# Chapter 3

# Supervised Clustering Competition Scheme.

## 3.1 General framework and main goal

The main goal in this work is to find an integrated framework for supervised and unsupervised classification techniques so that both can compete for the classification of a data point. Since clustering methods are usually based on the minimization of a dissimilarity measure or maximization of a similarity measure, it can be easily seen that an hybrid approach can take advantage of this fact and try to express both processes, the supervised and the unsupervised, as minimization of functionals. Therefore, our aim is to find a way to blend supervised and unsupervised classification schemes at the same time. Let us define,

$$L(\mathbf{x}) = h(min_{\mathbf{x}}(\alpha \cdot SF(\mathbf{x}) + (1-\alpha) \cdot UF(\mathbf{x}))) \tag{3.1}$$

where $SF$ stands for *supervised functional*, a functional the minimums of which are close to the centers of each class and that has an explicit maximum on the border between classes; $UF$ stands for *unsupervised functional*, expressing a dissimilarity measure; $\alpha$ is the mixing parameter; and the function $h(\cdot)$ is a decision scheme that allows the classification of each data sample.

Applying the gradient descent, we get,

$$\frac{\partial \mathbf{x}}{\partial t} = -\nabla F(\mathbf{x})$$

The iterative scheme is,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \triangle t \cdot \nabla F(\mathbf{x})$$

where $\nabla F(\mathbf{x}) = \{\partial F(\mathbf{x})/\partial x_i\}$. Since we want both techniques to compete for the data points, both functionals are combined by a mixing parameter $\alpha$,

$$\frac{\partial \mathbf{x}}{\partial t} = -\alpha \frac{\nabla SF(\mathbf{x})}{\|\nabla SF(\mathbf{x})\|} - (1-\alpha)\frac{\nabla UF(\mathbf{x})}{\|\nabla UF(\mathbf{x})\|} \tag{3.2}$$

Now, we have to define the minimization processes that represent the supervised classification and the clustering process.

## 3.2 Robust Similarity Clustering

Cluster analysis is concerned with grouping data by means of their similarity, therefore, it divides data in most dissimilar groups in different clusters [102] [107] [108]. Although different techniques have been proposed in the bibliography [114] [113], [105] [108], [103] [109] [111] [115] (just to mention a few), we are interested in tools for robust clustering related to invariability of the initial state, capability to differentiate among different volumes of different sizes and toleration to noise and outliers [100] [106] [97]. Recently, a simple but effective algorithm has been proposed to solve most of the problems of robust clustering [97]. This method solves the initialization and volume issues by means of a self-organizing technique of the data. This is the work we have chosen to use as a unsupervised classification technique. The method is based on a similarity clustering algorithm and an agglomerative hierarchical clustering algorithm to find the optimal cluster number and reject outliers. Since, we are not interested in finding the optimal number of clusters, because it is given *a priori*, we simply obviate the agglomerative hierarchical algorithm step, and replace it by a decision step that labels each cluster once it converges.

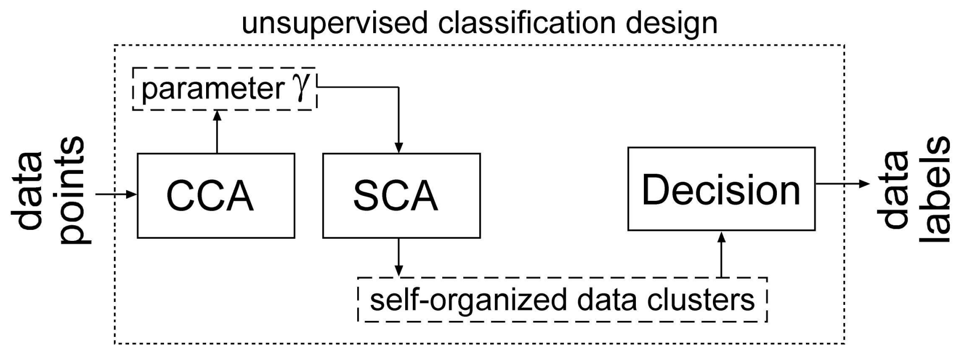

unsupervised classification design
parameter γ
data points
CCA
SCA
Decision
data labels
self-organized data clusters

**Figure 3.1:** Scheme of the unsupervised clustering classification algorithm.

Figure 3.1 shows the three steps of the unsupervised clustering design. The first step is the **correlation comparison algorithm (CCA)**. This algorithm is an automatic parameter extractor that analyzes exhaustively how different similarity-clustering results correlate when the algorithm parameter changes. The second step is **similarity clustering algorithm (SCA)**. It is the self-organizing technique, which output is a set of cluster centers around which the input data points are grouped. SCA is the iterative procedure that allows each center to converge according to the similarity measure. The more similar the data is, the more centers converge to the same point. The result is a set of points indicating where each original data point has converged. Though the number of final clusters is not known *a priori* the decision step will set them to the number of classes desired. The last step is a supervised classifier, in particular, the same classifier used as a model in the supervised classifier section. The decision of the classifier on the self-organized data allows the proper labelling of the data points.

### 3.2.1 Similarity clustering core

Most of the clustering algorithms procedures are based on minimization of a total dissimilarity measure, for instance, k-means or fuzzy c-means. In the approach we have chosen a *similarity* measure between two points $S(z_j, x_i)$ is used in a maximization framework. Our goal is to maximize the total similarity measure $J_s(\mathbf{x})$ defined as:

$$J_s(\mathbf{x}) = \sum_{i=1}^{c} \sum_{j=1}^{n} f(S(z_j, x_i))$$

where $f(\cdot)$ is a monotone increasing function, $\mathbf{x}$ represents the centers of each cluster and $\mathbf{z}$ is the original data set (where $\mathbf{z} = \{z_1, \dots, z_n\}$ and $z_i$ is a D-dimensional data point). As a similarity relation $S(z_j, x_i)$, we use,

$$S(z_j, x_i) = e^{-(\frac{\|z_j - x_i\|^2}{\beta})}$$

where $\beta$ is a normalization term. Let the monotone increasing function $f(\cdot)$ be,

$$f(\cdot) = (\cdot)^{\gamma} \quad , \quad \gamma > 0$$

Therefore, the complete similarity measure $J_s(\mathbf{x})$ is,

$$J_s(\mathbf{x}) = \sum_{i=1}^{c} \sum_{j=1}^{n} \left(e^{-\frac{\|z_j - x_i\|^2}{\beta}}\right)^{\gamma} \tag{3.3}$$

