A New Region-Based Parametric Deformable Model: Fuzzy Sets and Axis Independent Warps

By

Jean Honorio

B.S. in Systems Engineering, November 1997, Universidad de Lima

A dissertation submitted to

The Faculty of

The School of Engineering and Applied Science of

The George Washington University in partial satisfaction of

the requirements for the degree of Master of Science

August 7, 2006

Dissertation directed by

James K. Hahn

Professor of Engineering and Applied Science

Abstract

A new *region-based external energy* which is based on the *fuzzy sets theory* is presented. The proposed force converges fastest and asymptotically among different probability-based forces. Conditions for ensuring unique solution and convergence are analyzed as well. Additionally, a parametric contour representation is proposed by using *radial basis functions*.

Dedication

To my parents Mercedes and Jaime for instilling me the values of ethics, work and success.

To my sisters Karen and Josseline for their unconditional support.

To her for keeping my feet on the ground and for coming inside my life.

To another her for keeping my mind on heaven and for tenderly listening to me.

To some another her for accidentally giving me the idea which fired all the sequence of theorems and lemmas.

To yet another her for unintentionally saving me from my lack of inspiration and for giving me the strength in order to finish this research.

Table of Contents

List of Figures and Tables

Chapter 1. Introduction

1.1. Motivation

Image segmentation is the process of distinguishing objects from the background. Its importance has been increased in different applications in the past years. Medical image segmentation is one of the areas in which prior knowledge such as the number and shape of the objects to be segmented can be used. Even though image segmentation remains a difficult task due to the tremendous variability of object shapes, noise and sampling artifacts [35].

Techniques which only rely on edge detection and thresholding often fail in segmenting images on which the previous problems are present. On the other hand, d*eformable models* are contours (curves or surfaces) which are deformed in order to minimize their *internal* and *external energies*. The *internal energy* is computed from the curve itself and it ensures smoothness of the contour. It can also include prior information about the object shape. The *external energy* is computed from the image data and it ensures the convergence of the contour to the object boundaries.

The identification of image regions belonging to the object and background does not allow the understanding of the object in terms of its orientation and identification of constitutive parts. The use of *labeled deformable models* allows the easy identification of object parts after segmentation. For instance, a labeled model can contain the 17 left ventricular zones defined by the American Heart Association [6] for myocardial segmentation. Also, since *deformable models* are expressed in real-valued coordinates, they can obtain sub-pixel accuracy [35].

Deformable models can be classified as parametric and geometric. *Parametric deformable models* represent contours as explicit formulas in terms of a small set of parameters. For any given parameter values, the curve or surface which represents the object contour can be drawn. *Geometric deformable models* represent shapes as a level set (c.f. implicit function) over the image domain. After segmentation, the result is an image of positive and negative values corresponding to object and background pixels respectively. Topological adaptation such as splitting and merging parts during deformation are naturally handled in geometric models [35], while shape priors are easier to handle in parametric models.

Deformable models can be also classified as boundary-based and region-based. *Boundary-based deformable models* adapt by using an *external energy* driven by the gradient of the image. This is due to the fact that object boundaries correspond well to strong gradients. On the other hand, *region-based deformable models* adapt by using an *external energy* driven by region descriptors [19] which for instance can be based on the probability distributions of the intensities. Region-based approaches behave more stable and have non-trivial local minima that are often visually meaningful when compared with boundary-based approaches [7]. This is due to the fact that boundary-based methods are usually pulled towards noisy or fragmented edges.

1.2. Problem and Objective

In applications such as medical image segmentation, prior knowledge is available regarding the number and shape of the objects to be segmented. On the other hand, region-based information is very important in the absence of strong image gradients. The difference between the distributions of the intensity or some other texture property for the object and the background regions can be used in order to define more stable energy terms. *Region-based external forces* which make use of this prior knowledge vary in speed of convergence. Furthermore, convexity properties of the energy functions are poorly understood.

As a result, this research reviews several *internal* and *external forces*. A deeper analysis of the speed of convergence is performed for region-based forces which are based on probability measures. A new *external force* is proposed in order to obtain a faster convergence. Theoretical aspects such as conditions for ensuring unique solution and convergence are also analyzed.

1.3. Original Contribution

A deformable model which is based on the *fuzzy sets theory* is presented. A new *region-based external energy* is derived as a result of maximizing the *crisp probability* that the contour accurately separates the image into two regions. The proposed force converges fastest and asymptotically among different probability-based forces.

Unique solution and convergence is ensured for a *perfect point classifier*, an *axis-independent warp* and a convex contour.

A parametric contour representation is proposed by using *radial basis functions*, which allows using an arbitrary mesh as deformable model. As additional constraints, two simple *internal energies* are defined for ensuring topology preservation and temporal coherence.

1.4. Document Organization

In Chapter 2, the common framework for deformable models is explained. Several boundary-based and region-based *external energies* which were proposed by different authors are reviewed. Different shape parameterization techniques and their *internal energies* are presented in this chapter as well. Finally, results on the literature regarding unique solution, convexity of the energy function as well as speed of convergence are reviewed.

In Chapter 3, an *external energy* is derived as a result of maximizing the *crisp probability* that the contour accurately separates the image into two regions. Concepts such as *point classifier* as well as *axis-independent warps* are also introduced. Conditions for ensuring unique solution and convergence are analyzed in this chapter as well.

In Chapter 4, several implementation issues are presented in detail. *Radial basis functions warping* is demonstrated to be a class of *axis-independent warping*. The largely known Bayes classifier is used as a *point classifier*. Methods for approximating contour integrals are presented as well as additional steps for 2.5D segmentation.

In Chapter 5, segmentation of a 2D image, a sequence of few horizontal and vertical 2D cardiac magnetic resonance images (MRI) as well as a 3D computed tomography (CT) volume are shown. An analysis of the speed of convergence of the proposed force versus different probability-based forces is performed.

In Chapter 6, conclusions are drawn from the results which were obtained in the previous experiment. Several ways of extending this work are proposed as well.

Chapter 2. Related Work

2.1. Deformable Models

Deformable models were first introduced by [20] as a curve *C* over a 2D image $\Omega \subset \Re^2$ which minimizes the energy function:

$$E(C) = E_{\text{int}}(C) + E_{ext}(C) \quad \ldots (2.1)$$

Where E_{int} and E_{ext} are the *internal* and *external energy* respectively. The *internal energy* is computed from the curve itself and it ensures smoothness of the contour. The *external energy* is computed from the image data and it ensures the convergence of the contour to the object boundaries. The *internal* and *external energy* can be defined as a weighted sum of several energies.

The curve *C* which minimizes the energy function (2.1) is found by using the *gradient descent method*. The contour is made dependant on the time *t* such as *C*(*t*). Given an initial contour *C*(0) and the step size γ, the next approximations of the solution are computed as:

$$\begin{aligned} &\mathbf{w}(t+\partial t) = \mathbf{w}(t) + \gamma \frac{\partial \mathbf{w}}{\partial t}(t) \quad \text{for } \forall \mathbf{w}(t) \in C(t) \wedge t \geq 0 \wedge \gamma > 0 \\ &\frac{\partial \mathbf{w}}{\partial t}(t) = \mathbf{f}_{\text{int}}(\mathbf{w}(t)) + \mathbf{f}_{ext}(\mathbf{w}(t)) \end{aligned} \quad \ldots (2.2)$$

Where $\mathbf{f}_{int}$ and $\mathbf{f}_{ext}$ are the *internal* and *external forces* respectively. Forces are computed as the gradient of the equivalent energies with respect to the parameters which drive the deformation of the contour. In most of the cases, $\mathbf{f}_{ext}(\mathbf{w}) = -\nabla E_{ext}(\mathbf{w})$ is the negative of the gradient of the *external energy* with respect to each image axis. Because of this reason, some authors prefer to define forces instead of energies.

2.2. Common Notation

We want to segment a *d*-dimensional image defined over the region $\Omega \subset \Re^d$ into two disjoint regions Ω_{in} and Ω_{out} separated by a *simple closed contour* $C \subset \Re^d$ such that

$\Omega_{in} \cup \Omega_{out} = \Omega$. The region Ω_{in} represents the object of interest, while Ω_{out} represents the background.

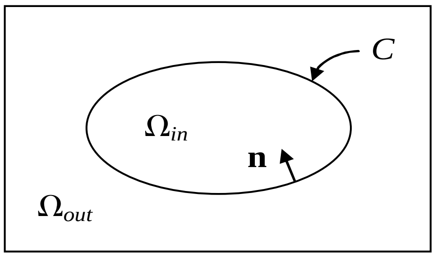


Figure 2.1: Contour and disjoint regions

A *simple closed contour* is a contour which does not contain end points neither self-intersections [2]. The region Ω_{in} would be not necessarily convex neither fully-connected, since the contour *C* would be a set of *simple closed contours*.

Given a region $R \subset \Re^d$, the area in 2D as well as the volume in 3D is called *hypervolume* in a generalized way. The *hypervolume* of the region is defined as $|R| = \int_R d\mathbf{w}$.

In the 2D case (*d* = 2) the images are composed by pixels and the contour is a curve. In the 3D case (*d* = 3) the images are composed by voxels and the contour is a surface. Image locations (pixels or voxels) are called *points* in this research, in order to define a general method for segmenting 2D and 3D images, as well as a sequence of them.

Let *I*(**w**) be the image intensity at the point **w**. Let $\mathbf{n}(\mathbf{w}): C \to \Re^d$ be the unit normal vector for a point **w** in the contour $C \subset \Re^d$, such that **n**(**w**) points inside the region of the object of interest Ω_{in} as shown in figure 2.1.

2.3. Boundary-Based External Energies

2.3.1. Line Detector

The energy function for detecting black or white lines is defined in [20] as:

$$E_{line}(C) = \pm \int_C I(\mathbf{w}) d\mathbf{w} \quad \dots (2.3)$$

Since the equation (2.1) is to be minimized, a positive sign on (2.3) allows detecting black lines, while a negative sign can be used if white lines are to be detected.

2.3.2. Edge Detector

Object boundaries correspond well to strong gradients. The energy function for detecting edges in the image is defined as a function of the image gradient:

$$E_{edge}(C) = \int_C g(|\nabla I(\mathbf{w})|)d\mathbf{w} \quad \ldots (2.4)$$

Since it is required that this energy be smaller at the edges and greater in homogeneous regions, one choice is $g(r) = -r^2$ as described in [20]. Another choices are $g(r) = 1/(1+r)$ or $g(r) = 1/(1+r^2)$ as defined in [5].

2.3.3. Dynamic Edge

The edge detector described in (2.4) has the drawback of creating oscillations of the deformable model across the pixels with high gradient due to the choice of the time step [14]. A dynamic edge force was created in order to deal with this problem. Let **g** the pixel of maximum gradient in a *m*×*m* window around **w**. The force is defined as:

$$\mathbf{f}_{dedge}(\mathbf{w}) = ((\mathbf{g} - \mathbf{w}) \bullet \mathbf{n}(\mathbf{w}))\mathbf{n}(\mathbf{w}) \quad \ldots (2.5)$$

This force can be seen as the projection of the direction of greatest gradient into the normal vector. As a result, the contour will be pushed outside or inside in order to intersect the pixel of greatest gradient. Not any oscillations in the mesh occur since the force is proportional of the necessary displacement in order to hit the pixel of maximum gradient (and not the gradient itself), and since it follows the normal direction. Since it is possible to pre-compute the corresponding pixel of maximum gradient **g** associated to the *m*×*m* window for every pixel **w**, it is not necessarily time-consuming.

2.3.4. Distance Map

The use of distance maps is proposed in [9] in order to make edges have a bigger area of influence. First, edge pixels are detected by using a *Canny-Deriche local edge detector*. Second, the value of the distance map at each pixel is defined as the Euclidean distance from the pixel to the closest edge pixel. Even though the distance map is calculated as a pre-processing step, a *Chamfer distance* is used for approximating the Euclidean distance in order to reduce the number of computations into a two-pass algorithm.

On the other hand, a initial *watershed segmentation* is proposed in [16]. This technique splits the image into several small regions with homogeneous color, which are divided by edges. But the main drawback of this algorithm is their high sensitivity to noise which results in an over-segmentation. Even though, it provides a good initial image partition which can be used in order to drive the evolution of the contour.

Let $d(\mathbf{w})$ be the distance between a pixel **w** and the nearest edge pixel. The energy function which attracts the contour towards the edges is defined as:

$$E_{dist}(C) = \int_C g(d(\mathbf{w}))d\mathbf{w} \;\ldots\; (2.6)$$

Since it is required that this energy be greater for larger distances, $g(r)$ is usually chosen as $g(r) = -\exp^{-r^2}$ or $g(r) = -1/r$ as in [9]. Another choice is $g(r) = r^2$ as defined in [16].

2.3.5. Dynamic Distance

The distance energy as defined in (2.6) entails very large deformation away from the edge pixels, which causes an unstable behavior and it is ambiguous at pixels which are equidistant from two edges [14]. For solving this problem, a dynamic distance force was created. Let **g** the nearest edge pixel in the direction of the normal at the pixel **w**. The force is defined as:

$$\mathbf{f}_{ddist}(\mathbf{w}) = \mathbf{g} - \mathbf{w} \;\ldots\; (2.7)$$

A maximum number of traversed pixels in the forward and backward direction of the normal is used. Notice that this force pushes the model in the direction of the normal since **g** is searched from the pixels on that direction. Since this operation is performed each time the model deforms, it is time-consuming.

2.3.6. Gradient Vector Flow

The distance energy defined in (2.6) can cause difficulties when deforming a contour into concavities. This is due to the fact that every pixel is attracted to the nearest edge. In a U shaped object, forces tend to point horizontally in opposite directions inside the concavity, but not any force pushes the contour downward. Therefore, the model does not converge inside the concavity. The edge detector (2.4) with a Gaussian filter with standard deviation σ can be used instead as proposed originally in [20]. Its range of influence can be increased by increasing the value of σ. Even though the boundary location becomes less accurate and distinct, ultimately eliminating the concavity itself when σ becomes too large [34].

In order to solve these problems, a new force is defined in [33] [34] as:

$$\mathbf{f}_{gvf}(\mathbf{w}) = \mathbf{v}(\mathbf{w}) \quad \ldots (2.8)$$

Where **v** is the pre-computed *gradient vector flow* at the pixel **w**. Let *m*(**w**) be an edge map derived from the image *I*(**w**) having the property that it is large near image edges, such as (2.3) or (2.4). The *gradient vector flow* field is a diffused version of the gradient field ∇m which keeps the desired property of having high gradients near the edges, but it extends the gradient field further away into homogeneous regions.

Given an initial value $\mathbf{v}(\mathbf{w},0) = \nabla m$, $\mathbf{v}$ is defined as the equilibrium solution to the following partial differential equation:

$$\mathbf{v}(\mathbf{w}, t + \partial t) = \mathbf{v}(\mathbf{w}, t) + \gamma \frac{\partial \mathbf{v}}{\partial t} \quad \text{for } \mathbf{w} \in \Omega \wedge t \geq 0 \wedge \gamma > 0$$

$$\frac{\partial \mathbf{v}}{\partial t} = g(|\nabla m|)\nabla^2 \mathbf{v} - h(|\nabla m|)(\mathbf{v} - \nabla m)$$

Where ∇^2 is the Laplacian operator applied to each spatial component, *g*(*r*) is the smoothing term since it produces a smoothly varying vector field, *h*(*r*) is the conformity term since it encourages the vector field to be as close to ∇m as possible. In [34] these terms are chosen to be $g(r) = \mu$ and $h(r) = r^2$, where μ should be set according to the amount of noise present in the image, and it governs the tradeoff between smoothing and conformity. When little smoothing is required in the presence of large gradients $g(r) = \exp^{-(r/\mu)^2}$ and

$h(r) = 1 - g(r)$ can be used instead as proposed in [33]. In this case μ determines to some extent the importance between smoothing and conformity.

2.3.7. Pressure Force

If there is not any edge detected by (2.4) as in a constant intensity area, the curve shrinks on itself and vanishes to a point. Also, if the initial contour is not close to the desired solution, the contour can fail to converge due to the presence of weak and spurious edges. In order to solve both of these problems, a pressure force which inflates or deflates the deformable model is defined in [10] as:

$$\mathbf{f}_{press}(\mathbf{w}) = \pm\mathbf{n}(\mathbf{w}) \quad \ldots (2.9)$$

A negative sign on (2.9) inflates the model, while a positive sign deflates it. The weighting parameter for this force should be selected so that it is smaller than the edge detector at significant edges, and it avoids weak and spurious edges. As shown in [9], this force is equivalent to the energy function:

$$E_{press}(C) = \mp|\Omega_{in}| \quad \ldots (2.10)$$

Minimizing this energy corresponds to obtain a contour C which minimizes its area inside it. Therefore, the force described in (2.9) pushes the model in the direction of the normal pointing outside the contour. The drawback is that the user needs to select the sign in order to inflate or deflate the model, so that the initial contour has to be inside or outside the solution, but not across [34].

