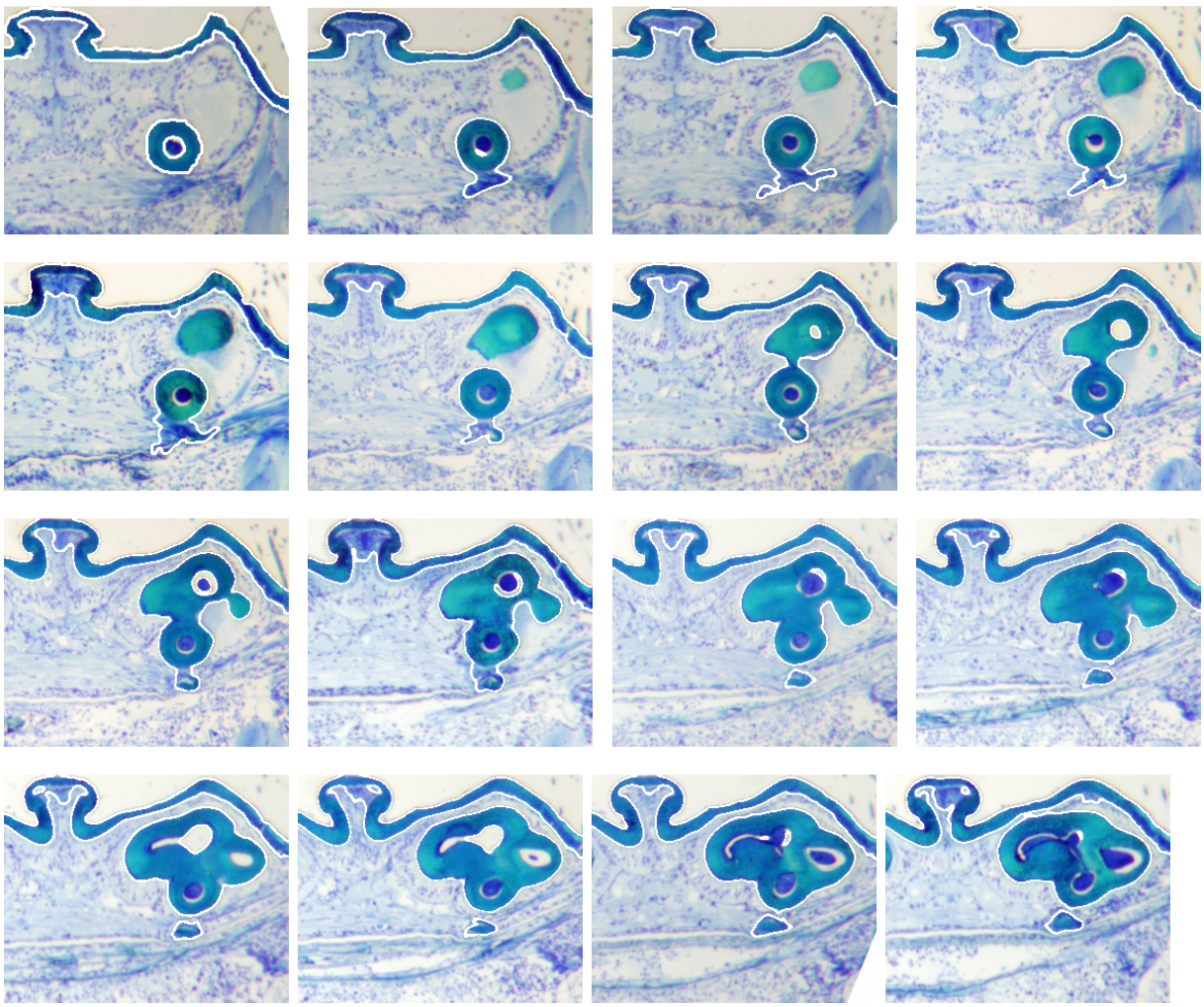


Figure A.7: *spider*: frames 1-16

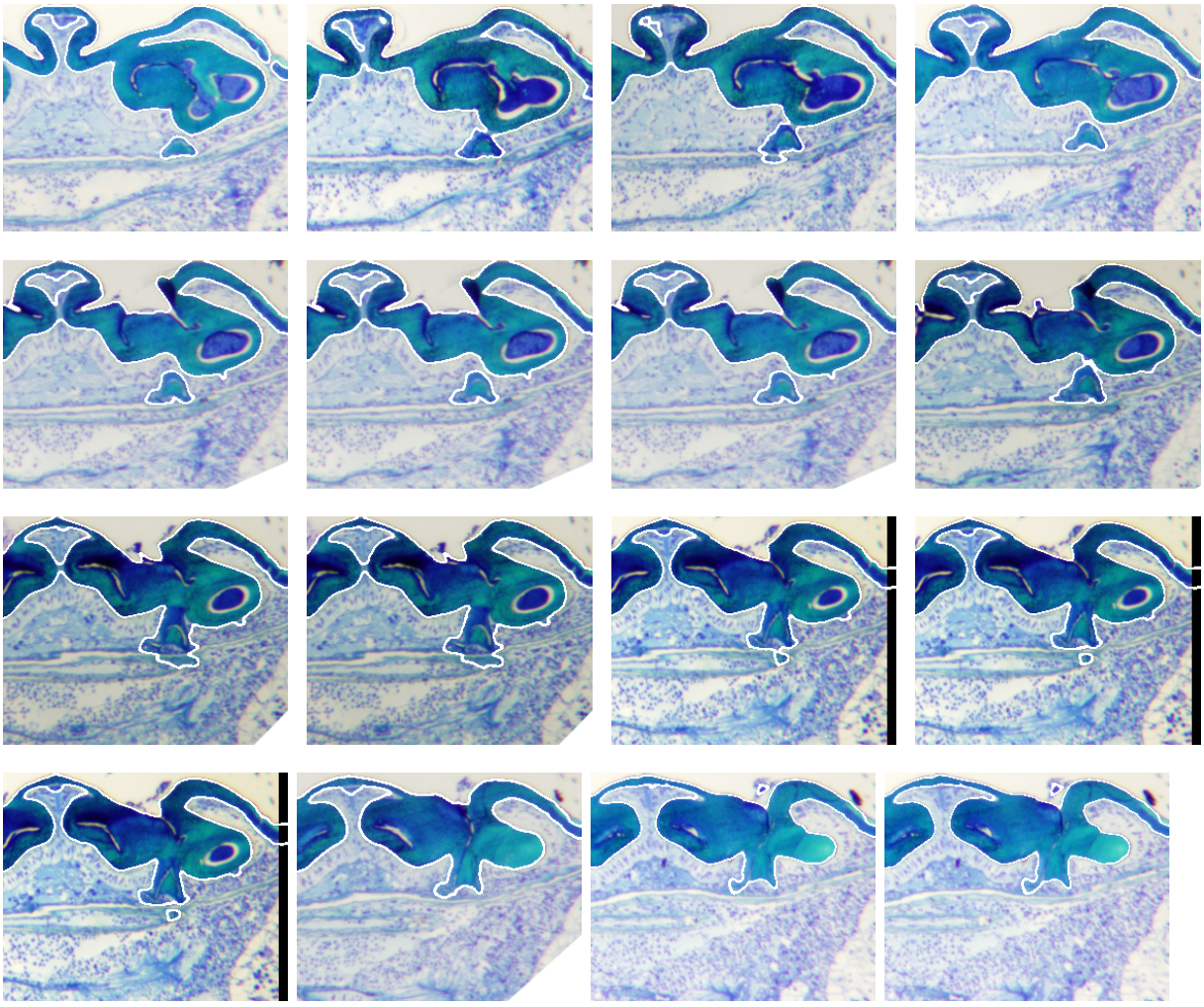


Figure A.8: *spider*: frames 17-33

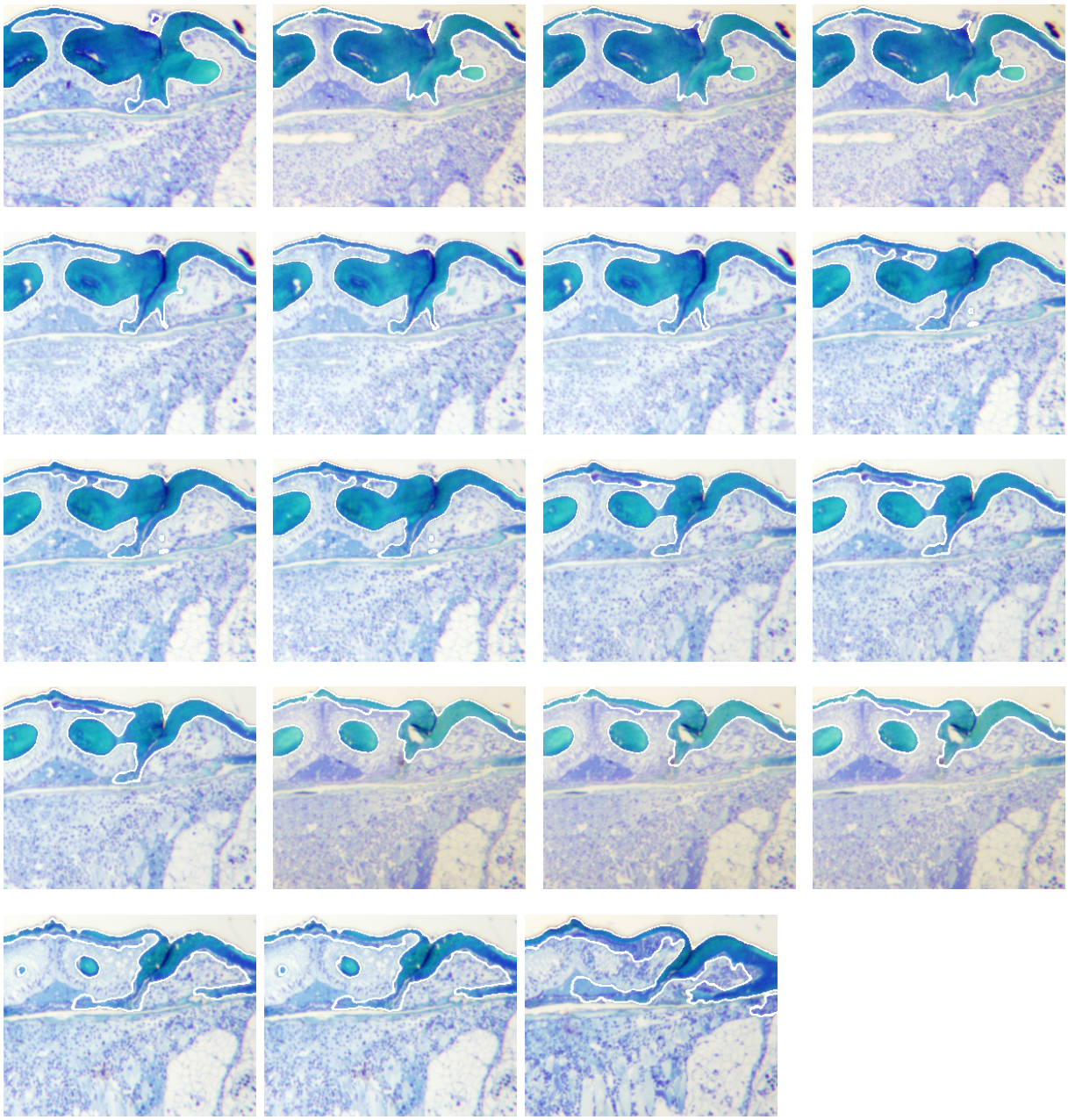


Figure A.9: *spider*: frames 34-52

A.5 Foot

Figure A.10: *Foot*: frames 1-24

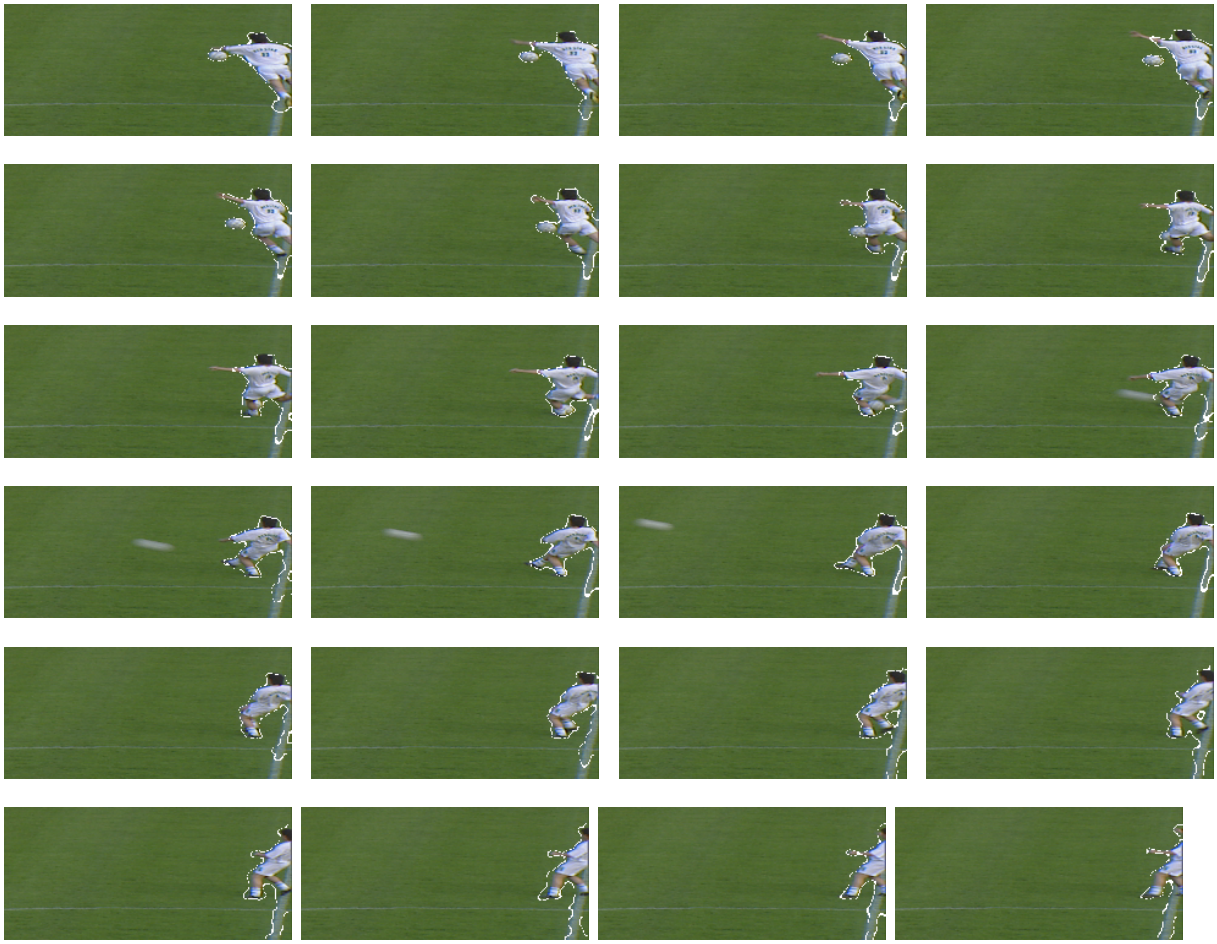


Figure A.11: *Foot*: frames 25-48

Bibliography

[1] G. Randall; A. Fernández; O. Trujillo; F. Malmierca; P. Morelli; G. Apelbaum; M. Bertalmío and L. Vázquez. Neuro3d: an interactive 3d reconstruction system of serial sections using automatic registration. *Proc. SPIE BiOS98, California*, 3621, March 1998.

[2] I. Carlbom; D. Terzopoulos and K. M. Harris. Computer-asisted registration, segmentation, and 3d reconstruction from images of neuronal tissue sections. *IEEE Trans. Med. Imag.*, pages 351–362, 1994.

[3] K. Montgomery; M. Ross. Improvements in semiautomated serial section reconstruction of neural tissue from tem images. *SPIE-94, California*, 1994.

[4] MIT Artificial Intelligence Laboratory. SLICER. http://www.slicer.org.

[5] Northern Eclipse. http://www.mvia.com/IASoftware/ia_software.html.

[6] M. Bertalmio. Morphing active contours: A geometric approach to topology independet image segmentation, tracking, interpolation and morphing. Master's thesis, Instituto de Ingenieria Eléctrica, Facultad de Ingenieria, Universidad de la República, 1998.