The parameter $\beta$ is superfluous in this scheme and can be defined as the sample variance,

$$\beta = \frac{\sum_{j=1}^{n} \|z_j - \bar{z}\|^2}{n} \quad \text{where} \quad \bar{z} = \frac{\sum_{j=1}^{n} z_j}{n}$$

The parameter $\gamma$ gains a considerable importance in this scheme since a good $\gamma$ estimate induces a good clustering result. The process of maximizing the total similarity measure is a way to find the peaks of the objective function $J_s(\mathbf{x})$. It is shown in [97] that the parameter $\gamma$ is used as a neighboring limiter, as well as a local density approximation. To find $\gamma$ one can use an exhaustive search of the correlation of the similarity function for each point when changing the parameter. If the correlation value is over a certain threshold, we can consider that the similarity measure represents the different variability and volumes of the data set accurately. The authors set this threshold experimentally to 0.97 but can change according to the application.

The similarity clustering approach uses the same similarity function but, as it is a self-organizing approach, we define the initial data and centers by the unlabelled data points $\mathbf{z}^0 = \mathbf{x}^0$,

$$UF(\mathbf{x}) = J_s(\mathbf{x}) = \sum_{j=1}^{n} \left(e^{-\frac{\|z_j - x_k\|^2}{\beta}}\right)^{\gamma}, \quad k = 1 \dots n$$

Getting its gradient, we obtain,

$$\nabla UF(\mathbf{x}) = -2\frac{\gamma}{\beta}\sum_{j=1}^{n}\left(e^{-\frac{\|z_j - x_k\|^2}{\beta}}\right)^{\gamma}(z_j - x_k), \quad k = 1\dots n \tag{3.4}$$

## 3.3 Supervised classifier functional

As our goal is to combine supervised and unsupervised classifiers in a fair competition scheme, we must take a common framework for both processes to be comparable. Since the clustering problem is solved as a self-organizing iterative process, we reformulate the supervised classifier process as a self-organizing iterative process.

Without loss of generality, we restrict our classifier design to a two class supervised classification process using a Bayesian framework with known class probability density functions. Assuming that we can estimate each class probability density function $f_A(\mathbf{x}|c = A)$ and $f_B(\mathbf{x}|c = B)$, the optimal discriminative rule using a Bayesian approach is given by,

$$h(\mathbf{x}) = \begin{cases} A & f(\mathbf{x}|c = A)P(A) > f(\mathbf{x}|c = B)P(B) \\ B & f(\mathbf{x}|c = A)P(A) \leq f(\mathbf{x}|c = B)P(B) \end{cases}$$

If *a priori* knowledge of the probability appearance is not known, we assume $P(A) = P(B)$.

The manifold we are looking for, must maintain the optimal borderline as a maximum, since we want the minimums to represent each of the classes. It can be easily seen that the following family of functions satisfies the requirement,

$$SF = -(f(\mathbf{x}|c = A) - f(\mathbf{x}|c = B))^{2N}, \qquad \forall N \in \{1..\infty\} \tag{3.5}$$

As a minimization tool, we apply the gradient descent method,

$$\frac{\partial \mathbf{x}}{\partial t} = -\nabla SF(\mathbf{x}),$$

the iterative scheme is,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \triangle t \cdot \nabla SF(\mathbf{x}) =$$

$$\mathbf{x}_t - 2N\triangle t \cdot (f_A(\mathbf{x}) - f_B(\mathbf{x}))^{2N-1}(f'_A(\mathbf{x}) - f'_B(\mathbf{x}))$$

In order to obtain a feasible closed form of the functional we can restrict the density estimation process to a Gaussian mixture model,

$$SF(\mathbf{x}) = \mathbf{f_K}(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{M_k}} \pi_{\mathbf{i}}\mathbf{g_i}(\mathbf{x}, \theta_{\mathbf{i}})$$

where $M_k$ is the model order and $g_i(\mathbf{x}, \theta_i)$ is the multidimensional gaussian function,

$$g_i(\mathbf{x}, \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|}e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)}$$

where $\theta_i = \{\mu_i, \Sigma_i\}$ are the parameters for the gaussian, the covariance matrix and the mean, and $d$ is the dimensionality of the data.

Therefore,

$$\nabla SF((\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{M_k}} (-\frac{\pi_\mathbf{i}}{\mathbf{2}} \mathbf{\Sigma_i^{-1}} (\mathbf{x} - \mu_\mathbf{i}) \mathbf{g_i}(\mathbf{x}, \theta_\mathbf{i})) \tag{3.6}$$

## 3.4 The procedure

Summarizing the overall procedure, we begin the process with three data sets, the labelled data, the unlabelled data and the test set. Labelled data is used to create the model for the supervised functional as well as used for the final decision step. Unlabelled data feeds the competition scheme and it is the data that will be labelled according to the balance of the supervised and unsupervised processes. The final algorithm is as follows:

**Step 1** Determine a supervised set $L$ of data among labelled data and a set $U$ of unlabelled data.

**Step 2** Estimate $\nabla SF(\mathbf{x})$ $x \in L$ from (3.6).

**Step 3** Apply the CCA step to set the parameter $\gamma$ of $UF(x)$ $x \in U$.

**Step 4** Feed the competition scheme by evolving the clusters (that come from unlabelled data $U$) according to (3.2) and let them converge.

**Step 5** If the error of the minimization process is less than a fixed threshold, stop the process. Otherwise, go to Step 3.