2.3.8. Interactive Constraints

In some cases, automated *external energies* fail to deform the model to the desired boundary. Interactive constraints allow the user to define points which are used for pushing or pulling the model. The s*pring energy* [20] which allows defining an attraction force between two points $\mathbf{w}_1$ and $\mathbf{w}_2$ is defined as the minimization of the square distance between them:

$$E_{spring}(\mathbf{w}_1, \mathbf{w}_2) = |\mathbf{w}_1 - \mathbf{w}_2|^2 \quad \ldots (2.11)$$

In the previous energy function, $\mathbf{w}_1$ is a point on the curve and $\mathbf{w}_2$ can be either another point on the curve or a fixed position. The *volcano energy* [20] which allows defining a repulsive force between two points $\mathbf{w}_1$ and $\mathbf{w}_2$ is defined as the maximization of the square distance between them:

$$E_{volcano}(\mathbf{w}_1, \mathbf{w}_2) = \frac{1}{\left|\mathbf{w}_1 - \mathbf{w}_2\right|^2} \quad \ldots (2.12)$$

2.4. Region-Based External Energies

2.4.1. Ward Distance

In order to include region information in the segmentation process, [26] proposed a heuristics that makes use of the *Ward distance*. This distance is defined as the amount of energy needed in order to disrupt a contour between two contiguous regions *A* and *B*:

$$d(A,B) = \int_{A \cup B} (I(\mathbf{w}) - \mu_{A \cup B})^2 d\mathbf{w} - \int_A (I(\mathbf{w}) - \mu_A)^2 d\mathbf{w} - \int_B (I(\mathbf{w}) - \mu_B)^2 d\mathbf{w}$$

Where μ_A, μ_B and $\mu_{A \cup B}$ are the intensity means on their corresponding areas. Let *M* a small rectangle of $l \times (2p - 1)$ pixels which follows the orientation of the normal $\mathbf{n}(\mathbf{w})$. The rectangle *M* is composed by three small rectangles: M_{in} and M_{out} of $l \times p$ pixels inside and outside the contour *C* respectively, as well as M_C of $l \times 1$ pixels crossing the contour *C*. The force is defined as:

$$\mathbf{f}_{ward}(\mathbf{w}) = (-d(M_{in}, M_C) + d(M_{out}, M_C))\mathbf{n}(\mathbf{w}) \quad \ldots (2.13)$$

If the *Ward distance* or the energy needed to disrupt the area inside the contour is greater than the one needed for the area outside, the contour will follow the opposite direction of the normal. This will cause the expansion of the deformable model which is a desirable result.

2.4.2. Mean Square Error

By assuming that the image is composed of two regions of approximately constant intensities, [8] proposed to reduce the mean square error of the intensities with respect to the intensity means μ_{in} and μ_{out}, inside and outside the contour *C* respectively. The energy function is defined as:

$$E_{mse}(C) = \int_{\Omega_{in}} (I(\mathbf{w}) - \mu_{in})^2 d\mathbf{w} + \int_{\Omega_{out}} (I(\mathbf{w}) - \mu_{out})^2 d\mathbf{w} \quad \ldots (2.14)$$

In this model, μ_{in} and μ_{out} are assumed to be constant with respect to the evolution of the contour *C* in order to derive the force. The force is defined as:

$$\mathbf{f}_{mse}(\mathbf{w}) = ((I(\mathbf{w}) - \mu_{in})^2 - (I(\mathbf{w}) - \mu_{out})^2)\mathbf{n}(\mathbf{w}) \quad \ldots (2.15)$$

When the intensity of a pixel in the contour is closer to the inside intensity mean μ_{in} than to the outside intensity mean μ_{out}, the force follows the opposite direction of the contour normal. This creates the desirable effect of making the deformable model to expand. In the opposite case, when the intensity of a pixel is closer to the outside intensity mean, the deformable model contracts. In [8] texture properties such as the curvature or the orientation were also used in (2.14) and (2.15) instead of intensity values.

2.4.3. Mumford-Shah Functional

It can be assumed that the objects in an image have smoothly varying surface and reflectance properties in small areas. Even though it is improper to assume that the image is piecewise smooth due to the presence of noise. The *Mumford-Shah model* approximates an original image *I*(**w**) by piecewise smooth approximations $\hat{I}_{in}(\mathbf{w})$ and $\hat{I}_{out}(\mathbf{w})$ inside and outside the boundary. The energy functional is defined in [4] as:

$$\begin{aligned} E_{mshah}(C) = \int_{\Omega_{in}} \left[(I(\mathbf{w}) - \hat{I}_{in}(\mathbf{w}))^2 + \mu \left| \nabla \hat{I}_{in}(\mathbf{w}) \right|^2 \right] d\mathbf{w} + \\ \int_{\Omega_{out}} \left[(I(\mathbf{w}) - \hat{I}_{out}(\mathbf{w}))^2 + \mu \left| \nabla \hat{I}_{out}(\mathbf{w}) \right|^2 \right] d\mathbf{w} \end{aligned} \quad \ldots (2.16)$$

Where the gradients $\nabla\hat{I}_{in}(\mathbf{w})$ and $\nabla\hat{I}_{out}(\mathbf{w})$ are computed with respect to each image axis. The first terms on both integrals reduce the amount of noise, which is measured as the square difference between the original image and its piecewise smooth approximations. The second terms on both integrals prevent high intensity gradients inside and outside the contour, therefore generating smooth regions. The factor μ governs the tradeoff between noise and smoothness.

The functions $\hat{I}_{in}(\mathbf{w})$ and $\hat{I}_{out}(\mathbf{w})$ in [4] are chosen to be the intensity means of several small areas in the image. Notice that one specific case is the model proposed in (2.14), when the smooth approximations are chosen to be the intensity means inside and outside the contour *C* and therefore the gradients vanish.

In this model, $\hat{I}_{in}(\mathbf{w})$ and $\hat{I}_{out}(\mathbf{w})$ are assumed to be constant with respect to the evolution of the contour *C* in order to derive the force. The force is defined as:

$$\mathbf{f}_{mshah}(\mathbf{w}) = \begin{pmatrix} (I(\mathbf{w}) - \hat{I}_{in}(\mathbf{w}))^2 + \mu \left|\nabla \hat{I}_{in}(\mathbf{w})\right|^2 \\ -(I(\mathbf{w}) - \hat{I}_{out}(\mathbf{w}))^2 - \mu \left|\nabla \hat{I}_{out}(\mathbf{w})\right|^2 \end{pmatrix} \mathbf{n}(\mathbf{w}) \; \ldots \; (2.17)$$

Notice that texture properties such as the curvature or the orientation can also be used in (2.16) and (2.17) instead of intensity values.

2.4.4. Pairwise Dissimilarity

A pairwise dissimilarity measure is proposed in [28] in order to encourage similarity within the regions while discouraging inter-region similarity. The energy function is proposed in two different versions, even though the derived force is the same. The first version of the energy function maximizes inter-region dissimilarity by minimizing:

$$E_{pwd}(C) = -\int_{\Omega_{in}} \int_{\Omega_{out}} g(\mathbf{w}, \mathbf{v}) d\mathbf{v} d\mathbf{w} \; \ldots \; (2.18)$$

Where $g(\mathbf{w},\mathbf{v})$ is a dissimilarity measure between the pixels **w** and **v**, such that a greater value indicates a lower similarity. The second version of the energy function minimizes dissimilarity inside each region and is given by:

$$E_{pwd}(C) = \int_{\Omega_{in}} \int_{\Omega_{in}} g(\mathbf{w}, \mathbf{v}) d\mathbf{v} d\mathbf{w} + \int_{\Omega_{out}} \int_{\Omega_{out}} g(\mathbf{w}, \mathbf{v}) d\mathbf{v} d\mathbf{w} \; \ldots \; (2.19)$$

As demonstrated in [28], both (2.18) and (2.19) arrive to the same force for the curve evolution. The force is defined as:

$$\mathbf{f}_{pwd}(\mathbf{w}) = \left(\int_{\Omega_{in}} g(\mathbf{w},\mathbf{v})d\mathbf{v} - \int_{\Omega_{out}} g(\mathbf{w},\mathbf{v})d\mathbf{v} \right)\mathbf{n}(\mathbf{w}) \quad \ldots (2.20)$$

Where the dissimilarity measure is chosen to be the absolute difference of the intensity values, such that $g(\mathbf{w},\mathbf{v}) = |I(\mathbf{w}) - I(\mathbf{v})|$. Other more sophisticated versions of dissimilarity measures based on texture properties and pixel distance can also be defined such as in [28]. Notice that the dissimilarity map $g(\mathbf{w},\mathbf{v})$ between every pixel **w** and **v** can be pre-computed. A reduced resolution of the image is used for **v** while the original resolution is used for **w**. This is done in order to reduce the amount of required memory and to speed up the computation of the integrals in the force (2.20).

2.4.5. Adaptive Fuzzy C-Means

Adaptive fuzzy C-means segmentation allows partitioning an image into *n* fixed different classes. The result of this algorithm is a membership value $\mu_i(\mathbf{w}) \geq 0$ for *i* = 1…*n* which measures the degree of membership of the pixel **w** to the class *i*. One requirement for membership values is that $\sum_{i=1}^{n} \mu_i(\mathbf{w}) = 1$. Let $\mu_1(\mathbf{w})$ be the membership value for the class which corresponds to the object of interest. The energy function is defined in [22] as:

$$\mathbf{f}_{afcm}(\mathbf{w}) = \left(\frac{\max\limits_{i=2}^{n} \mu_i(\mathbf{w})}{\mu_1(\mathbf{w})} - 1 \right)\mathbf{n}(\mathbf{w}) \quad \ldots (2.21)$$

Notice that when the pixel **w** is more likely to be part of the object of interest than to the background $\mu_1(\mathbf{w}) > \max \mu_i(\mathbf{w})$, the factor is negative and the force follows the opposite direction of the normal. As a desirable result, the deformable model is inflated in order to contain the pixel **w**. In the opposite case when $\mu_1(\mathbf{w}) < \max \mu_i(\mathbf{w})$, the deformable model is deflated.

2.4.6. Region Probability

Maximization of the a posteriori segmentation probability is proposed in [25]. The authors follow the Bayes rule while several assumptions are made. First, every way of partitioning the image into two regions (inside and outside the contour C) is assumed to be equally probable. Second, both regions are assumed to be statistically independent since they depend only on the pixels contained inside each region. Finally, the intensity of each pixel is assumed to be statistically independent. After these assumptions, the a posteriori probability of partitioning the image into the two regions becomes:

$$p(\Omega_{in} \cap \Omega_{out} \mid I) = \prod_{\mathbf{w}\in\Omega_{in}} p_{in}(I(\mathbf{w})) \prod_{\mathbf{w}\in\Omega_{out}} p_{out}(I(\mathbf{w}))$$

Where $p_{in}(I(\mathbf{w}))$ and $p_{out}(I(\mathbf{w}))$ are the probability density functions of the intensities for the object and the background. These probability density functions are approximated by Normal distributions or non-parametrically estimated by using the *Parzen window method*. Since the maximization of this formula corresponds to the minimization of its negative logarithm, the energy function is defined as:

$$E_{rprob}(C) = -\int_{\Omega_{in}} \log p_{in}(I(\mathbf{w}))d\mathbf{w} - \int_{\Omega_{out}} \log p_{out}(I(\mathbf{w}))d\mathbf{w} \quad \ldots (2.22)$$

In this model, $p_{in}(I(\mathbf{w}))$ and $p_{out}(I(\mathbf{w}))$ are assumed to be constant with respect to the evolution of the contour C in order to derive the force. The force is defined as:

$$\mathbf{f}_{rprob}(\mathbf{w}) = \log\left(\frac{p_{out}(I(\mathbf{w}))}{p_{in}(I(\mathbf{w}))}\right)\mathbf{n}(\mathbf{w}) \quad \ldots (2.23)$$

When the intensity of a pixel in the contour is more probable to be part of the object than of the background $p_{in}(I(\mathbf{w})) > p_{out}(I(\mathbf{w}))$, the logarithm becomes negative and the force follows the opposite direction of the contour normal. This creates the desirable effect of making the deformable model to expand. In the opposite case, when the intensity of a pixel is more probable to be part of the background than of the object, the deformable model contracts. In [25] texture properties were also used in (2.22) and (2.23) instead of intensity values.

2.4.7. Region-Dependent Descriptor

Previous works on region-based deformable models do not focus on the dependence of the energy function on the evolution of the regions [19]. This occurs for statistical descriptors such as the mean as used in (2.14). The remarkable work done in [19] provides a general framework for region-dependent energy functions, as well as it demonstrates that additional terms should be used for the derived region-based forces. The energy function is defined as:

$$E_{rdesc}(C(t)) = \int_{\Omega_{in}(t)} k_{in}(\mathbf{w})d\mathbf{w} + \int_{\Omega_{out}(t)} k_{out}(\mathbf{w})d\mathbf{w} \quad \ldots (2.24)$$

Where $k_{in}(\mathbf{w})$ is the descriptor for the object region and $k_{out}(\mathbf{w})$ is the descriptor for the background region. Notice that the dependence on the evolution variable t has been made more notorious in this framework. The derived force becomes:

$$\mathbf{f}_{rdesc}(\mathbf{w}) = \left(k_{in}(\mathbf{w}) - k_{out}(\mathbf{w}) + \int_{\Omega_{in}(t)} \frac{\partial k_{in}}{\partial t} d\mathbf{w} + \int_{\Omega_{out}(t)} \frac{\partial k_{out}}{\partial t} d\mathbf{w} \right) \mathbf{n}(\mathbf{w}) \quad \ldots (2.25)$$

The first two terms have an intuitive interpretation. For a pixel $\mathbf{w}$, $k_{in}(\mathbf{w}) < k_{out}(\mathbf{w})$ indicates that the pixel is more likely to belong to the object than to the background. In that case, the sum of the first two terms is negative and the deformable model will follow the opposite direction of the normal. This will cause a contraction which is the desirable effect. In [19] several implementations based on (2.25) are described, such as descriptors based on means and variances.

2.4.8. Information Entropy

The minimization of the *entropy* is proposed in [17] in order to segment the image into two regions with dominant intensities. The *information entropy* is a negative measure, since it is related with the amount of randomness or information in an event. Recall that the *information entropy* is maximized when every intensity level is equally probable. The *entropy* is lower when only few intensity levels dominate. Therefore, the following energy function is defined in order to minimize the *entropy* in both regions:

$$\begin{aligned} E_{entropy}(C) = &-\int_{\Omega_{in}} p_{in}(I(\mathbf{w}))\log p_{in}(I(\mathbf{w}))d\mathbf{w} \\ &-\int_{\Omega_{out}} p_{out}(I(\mathbf{w}))\log p_{out}(I(\mathbf{w}))d\mathbf{w} \end{aligned} \quad \dots (2.26)$$

Where $p_{in}(I(\mathbf{w}))$ and $p_{out}(I(\mathbf{w}))$ are the probability density functions of the intensities for the object and the background. These probability density functions are non-parametrically estimated by using the *Parzen window method*. The forces are derived by following the framework described in (2.24) and (2.25).

2.4.9. Mutual Information

The maximization of the *mutual information* between the intensity and the partition is proposed in [21]. Recall that the *mutual information* is a measure of the mutual dependence between two variables, such that their *mutual information* is zero when they are independent. In that sense, the *mutual information* measures how well the partition into two regions explains well the probability density functions of the intensities inside each region. As found in [21], the maximization of the *mutual information* is equivalent to the minimization of the *conditional entropy* due to the fact that the true intensity distributions do not depend on the contour. Therefore, the following energy function was defined in [17] and [21]:

$$\begin{aligned} E_{mutual}(C) = &-\frac{|\Omega_{in}|}{|\Omega|}\int_{\Omega_{in}} p_{in}(I(\mathbf{w}))\log p_{in}(I(\mathbf{w}))d\mathbf{w} \\ &-\frac{|\Omega_{out}|}{|\Omega|}\int_{\Omega_{out}} p_{out}(I(\mathbf{w}))\log p_{out}(I(\mathbf{w}))d\mathbf{w} \end{aligned} \quad \dots (2.27)$$

Where $p_{in}(I(\mathbf{w}))$ and $p_{out}(I(\mathbf{w}))$ are the probability density functions of the intensities for the object and the background. These probability density functions are non-parametrically estimated by using the *Parzen window method*. Notice that the energy function (2.27) is a weighted average version of (2.26) by using the proportion of the areas of the regions inside and outside the contour *C*. The forces are derived by following the framework described in (2.24) and (2.25).