[7] M. de los Heros; J. Preciozzi; A. Martin. Bio3D. http://iie.fing.edu.uy/bio3d.

[8] S. C. Zhu; A. Yuille. Region competition: Unifying snakes, region growing and bayes/mdl for multi-band image segmentation. *IEEE Trans. Patt. Anal. Machine Intell.*, 79:12–49, September 1996.

[9] M. Kass; A. Witkin; D. Terzopoulos. Snakes: Active contour models. *Int. J. Comp. Vision*, 1:321–332, 1988.

[10] R. M. Haralick. Digital step edges from zero crossing of the second directional derivative. *IEEE Trans. Patt. Anal. Machine Intell.*, 6:58–68, 1984.

[11] R. Ronfard. Region-based strategies for active contour models. *Int. J. Comp. Vision*, 2:229–251, 13 1994.

[12] T. Chan; L. Vese. Active contour without edges. *IEEE Trans. Image Proc.*, 2(10):266–277, February 2001.

[13] N. Paragios; R. Deriche. Geodesic active regions: A new framework to deal with frame partition problems in computer vision. *J. of Visual Comm. and Image Representation*, 13(1):249–268, June 2002.

[14] S. Jehan-Besson; M. Barlaud; G. Aubert. Detection and tracking of moving objects using a new level set based method. *Int. J. Comp. Phys.*, September 2000.

[15] T. Brox; M. Rousson; R. Deriche and J. Weickert. Unsupervised segmentation incorporating colour, texture, and motion. *10th Int. Conf. on Comp. Analysis of Images and Patterns,*, 2756:353–360, August 2003.

[16] G. Medioni; C-K Tang; M-S Lee. Tensor voting: Theory and applications.

[17] A. Shashua; S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. *ICCV*, 1988.

[18] A. Amir; M. Lindenbaum. Quantitative analysis of grouping processes. *IEEE Trans. Patt. Anal. Machine Intell.*, 20(2):3168–185, 1998.

[19] Normalized Cuts and Image Segmentation. J. shi; j . malik. *IEEE Trans. Patt. Anal. Machine Intell.*, 22(8), August 2000.

[20] F. Cao; P. Musé; F. Sur. Extracting meaningful curves from images. *preprint INRIA No.5067*, December 2003.

[21] A. Pardo. Semantic image segmentation using morphological tools. *ICIP*, 2002.

[22] A. Desolneux; L. Moisan; J-M. Morel. Edge detection by helmholtz principle. *J. of Math. Imaging And Vision*, 14:271–284, August 2001.

[23] J. M. Morel; S. Solimini. *Variational Methods in Image Segmentation*. Birkhuser, 2001.

[24] V. Caselles; R. Kimmel; G. Sapiro. Geodesic active contours. *Int. J. Comp. Vision*, 22:61–79, March 1997.

[25] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and variational methods. *Comm. on Pure and Applied Math.*, XLII:577–685, August 1988.

[26] L. Cohen. On active contour models and balloons. *Comp. Vision, Graphics and Image Processing: Im. Understanding*, (53):211–218, 1991.

[27] T.F. Chan, B.Y. Sandberg, and L.A. Vese. Active contours without edges for vector-valued images. *Journal of Visual Communication and Image Representation*, 11(2):130–141, June 2000.

[28] R. Paragios, N.; Deriche. Geodesic active regions for supervised texture segmentation. *ICCV*, September 1999.

[29] J. Bign; G. H. Granlund; and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Trans. Patt. Anal. Machine Intell.*, 13(8):775–790, August 1991.

[30] J-F. Aujol; G. Aubert; L. Blanc-Feraud. Wavelet-based level set evolution for classification of textured images. *IEEE Trans. Image Proc.*, 12(12), December 2003.

[31] B. Sandberg; T. Chan; L. Vese. A level-set and gabor-based active contour algorithm for segmenting textured images. *UCLA CAM Report*, 02-39, July 2002.

[32] M. Rousson; N. Paragios. Geodesic active regions for image segmentation. *Int. J. Comp. Vision*, 22:61–79, March 2002.