### 3.4.1 Behavior of the Supervised Clustering Competition

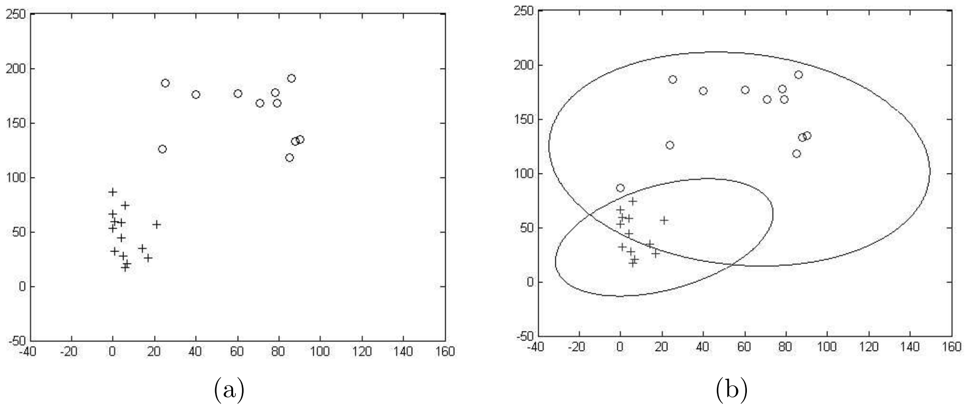
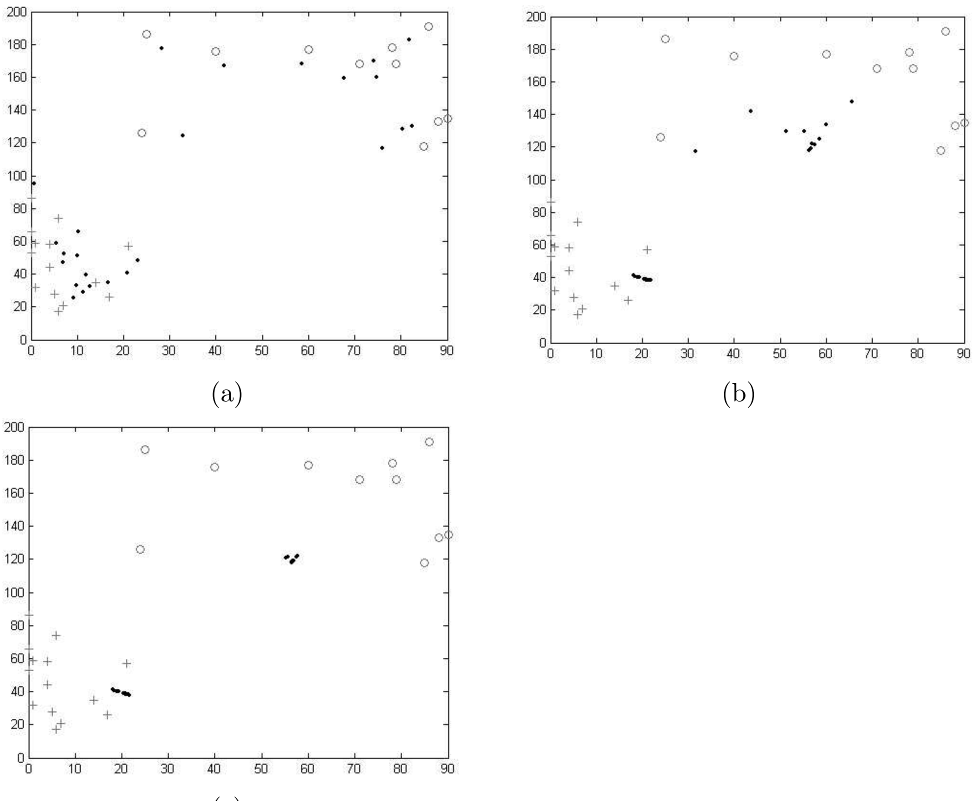


Figure 3.2: (a) Original labels for the experiment. (b) Maximum likelihood classifier performance using a mixture of 1 gaussian per class.

This section shows an example to compare supervised classification, clustering and supervised clustering competition. In figure 3.2 a classical supervised classification is performed, the classifier is estimated and the corresponding Gaussian functions are drawn. Figure 3.2.a show the original labels of the data. The points noted by + and *o* are the labels assigned to the test data by the estimated already supervised classifier. Note that the supervised process misses one of the data points (see the bottom left *o* in the figure should be + to be correctly classified) assigning it an incorrect label.



(a) (b)

(c)

**Figure 3.3:** Behavior of the supervised self-organizing schemes, (a) Supervised functional minimization at t=3. (b) Supervised functional minimization at t=14. (c) Unsupervised functional minimization at t=20.

Figures 3.3, 3.4 and 3.5 show the supervised self-organization procedure, the unsupervised clustering and the supervised clustering competition scheme respectively. In those figures, the + and the *o* displayed are the labels of each data point from the labelled training data and the . denote the clusters initialized by the unlabelled data. Figures 3.3.a, 3.3.b and 3.3.c show the motion of the clusters by time when only the supervised part of the functional in (3.1) is minimized. These figures show how the data points mimic the supervised classification scheme as a self-organization procedure. At each stage the points gather near the center of each of the classes. If we follow the path of each of the points we easily see that the resulting classification is that displayed in figure 3.2, achieving the objective of having the same behavior than the supervised classification process. This means that as the supervised as the unsupervised frameworks can misclassify some of the points.