2.4.10. Active Appearance Model

A model which deals with shape and appearance (intensity) fitness with respect to training images is proposed in [11] [23]. Shape and appearance are represented a weighted sum of few terms involving the most significant variability in the training data. Shapes are parameterized by a fixed set of landmarks. These landmarks are manually identified on several training images. Subsequently, these shapes are aligned to one another with respect to scaling, rotation and translation by using the *Procrustes method*, which allows analyzing the landmark variability in a common coordinate frame. Given *N* aligned shapes $\mathbf{S}_i = (x_{i,1}, y_{i,1} \ldots x_{i,L}, y_{i,L})^T$ for *i* = 1…*N*, where *L* is the number of landmarks in the shape. The mean shape $\overline{\mathbf{S}}$ and covariance matrix $\hat{\mathbf{S}}$ are computed as:

$$\overline{\mathbf{S}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{S}_i$$
$$\hat{\mathbf{S}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{S}_i - \overline{\mathbf{S}})(\mathbf{S}_i - \overline{\mathbf{S}})^T$$

Given *N* training images A_i and by using the mean shape $\overline{\mathbf{S}}$ as a common coordinate frame, the mean appearance $\overline{A}$ and covariance matrix $\hat{A}$ are computed as:

$$\overline{A} = \frac{1}{N}\sum_{i=1}^{N}A_i$$
$$\hat{A} = \frac{1}{N-1}\sum_{i=1}^{N}(A_i - \overline{A})(A_i - \overline{A})^T$$

By using *principal component analysis*, the eigenvectors $\hat{\mathbf{S}}_i$ for *i* = 1…*n* corresponding to the largest *n* eigenvalues λ_i of the covariance matrix $\hat{\mathbf{S}}$ describe the most significant variability of the landmarks in the training data. Therefore, any shape **S** can be approximated as the mean shape plus a weighted sum of the first *n* eigenvectors:

$$\mathbf{S}(\mathbf{p}) = \overline{\mathbf{S}} + \sum_{i=1}^{n} p_i \hat{\mathbf{S}}_i$$

Where p_i for *i* = 1…*n* are the shape parameters. In a similar fashion, any appearance *A* can be approximated as the mean appearance plus a weighted sum of the first *m* eigenvectors:

$$A(\mathbf{q}) = \overline{A} + \sum_{i=1}^{m} q_i \hat{A}_i$$

Where q_i for $i = 1 \ldots m$ are the appearance parameters. Additional parameters are added into the model in order to account for global transformations, such as a scaling factor s, a rotation matrix $\mathbf{R}$ with angle θ and a translation vector $\mathbf{t}$. A landmark on the contour C can be calculated from a landmark (x_i, y_i) in the undeformed common coordinate frame as:

$$\mathbf{l}_i(\mathbf{p}) = s\mathbf{R}(\theta)(x_i, y_i)^T + \mathbf{t} \quad \text{for } (x_i, y_i) \in \mathbf{S}(\mathbf{p})$$

This landmark set $\mathbf{L}(\mathbf{p}) = (\mathbf{l}_1(\mathbf{p}), \mathbf{l}_2(\mathbf{p}), \ldots, \mathbf{l}_L(\mathbf{p}))^T$ is then used to deform the contour by using a warping function $\mathbf{w}(\mathbf{L}, \mathbf{x}) : \Re^{2L} \times \Re^2 \to \Re^2$, which maps a pixel $\mathbf{x}$ from the mean shape $\overline{\mathbf{S}}$ into the current shape of the curve C.

In order to maximize the similarity between the intensities inside the curve C and the model driven by the shape and appearance parameters, the following energy function is proposed:

$$E_{aam}(C) = \int_{\overline{S}} \left(I(\mathbf{w}(\mathbf{L}(\mathbf{p}), \mathbf{x})) - A(\mathbf{q}, \mathbf{x}) \right)^2 d\mathbf{x} \quad \ldots (2.28)$$

Notice that this corresponds to the minimization over the mean shape space $\overline{\mathbf{S}}$ of the square error between the current image and the model, with respect to the shape parameters as well as to the appearance parameters.

2.5. Shape Parameterization and Internal Energies

2.5.1. Spline

As originally introduced by [20] the deformable curve C is defined as a function $\mathbf{w}(u) : [0,1] \to \Re^2$, such that for every value of u, $\mathbf{w}(u)$ is the position on the curve associated with that value. The energy function is defined as:

$$E_{spline}(C) = \frac{1}{2} \int_0^1 \left(\alpha(u) \left| \frac{\partial \mathbf{w}}{\partial u} \right|^2 + \beta(u) \left| \frac{\partial^2 \mathbf{w}}{\partial u^2} \right|^2 \right) du \quad \ldots (2.29)$$

Where the first term inside the integral controls the tension by minimizing the length of the curve and therefore prevents the curve to stretch. The second term controls the rigidity by minimizing the curvature and therefore it prevents the curve to bend. The weights $\alpha(u)$ and $\beta(u)$ are usually chosen to be constant with respect to *u*. The force is defined as:

$$\mathbf{f}_{spline}(\mathbf{w}) = \frac{\partial}{\partial u}\left(\alpha(u)\frac{\partial \mathbf{w}}{\partial u}\right) - \frac{\partial^2}{\partial u^2}\left(\beta(u)\frac{\partial^2 \mathbf{w}}{\partial u^2}\right) \quad \ldots (2.30)$$

The equivalent equations to (2.2), (2.29) and (2.30) for 3D segmentation are presented in [9], where another factor which controls the amount of twisting is added. In order to reduce the degrees of freedom, nodes can be prevented from moving between slices as proposed in [9]. A more simplified method involves joining 2D segmented contours as in [9] [10]. The intermediate cross section is chosen for performing a first segmentation and each solution is propagated as the initial contour for the neighbor section.

2.5.2. B-Spline

B-Splines are proposed in [24] as an efficient and natural way for representing smoothly curved objects. B-Splines are piecewise cubic polynomials which shape is driven by control points, even though they do not interpolate between them. Given *n* control points $\mathbf{p}_i$ for $i = 1 \ldots n$, the curve *C* is defined as a collection of functions $\mathbf{w}_i(u):[0,1] \rightarrow \Re^2$ which return the position on the curve for a value of *u* in the segment from $\mathbf{p}_{i-1}$ to $\mathbf{p}_i$:

$$\mathbf{w}_i(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \begin{bmatrix} -1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/6 & 2/3 & 1/6 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-2} \\ \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix} \quad \ldots (2.31)$$

$$\text{for } s \in [0,1] \wedge i = 1 \ldots n$$

Where $\mathbf{p}_{-1} = \mathbf{p}_{n-1}$, $\mathbf{p}_0 = \mathbf{p}_n$ and $\mathbf{p}_{n+1} = \mathbf{p}_0$. The energy function and force follow the same derivation as in (2.29) and (2.30). One of the drawbacks of this parameterization is that it is not suitable for interactive delineation of the contour, since it does not interpolate between the control points [15].

2.5.3. Hermite

B-Splines are less suited for objects with sharp corners [15]. Hermite contours are piecewise cubic polynomials which interpolate control points as well as allow specifying tangent vectors. Therefore they can efficiently represent both smooth and sharp contours by the adjustment of the tangent vector parameters. Given n control points $\mathbf{p}_i$ and tangents $\mathbf{t}_i$ at the control points for $i = 1\ldots n$, the curve C is defined as a collection of functions $\mathbf{w}_i(u):[0,1]\rightarrow\Re^2$ which return the position on the curve for a value of u in the segment from $\mathbf{p}_{i-1}$ to $\mathbf{p}_i$:

$$\mathbf{w}_i(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{t}_{i-1} \\ \mathbf{t}_i \end{bmatrix} \quad \ldots (2.32)$$
$$\text{for } s \in [0,1] \wedge i = 1 \ldots n$$

Where $\mathbf{p}_0 = \mathbf{p}_n$ and $\mathbf{t}_0 = \mathbf{t}_n$. It can be observed that under this definition $\mathbf{w}_i(0) = \mathbf{p}_{i-1}$, $\mathbf{w}_i(1) = \mathbf{p}_i$, $\partial\mathbf{w}_i/\partial s(0) = \mathbf{t}_{i-1}$, $\partial\mathbf{w}_i/\partial s(1) = \mathbf{t}_i$. The energy function and force follow the same derivation as in (2.29) and (2.30).

2.5.4. Superquadric

A model which accounts for global as well as local deformations is proposed in [30]. The global shape is modeled by a deformable superquadric $\mathbf{q}(u,v)$ in $\Re^3$ with three aspect ratios a_1, a_2 and a_3 and the squareness parameters b_1 and b_2:

$$\mathbf{q}(u,v) = \begin{bmatrix} a_1 \operatorname{sgn}(\cos u)\left|\cos u\right|^{b_1} \operatorname{sgn}(\cos v)\left|\cos v\right|^{b_2} \\ a_2 \operatorname{sgn}(\cos u)\left|\cos u\right|^{b_1} \operatorname{sgn}(\sin v)\left|\sin v\right|^{b_2} \\ a_3 \operatorname{sgn}(\sin u)\left|\sin u\right|^{b_1} \end{bmatrix}$$
$$\text{for } u \in [-\pi/2, \pi/2] \wedge v \in [\pi, \pi]$$

The aspect ratios allow stretching and shrinking the superquadric with respect to the three different axes. For aspect ratios $a_1 = a_2 = a_3 = 1$ and squareness parameters $b_1 = b_2 = 1$, a sphere is generated. For squareness parameters smaller than one, a rounded cube is generated. For squareness parameters greater than one, a diamond shape is generated. Additional parameters are added into the model in order to account for global deformations, such as a scaling factor *s*, a rotation matrix **R** and a translation vector **t**. In order to account for local deformations, a displacement function **d** is also imposed. Therefore, the curve *C* is defined by the function **w**(*u*,*v*) as:

$$\begin{aligned}&\mathbf{w}(u,v) = \mathbf{t} + \mathbf{R}(s\mathbf{q}(u,v) + \mathbf{d}(u,v)) \\ &\text{for } u \in [-\pi/2, \pi/2] \wedge v \in [\pi, \pi]\end{aligned} \quad \ldots (2.33)$$

In [30] the rotation matrix **R** is modeled as a quaternion (r_1, r_2, r_3, r_4) and the displacement function **d** is expressed as a weighted sum of basis functions. The parameters which govern the global deformation $(a_1 \ldots a_3, b_1, b_2, s, t_1 \ldots t_3, r_1 \ldots r_4)^T$ are allow to change freely in order to account for as much as the data as possible. Consequently, the authors do not impose any energy function over the behavior of these parameters. Although, the local deformation **q**(*u*,*v*) must be small and continuous, which is expressed as minimizing the function:

$$E_{squad}(C) = \int_{-\pi}^{\pi} \int_{-\pi/2}^{\pi/2} \left(\alpha(u,v) \left| \mathbf{d}(u,v) \right|^2 + \beta(u,v) \left(\left| \frac{\partial \mathbf{d}}{\partial u} \right|^2 + \left| \frac{\partial \mathbf{d}}{\partial v} \right|^2 \right) \right) du dv \quad \ldots (2.34)$$

Where the first term allows minimizing the amount of local deformation, and the second term controls the local variation of the deformation.

2.5.5. Simplex Mesh

The use of simplex meshes for representing contours in $\Re^3$ is proposed in [14]. Simplex meshes allow smooth deformations in a simple and efficient manner, since they have constant vertex connectivity. In 2-simplex meshes, each vertex is shared by only three edges, therefore each vertex **w** has three neighbor vertices $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$.

A tangential force allows controlling the vertex position with respect to its three neighbors in the tangent plane. Let ε_1, ε_2 and ε_3 be the barycentric coordinates of the projection $\mathbf{p}(\mathbf{w})$ of the vertex **w** into the triangle $\Delta(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ in the normal direction, such that $\mathbf{p}(\mathbf{w}) = \varepsilon_1 \mathbf{w}_1 + \varepsilon_2 \mathbf{w}_2 + \varepsilon_3 \mathbf{w}_3$ where by definition $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = 1$. The tangential force is defined as:

$$\mathbf{f}_{stgt}(\mathbf{w}) = \tilde{\mathbf{p}}(\mathbf{w}) - \mathbf{p}(\mathbf{w}) = (\tilde{\varepsilon}_1 - \varepsilon_1)\mathbf{w}_1 + (\tilde{\varepsilon}_2 - \varepsilon_2)\mathbf{w}_2 + (\tilde{\varepsilon}_3 - \varepsilon_3)\mathbf{w}_3 \quad \ldots (2.35)$$

Where $\tilde{\varepsilon}_1$, $\tilde{\varepsilon}_2$ and $\tilde{\varepsilon}_3$ are the desired barycentric coordinates. In order to have vertices uniformly spread over the surface of a simplex mesh, the control parameters can be set to $\tilde{\varepsilon}_1 = \tilde{\varepsilon}_2 = \tilde{\varepsilon}_3 = 1/3$.

A normal force allows controlling the mean curvature of the surface through the simplex angle. The normal vector **n**(**w**) is assumed to be normal to the triangle $\Delta(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$. Also, let h(**w**) be the height of the vertex **w**, which is the distance between the vertex and its projection into the triangle $\Delta(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$, such that $\mathbf{w} = \mathbf{p}(\mathbf{w}) - h(\mathbf{w})\mathbf{n}(\mathbf{w})$. As described in [14], the height of the vertex can be expressed as a function of the simplex angle φ which measures the local curvature at the vertex **w**. The simplex angle is computed by first finding both a circle which crosses the vertices $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ as well as a sphere which crosses the vertices **w**, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$. The simplex angle $\varphi = \pi/2$ indicates that the center of the circle coincides with the center of the sphere, $\varphi < \pi/2$ indicates that the center of the circle is in front of the center of the sphere, and $\varphi > \pi/2$ indicates that the center of the circle is behind the center of the sphere. Finally, the normal force is defined as:

$$\mathbf{f}_{snor}(\mathbf{w}) = \tilde{\mathbf{w}} - \mathbf{w} = (-h(\tilde{\varphi}) + h(\varphi))\mathbf{n}(\mathbf{w}) \quad \ldots (2.36)$$

Where $\tilde{\varphi}$ is the desired simplex angle. As explained in [14], several choices can be selected for this control parameter. The desired simplex angle can be set to $\tilde{\varphi} = \varphi$ in order to nullify the normal force and make the surface freely bend around the vertex. The angle can be set to $\tilde{\varphi} = 0$ in order to minimize the curvature. By setting the angle to an arbitrary constant for every vertex, only global deformations are allowed up to a rotation, translation and scale transformations. The desired simplex angle can also be set to the average of the neighbor vertices angles in order to entail curvature continuity over the surface.