[33] D. Freedman; T. Zhang. Active contours for tracking distributions. *IEEE Trans. Image Proc.*, 13, April 2004.

[34] S. Jehan-Besson; M. Barlaud; G. Aubert. Deformable regions driven by an eulerian acccurate minimization method for image and video segmentation. *Int. J. Comp. Vision*, 2002.

[35] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, January 2001.

[36] S. Osher; N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer-Verlag, 2003.

[37] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Science*. Cambridge University Press, 1998.

[38] F. Mémoli; A. Bartesaghi. *Toolbox FLUJOS: Evolución de curvas en Tratamiento de Imágenes (In Spanish)*. PhD thesis, Instituto de Ingeniería Eléctrica. Facultad de Ingeniería. Universidad de la República., 1999.

[39] C. Epstein; M. Gage. The curve shortening flow. *Wave Motion: Theory, Modelling and Computation*, 1987.

[40] M. Grayson. The hear equation shrinks embedded plane curves to round points. *J. Diff. Geom*, pages 285–314, 1987.

[41] Scale Space and Edge Detection using Anisotropic diffusion. P. perona; j. malik. *IEEE Trans. Patt. Anal. Machine Intell.*, (7):629–639, July 1990.

[42] Guillermo Sapiro and Allen Tannenbaum. Affine invariant scale-space. *Int. J. Comput. Vision*, 11(1):25–44, 1993.

[43] S. Osher; J. Sethian. Fronts propagating with curvature dependent speed, algorithms based on a hamilton-jacobi formulation. *Int. J. Comp. Phys.*, 79:12–49, 1988.

[44] W. Lorensen; H. Cline. Marching cubes: a high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21(4), July 1987.

[45] D. Peng; B. Merriman; S. Osher; H.K. Zhao; M. Kang. A pde based fast local level set method. *Int. J. Comp. Phys.*, 155:410–438, 1999.

[46] M. Sussman; P. Smereka; S. Osher. A level set method for computing solutions to incompressible twophase flow. *J. Comp. Phys.*, pages 119–146, December 1994.

[47] P. D. Lax. Hyperbolic systems of conservation laws and the mathematical theory of shock waves. *SIAM Regional Conference Series in Applied Mathematics, Philadelphia*, 1973.

[48] M. G. Crandall and P. Lions. Two approximations of solutions of hamilton-jacobi equations. *Mathematics of Computation*, 43(167):1–19, 1984.

[49] L. C. Evans. *Partial Differential Equations*. Berkeley Mathematics Lecture Notes Series, 1994.

[50] C. Hirsch. *Numerical Computation of Internal and External Flows.* John Wiley & Sons, Inc., New York. NY. USA, 1988.

[51] V. Caselles; V. Catte; F. Coll; F. Dibos. A geometric model for active contours. *Numersiche Mathematik*, 1993.

[52] R. Malladi; J. Sethian; B. Vemuri. Shape modeling with front propagation: a level set approach. *IEEE Trans. Patt. Anal. Machine Intell.*, 17:158–175, March 1995.

[53] H. Zhao; T. Chan ; B. Merriman; and S. Osher. A variational level set approach to multiphase motion. *Int. J. Comp. Phys.*, 143:495–518, 1998.

[54] N. Paragios; R. Deriche. Geodesic active regions for motion estimation and tracking. *ICCV*, 1999.

[55] N. Paragios; R. Deriche. Coupled geodesic active regions for image segmentation. *Inria Research Report*, 1999.

[56] S. Jehan-Besson; M. Barlaud; G. Aubert. Region-based active contours using geometrical and statistical features for image segmentation. *ICIP 03*, 2003.

[57] National Library of Medicine. Itk: Insight toolkit. http://www.itk.org.

[58] David Tschumperlé and Rachid Deriche. Vector-valued image regularization with pde's : A common framework for different applications. *IEEE Trans. Patt. Anal. Machine Intell.*, 27(4):506–517, April 2005.

[59] D. Tschumperlé. *Vector-Valued Image Regularization with PDE's : A Common Framework for Different Applications.* PhD thesis, Universit de Nice - Sophia Antipolis, 2002.

[60] S. Di Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics, and Image Processing*, (33):116–125, 1986.