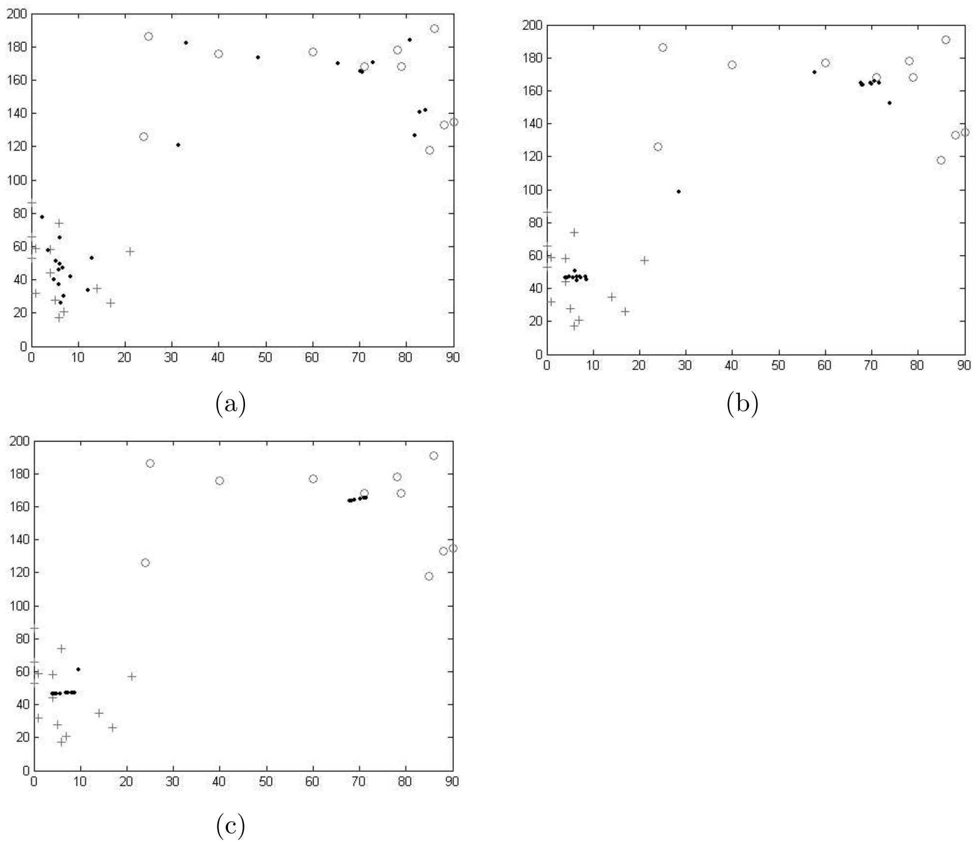
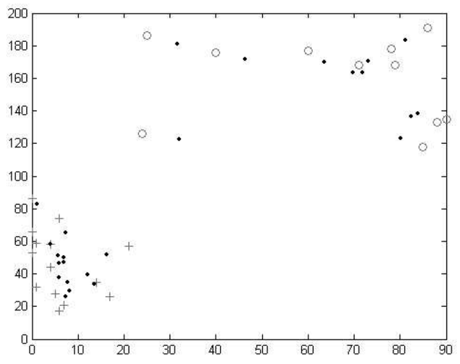
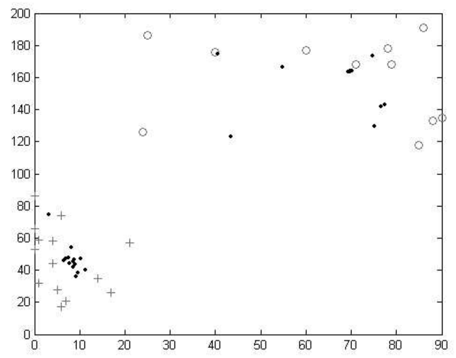


Figure 3.4: Behavior of the unsupervised self-organizing schemes, (a) Unsupervised functional minimization at t=3. (b) Unsupervised functional minimization at t=14. (c) Unsupervised clustering functional minimization at t=20.

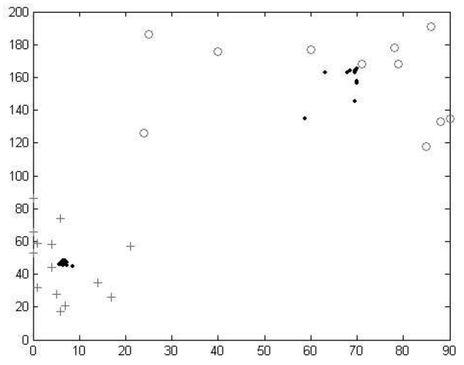
Figure 3.4 shows the result of the unsupervised clustering on the same example. In this figure, the + and the *o* displayed are the real labels of each data point, and the dots are the points as they gather around the centers of each class. Figures 3.4 show how the data points gathers in a unsupervised way. Note that the unsupervised process misses one of the points, in particular the isolated point near the center of the feature space converge to the wrong class due to the unsupervised procedure. Finally, figures 3.5 show the Supervised Clustering Competition in action. The process uses a mixing value of $\alpha = 0.3$, and following the points as they evolve we can see that each of the data points converges to its real class.



(a)



(b)



(c)

**Figure 3.5:** Behavior of the hybrid self-organizing schemes, (a) Supervised clustering functional minimization at t=3. (b) Supervised clustering functional minimization at t=7. (c) Supervised clustering functional minimization at t=14.

# Chapter 4

# Deformable models

## 4.1 Statistic Deformable Models: Generative Snakes

Generative snakes is the first proposal of this work. It is based on the definition of a new external potential that describes the image features of interest. This potential is derived from a generative model, and creates the likelihood map of an image. In this context the feature modelling process is the first step for the likelihood map computation. The likelihood map measures how likely each of the pixels of the image is to belong to the desired pattern. This approach addresses the problems of unfilled and unconnected regions in region classification as well as provide a constrained model for searching in the likelihood space. It is worth mention that this scheme does not depends on a previous classification since the classification step and the high-level integration step are performed at the same time.

### 4.1.1 Background of parametric snakes

The basic goal of active contours is to find a parameterized curve that minimizes the weighted sum of its internal energy ($E_i$) and external energy ($E_e$). Given a traditional snake curve $\mathbf{x}(s) = (x(s), y(s)), s \in [0, 1]$, the snake can be formulated as the minimization of the following equation:

$$S(\mathbf{x}) = \int E_i + E_e ds = \int_0^1 (\alpha(s)\left|\frac{\partial x}{\partial s}\right|^2 + \beta(s)\left|\frac{\partial^2 x}{\partial s^2}\right|^2 + E_e)ds \tag{4.1}$$

The internal energy specifies the tension and smoothness of the contour. The first order derivative prevents stretching (elasticity, controlled by $\alpha$) while the second order derivative discourages bending (stiffness, controlled by $\beta$). The external energy is derived from the image data. Regardless of the external function the problem of finding the curve $\mathbf{x}(s)$ that balances the internal and external forces must satisfy the Euler-Lagrange equation which can be solved iteratively as follows:

$$\frac{\partial x}{\partial t} = \frac{\partial}{\partial s}\left(\alpha\frac{\partial x}{\partial s}\right) - \frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2 x}{\partial s^2}\right) - \nabla E_e(x) \tag{4.2}$$

When $x(s,t)$ stabilizes $x_t(s,t)$ vanishes, leading to the solution of the system. This system can be seen as a gradient descent algorithm; the solution can be found by discretizing the equation and solving it iteratively.

The classic approach to the external energy is the potential energy which takes smaller values at object boundaries as well as other features of interest. The typical potential function designed to lead a deformable contour toward step edges is a function of the high-gradient location of the original image pixels.

$$P(x,y) = -\gamma|\nabla(G_\sigma(x,y) * I(x,y))|^2 \tag{4.3}$$

where $\nabla$ is the gradient operator, $\gamma$ is a weighting parameter, $G_\sigma(x,y)$ is the gaussian filter of standard deviation $\sigma$, and $I(x,y)$ is the image data. As it can be observed, greater $\sigma$ will increase the attraction range but will blur the edges.