2.5.6. Fourier Decomposition

A probabilistic model based on a Fourier parameterization of training curves is proposed in [27]. This allows a compact representation of smooth shapes where the first terms in the Fourier expansion describe global properties of the shape such as translation, while the following terms describe local deformations. Let *C* be a curve defined as a function $\mathbf{w}(u):[0,1] \rightarrow \Re^2$, such that for every value of *u*, **w**(*u*) is the position on the curve associated with that value. The approximation of the curve *C* by using the first *n* terms in the Fourier expansion is given by:

$$\begin{aligned}
&\mathbf{w}(u) \approx \mathbf{a}_1 + \sum_{k=1}^{n} \left[\mathbf{a}_{2k} \cos(2\pi k u) + \mathbf{a}_{2k+1} \sin(2\pi k u)\right] \quad \text{for } u \in [0,1] \\
&\mathbf{a}_1 = \frac{1}{2\pi} \int_0^1 \mathbf{w}(u)\, du \\
&\mathbf{a}_{2k} = \frac{1}{\pi} \int_0^1 \mathbf{w}(u) \cos(2\pi k u)\, du \quad \text{for } k = 1 \ldots n \\
&\mathbf{a}_{2k+1} = \frac{1}{\pi} \int_0^1 \mathbf{w}(u) \sin(2\pi k u)\, du \quad \text{for } k = 1 \ldots n
\end{aligned} \qquad \ldots (2.37)$$

Where **A** is the parameter vector of Fourier coefficients $\mathbf{a}_k \in \Re^2$ for *k* = 1…2*n*+1, and *n* is usually chosen between four and six. Several images are manually segmented in a training phase in order to capture the statistical variability of shapes and to generate an energy function which allows measuring the fitness of a curve with respect to the training data. Statistical independence is assumed between Fourier coefficients as well as between components of one coefficient. Therefore, the probability of the parameter vector **A** is expressed as:

$$p(\mathbf{A}) = \prod_{k=1}^{2n+1} p(\mathbf{a}_k) = \prod_{k=1}^{2n+1} \prod_{j=1,2} p(a_{k,j})$$

Where each component is assumed to follow a Normal distribution such that $p(a_{k,j}) \sim N(\mu_{k,j}, \sigma_{k,j}^2)$ for *k* = 1…2*n*+1, *j* = 1,2. Since the maximization of this expression corresponds to the minimization of its negative logarithm, the energy function is defined as:

$$E_{fdec}(C) = -\sum_{k=1}^{2n+1} \sum_{j=1,2} \log p(a_{k,j}) \quad \ldots (2.38)$$

2.5.7. Active Shape Model

Landmark based shape priors are proposed in [12] in order to represent shapes as a weighted sum of few terms involving the most significant variability in the shape population. Shapes are parameterized by a fixed set of landmarks. These landmarks are manually identified on several training images. Subsequently, these shapes are aligned to one another with respect to scaling, rotation and translation by using the *Procrustes method*, which allows analyzing the landmark variability in a common coordinate frame. Given N aligned shapes $\mathbf{S}_i = (x_{i,1}, y_{i,1} ... x_{i,L}, y_{i,L})^T$ for i = 1…N, where L is the number of landmarks in the shape. The mean shape $\overline{\mathbf{S}}$ and covariance matrix $\hat{\mathbf{S}}$ are computed as:

$$\overline{\mathbf{S}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{S}_i$$

$$\hat{\mathbf{S}} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{S}_i - \overline{\mathbf{S}})(\mathbf{S}_i - \overline{\mathbf{S}})^T$$

By using *principal component analysis*, the eigenvectors $\hat{\mathbf{S}}_i$ for i = 1…n corresponding to the largest n eigenvalues λ_i of the covariance matrix $\hat{\mathbf{S}}$ describe the most significant variability of the landmarks in the training data. Therefore, any shape **S** can be approximated as the mean shape plus a weighted sum of the first n eigenvectors:

$$\mathbf{S}(\mathbf{p}) = \overline{\mathbf{S}} + \sum_{i=1}^{n} p_i\hat{\mathbf{S}}_i \quad \text{… (2.39)}$$

Where p_i for i = 1…n are the shape parameters. Additional parameters are added into the model in order to account for global transformations, such as a scaling factor s, a rotation matrix **R** with angle θ and a translation vector **t**. A landmark on the contour C can be calculated from a landmark (x_i,y_i) in the undeformed common coordinate frame as:

$$\mathbf{l}_i(\mathbf{p}) = s\mathbf{R}(\theta)(x_i, y_i)^T + \mathbf{t} \quad \text{for } (x_i, y_i) \in \mathbf{S}(\mathbf{p})$$

This landmark set $\mathbf{L}(\mathbf{p}) = (\mathbf{l}_1(\mathbf{p}), \mathbf{l}_2(\mathbf{p}), ..., \mathbf{l}_L(\mathbf{p}))^T$ is then used to deform the contour by using a warping function $\mathbf{w}(\mathbf{L}, \mathbf{x}) : \Re^{2L} \times \Re^2 \to \Re^2$, which maps a pixel **x** from the mean shape $\overline{\mathbf{S}}$ into the current shape of the curve *C*.

No energy function is defined in [12] since the global parameters are allowed to change freely and the shape parameters p_i are only constrained not to fall outside the range $[-3\sqrt{\lambda_i}, +3\sqrt{\lambda_i}]$. Since the eigenvalues λ_i for i = 1…n describe the amount of variance for each eigenvector $\hat{\mathbf{S}}_i$, this rule is equivalent to allow the shape to vary not more than three standard deviations from the mean shape.

2.5.8. Statistical Shape Model

An energy function is built upon (2.39) in [32] by collecting global and shape parameters into one parameter set $\mathbf{a} = (s, \theta, t_1, t_2, p_1 \ldots p_n)^T$ where s is the scaling factor, θ is the rotation angle, **t** is the translation vector and $p_1 \ldots p_n$ are the shape parameters as described in (2.39). Since the shape parameters p_i are weights of different orthogonal eigenvectors, they are statistically independent by definition. Also, statistically independence is assumed between scaling, rotation and translation parameters. Therefore, the probability of the parameter vector **a** can be expressed as:

$$p(\mathbf{a}) = \prod_{k=1}^{n+4} p(\mathbf{a}_k)$$

Where each component is assumed to follow a Normal distribution such that $p(a_k) \sim N(\mu_k, \sigma_k^2)$ for k = 1…n+4. Since the maximization of this probability corresponds to the minimization of its negative logarithm, the energy function is defined as:

$$E_{ssm}(C) = -\sum_{k=1}^{n+4} \log p(a_k) \quad \ldots (2.40)$$

2.6. Unique Solution and Convergence

The main disadvantage of deformable models is that convexity properties of the energy function are poorly understood. Due to this fact, solutions are often locally rather than globally optimal [13].

As stated in the original paper [20], *scale-space theory* can be used in order to avoid non-significant local minima when performing image segmentation. This requires a process for blurring the image to a controllable degree in order to compute energies such as the line (2.3) and edge detectors (2.4). The Gaussian filter is the only convolution kernel which meets the *minimum-maximum principle* as well as the *semi-group property* [18]. The first principle states that local maximum or minimum must not increase or decrease, respectively. The second property states that the process can start at any scale and still get the same scale space. Therefore, [20] proposed using a Gaussian filter with standard deviation σ. The value for σ should be decreased for successive steps in the iterative energy minimization. Even though the weakness of this approach is that there is not any established theorem for how to schedule changes in σ, which leads to unreliable results [35].

Proof of unique solution and convergence conditions are presented in [5] for *geodesic active contours*. This class of contours only includes the edge detector as *external energy* (2.4) as well as the tension term in the *internal energy* (2.29) in order to minimize the curve length. If the initial contour encloses the object of interest, the curve will evolve to the object boundaries. But notice that since the proof was presented in the framework of *geometric deformable models*, the topology of the curve evolves freely without any control of the final shape.

Convexity of the energy function is analyzed in [13] for the *finite differences approximation* of deformable models. A general boundary-based *external energy* is assumed as well as the *internal energy* (2.29). As a remarkable result it is demonstrated that the convexity of the energy function depends on the parameters $\alpha(u)$ and $\beta(u)$ of the *internal energy* (2.29) which control the tension and rigidity of the curve. The feasible values for the regularization parameters are determined by the nature of the *external energy*. Local characteristics of the object of interest such as its curvature also affect the convexity of the formulation. This phenomenon produces in practice a fast convergence in some regions of the object and poor convergence in some other ones.

Speed of convergence is analyzed in [31] for the *finite differences approximation* of deformable models. A general boundary-based *external energy* is assumed as well as the *internal energy* (2.29). As a main result it is theoretically proven that the speed of convergence depends on the parameters $\alpha(u)$ and $\beta(u)$ of the *internal energy* (2.29) which allows controlling the tension and rigidity of the deformable model.

Chapter 3. The Proposed Deformable Model

3.1. External Energy

3.1.1. Point Classifier

Definition 3.1:

Let Ω^*_{in} and Ω^*_{out} be the true object and background regions respectively. A *point classifier* $f(\mathbf{w}):\Omega \to [0,1]$ is a continuous function such that:

$$f(\mathbf{w}) = \Pr[\mathbf{w} \in \Omega^*_{in}] \quad \ldots (3.1)$$

from (3.1) it can be easily derived:

$$\Pr[\mathbf{w} \in \Omega^*_{out}] = 1 - \Pr[\mathbf{w} \in \Omega^*_{in}] = 1 - f(\mathbf{w}) \quad \ldots (3.2)$$

Assumption 3.1:

Since the object is usually completely contained in the image, it is safe to assume that *f*(**w**) = 0 for every $\mathbf{w} \notin \Omega$.

Lemma 3.1:

In order to decide to which region a point **w** belongs to, the following rule is used:

$$\begin{array}{l} \mathbf{w} \in \Omega^*_{in} \quad \text{if } f(\mathbf{w}) \geq 0.5 \\ \mathbf{w} \in \Omega^*_{out} \quad \text{if } f(\mathbf{w}) < 0.5 \end{array} \quad \ldots (3.3)$$

Proof:

For a point **w**, the best action is: “decide that $\mathbf{w} \in \Omega^*_{in}$ if $\Pr[\mathbf{w} \in \Omega^*_{in}] \geq \Pr[\mathbf{w} \in \Omega^*_{out}]$” and “decide $\mathbf{w} \in \Omega^*_{out}$ if $\Pr[\mathbf{w} \in \Omega^*_{in}] < \Pr[\mathbf{w} \in \Omega^*_{out}]$”. By replacing (3.1) and (3.2), the rule (3.3) is found.

$\sqcup$

3.1.2. Fuzzy Sets

Definition 3.2:

Let **T** be a set of parameters which control the shape of the contour *C*(**T**) and therefore the regions $\Omega_{in}(\mathbf{T})$ and $\Omega_{out}(\mathbf{T})$. A *region indicator function* $H(\mathbf{w},\mathbf{T}):\Omega \to \{0,1\}$ is defined as:

$$H(\mathbf{w},\mathbf{T}) = \begin{cases} 1 & \text{if } \mathbf{w} \in \Omega_{in}(\mathbf{T}) \\ 0 & \text{if } \mathbf{w} \in \Omega_{out}(\mathbf{T}) \end{cases} \quad \ldots (3.4)$$

Theorem 3.1:

In order to maximize the *crisp probability* that the contour $C(\mathbf{T})$ accurately separates Ω into two regions $\Omega_{in}(\mathbf{T})$ and $\Omega_{out}(\mathbf{T})$, the following energy function should be minimized:

$$E_{fuz}(\Omega \mid \mathbf{T}) = \int_{\Omega_{in}(\mathbf{T})} (1 - 2f(\mathbf{w},\mathbf{T}))d\mathbf{w} + \int_{\Omega} f(\mathbf{w},\mathbf{T})d\mathbf{w} \quad \ldots (3.5)$$

Notice that $f(\mathbf{w},\mathbf{T})$ is used instead of $f(\mathbf{w})$ in order to show the possible dependence of the *point classifier* on the evolution of the regions.

Proof:

In *fuzzy sets theory* [1], $\Pr[\mathbf{w} \in \Omega^*_{in}]$ and $\Pr[\mathbf{w} \in \Omega^*_{out}]$ can be interpreted as *membership functions*. The *crisp probability* that the contour $C(\mathbf{T})$ accurately separates Ω into two regions $\Omega_{in}(\mathbf{T})$ and $\Omega_{out}(\mathbf{T})$ should be maximized:

$$\Pr[\Omega \mid \mathbf{T}] = \int_{\Omega_{in}(\mathbf{T})} \Pr[\mathbf{w} \in \Omega^*_{in}]P(\mathbf{w})d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T})} \Pr[\mathbf{w} \in \Omega^*_{out}]P(\mathbf{w})d\mathbf{w}$$

Where $P(\mathbf{w})$ is the probability of occurrence of the point $\mathbf{w}$. Since by *probability theory* $\int_{\Omega} P(\mathbf{w})d\mathbf{w} = 1$ and since every point over the region Ω is equally probable, we can conclude that $P(\mathbf{w}) = 1/|\Omega|$ and therefore:

$$\Pr[\Omega \mid \mathbf{T}] = \frac{\int_{\Omega_{in}(\mathbf{T})} \Pr[\mathbf{w} \in \Omega^*_{in}]d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T})} \Pr[\mathbf{w} \in \Omega^*_{out}]d\mathbf{w}}{|\Omega|}$$

In the previous formula, the denominator can be cancelled since it is constant with respect to $\mathbf{T}$. Also, by replacing (3.1) and (3.2):

$$\Pr[\Omega \mid \mathbf{T}] \propto \int_{\Omega_{in}(\mathbf{T})} f(\mathbf{w},\mathbf{T})d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T})} (1 - f(\mathbf{w},\mathbf{T}))d\mathbf{w}$$

This formula can be rewritten by using the *region indicator function* defined in (3.4), since $H(\mathbf{w},\mathbf{T})$ is not zero only inside $\Omega_{in}(\mathbf{T})$ and $(1 - H(\mathbf{w},\mathbf{T}))$ is not zero only inside $\Omega_{out}(\mathbf{T})$:

$$\Pr[\Omega \mid \mathbf{T}] \propto \int_{\Omega} \left[f(\mathbf{w},\mathbf{T})H(\mathbf{w},\mathbf{T}) + (1 - f(\mathbf{w},\mathbf{T}))(1 - H(\mathbf{w},\mathbf{T})) \right] d\mathbf{w}$$

By reordering the factors in this expression:

$$\Pr[\Omega \mid \mathbf{T}] \propto \int_{\Omega} \left[(2f(\mathbf{w},\mathbf{T})-1)H(\mathbf{w},\mathbf{T}) - f(\mathbf{w},\mathbf{T}) \right] d\mathbf{w} + |\Omega|$$

The second integral is constant with respect to **T**, therefore it can be removed:

$$\Pr[\Omega \mid \mathbf{T}] \propto \int_{\Omega} \left[(2f(\mathbf{w},\mathbf{T})-1)H(\mathbf{w},\mathbf{T}) - f(\mathbf{w},\mathbf{T}) \right] d\mathbf{w}$$

Since $H(\mathbf{w},\mathbf{T})$ is not zero only inside $\Omega_{in}(\mathbf{T})$, the first term can be integrated only over the region $\Omega_{in}(\mathbf{T})$. Finally, the maximization of the previous formula is equivalent to the minimization of its negative.

□

Observation 3.1:

The *external energy* (3.5) can also be derived from the framework (2.24) by making $k_{in}(\mathbf{w}) = 1 - f(\mathbf{w},\mathbf{T})$ and $k_{out}(\mathbf{w}) = f(\mathbf{w},\mathbf{T})$.

3.1.3. Gradient Descent

Let $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_n)^T$ be a set of parameters such that $\mathbf{t}_i \in \Re^d$ for i = 1…n. The minimization of the energy function (3.5) can be done by using the *gradient descent method*. Given initial values for the set of parameters $\mathbf{T}^{(0)}$ and the step size γ, the next approximations of the solution **T*** which minimizes (3.5) are computed as:

$$\mathbf{T}^{(k+1)} = \mathbf{T}^{(k)} + \gamma \nabla E \quad \text{for } k \geq 0 \wedge \gamma > 0$$
$$\nabla E = \begin{bmatrix} \frac{\partial E}{\partial t_{1,1}} & \cdots & \frac{\partial E}{\partial t_{1,d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial t_{n,1}} & \cdots & \frac{\partial E}{\partial t_{n,d}} \end{bmatrix} \qquad \text{… (3.6)}$$

Theorem 3.2:

For a *point classifier* $f(\mathbf{w},\mathbf{T})$ the update rule for the *gradient descent method* becomes:

$$\frac{\partial E_{fuz}}{\partial t_{i,j}} = \int_{C(\mathbf{T})} \left[(1 - 2f(\mathbf{w},\mathbf{T})) \frac{\partial H}{\partial t_{i,j}} \right] d\mathbf{w} - \int_{\Omega_{in}(\mathbf{T})} \frac{\partial f}{\partial t_{i,j}} d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T})} \frac{\partial f}{\partial t_{i,j}} d\mathbf{w} \quad \text{… (3.7)}$$
$$\text{for } i = 1...n \wedge j = 1...d$$

Proof:

By deriving (3.5) with respect to $t_{i,j}$ and by regrouping the terms:

$$\frac{\partial E_{fuz}}{\partial t_{i,j}} = \int_{\Omega} \left[(1-2f(\mathbf{w},\mathbf{T}))\frac{\partial H}{\partial t_{i,j}} - \frac{\partial f}{\partial t_{i,j}} H(\mathbf{w},\mathbf{T}) + \frac{\partial f}{\partial t_{i,j}}(1-H(\mathbf{w},\mathbf{T})) \right] d\mathbf{w}$$

For the first term in the integral, notice that $\partial H / \partial t_{i,j}$ is defined only in the contour $C(\mathbf{T})$ and it is zero everywhere else. Therefore the integral over the region Ω becomes the integral over $C(\mathbf{T})$. For the second and third term, recall that $H(\mathbf{w},\mathbf{T})$ is not zero only inside $\Omega_{in}(\mathbf{T})$ and $(1 – H(\mathbf{w},\mathbf{T}))$ is not zero only inside $\Omega_{out}(\mathbf{T})$.