[61] Charles Poynton. The color faq. http://www.poyton.com.

[62] Y. Ohta; T. Kanade; T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, (13):222–241, 1980.

[63] V. Caselles; B. Coll; J. M. Morel. Geometry and color in natural images. *Journal of Mathematical Imaging and Vision*, (2):89–105, March 2002.

[64] M. Tuceryan; A. K. Jain. *Texture Analysis, Chapter 2.1.* World Scientific, 1998.

[65] K. Brady. *A probabilistic framework for adaptive texture description.* PhD thesis, Univeristy of Nice, Sofia Antipolis, 2003.

[66] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1998.

[67] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Trans. Image Proc.*

[68] M. Do; M. Vetterli. Wavelet-based texture retrieval using generalized gaussian density and Kullback-Leilbler distance. *IEEE Trans. Image Proc.*, (2), February 2002.

[69] G. Van de Wouwer; P. Scheunders; S. Livens; D. Van Dyck. Wavelet correlation signatures for color texture characterization. *Pattern recognition*, 32(3):443–451, 1999.

[70] J. L. Barron; D. J. Fleet; S. S. Beauchemin. Performance of optical flow techniques. *Int. J. Comp. Vision*, 12(1):43–77, February 1994.

[71] B. Lucas; T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. of Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada:674–679, August 1981.

[72] B. Horn; B. Schunck. Determining optical flow. *AI*, 17:185–204, August 1981.

[73] M. Nicolescu; G. Medioni. Tensor voting: Theory and applications. *Proc. IEEE CVPR*, 2003.

[74] T. Brox; J. Weickert. Nonlinear matrix diffusion for optic flow estimation. *In L. Van Gool, editor, Pattern Recognition, volume 2449 of Lecture Notes in Computer Science. Springer. Berlin*, 2449:446–453, 2002.

[75] T. Brox; A. Bruhn; N. PapenBerg; J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Special Issue - European Conference on Computer Vision 2004 International Journal of Computer Vision, to appear.*

[76] D. Johnson; S. Sinanovic. Symmetrizing the kullback-leibler distance. *IEEE Trans. on Information Theory (submitted)*, 2001.

[77] F. Mémoli; A. Bartesaghi. Toolbox flujos: Evolución de curvas en tratamiento de imágenes. http://www.iie.edu.uy/investigacion/grupos/gti/flujos/flujos.html.

[78] Blitz++ library. http://blitz.sourceforge.net.

[79] Image magick. http://www.imagemagick.org.

[80] Kitware Inc. Vtk: The visualization toolkit. http://www.vtk.org.

[81] Gnu scientifc library. http://www.gsl.org.

[82] G. Useta; B. A. Huber. Estructura de las espermatecas y dinámica del esperma en schizocosa malitiosa (aranae, licosidae): Una primera aproximación (in spanish). *Resumen. Cuarto Encuentro de Aracnólogos del Cono Sur, Sao Paulo-Brasil.*, December 2003.

[83] Alejandro Bielli; Raquel Pérez; Graciela Pedrana; John T. B. Milton; Álvaro Lopez; Margaret A. Blackberry; Gregory Duncombe; Heriberto Rodriguez-Martinez and Graeme B. Martin. carneros. *Reproduction Fertility and Development*, (6):333–337, October 2002.

[84] Delgado R; Daz G; Ganuzza F; Olivera K; Pais B; Perdomo A; Pintos J; Rodrguez S; Bielli A. Katz H; Cortazzo N. Histology of prepuberal ovaries in the southamerican fur seal (arctocephalus australis zimmerman, 1783): three-dimensional reconstruction of crypts and follicles. *International Congress in Animal Reproduction (ICAR 2004). Porto Seguro, Bahia, Brasil*, (2), August 2004.

[85] C. Martinez; R. Paredes; R. Stock; A. Saralegui; M. Andreu; C. Cabezń; R. Ehrlich;and N. Galanti. Cellular organization and appearance of differentiated structures in developing stages of the parasitic platyhelminth echinococcus granulosus. *Journal of Biological Chemistry*, January 2005.

[86] T. Brox; M. Rousson; R. Deriche and J. Weickert. Unsupervised segmentation incorporating colour, texture, and motion. *INRIA Research Report*, March 2003.