In contrast to physics-based snakes, Geodesic snakes take the following form:

$$\frac{\partial x}{\partial t} = c(\kappa + V_0)|\nabla x| \tag{4.4}$$

where

$$c = \frac{1}{1 + |\nabla(G_\sigma(x,y) * I(x,y))|} \tag{4.5}$$

where $V_0$ is constant and $\kappa$ is the curvature of the deformable model at a given point. To solve the segmentation problem, the snake curve is embedded in a higher order manifold that deforms to adjust to image features. The final segmentation result is obtained as the level set curve of the stabilized manifold. Positive $V_0$ shrinks the curve. It must be noted that both approaches have a term dependent on the image, the external energy ($E_e$) in parametric snakes and the speed function ($c$) in geometric snakes. Our approach focuses on these terms, therefore, it can be applied to both methods by changing the defined image energy in equations (4.2) or the speed in equation (4.4).

Different convergence problems affect the performance of snakes: a need for close initialization, stopping criterion, impossibility of converging to concave boundaries, etc. One of the most standard approaches used to try to solve problems in parametric snakes is the Generalized Gradient Vector Flow (GGVF) [81] [80]. The GGVF begins with the definition of an edge map $f(x,y)$ derived from the image $I(x,y)$. Therefore, the $\nabla f$ vector is directed towards the edges with a narrow capture range, while leaving the rest with no information. A regularization process is applied in order to propagate information from the contours to assure continuity of the vector flow. Formally, GGVF is defined as a vector field $\mathbf{v}(x,y) = (u(x,y), v(x,y))$ that minimizes the energy functional:

$$F = \int\int g(|\nabla f|)(u_x^2 + u_y^2 + v_x^2 + v_y^2) + h(|\nabla f|)|\mathbf{v} - \nabla f|^2 dxdy \tag{4.6}$$

This equation follows the principle of propagating data at image regions where the gradient, $|\nabla f|$, cancels. Usually, $h(.)$ is a monotonically decreasing function, with maximum value on the edges provided by $|\nabla f|$. This means that information from the contours is propagated smoothly where $f$ contains no data, while near image edges should be as similar as possible to $\nabla f$. Function $g(|\nabla f|)$ is chosen as complementary to $h(.)$ and governs the trade-off between both terms. Xu et al.[80] use $g(|\nabla f|) = e^{-(|\nabla f|/K)}$ and $h(|\nabla f|) = 1 - g(|\nabla f|)$ to cope with narrow concavities and some speckle noise when deforming.

The solution of this system can be found iteratively by discretizing the Euler equations:

$$u_t(x,y,t) = g(|\nabla f|)\nabla^2 u(x,y,t) - h(|\nabla f|)(u(x,y,t) - f_x(x,y)) \tag{4.7}$$

$$v_t(x,y,t) = g(|\nabla f|)\nabla^2 v(x,y,t) - h(|\nabla f|)(v(x,y,t) - f_y(x,y)) \tag{4.8}$$

where $\nabla^2$ is the Laplacian operator. Note that in homogeneous regions the second term of the former equations is zero, thus making the Laplace term govern the evolution. The result is somewhat similar to a filling-in and spreading of information taken from the boundaries. After the computation of $\mathbf{v}(x,y)$, the external force $-\nabla E_e$ of the snake is replaced by $\mathbf{v}(x,y)$ in (4.2) or (4.4).

### 4.1.2 Enhanced Likelihood Map for Statistical Snakes

Instead of using a heuristically defined potential field or its regularized version, we are interested in defining the external energy of the snake as a likelihood map. To achieve this purpose, given a set of target regions $S = \{s_i\}, s_i \in c$ a feature extractor is used to take out representative information of this set, which results in a feature vector for each of the regions ($F_i = \{f_1(s_i), f_2(s_i), \dots, f_n(s_i)\}$). This allows us to model the likelihood ($L(F_i)$) of the data in the feature space given a set of target feature points. This measure is related to the probability density function corresponding to the texture region of interest.

The likelihood function is used to create a *likelihood map* ($L(x,y) = L(F(x,y)|M_i)$). Each value in the likelihood map indicates how likely a pixel of the image is to belong to the likelihood model $M_i$ ($M_i$ is the model of the target region of interest). This process is done by applying the aforementioned linear discriminant analysis and the gaussian mixture model on the feature vector (see section 2.2).

Usually, in general analysis, once the likelihood has been estimated, a classification map is constructed by applying a fixed threshold estimated from a ROC curve. An alternative for the classification of pixels consists in comparing distances to the class centers, using the class membership of the nearest neighbor, and so for.

The classification map can be extremely irregular, containing holes and "islands", making its direct use as an external energy map not straightforward. Our approach relies on substituting the classification step with organizing pixels in regions according to their likelihood and spatial distribution. Analyzing the likelihood map, its contours are indicative of likelihood changes, and therefore, are candidates to represent object boundaries. The contours of the likelihood map suffer from some shortcomings: a) contour displacement, b) irregular contour values, c) strong contours between regions ( we want to have significant contours near the real boundaries and low values otherwise) and d) appearance of non-desired edges inside the regions. This leads us to define an enhancing procedure for the likelihood map. Hence, we define a new **Enhanced Likelihood Map** ($ELM$) for the two class problem as follows:

$$\tilde{L} = \left(\frac{\lambda}{\lambda + |L(I|M_i) - L(I|\overline{M}_i)|} \cdot L(I|M_i)\right) \quad (4.9)$$

where $L(I|M_i)$ is a compact notation for $L(F(I(x,y))|M_i), \forall x, y \in I$, $M_i$ is the model of the target region, $\overline{M}_i$ is the model of the complement of the target region and $\lambda$ is a weighing parameter. Here $L(F(I(x,y))|M_i)$ means the likelihood that the projected features of pixel $(x,y)$ of image $I$ belongs to the target model $M_i$. Note that when both likelihood values are similar $(L(I|M_i) \sim L(I|\overline{M}_i))$, the likelihood map $(L(I|M_i))$ is emphasized. It keeps smooth elsewhere. This term highlights the likelihood map in a range near the border given by $L(I|M_i) = L(I|\overline{M}_i)$. In this way, we enhance edges near the classification border weighted by their likelihood value in the map. The proposed ELM partially overcomes the main drawbacks of the usage of the contours of the likelihood map, giving better accuracy and removing non-prominent edges.