$\sqcup$

Observation 3.2:

The *external force* (3.7) can also be derived from the framework (2.25) by making $k_{in}(\mathbf{w}) = 1 – f(\mathbf{w},\mathbf{T})$ and $k_{out}(\mathbf{w}) = f(\mathbf{w},\mathbf{T})$. The first term of this *external force* is remarkably similar to the one used for experiments in [36], even though not any formula derivation neither theoretical proof was presented there. Authors defined it as a pressure force as (2.9) with a weight inside the range $[-1,+1]$.

3.1.4. Axis Independent Warps

Definition 3.3:

Let $\mathbf{w}(\mathbf{x},\mathbf{T}) : \Re^d \times \Re^{n \times d} \to \Re^d$ be a warping function which allows calculating the deformed location **w** for every point **x**. The warping function **w** is *axis independent* if:

$$\frac{\partial w_k}{\partial t_{i,j}} = 0 \quad \text{for } i = 1...n \wedge j \neq k \wedge j,k = 1...d \quad \cdots (3.8)$$

Lemma 3.2:

For a contour generated by an *axis independent* warping function **w(x,T)**, the *region indicator derivative* in (3.7) is given by:

$$\frac{\partial H}{\partial t_{i,j}} = n_j(\mathbf{w})\frac{\partial w_j}{\partial t_{i,j}} \quad \text{for } i = 1...n \wedge j = 1...d \quad \cdots (3.9)$$

Proof:

The chain rule can be used with the variable **w** in the formulation:

$$\frac{\partial H}{\partial t_{i,j}} = \frac{\partial H}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial t_{i,j}} = \sum_{k=1}^{d} \frac{\partial H}{\partial w_k} \frac{\partial w_k}{\partial t_{i,j}}$$

Since **w** is an *axis independent warping*, the derivatives with $j \neq k$ are zero. Therefore this equation is reduced to:

$$\frac{\partial H}{\partial t_{i,j}} = \frac{\partial H}{\partial w_j} \frac{\partial w_j}{\partial t_{i,j}}$$

Notice that $\partial H / \partial w_j$ is a Dirac delta pointing outside the region $\mathbf{\Omega}_{in}(\mathbf{T})$ in the contour $C(\mathbf{T})$ and zero everywhere else. Therefore, it is equal to the j-th component of the normal vector $\mathbf{n}(\mathbf{w})$.

⊔

3.1.5. Axis Independent Warps in 2.5D

In 2D and 3D since the dimensionality of the image and the dimensionality of the contour are the same, not any further analysis is needed. In the 2D case (d = 2) the images are composed by pixels and the contour is a curve. In the 3D case (d = 3) the images are composed by voxels and the contour is a surface. In the 2.5D case, we have a small set of 2D images spatially located in the 3D space, and the contour is a surface. This problem arises in the medical field, when a few 2D magnetic resonance images are used as input (for instance three horizontal and one vertical), and the reconstruction of the 3D surface of the organ is required.

In order to attack this problem, the first step is to find the intersection between the 3D surface and each image in order to compute the 2D contour $C(\mathbf{T})$ as shown in the following figure:

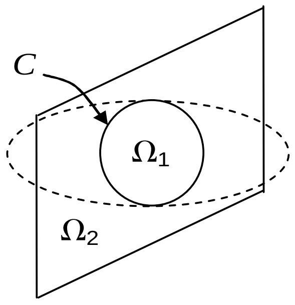


Figure 3.1: Contour and disjoint regions in 2.5D

The warping function $\mathbf{w}(\mathbf{x},\mathbf{T}):\Re^3\times\Re^{3n}\rightarrow\Re^2$ maps a point **x** in the 3D space to a point **w** in the 2D image under the influence of the set of *n* parameters **T** in the 3D space. Notice that the 3D surface deforms according to **T**, and the coordinate change from 3D to 2D can be expressed as a linear transformation. Therefore, the function **w** can be written as the composition of two functions such as:

$$\mathbf{w}(\mathbf{x},\mathbf{T})=\mathbf{w}^{2D}(\mathbf{w}^{3D}(\mathbf{x},\mathbf{T}))$$
$$\mathbf{w}^{2D}(\mathbf{w}^{3D})=\begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4}\end{bmatrix}\begin{bmatrix} w_1^{3D} \\ w_2^{3D} \\ w_3^{3D} \\ 1\end{bmatrix} \quad \ldots (3.10)$$

In this formula $\mathbf{w}^{3D}(\mathbf{x},\mathbf{T}):\Re^3\times\Re^{3n}\rightarrow\Re^3$ is the warping function working in the 3D space, and $\mathbf{w}^{2D}(\mathbf{w}^{3D}):\Re^3\rightarrow\Re^2$ is the linear transformation that maps a point in 3D space to a location in the 2D image, and it only depends on the location of the 2D image in the 3D space.

Lemma 3.3:

In the 2.5D case, for a *point classifier* $f(\mathbf{w}^{2D},\mathbf{T})$ and a contour generated by an *axis independent* warping function $\mathbf{w}^{3D}(\mathbf{x},\mathbf{T})$, the *region indicator derivative* in (3.7) is given by:

$$\frac{\partial H}{\partial t_{i,j}}=\left[m_{1,j}n_1(\mathbf{w}^{2D})+m_{2,j}n_2(\mathbf{w}^{2D})\right]\frac{\partial w_j^{3D}}{\partial t_{i,j}} \quad \text{for } i=1...n \wedge j=1...3 \quad \ldots (3.11)$$

Proof:

The chain rule can be used with the variables $\mathbf{w}^{2D}$ and $\mathbf{w}^{3D}$ in the formulation:

$$\frac{\partial H}{\partial t_{i,j}}=\frac{\partial H}{\partial \mathbf{w}^{2D}}\frac{\partial \mathbf{w}^{2D}}{\partial \mathbf{w}^{3D}}\frac{\partial \mathbf{w}^{3D}}{\partial t_{i,j}}=\frac{\partial H}{\partial \mathbf{w}^{2D}}\sum_{k=1}^{3}\frac{\mathbf{w}^{2D}}{\partial w_k^{3D}}\frac{\partial w_k^{3D}}{\partial t_{i,j}}$$

Since $\mathbf{w}^{3D}$ is an *axis independent warping*, the derivatives with $j \neq k$ are zero. Therefore this equation is reduced to:

$$\frac{\partial H}{\partial t_{i,j}}=\frac{\partial H}{\partial \mathbf{w}^{2D}}\frac{\partial \mathbf{w}^{2D}}{\partial w_j^{3D}}\frac{\partial w_j^{3D}}{\partial t_{i,j}}$$

The first two terms can be expanded as:

$$\frac{\partial H}{\partial t_{i,j}} = \left[\frac{\partial H}{\partial w_1^{2D}} \frac{\partial w_1^{2D}}{\partial w_j^{3D}} + \frac{\partial H}{\partial w_2^{2D}} \frac{\partial w_2^{2D}}{\partial w_j^{3D}} \right] \frac{\partial w_j^{3D}}{\partial t_{i,j}}$$

Notice that $\partial w_1^{2D} / \partial w_j^{3D}$ and $\partial w_2^{2D} / \partial w_j^{3D}$ are Dirac deltas pointing outside the region $\Omega_{in}(\mathbf{T})$ in the contour *C*(**T**) and zero everywhere else. Therefore, they are equal to the first and second components of the normal vector **n**(**w**^{2D}) respectively. Finally, $\partial H / \partial w_1^{2D}$ and $\partial H / \partial w_2^{2D}$ are replaced from the definition (3.10).

⊔

3.1.6. Region-Dependent Descriptors

Definition 3.4:

For a *point classifier f*(**w**,**T**) is based on region-dependent descriptors if can be expressed in the form *f*(**w**,**T**) = *f*(**w**,**p**(**T**)) where:

$$\begin{aligned} &\mathbf{p}(\mathbf{T}) = (p_1(\mathbf{T}),...,p_m(\mathbf{T}))^T \\ &p_k(\mathbf{T}) = \int_{R(\mathbf{T})} r_k(\mathbf{w}) d\mathbf{w} \quad \text{for } k = 1...m \end{aligned} \quad \dots (3.12)$$

Where **p**(**T**) is the vector of *m* region-dependent descriptors *p$_k$*(**T**) for *k* = 1…*m*. Each of these descriptors *p$_k$*(**T**) are associated with a function $r_k(\mathbf{w}) : \Re^d \to \Re$ which is evaluated on a region *R*(**T**) which can be $\Omega_{in}(\mathbf{T})$ or $\Omega_{out}(\mathbf{T})$. For instance, in order to generate a *point classifier* based on means and variances of intensity, descriptors such as area, sum of intensities and sum of square intensities are required. These descriptors correspond to the descriptor functions $r_k(\mathbf{w}) = 1$, $r_k(\mathbf{w}) = I(\mathbf{w})$ and $r_k(\mathbf{w}) = [I(\mathbf{w})]^2$ respectively.

Lemma 3.4:

The derivative for a *region-dependent descriptor p$_k$*(**T**) on $\Omega_{in}(\mathbf{T})$ is given by:

$$\frac{\partial p_k}{\partial t_{i,j}} = \int_{C(\mathbf{T})} r_k(\mathbf{w}) \frac{\partial H}{\partial t_{i,j}} d\mathbf{w} \quad \text{for } i = 1...n \wedge j = 1...d \quad \dots (3.13)$$

The derivative for a *region-dependent descriptor p$_k$*(**T**) on $\Omega_{out}(\mathbf{T})$ is given by:

$$\frac{\partial p_k}{\partial t_{i,j}} = -\int_{C(\mathbf{T})} r_k(\mathbf{w}) \frac{\partial H}{\partial t_{i,j}} d\mathbf{w} \quad \text{for } i = 1...n \wedge j = 1...d \quad \dots (3.14)$$

Proof:

Straightforward by deriving the region-dependent descriptor definition in (3.12) and by using the *region indicator function* defined in (3.4).

⊔

Lemma 3.5:

For a *point classifier f*(**w**,**T**) which is based on region-dependent descriptors. The integral of the *point classifier derivative* over a region *R*(**T**) which can be $\boldsymbol{\Omega}_{in}(\mathbf{T})$ or $\boldsymbol{\Omega}_{out}(\mathbf{T})$ needs to be computed in (3.7). For a small step size *h*, this integral can be approximated as:

$$\int_{R(\mathbf{T})} \frac{\partial f}{\partial t_{i,j}} d\mathbf{w} \approx \frac{1}{2h} \int_{R(\mathbf{T})} \left[f\left(\mathbf{w}, \mathbf{p}(\mathbf{T}) + h \frac{\partial \mathbf{p}}{\partial t_{i,j}} \right) - f\left(\mathbf{w}, \mathbf{p}(\mathbf{T}) - h \frac{\partial \mathbf{p}}{\partial t_{i,j}} \right) \right] d\mathbf{w}$$
$$\frac{\partial \mathbf{p}}{\partial t_{i,j}} = \left(\frac{\partial p_1}{\partial t_{i,j}}, \ldots, \frac{\partial p_m}{\partial t_{i,j}} \right)^T \qquad \ldots (3.15)$$
$$\text{for } i = 1 \ldots n \wedge j = 1 \ldots d$$

Proof:

Straightforward by replacing $\partial f / \partial t_{i,j}$ by its *finite differences approximation*, and extracting the factor 1/2*h* outside the integral.

⊔

3.2. Internal Energies

3.2.1. Topology Preservation

In order to add topology preservation to the deformation model, we need to produce a topology preservation measure which is not affected by any affine transformation (including but not limited to translation, rotation and scaling) over the prototypical configuration $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)^T$ such that $\mathbf{s}_i \in \Re^d$. Recall that $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n)^T$ such that $\mathbf{t}_i \in \Re^d$ are the current values for the set of parameters. The energy function for ensuring topology preservation is defined as:

$$E_{top}(\mathbf{T}, \mathbf{G}) = \sum_{i=1}^{n} \left\| \mathbf{t}_i - \mathbf{G} \cdot \begin{bmatrix} \mathbf{s}_i \\ 1 \end{bmatrix} \right\|^2$$

Where **G** is the $d\times(d+1)$ transformation matrix which minimizes E_{top}. In other words, **G** allows converting every point $\mathbf{s}_i$ into $\mathbf{t}_i$ for $i = 1\ldots n$ with a minimum error, such that if every $\mathbf{t}_i$ can be expressed as a linear transformation of each correspondent $\mathbf{s}_i$, the error would be zero. The update rule for the *gradient descent method* will become:

$$\frac{\partial E_{top}}{\partial t_{i,j}} = 2(t_{i,j} - \sum_{k=1}^{d} g_{j,k}s_{i,k} - g_{j,d+1}) \ \ldots\ (3.16)$$

Let $\mathbf{\Psi}(\mathbf{A},\mathbf{B})$ be a $d\times d$ matrix such that $\psi_{i,j}(\mathbf{A},\mathbf{B}) = \sum_{k=1}^{n} a_{k,i}b_{k,j}$ and let $\zeta(\mathbf{A})$ be a d-dimensional vector such that $\zeta_i(\mathbf{A}) = \sum_{k=1}^{n} a_{k,i}$. A rearrangement of the equations derived in [29] for computing the matrix **G** leads to the following linear equation system:

$$\begin{aligned} \mathbf{L}(\mathbf{S})\mathbf{G}^T &= \begin{bmatrix} \mathbf{\Psi}(\mathbf{S},\mathbf{T}) \\ \zeta^T(\mathbf{T}) \end{bmatrix} \\ \mathbf{L}(\mathbf{S}) &= \begin{bmatrix} \mathbf{\Psi}(\mathbf{S},\mathbf{S}) & \zeta(\mathbf{S}) \\ \zeta^T(\mathbf{S}) & n \end{bmatrix} \end{aligned} \ \ldots\ (3.17)$$

Observation 3.3:

Since only **T** changes trough all the optimization process and **S** is constant, it is more efficient to invert the matrix **L**(**S**) in the beginning of the process and to calculate **G** by matrix multiplication, rather than solving the linear equation system (3.17) at each iteration.

3.2.2. Temporal Coherence

Whether a sequence of d-dimensional images is used in the segmentation process, we can add temporal coherence so that the parameters do not drastically change between frames. This rule can be stated in terms minimizing the difference between the current frame and the previous one, as well as between the current frame and the next one. Let $\mathbf{T}^{(q)} = (\mathbf{t}_1^{(q)}, \mathbf{t}_2^{(q)}, ..., \mathbf{t}_2^{(q)})^T$ be the parameter set for the frame q such that $\mathbf{t}_i^{(q)} \in \Re^d$ for $i = 1\ldots n$. The energy function for ensuring temporal coherence is defined as:

$$E_{temp}(\mathbf{T}, q) = \sum_{i=1}^{n} \left(\left| \mathbf{t}_i^{(q)} - \mathbf{t}_i^{(q-1)} \right|^2 + \left| \mathbf{t}_i^{(q)} - \mathbf{t}_i^{(q+1)} \right|^2 \right)$$

The update rule for the *gradient descent method* will become:

$$\frac{\partial E_{temp}}{\partial t_{i,j}^{(q)}} = -2t_{i,j}^{(q-1)} + 4t_{i,j}^{(q)} - 2t_{i,j}^{(q+1)} \quad \dots (3.18)$$

3.3. Unique Solution and Convergence

As can be noticed, the topology preservation energy function is convex with respect to $t_{i,j}$ since its derivative (3.16) has only one minimum. In the same way, the temporal coherence energy function is convex with respect to $t_{i,j}^{(q)}$ due to the behavior of its derivative (3.18). Therefore, only the *external energy* (3.5) of the proposed model needs further convergence analysis.

For some specific warping functions, different values for the set of parameters **T** can lead to the same shape of the contour *C*(**T**). For instance in the case of *radial basis functions warping*, a circular shape parameterized by four equidistant points admits infinite different values for the parameters for generating the same shape by rotating the points around the center of the circumference. Therefore, the proof of unique solution and convergence should be based not on the values of the parameters but on the shape of the contour *C*(**T**).

3.3.1. Unique Solution

Definition 3.5:

Let **T*** be the values for a set of parameters which minimizes the energy function (3.5). According to (3.3), a *perfect point classifier f* *(**w**) is defined as:

$$\begin{aligned} f^*(\mathbf{w}) &= 0.5 \quad \text{if } \mathbf{w} \in C(\mathbf{T}^*) \\ f^*(\mathbf{w}) &> 0.5 \quad \text{if } \mathbf{w} \in \Omega_{in}(\mathbf{T}^*) \quad \dots (3.19) \\ f^*(\mathbf{w}) &< 0.5 \quad \text{if } \mathbf{w} \in \Omega_{out}(\mathbf{T}^*) \end{aligned}$$

Theorem 3.3:

For a *perfect point classifier f* *(**w**):

$$E_{fuz}(\Omega \mid \mathbf{T}^*) < E_{fuz}(\Omega \mid \mathbf{T}) \quad \text{for all } C(\mathbf{T}) \neq C(\mathbf{T}^*) \quad \dots (3.20)$$

This states that there is not any other contour that produces a greater value of the energy function, and therefore there is only one global minimum.