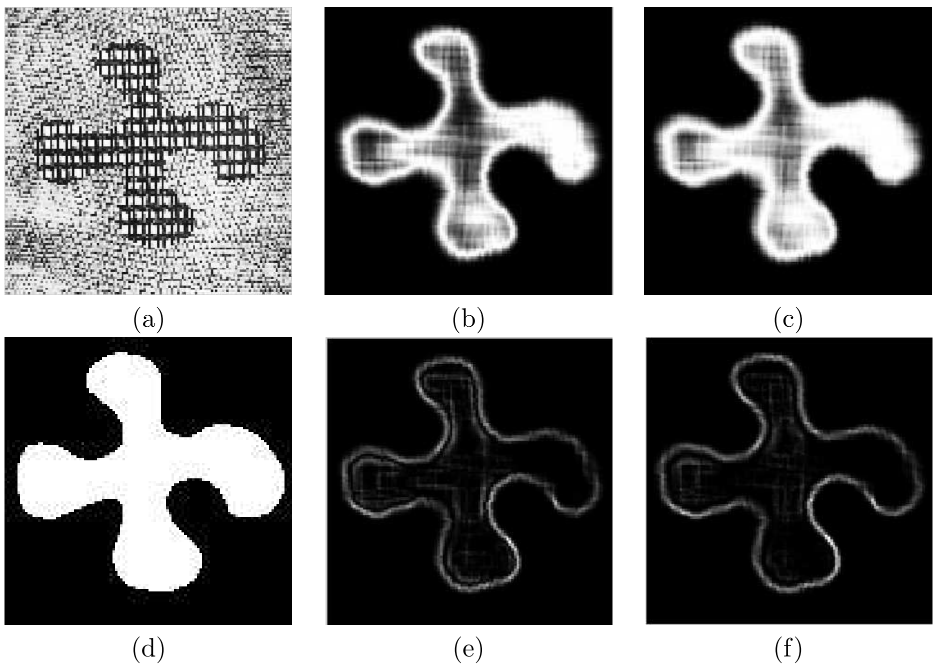


**Figure 4.1:** (a) Original image, (b) Likelihood Map for target texture, (c) Enhanced likelihood map, (d) Mask used to create (a), (e) Contours of the likelihood map, (f) Contours of the enhanced likelihood map.

Figure 4.1 shows the process of constructing the ELM and obtaining its contours for different texture prototypes. Figure 4.1.(a) shows a synthetic image generated with

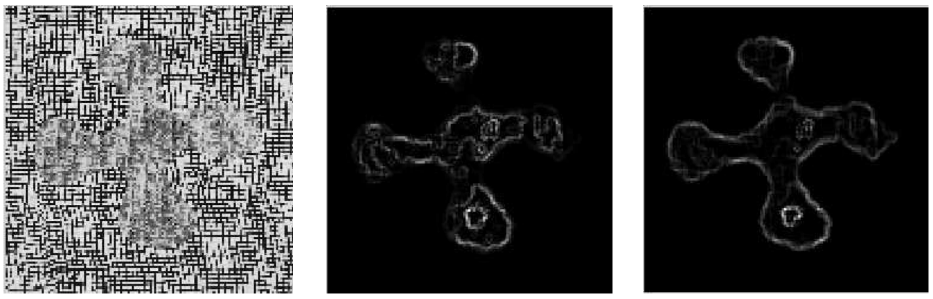


**Figure 4.2:** (a) Original image, (b) Contours of the likelihood map, (c) Contours of the proposed enhanced likelihood map.

the mask of figure 4.1.(d). Figure 4.1.(b) shows the likelihood map obtained for the target texture corresponding to the central region of the figure 4.1.(a) and figure 4.1.(e) displays its contours. Figure 4.1.(c) shows the enhanced likelihood map obtained for the target texture corresponding to the central region of the figure 4.1.(a) and figure 4.1.(f) illustrates its contours. Note that the proposed enhanced likelihood map (4.1.(c)) is smoother and more compact than the original likelihood map (4.1.(b)). Analogously, it can be observed that the contours of the enhanced likelihood map (4.1.(f)) are more accurate than the ones of the original likelihood map (4.1.(e)).

Figure 4.2 illustrates another comparison between the traditional likelihood map and our enhanced version. The texture image of figure 4.2.(a) has been generated using the same mask but a different pattern. Comparing contours of fig.4.2.(b) with those of the enhanced likelihood map (fig.4.2.(c)), we see that not only are contours of fig.4.2.(c) smoother but also they are more reliable than those shown in fig.4.2.(b). Moreover, we have removed superfluous edges inside the textured object, which is essential for a good snake convergence.

### 4.1.3 Snakes Maximizing the Likelihood (SML)

The contours of the enhanced likelihood map represent an approximation of the region of interest boundaries, which serves to define explicitly a potential map for the snake. Taking into account that physics-based and geodesic snakes present a poor convergence to concave boundaries, we use the generalized gradient vector flow approach applied over the likelihood map instead of using a heuristic potential field. Note that this process is equivalent to a regularization procedure performed on the likelihood map.

It consists of minimizing the following functional:

$$E(\mathbf{v}) = \int\int \mu e^{-\frac{|\nabla \tilde{L}|}{K}} |\nabla \mathbf{v}|^2 + (1 - e^{-\frac{|\nabla \tilde{L}|}{K}}) \cdot |\mathbf{v} - \nabla \tilde{L}|^2 dxdy \tag{4.10}$$

where $\tilde{L}$ is the enhanced likelihood map, $K$ and $\mu$ are constants. The minimum of the functional (4.10), $\mathbf{v}$, replaces the image force in the snake equation (4.2). Recall that $\mathbf{v}$ is a smooth vector field on the whole image domain that preserves the gradient of the enhanced likelihood map at significant contours. As a result, this regularization allows the snake surpassing weak likelihood contours giving a much better approximation to the region of interest. The definition of SML is not constrained to parametric snakes and can be generalized to geometric deformable models. However, geometric deformable models as they are formulated are not suitable for operating under non-binary derived external force. This fact lead us to reformulate the geodesic formulation and incorporate several improvements in the way to the final theory.

## 4.2 Stop and Go snakes

Most of current snakes define curve evolution within an energy minimization framework. In this context, the energy functional should achieve a compromise between adjusting to image features and achieving curve regularity. Two are the main tendencies for the definition of the minimizing energy.

- **Geodesic formulations in a contour space**

  General geodesic snake formulation defines the evolution of a snake within an energy minimization framework. In particular, the solution to the problem is the curve ($\Gamma$) of minimum length in a Riemannian surface with a metric ($g$) depending on the image contrast changes. It follows that if we note by $u$ the image to be segmented, the geodesic functional is given by:

  $$E_{geod} = \int_\Gamma g \; ds \qquad \text{with} \qquad g = \frac{1}{1 + |\nabla u|^2} \tag{4.11}$$

  The normal component of the Euler-Lagrange formulation characterizes geodesic snakes as the curve that satisfies:

  $$< \nabla g, \vec{n} > -g \cdot \kappa = 0$$

  where $\kappa$ is the curvature of $\Gamma$, $\vec{n}$ its inward unit normal and $<,>$ stands for the scalar product of two vectors. Therefore, using the gradient descent flow we obtain the following evolution equation:

  $$\frac{\partial \Gamma}{\partial t} = (g \cdot \kappa - < \nabla g, \vec{n} >) \cdot \vec{n} \tag{4.12}$$

  We can give the following interpretation to each of the terms involved in the above formula. The role of $< \nabla g, \vec{n} > \vec{n}$ is pretty clear: it is a vector field defined on the curve pointing to the region of interest that attracts the snake to the object boundary. Since its computation essentially relies on image edges, from a vector flow point of view, it can be considered as a *Static Vector Field* locally defining the target object. Notice that in the standard formulation (4.11) the scope of this static term reduces to a narrow surrounding of the boundaries of interest. The curvature term, $g \cdot \kappa \vec{n}$, influences different aspects of the snake evolution. On one hand, it defines its motion when it is located far away from the object boundaries. Since it depends on the evolving snake, it acts as a *Dynamic Vector Field* in the convergence process. On the other hand, it serves as a curve regularizing term, ensuring continuity of the final segmenting snake in a similar fashion [94] the membrane term of parametric snakes does. Finally, it gives the process a smooth behavior and ensures continuity during the deformation, in the sense that it prevents shock formation [83]. However, incorporating the curvature term into the convergence scheme has some disadvantages. First, it difficulties the snake convergence to concave areas. Second, guidance through the curvature is extremely slow, so in spite of giving regularity to the evolution equation, it hinders the numerical scheme since time increment is bounded by the second order term [91].

The main problem of (4.12) is that convergence to the object of interest relies on the properties of the external field. It is well known [84] that current external potentials do not ensure convergence to concave regions [84]. Even considering a regularization [93] of the external force, concave regions such that the unit tangent turns around more than $\pi$ between consecutive inflexion points of the object contour, can not be reached [84]. In order to increase convergence to concavities and to speed up the evolution a constant velocity term in the direction of the normal component is usually added to the evolution equation giving rise to 'balloon'-like snakes. The new term corresponds to area minimization and results in a constant dynamic speed in the gradient descent:

$$\frac{\partial \Gamma}{\partial t} = (g \cdot \kappa + V_0 - < \nabla g, \vec{n} >) \cdot \vec{n} \tag{4.13}$$

for $V_0$ a given constant. Notice that, in order to ensure that the scheme will stop at the boundary of interest, an equilibrium between the constant shrinking velocity, $V_0$, and the static vector field, $\nabla g$, must be achieved. One easily realizes that, should this condition be satisfied, incorporating the curvature term into the convergence scheme constitutes a significant drawback. For $V_0$ must overpass the magnitude of $\kappa$ to enter into concave regions but, at the same time, it should be kept under $min|\langle \nabla g, \vec{n} \rangle|$ (minimum taken on the curve to detect!) to guarantee non trivial steady states. This dichotomy motivates the restriction of the bounding the scope of $V_0$ to a given image region. On the other hand, $V_0$ can be thought as a minimizing area term. The concept of minimizing area has been studied further leading to the so called "region schemes" [85]:

- **Snake formulation in a region scheme**

  The "region terms" [85] are added to the minimization scheme as follows:

$$E(\Omega_{in}, \Omega_{out}, \Gamma) = \int\int_{\Omega_{in}} g^{(\Omega_{in})} dxdy + \int\int_{\Omega_{out}} g^{(\Omega_{out})} dxdy + \int_{\Gamma} g^{(\Gamma)} ds \tag{4.14}$$

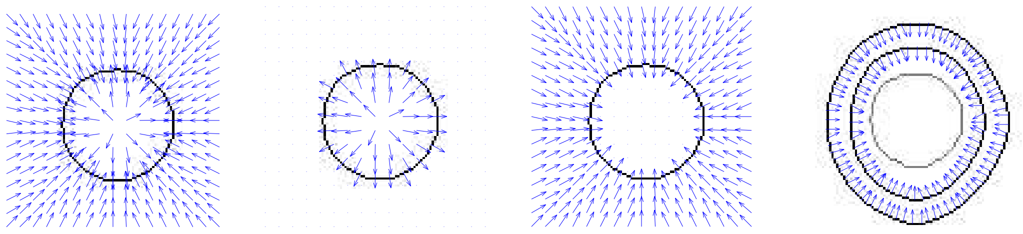


**Figure 4.3:** External force decoupling into Stop and Go terms. (a) Static external force. (b) Repulsive term. (c) Attractive term. (d) Dynamic go term.

where $\Omega_{in}$ and $\Omega_{out}$ refer to the inside and outside of the region of interest. There are two different approaches to determine the region: a "pseudo-static" approach and a dynamic one. In the first case, the attraction term that guides the evolution of each point in the curve is previously computed and kept fixed during the evolution [71], [82]. In the second one, measures of the regions descriptors depend on the evolving curve [92] [95], so that all parameters must be updated at each iteration.

Region-based approaches usually rely on a pseudo mask behavior. This effect can be seen in Zhu and Paragios region term

$$E = \alpha \log \frac{P_{Background}}{P_{Target}}$$

The last equation creates a static vector field where the contours of interest are located at $P_{Background} = P_{Target}$. The same effect could have been obtained if we have a mask defined:

$$M(x,y) = \begin{cases} \alpha & \text{if } P_{Background} > P_{Target} \\ -\alpha & \text{otherwise} \end{cases}$$

Hence, for any region-based approach we can define a mask which emulates the behavior of the region scheme and defines the object of interest.