Proof:

By replacing (3.5) in the previous formula:

$$\int_{\Omega_{in}(\mathbf{T}^*)}(1-2f^*(\mathbf{w}))d\mathbf{w}+\int_{\Omega}f^*(\mathbf{w})d\mathbf{w}<\int_{\Omega_{out}(\mathbf{T})}(1-2f^*(\mathbf{w}))d\mathbf{w}+\int_{\Omega}f^*(\mathbf{w})d\mathbf{w}$$

Where the second terms on both sides can be removed. Notice that by definition (3.19) $f^*(\mathbf{w}) > 0.5$ in the region $\Omega_{in}(\mathbf{T}^*)$ and therefore $1 - 2f^*(\mathbf{w}) < 0$. In the same way, $f^*(\mathbf{w}) < 0.5$ in the region $\Omega_{out}(\mathbf{T}^*)$ and therefore $1 - 2f^*(\mathbf{w}) > 0$.

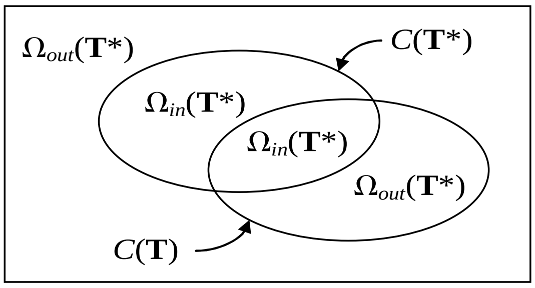

$\Omega_{out}(\mathbf{T}^*)$
$C(\mathbf{T}^*)$
$\Omega_{in}(\mathbf{T}^*)$
$\Omega_{in}(\mathbf{T}^*)$
$\Omega_{out}(\mathbf{T}^*)$
$C(\mathbf{T})$

Figure 3.2: Areas of integration

By noticing this fact, we can split the integral in the right side of the previous equation into two terms:

$$\int_{\Omega_{in}(\mathbf{T}^*)}(1-2f^*(\mathbf{w}))d\mathbf{w}<\int_{\Omega_{in}(\mathbf{T})-\Omega_{in}(\mathbf{T}^*)}(1-2f^*(\mathbf{w}))d\mathbf{w}+\int_{\Omega_{in}(\mathbf{T})\cap\Omega_{in}(\mathbf{T}^*)}(1-2f^*(\mathbf{w}))d\mathbf{w}$$

In the right side of the inequality, the first integral is performed in the region $\Omega_{in}(\mathbf{T})-\Omega_{in}(\mathbf{T}^*)\subset\Omega_{out}(\mathbf{T}^*)$ and therefore it is positive. Also, notice that the other two integrals are negative since they are performed in the regions $\Omega_{in}(\mathbf{T}^*)$ and $\Omega_{in}(\mathbf{T})\cap\Omega_{in}(\mathbf{T}^*)\subset\Omega_{in}(\mathbf{T}^*)$ respectively.

In the extreme case of the inequality, the positive term should be removed by choosing $\Omega_{in}(\mathbf{T})\subset\Omega_{in}(\mathbf{T}^*)$, and therefore $\Omega_{in}(\mathbf{T})-\Omega_{in}(\mathbf{T}^*)=\phi$ and $\Omega_{in}(\mathbf{T})\cap\Omega_{in}(\mathbf{T}^*)=\Omega_{in}(\mathbf{T})$. This converts the previous formula into:

$$\int_{\Omega_{in}(\mathbf{T}^*)}(1-2f^*(\mathbf{w}))d\mathbf{w}<\int_{\Omega_{in}(\mathbf{T})}(1-2f^*(\mathbf{w}))d\mathbf{w}$$

Finally, this is true since the formula being integrated is the same, and the region on the right side is contained into the region in the left side.

⊔

3.3.2. Convergence

Theorem 3.4:

For a *perfect point classifier f* *(**w**) and a convex contour:

$$\begin{aligned}&\frac{\partial E_{fuz}}{\partial t_{i,j}} = 0 \quad \text{if and only if } C(\mathbf{T}) = C(\mathbf{T}^*) \\ &\text{for } i = 1...n \wedge j = 1...d\end{aligned} \quad \ldots (3.21)$$

This states that the first derivative of the energy function is zero only in the global minimum. Consequently, not any local minimum exist and therefore the *gradient descent method* will return the correct contour.

Proof:

Since we do not have an explicit representation of the *perfect point classifier f* *(**w**) nor the warping function **w**(**x**,**T**), it is not possible to evaluate the zeroes of the first derivative. Therefore, the previous statement is divided into two equivalent statements:

$$\begin{aligned}&\frac{\partial E_{fuz}}{\partial t_{i,j}} = 0 \quad \text{for all } C(\mathbf{T}) = C(\mathbf{T}^*) \wedge \text{for all } i, j \\ &\text{for } i = 1...n \wedge j = 1...d\end{aligned} \quad \ldots (3.22)$$

and:

$$\begin{aligned}&\frac{\partial E_{fuz}}{\partial t_{i,j}} \neq 0 \quad \text{for all } C(\mathbf{T}) \neq C(\mathbf{T}^*) \wedge \text{for some } i, j \\ &\text{for } i = 1...n \wedge j = 1...d\end{aligned} \quad \ldots (3.23)$$

By expanding the formula (3.22) by using (3.7) and (3.9) and by replacing *C*(**T***) instead of *C*(**T**) since the contours are equivalent:

$$\int_{C(\mathbf{T}^*)} (1 - 2f^*(\mathbf{w}))n_j(\mathbf{w})\frac{\partial w_j}{\partial t_{i,j}} d\mathbf{w} - \int_{\Omega_{in}(\mathbf{T}^*)} \frac{\partial f^*}{\partial t_{i,j}} d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T}^*)} \frac{\partial f^*}{\partial t_{i,j}} d\mathbf{w} = 0$$

By definition (3.19) *f* *(**w**) = 0.5 in the contour *C*(**T***) and therefore 1 – 2 *f* *(**w**) = 0. Also, since the *perfect point classifier f* *(**w**) does not depend on the evolution of the regions $\partial f^* / \partial t_{i,j} = 0$. This fact makes the three integrals equal to zero. Therefore, the statement (3.22) is true.

On the other hand, the statement (3.23) can be proved by contradiction:

$$\frac{\partial E_{fuz}}{\partial t_{i,j}} = 0 \quad \text{for all}\, C(\mathbf{T}) \neq C(\mathbf{T}^*) \wedge \text{for all}\, i,j \qquad \ldots (3.24)$$
$$\text{for}\, i = 1...n \wedge j = 1...d$$

Let assume that we parameterize any d-dimensional point $\mathbf{w} \in C(\mathbf{T})$ in a lower dimensional space $U = [0,1] \times ... \times [0,1] \subset \Re^{d-1}$. In 2D images, the shape of a curve is parameterized by a variable u in the range $[0,1]$. In 3D images, the shape of a surface is parameterized by two variables u_1 and u_2 both of them in the range $[0,1]$. Let $\mathbf{x}(\mathbf{u}) : U \to \Re^d$ be the position of a point in the prototypical undeformed contour associated with the parameter **u**. Since C(**T**) is a *simple closed contour*, it allows a one-to-one parameterization such that only one position **x**(**u**) is related with a value for **u**, and viceversa [2]. Equations (3.7) and (3.9) can be rewritten as:

$$\frac{\partial E_{fuz}}{\partial t_{i,j}} = \int_U (1 - 2f^*(\mathbf{w}(\mathbf{u})))n_j(\mathbf{w}(\mathbf{u}))\frac{\partial w_j}{\partial t_{i,j}} d\mathbf{u} - \int_{\Omega_{in}(\mathbf{T})} \frac{\partial f^*}{\partial t_{i,j}} d\mathbf{w} + \int_{\Omega_{out}(\mathbf{T})} \frac{\partial f^*}{\partial t_{i,j}} d\mathbf{w}$$
$$\text{for}\, i = 1...n \wedge j = 1...d \qquad \ldots (3.25)$$
$$\mathbf{w}(\mathbf{u}) \equiv \mathbf{w}(\mathbf{x}(\mathbf{u}), \mathbf{T})$$

Since the *perfect point classifier f**(**w**) does not depend on the evolution of the regions $\partial f^* / \partial t_{i,j} = 0$ and the last two terms can be removed. Let $\mathbf{m}(\mathbf{u}) : U \to \Re^d$ be the normal vector pointing inside at the position **x**(**u**) in the same prototypical contour. The unit normal vector **n**(**w**(**u**)) pointing inside the region $\Omega_{in}(\mathbf{T})$ in the deformed contour is defined as:

$$\mathbf{n}(\mathbf{w}(\mathbf{u})) = \frac{\tilde{\mathbf{w}}(\mathbf{u}) - \mathbf{w}(\mathbf{u})}{\left|\tilde{\mathbf{w}}(\mathbf{u}) - \mathbf{w}(\mathbf{u})\right|}$$
$$\tilde{\mathbf{w}}(\mathbf{u}) \equiv \mathbf{w}(\mathbf{x}(\mathbf{u}) + \mathbf{m}(\mathbf{u}), \mathbf{T})$$

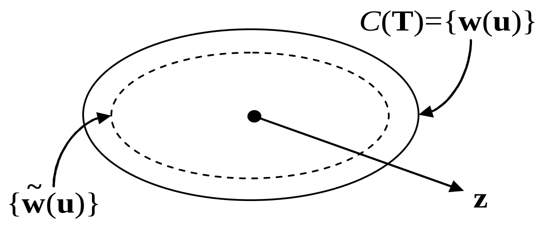
By replacing the previous formula in (3.25):

$$\frac{\partial E_{fuz}}{\partial t_{i,j}} = \int_U (1 - 2f^*(\mathbf{w}(\mathbf{u})))\frac{\partial w_j}{\partial t_{i,j}} \left[\frac{\tilde{w}_j(\mathbf{u}) - w_j(\mathbf{u})}{\left|\tilde{\mathbf{w}}(\mathbf{u}) - \mathbf{w}(\mathbf{u})\right|}\right] d\mathbf{u} \quad \text{for}\, i = 1...n \wedge j = 1...d$$

By replacing this in (3.24) and separating the term inside the brackets to both sides of the equality:

$$\int_U \xi(\mathbf{u})\tilde{w}_j(\mathbf{u})d\mathbf{u} = \int_U \xi(\mathbf{u})w_j(\mathbf{u})d\mathbf{u} \quad \text{for } i = 1...n \wedge j = 1...d$$
$$\xi(\mathbf{u}) \equiv \frac{(1-2f^*(\mathbf{w}(\mathbf{u})))\partial w_j / \partial t_{i,j}}{|\tilde{\mathbf{w}}(\mathbf{u}) - \mathbf{w}(\mathbf{u})|} \quad \ldots (3.26)$$

Notice that the points $\tilde{\mathbf{w}}(\mathbf{u})$ draw a shape which is smaller than the one drawn by the points $\mathbf{w}(\mathbf{u})$ which form the contour $C(\mathbf{T})$, as shown in the following figure:



$C(\mathbf{T})=\{\mathbf{w}(\mathbf{u})\}$
$\{\tilde{\mathbf{w}}(\mathbf{u})\}$
$\mathbf{z}$

Figure 3.3: Convex contour and its displaced version in normal direction

Also, if the deformed contour is convex, the first shape is always inside the second one. Moreover, if we assume the center **z** of the contour $C(\mathbf{T})$ as the center of our coordinate system, any line drawn with origin in **z** will intersect the contour created by the points $\tilde{\mathbf{w}}(\mathbf{u})$ before than the contour $C(\mathbf{T})$. Specifically, for lines traveling in the axis direction:

$$|\tilde{w}_j(\mathbf{u}) - z_j| < |w_j(\mathbf{u}) - z_j| \quad \text{for } j = 1...d \quad \ldots (3.27)$$

Notice that since $\tilde{\mathbf{w}}(\mathbf{u})$ is the displacement of a point in the contour in the normal direction of the prototypical undeformed contour, $\tilde{w}_j(\mathbf{u}) - z_j$ has zeroes in the same values for **u** than $w_j(\mathbf{u}) - z_j$ and they both have the same sign. By the additive property of the integrals, the term z_j can be added into both sides of equation (3.26):

$$\int_U \xi(\mathbf{u})(\tilde{w}_j(\mathbf{u}) - z_j)d\mathbf{u} = \int_U \xi(\mathbf{u})(w_j(\mathbf{u}) - z_j)d\mathbf{u} \quad \text{for } j = 1...d$$

Which leads to an absurd by the property (3.27).

⊔

Chapter 4. Implementation Issues

4.1. Radial Basis Functions Warping

Given two sets of *n* corresponding source and target points in a *d*-dimensional space called $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n)^T$ and $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_n)^T$ such that $\mathbf{s}_i \in \Re^d$ and $\mathbf{t}_i \in \Re^d$ for $i = 1 \ldots n$. We are looking for a warping function $\mathbf{w}(\mathbf{x}, \mathbf{T}) : \Re^d \times \Re^{n \times d} \to \Re^d$ which allows calculating the deformed location **w** for every point **x**, such that the source points are mapped exactly into the target points. For solving this problem the *radial basis functions warping* [3] is employed:

$$\mathbf{w}(\mathbf{x}, \mathbf{T}) = \sum_{j=1}^{n} \boldsymbol{\lambda}_j(\mathbf{S}, \mathbf{T}) \phi(|\mathbf{x} - \mathbf{s}_j|) + \boldsymbol{\lambda}_{n+1}(\mathbf{S}, \mathbf{T}) + \sum_{j=1}^{d} \boldsymbol{\lambda}_{n+j+1}(\mathbf{S}, \mathbf{T}) x_j \quad \ldots (4.1)$$

In the previous formula (4.1) each $\boldsymbol{\lambda}_j(\mathbf{S}, \mathbf{T}) \in \Re^d$ is a weight for the radial basis function $\phi(r) = r^2 \log(r)$ for $j = 1 \ldots n+d+1$. The last two terms account for the constant and linear portions of **w**. As it was stated before, we want a function **w**(**x**,**T**) such that it maps each source point to its correspondent target point:

$$\begin{aligned} &\mathbf{w}(\mathbf{s}_i, \mathbf{T}) = \mathbf{t}_i \quad \text{for } i = 1 \ldots n \\ &\sum_{j=1}^{n} \boldsymbol{\lambda}_j(\mathbf{S}, \mathbf{T}) = 0 \qquad\qquad \ldots (4.2) \\ &\sum_{j=1}^{n} \boldsymbol{\lambda}_j(\mathbf{S}, \mathbf{T}) s_{j,i} = 0 \quad \text{for } i = 1 \ldots d \end{aligned}$$

Let $\mathbf{0}_{p \times q}$ and $\mathbf{1}_{p \times q}$ be $p \times q$ matrices of zeroes and ones respectively. Let $\boldsymbol{\Lambda}(\mathbf{S}, \mathbf{T}) = (\boldsymbol{\lambda}_1(\mathbf{S}, \mathbf{T}), \boldsymbol{\lambda}_2(\mathbf{S}, \mathbf{T}), ..., \boldsymbol{\lambda}_n(\mathbf{S}, \mathbf{T}))^T$, $\tilde{\mathbf{T}} = (\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_n, \mathbf{0}_{d \times d+1})^T$ and $\boldsymbol{\Phi}(\mathbf{S})$ be a $n \times n$ matrix such that $\phi_{i,j}(\mathbf{S}) = \phi(|\mathbf{s}_i - \mathbf{s}_j|)$. Formulas (4.1) and (4.2) can be expressed as a linear equation system:

$$\begin{aligned} &\mathbf{K}(\mathbf{S}) \boldsymbol{\Lambda}(\mathbf{S}, \mathbf{T}) = \tilde{\mathbf{T}} \\ &\mathbf{K}(\mathbf{S}) = \begin{bmatrix} \boldsymbol{\Phi}(\mathbf{S}) & \mathbf{1}_{n \times 1} & \mathbf{S} \\ \mathbf{1}_{1 \times n} & \mathbf{0}_{d+1 \times d+1} & \\ \mathbf{S}^T & & \end{bmatrix} \ldots (4.3) \end{aligned}$$

Lemma 4.1:

Every *radial basis functions warping* **w** linearly depends on the set of parameters **T** and it is *axis independent*.

Proof:

Formula (4.3) can be rewritten as:

$$\begin{aligned}\boldsymbol{\Lambda}(\mathbf{S},\mathbf{T}) &= \mathbf{R}(\mathbf{S})\tilde{\mathbf{T}}\\ \mathbf{R}(\mathbf{S}) &= \mathbf{K}^{-1}(\mathbf{S})\end{aligned}$$

Let $\mathbf{r}_j(\mathbf{S}) \in \Re^{n+d+1}$ be the *j*-th row of the matrix **R**(**S**). The previous formula for the *j*-th weight can be written as:

$$\boldsymbol{\lambda}_j(\mathbf{S},\mathbf{T}) = \mathbf{r}_j(\mathbf{S})\tilde{\mathbf{T}} \quad \text{for } j = 1...n+d+1$$

Since the last *d*+1 rows of the matrix $\tilde{\mathbf{T}}$ are zeroes, the previous formula can be reduced to:

$$\boldsymbol{\lambda}_j(\mathbf{S},\mathbf{T}) = \sum_{i=1}^{n} r_{j,i}(\mathbf{S})\mathbf{t}_i \quad \text{for } j = 1...n+d+1 \;\ldots\; (4.4)$$

By replacing (4.4) in (4.1), the warping function **w** can be expressed in terms of the target points $\mathbf{t}_i$:

$$\begin{aligned}\mathbf{w}(\mathbf{x},\mathbf{T}) &= \sum_{i=1}^{n} \mathbf{t}_i b_i(\mathbf{x},\mathbf{S})\\ b_i(\mathbf{x},\mathbf{S}) &= \sum_{j=1}^{n} r_{j,i}(\mathbf{S})\phi(|\mathbf{x}-\mathbf{s}_j|) + r_{n+1,i}(\mathbf{S}) + \sum_{j=1}^{d} r_{n+j+1,i}(\mathbf{S})x_j\end{aligned}$$

Finally:

$$\left.\begin{aligned}\frac{\partial w_j}{\partial t_{i,j}} &= b_i(\mathbf{x},\mathbf{S}) \quad \text{for } i = 1...n \wedge j = 1...d\\ \frac{\partial w_k}{\partial t_{i,j}} &= 0 \quad \text{for } i = 1...n \wedge j \neq k \wedge j,k = 1...d\end{aligned}\right. \;\ldots\; (4.5)$$

⊔

Observation 4.1:

Since only **T** changes trough all the optimization process and **S** is constant, it is more efficient to invert the matrix **K**(**S**) in the beginning of the process and to recalculate $\boldsymbol{\Lambda}(\mathbf{S},\mathbf{T})$

by using (4.4) at each step, rather than solving the linear equation system (4.3) at each iteration.

4.2. Bayes Classifier

Remember that by definition (3.1) the *point classifier* *f*(**w**,**T**) represents the probability that the point **w** belongs to the true object region Ω_{in}^*. In order to make use of the Bayes rule, we introduce an equivalent notation:

$$\begin{aligned}&P(\Omega_k^* \mid \mathbf{w}) \equiv \Pr[\mathbf{w} \in \Omega_k^*]\\ &p(\mathbf{w} \mid \Omega_k^*) \equiv p_k(I(\mathbf{w}))\\ &\text{for } k = in, out\end{aligned}$$

Where $P(\Omega_k^* \mid \mathbf{w})$ is the so called *posterior probability* and $p(\mathbf{w} \mid \Omega_k^*)$ is the *likelihood*. The first one corresponds to the probability of belonging to a region given an intensity level, while the second one is the probability density function of the intensity levels on each region. Let $P(\Omega_k^*)$ the *prior probability*, the *point classifier* *f*(**w**,**T**) is defined as:

$$f(\mathbf{w},\mathbf{T}) = P(\Omega_{in}^* \mid \mathbf{w}) = \frac{p_{in}(I(\mathbf{w}))P(\Omega_{in}^*)}{p_{in}(I(\mathbf{w}))P(\Omega_{in}^*) + p_{out}(I(\mathbf{w}))P(\Omega_{out}^*)} \quad \ldots (4.6)$$

In our implementation of (4.6) we assumed that the *prior probabilities* are equal such that $P(\Omega_{in}^*) = P(\Omega_{out}^*) = 0.5$. The *likelihoods* follow a Normal distribution in both regions and can be modeled in two different ways. When no prior knowledge is available regarding the intensities for the object and background regions, the means and variances are approximated by using the *maximum likelihood estimation*. In this case, the sample means and variances approximate the real means and variances:

$$p_k(I(\mathbf{w})) \sim N(\mu_k(\mathbf{T}), \sigma_k^2(\mathbf{T})) \quad \text{for } k = in, out \quad \ldots (4.7)$$

When prior knowledge is available regarding the intensities for the object and background regions, the means can be approximated by using the *Bayesian parameter estimation*. In this case, we have an initial guess of the means $\mu_{0,k}$ with an uncertainty $\sigma_{0,k}^2$ for *k* = *in*,*out*. After *n* iterations, the sample means are averaged with the initial guess in order to approximate the real means:

$$p_k(I(\mathbf{w})) \sim N\left(\frac{n\sigma_{0,k}^2 \mu_k(\mathbf{T}) + \sigma_k^2(\mathbf{T})\mu_{0,k}}{n\sigma_{0,k}^2 + \sigma_k^2(\mathbf{T})}, \sigma_k^2(\mathbf{T})\right) \text{ for } k = in, out \quad \ldots (4.8)$$

Image classification is usually performed for discrete points, such as pixels or voxels. Bilinear interpolation is performed for 2D images in order to evaluate non-discrete locations. Trilinear interpolation is performed for 3D images.

4.3. Integral Approximation

A contour integral needs to be computed in order to compute the update rule for the *gradient descent method* as presented in (3.7). For segmenting 2D images, the contour is a curve which is approximated by using a set of connected segments. The integral over the contour is evaluated as the summation of the integral for each segment. If we assume that the size of each segment is smaller than the size of a pixel, the *trapezoidal rule* will generate a good approximation. Let $\mathbf{a}, \mathbf{b} \in \Re^2$ be the coordinates of the extremes of the segment $\ell(\mathbf{a}, \mathbf{b})$ and $g(\mathbf{w}) : \Re^2 \rightarrow \Re$ be the function to be integrated. The integral is approximated by:

$$\int_{\ell(\mathbf{a},\mathbf{b})} g(\mathbf{w}) d\mathbf{w} \approx |\mathbf{a} - \mathbf{b}| \frac{(g(\mathbf{a}) + g(\mathbf{b}))}{2} \quad \ldots (4.9)$$

For segmenting 3D images, the contour is a surface which is approximated by using a triangle mesh. The integral over the contour is evaluated as the summation of the integral for each triangle. If we assume that the size of each triangle is smaller than the size of a voxel, the *trapezoidal rule* will generate a good approximation. Let $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \Re^3$ be the coordinates of the vertices for the triangle $\Delta(\mathbf{a}, \mathbf{b}, \mathbf{c})$ and $g(\mathbf{w}) : \Re^3 \rightarrow \Re$ be the function to be integrated. The integral is approximated by:

$$\int_{\Delta(\mathbf{a},\mathbf{b},\mathbf{c})} g(\mathbf{w}) d\mathbf{w} \approx \frac{|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{b})|}{2} \frac{(g(\mathbf{a}) + g(\mathbf{b}) + g(\mathbf{c}))}{3} \quad \ldots (4.10)$$

The first part of the equation corresponds to the absolute value of the signed area of the triangle $\Delta(\mathbf{a}, \mathbf{b}, \mathbf{c})$. Finally, recall that in order to compute the update rule (3.7), the derivative of the *warping function* (3.9) needs to be computed as in (4.5). This formula returns a value which depends on the undeformed coordinates of the prototypical contour. Notice that we are evaluating the function only on known vertices of the contour, such as segment extremes and triangle vertices. Therefore, the underformed coordinate **x** for every deformed coordinate **w** after applying the *warping function* is known.

4.4. Additional Steps for 2.5D

As explained in 3.5, the first step in 2.5D segmentation is to find the intersection between the 3D surface and each 2D image in order to compute the 2D contour $C(\mathbf{T})$ as shown in figure 3.1. This is performed by intersecting every triangle in the deformed mesh with each 2D image. A triangle $\Delta(\mathbf{w}^a, \mathbf{w}^b, \mathbf{w}^c)$ intersects the image plane when two segments of the triangle intersect that plane, as shown in the following figure:

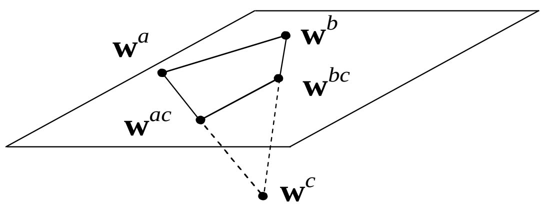


Figure 4.1: Intersection between a triangle and image plane

In the previous figure, the point $\mathbf{w}^{ac}$ is the intersection of the segment $\ell(\mathbf{w}^a, \mathbf{w}^c)$ with the image plane, and $\mathbf{w}^{bc}$ is defined in a similar way. By using these two coordinates, the integral of the segment $\ell(\mathbf{w}^{ac}, \mathbf{w}^{bc})$ is evaluated by using (4.9). The 3D normal vector of the triangle $\Delta(\mathbf{w}^a, \mathbf{w}^b, \mathbf{w}^c)$ is also projected into the image plane as needed in (3.11).

Finally, recall that in order to compute the update rule (3.7), the derivative of the *warping function* (3.11) needs to be computed as in (4.5). This formula returns a value which depends on the undeformed coordinates of the prototypical contour. Notice that we are not relying on known vertices of the contour due to the intersection process, but on the coordinates $\mathbf{w}^{ac}$ and $\mathbf{w}^{bc}$. Let $\mathbf{x}^a$, $\mathbf{x}^b$, $\mathbf{x}^c$ be the undeformed coordinates for the deformed coordinates $\mathbf{w}^a$, $\mathbf{w}^b$, $\mathbf{w}^c$, we can compute the linear approximation:

$$\mathbf{x}^{ac} \approx \mathbf{x}^{a} + \frac{\left|\mathbf{w}^{ac} - \mathbf{w}^{a}\right|}{\left|\mathbf{w}^{c} - \mathbf{w}^{a}\right|}(\mathbf{x}^{c} - \mathbf{x}^{a})$$

While $\mathbf{x}^{bc}$ is computed in a similar way. Notice that this is an approximation since the *radial basis functions warping* is not linear on the undeformed coordinates $\mathbf{x}$.

Chapter 5. Results

5.1. 2D Segmentation of an Image

The image to be segmented was immersed in the coordinate system $[-1,+1]\times[-1,+1]$. The initial configuration was set as the known solution after rotating 10° and translating (0.1, 0.1) units as shown in figure 5.1. The step size for the *gradient descent method* (3.6) was set to $\gamma = 0.05$ and the weights for the fuzzy set and the topology preservation energies were both set to 1.0. The step size for the derivative approximation (3.15) was set to h = 0.001. *Maximum likelihood estimation* as in (4.7) was used for approximating the means and variances for the object and background regions. Figure 5.2 shows the curve after 35 iterations.

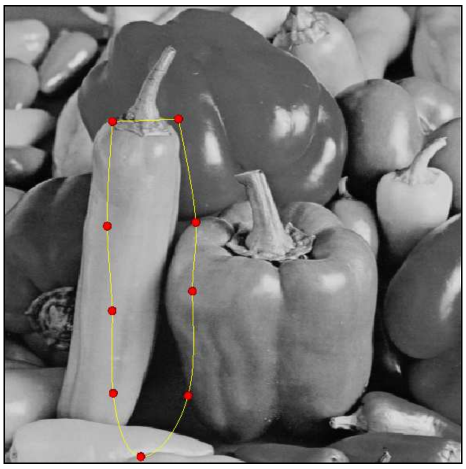


Figure 5.1: Initial configuration for the 2D segmentation

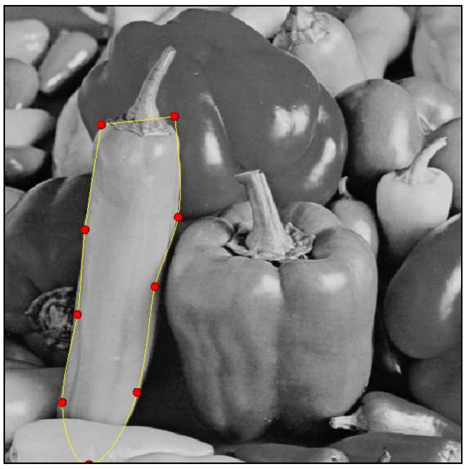


Figure 5.2: Final configuration for the 2D segmentation

5.2. 2.5D Segmentation of a Cardiac MRI Sequence

The sequence of 25 magnetic resonance images (MRI) to be segmented were immersed in the coordinate system $[-1,+1]\times[-1,+1]\times[-1,+1]$. The initial configuration for all frames was set as the known solution in the mid frame after rotating 5° and translating (0.1, 0.1, 0.1) units as shown in figure 5.3. The step size for the *gradient descent method* (3.6) was set to $\gamma = 0.01$ and the weights for the fuzzy set, topology preservation and temporal coherence energies were set to 1.0, 1.0 and 20.0 respectively. The step size for the derivative approximation (3.15) was set to h = 0.001. *Bayesian parameter estimation* as in (4.8) was used for approximating the means and variances for the object and background regions. The initial guesses of the means were set to $\mu_{0,in} = 0.4$ and $\mu_{0,out} = 0.05$ with an uncertainty $\sigma^2_{0,in} = \sigma^2_{0,out} = 0.025$. Figure 5.4 shows the surface after 80 iterations for three different frames.

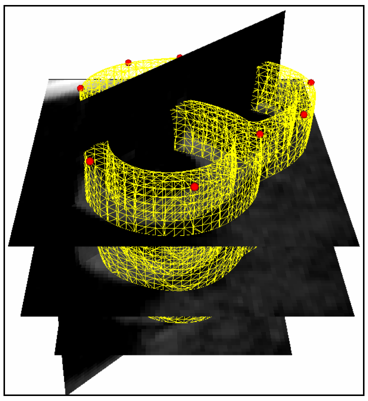


Figure 5.3: Initial configuration for the 2.5D segmentation (mid frame)

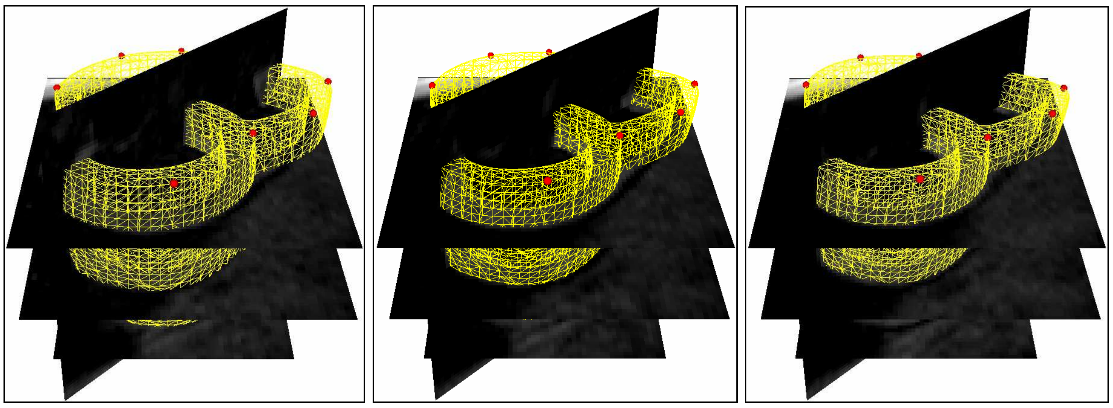


Figure 5.4: Final configuration for the 2.5D segmentation (three different frames)

5.3. 3D Segmentation of a Laryngeal CT

The computed tomography (CT) volume to be segmented was immersed in the coordinate system $[-1,+1]\times[-1,+1]\times[-1,+1]$. The initial configuration was set as the known solution after rotating 2.5° and translating (0.05, 0.05, –0.05) units as shown in figure 5.5. The step size for the *gradient descent method* (3.6) was set to $\gamma = 0.15$ and the weights for the fuzzy set and the topology preservation energies were both set to 1.0. The step size for the derivative approximation (3.15) was set to *h* = 0.001. *Maximum likelihood estimation* as in (4.7) was used for approximating the means and variances for the object and background regions. Figure 5.6 shows the surface after 50 iterations.

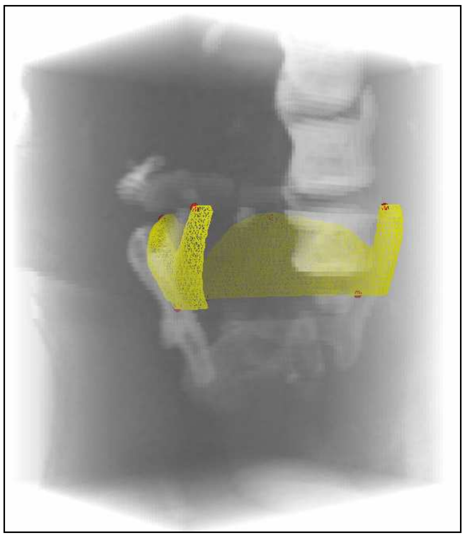


Figure 5.5: Initial configuration for the 3D segmentation

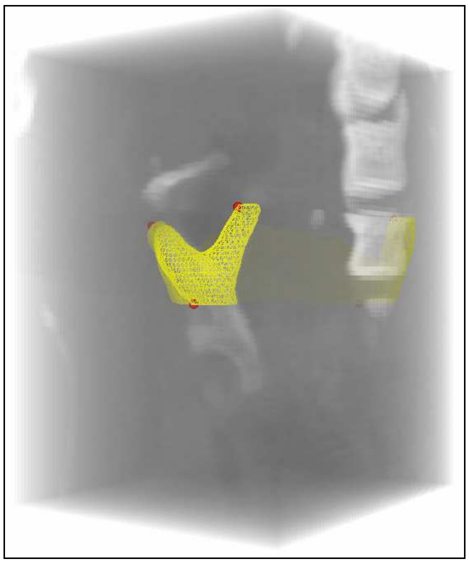


Figure 5.6: Final configuration for the 3D segmentation

5.4. Comparison of Probability-Based Forces

A comparison between different probability-based external forces was performed in order to measure the advantage of using the proposed force. The initial configuration, step sizes and topology preservation factor were the same as the ones used in the 2D implementation. The region-dependent descriptor framework as defined in (2.24) and (2.25) was used.

The following table shows the different external forces as well as their object and background descriptors (2nd and 3rd columns). The probability density functions of the intensity levels $p_{in}(I(\mathbf{w}))$ and $p_{out}(I(\mathbf{w}))$ were approximated as Normal distributions. Different force factors (4th column) were used since the magnitudes of the generated forces are different.

External force	**Object descriptor** k_{in}	**Background descriptor** k_{out}	**Force Factor**
Adaptive fuzzy C-means in (2.21)	$\frac{p_{out}}{p_{in}}$ (maximum value: 8)	1	0.1774
Region probability in (2.22) and (2.23)	$-\log p_{in}$	$-\log p_{out}$	0.2310
Entropy in (2.26)	$-p_{in}\log p_{in}$	$-p_{out}\log p_{out}$	0.2572
Mutual information in (2.27)	$-\frac{\lvert\Omega_{in}\rvert}{\lvert\Omega\rvert}p_{in}\log p_{in}$	$-\frac{\lvert\Omega_{out}\rvert}{\lvert\Omega\rvert}p_{out}\log p_{out}$	0.4476
Proposed: Fuzzy sets in (3.5) and (3.7)	$\frac{p_{out}}{p_{in}+p_{out}}$	$\frac{p_{in}}{p_{in}+p_{out}}$	1.0000

Table 5.1: Description of probability-based forces

The rationale for computing the force factors is to generate the same average magnitude for every different force. In order to do this, the total absolute force magnitude over all intensities should be equal for every force. This is given by:

$$\int_0^1 \left| (k_{in}(v) - k_{out}(v)) \left(\frac{|\Omega_{in}|}{|\Omega|} p_{in}(v) + \frac{|\Omega_{out}|}{|\Omega|} p_{out}(v) \right) \right| dv$$

Where the first term is the force magnitude for an intensity level *v* by assuming that it does not depend on the evolution of the contour. The second term is the probability density function of the intensities in the overall image. From the known solution $p_{in}(I(\mathbf{w})) \sim N(0.6404, 0.0085)$, $p_{out}(I(\mathbf{w})) \sim N(0.4320, 0.0177)$, $|\Omega_{in}|/|\Omega| = 0.1217$ and $|\Omega_{out}|/|\Omega| = 0.8783$.

In order to compare the results, the mean square error of the landmarks was chosen as a measure of convergence. Given the known solution **T*** and its approximation **T**, each of them composed of *n* two-dimensional landmarks, the mean square error is defined as:

$$MSE(\mathbf{T}, \mathbf{T}^*) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1,2} (t_{i,j} - t_{i,j}^*)^2$$

The following chart shows the mean square error in the Y axis versus the number of iterations in the X axis for each technique:

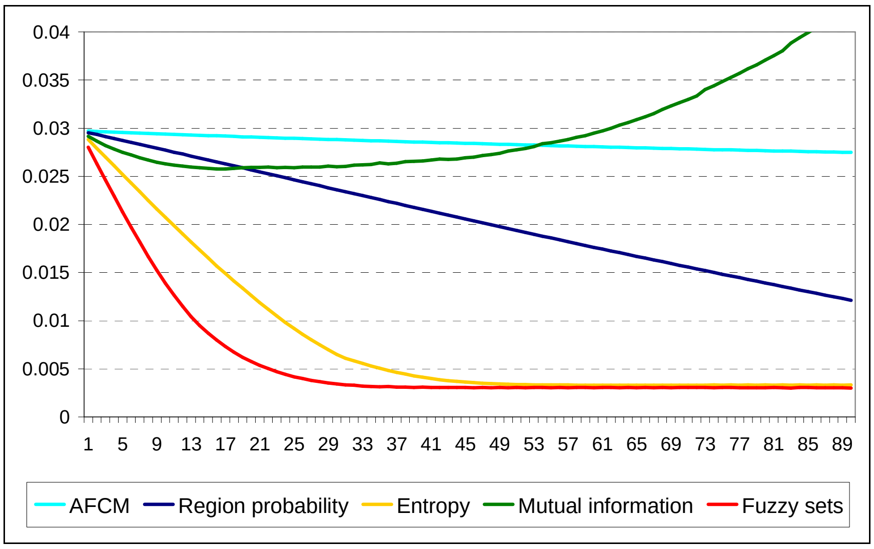


Figure 5.7: Convergence of probability-based forces

The fastest convergence was reached by the proposed *fuzzy sets* force. Only the *entropy* force behaved similar to the proposed force, since both converge asymptotically to the known solution. *Adaptive fuzzy C-means* as well as *region probability* have a slow linear convergence. It is also shown that the *mutual information* force is sensitive to the initial contour, since it did not converge to the known solution.

In order to graphically understand this behavior, the force magnitudes $k_{in} - k_{out}$ were computed from the known solution. Figure 5.8 shows the probability density functions, where the X axis corresponds to the intensity levels. Notice that the two curves intersect each other when the intensity value is approximately 0.55.

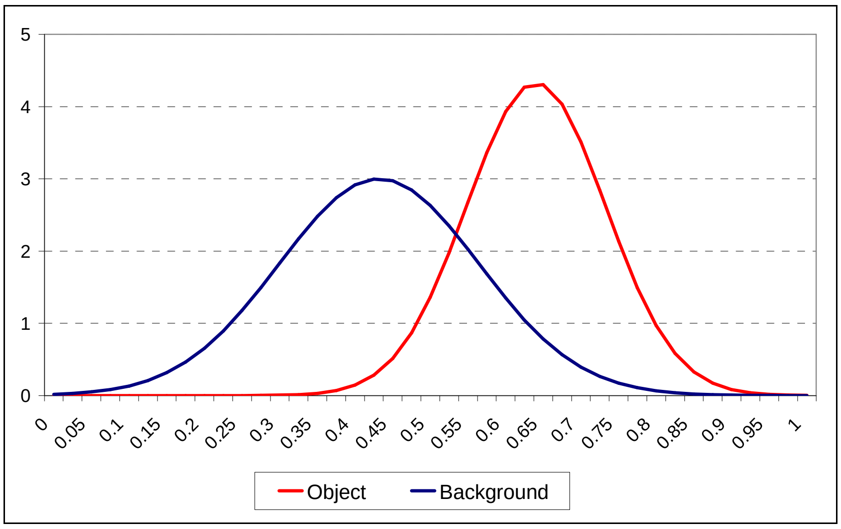


Figure 5.8: Probability density functions for the image

Figure 5.9 shows the force magnitudes for each probability-based force, where the X axis corresponds to the intensity levels. From figure 5.8, it is expected that force magnitudes are positive for intensity values less than 0.55 and negative for intensity values greater than 0.55. The rationale is that the curve should shrink when a background pixel is detected, and expand when an object pixel is detected. Even though, as it can be noticed the *information entropy* as well as the *mutual information* forces does not follow this rule for very low as well as very high intensity values. Notice also that the *mutual information* force do not cross the X axis in 0.55. The *adaptive fuzzy C-means* as well as the *region probability* shrink and expand the deformable model properly, but the magnitude is not balanced. A small negative value at the right compared with a larger positive value at the left indicates a small expansion compared with a larger contraction. Only the proposed *fuzzy sets* force properly equilibrates the amount of expansion and contraction.

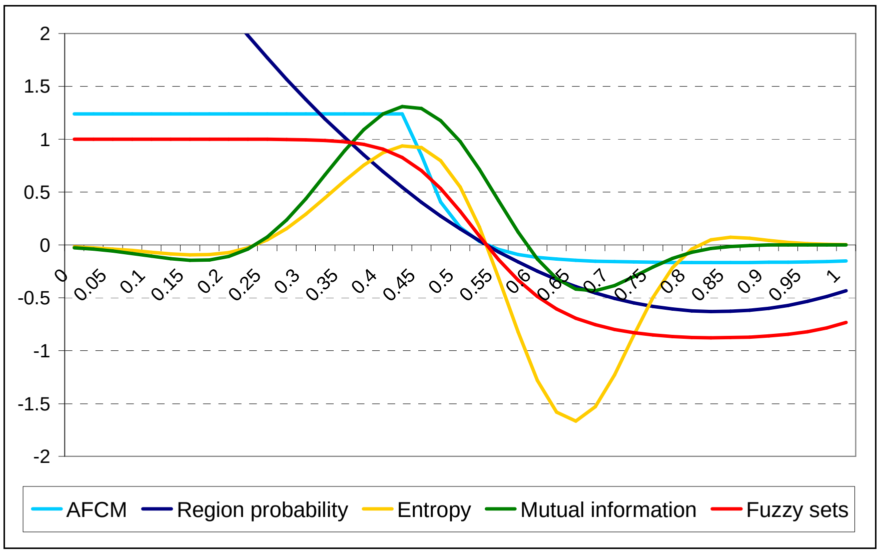


Figure 5.9: Magnitude of probability-based forces

Chapter 6. Conclusion and Future Work

6.1. Conclusion

A deformable model which is based on the *fuzzy sets theory* was presented. A new *external energy* was derived as a result of maximizing the *crisp probability* that the contour accurately separates the image into two regions. Unique solution and convergence is ensured for a *perfect point classifier*, an *axis-independent warp* and a convex contour.

The proposed force obtained the fastest convergence when compared versus different probability-based forces. This force asymptotically converged to the known solution. It was also shown that the proposed force properly equilibrates the amount of expansion and contraction of the deformable model.

Radial basis functions were used for contour parameterization, which allows using an arbitrary mesh as deformable model. As additional constraints, two simple *internal energies* were defined for ensuring topology preservation and temporal coherence.

6.2. Future Work

There are several ways of extending this work. A more statistically rigorous comparison of convergence speed of the proposed force versus the different probability-based forces should be performed.

A more general proof of unique solution and convergence should be developed. This general proof should take into account the use of non-convex contours as well as noisy point classifiers, which can formally include non-spatially and spatially correlated error.

Boundary-based external energies based on fuzzy sets can be defined. Finally, regularized region indicator functions can be applied instead of the one used in our formulation. This allows integration over the contour neighborhood instead on only in the infinitesimally thin contour. As a result, this enhancement can add a flavor of smoothing and give more hints for the evolution of the contour.

References

[1] Bandemer and Gottwald, "Fuzzy Sets, Fuzzy Logic, Fuzzy Methods With Applications", *John Wiley & Sons*, 1995.

[2] Baxandall and Liebeck, "Vector Calculus", *Oxford University Press*, 1986.

[3] Bookstein, "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1989.

[4] Bresson, Vandergheynst and Thiran, "A Variational Model for Object Segmentation Using Boundary Information and Shape Prior Driven by the Mumford-Shah Functional", *International Journal of Computer Vision*, 2006.

[5] Caselles, Kimmel and Sapiro, "Geodesic Active Contours", *International Journal of Computer Vision*, 1997.

[6] Cerqueira, Weissman, Dilsizian, Jacobs, Kaul, Laskey, Pennell, Rumberger, Ryan and Verani, "Standardized Myocardial Segmentation and Nomenclature for Tomographic Imaging of the Heart", *American Heart Association*, 2002.

[7] Chan and Shen, "Image Processing and Analysis: Variational, PDE, Wavelet and Stochastic Methods", *Society for Industrial and Applied Mathematics*, 2005.

[8] Chan and Vese, "Active Contours Without Edges", *IEEE Transactions on Image Processing*, 2001.

[9] Cohen and Cohen, "Finite Element Methods for Active Contour Models and Balloons for 2D and 3D Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

[10] Cohen, "On Active Contour Models and Balloons", *CVGIP: Image Understanding*, 1991.

[11] Cootes, Edwards and Taylor, "Active Appearance Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[12] Cootes, Taylor, Cooper and Graham, "Active Shape Models: Their Training and Application", *Computer Vision and Image Understanding*, 1995.

[13] Davatzikos and Prince, "Convexity Analysis of Active Contour Problems", *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[14] Delingette, "General Object Reconstruction Based on Simplex Meshes", *International Journal of Computer Vision*, 1999.

[15] Gavrila, "Hermite Deformable Contours", *IEEE International Conference on Pattern Recognition*, 1996.

[16] Gouze, De Roover, Macq, Herbulot, Debreuve and Barlaud, "Watershed-Driven Active Contours for Moving Object Segmentation", *IEEE International Conference on Image Processing*, 2005.

[17] Herbulot, Jehan-Besson, Barlaud and Aubert, "Shape Gradient for Image Segmentation Using Information Theory", *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2004.

[18] Jahne, "Digital Image Processing", Sixth Edition, *Springer*, 2005.

[19] Jehan-Besson, Barlaud and Aubert, "DREAM^{2}S: Deformable Regions Driven by an Eulerian Accurate Minimization Method for Image and Video Segmentation", *International Journal of Computer Vision*, 2003.

[20] Kass, Witkin and Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, 1988.

[21] Kim, Fisher, Yezzi, Cetin and Willsky, "A Nonparametric Statistical Method for Image Segmentation Using Information Theory and Curve Evolution", *IEEE Transactions on Image Processing*, 2005.

[22] Limin, Shiwen, Xinquan and Yaoping, "3D Surface Reconstruction Using Fuzzy Deformable Models", *IEEE International Conference on Signal Processing*, 2000.

[23] Matthews and Baker, "Active Appearance Models Revisited", *International Journal of Computer Vision*, 2004.

[24] Menet, Saint-Marc and Medioni, "B-Snakes: Implementation and Application to Stereo", *International Conference on Computer Vision*, 1990.

[25] Paragios and Deriche, "Geodesic Active Regions: A New Framework to Deal With Frame Partition Problems in Computer Vision", *Journal of Visual Communication and Image Representation*, 2002.

[26] Ronfard, "Region-Based Strategies for Active Contour Models", *International Journal of Computer Vision*, 1994.

[27] Staib and Duncan, "Boundary Finding With Parametrically Deformable Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[28] Sumengen and Manjunath, "Graph Partitioning Active Contours (GPAC) for Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[29] Tanacs, Czedli, Palagyi and Kuba, "Affine Matching of Two Sets of Points in Arbitrary Dimensions", *Acta Cybernetica*, 2001.

[30] Terzopoulos and Metaxas, "Dynamic 3D Models With Local and Global Deformations: Deformable Superquadrics", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[31] Verdu, Morales, Gonzalez and Weruaga, "Convergence Analysis of Active Contours in Image Segmentation", *IEEE International Conference on Image Processing*, 2004.

[32] Wang and Staib, "Boundary Finding With Correspondence Using Statistical Shape Models", *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[33] Xu and Prince, "Generalized Gradient Vector Flow External Forces for Active Contours", *Signal Processing*, 1998.

[34] Xu and Prince, "Snakes, Shapes and Gradient Vector Flow", *IEEE Transactions on Image Processing*, 1998.

[35] Xu, Pham and Prince, "Image Segmentation Using Deformable Models", *Handbook of Medical Imaging, The International Society for Optical Engineering*, 2000.

[36] Xu, Yezzi and Prince, "On the Relationship Between Parametric and Geometric Active Contours", *Asilomar Conference on Signals, Systems and Computers*, 2000.