We propose to reformulate (4.13) decoupling the regularity and convergence terms and embedding the scheme in a region-based framework.

Putting aside the energy minimization interpretation, the evolution of the curve is basically guided by an external force which defines an equilibrium state of the evolution. Whatever the external vector field, from the point of view of the evolving curve, achieving an equilibrium state can be decoupled into two stages: a straight forward advancing front defined outside the region of interest, and an inside region term opposed to it. Evolution stops if these two forces cancel along the curve of interest. Standard snake vector fields will serve to build an external force ensuring convergence and a mask of the region of interest, $I$, will be used to perform any vector decoupling /restriction.

### 4.2.1 Basics of Stop and Go formulation

To get an intuitive idea of the underlying mechanism of *Stop and Go*, let us assume that there is a mask representation (characteristic function) of the object we want to detect. That is, we have a function $I$ defining the region of interest R as:

$$I(x,y) = \begin{cases} 1 & \text{if } (x,y) \in R \\ 0 & \text{otherwise} \end{cases}$$

Then, a simplistic region evolution could be formulated as:

$$\frac{\partial \Gamma}{\partial t} = \alpha \operatorname{sign}(I) \cdot \vec{n} \tag{4.15}$$

where $\alpha$ is a parameter and $\operatorname{sign}(I)$ is a sign function defined for each pixel by $(-1)^{I(x,y)}$. This equation defines a simple dynamic vector field with an equilibrium state at the boundary of the object. The dynamic field shrinks the snake if it is located outside the object and blows it in its inside. Assuming that the initial snake is placed outside $R$, the former evolutions correspond, respectively, to an "inward" motion in the curve normal direction and an "outward" one in the opposite direction.

Although, unfortunately, we do not dispose of such a mask in practical applications, we can emulate the behavior of the former evolution. We will, first, derive the snake evolution equation in an ideal case (that is, still assuming we can produce a mask) and then describe in the next section how adapt *Stop and Go* formulation to the schemes (contour and region-based spaces) described in section 4.2 and to the likelihood map space, which estimates the characteristic functions/masks space.

First notice that in any minimization process, the snake deforms under two different vector flows: an attractor vector field (GO) moving the curve towards the target and a repulsive one (STOP) making that evolution stop. By means of the characteristic function $I$ these two motions can be decoupled and bounded like in (4.15). That is, the scope of the attractor GO field will be restricted to the outside of $R$ by means of the function $(1 - I)$, and the repulsive STOP to the inside through $I$. Let us assume once again that the evolving curve is outside the region of interest. Then, in a region-based approach, like the one given by (4.15), the GO term corresponds to an area minimization process restricted to the outside of $R$:

$$V_{GO} = (1 - I) \cdot V_0 \cdot \vec{n} \tag{4.16}$$

The above equation creates a dynamic "inward" motion to the region of interest, in the same fashion of that of 'balloon' snakes. In order to define the outward "motion", notice that there is no need to define the STOP field on the whole image. The scheme will work, as long as this vector is well defined in the environment of the contour we are looking for. This forces using a static vector field defined by the image features. Therefore, the STOP term can be defined by the "outward" gradient of any function, namely $g$, locally defining the contours of the object of interest:

$$V_{STOP} = I \cdot \langle \nabla g, \vec{n} \rangle \vec{n} \tag{4.17}$$

It follows that the evolution of an outward initial curve to the region of interest in Stop and Go formulation is given in the following terms:

$$\frac{\partial\Gamma}{\partial t} = \underbrace{< I \cdot \nabla g, \vec{n} > \vec{n}}_{\text{Stop}} + \underbrace{V_0 \cdot (1 - I) \cdot \vec{n}}_{\text{Go}} \tag{4.18}$$

As in the case of "balloon" snakes, the sum of both terms defines an (oscillating) equilibrium solution if we assure that $V_0 + < \nabla g, \vec{n} > \leq 0$ on the boundary of $R$.

The drawings in fig.4.3 illustrate the grounds of a Stop and Go field. A standard static field, $\nabla g$, having the circle as minimum is displayed in fig.4.3 (a). Its decomposition into the repulsive, $I \cdot \nabla g$ and attractive, $(1 - I) \cdot \nabla g$, vector fields is shown, respectively, in fig.4.3 (b) and (c). We observe that, like in any geodesic formulation, the $V_{STOP}$ term given by formula (4.17) corresponds to the projection of the repulsive vector field onto the snake unit normal, $\vec{n}$. Finally, fig.4.3 (d) represents a dynamic $V_{GO}$ field for the case of a shrinking ellipse placed outside the target gray circle.

### 4.2.2 Improving stop and go with a regularizing term

The "stop and go" approach leads a curve to the desired boundary, however some smoothness and continuity is desired on the final model. We will introduce the standard length minimizing curvature term to prevent high curvature segments in the snake and to prevent it from leaking into small holes of the object contour. Because regularity is only necessary in the final steps of the snake deformation, we will bound its scope to a neighborhood of the target object. Such restriction can be performed by means of a smoothed version of the mask $I$, given, for instance, by $\check{I} = G_\sigma * I$; for $G_\sigma$ a gaussian filter with standard deviation $\sigma$. Adding this regularity term to (4.18) the final evolution equation of Stop and Go snakes yields:

$$\Gamma_t = \underbrace{(I < \nabla g, \vec{n} > + V_0(1 - I)) \cdot \vec{n}}_{\text{Stop and Go}} + \underbrace{\alpha\kappa\check{I}\vec{n}}_{\text{Reg. term}} \tag{4.19}$$

The above formulation can be interpreted as selecting among all curves approaching the boundary of the target object, those complying to a given degree of regularity. Hence our formulation highly resembles that of parametric snakes, in the sense that regularity and convergence have been decoupled.

Reducing the action range of the regularizing term, has the following consequences. First of all, the curvature term has a radically different role than it had in the classical geodesic snakes formulation. The main difference to existent geodesic snakes schemes is that the snake only evolves under curvature near the object of interest. Removing curvature from the convergence phase endows snakes evolving under (4.19) with some useful properties. First, curvature is a strictly regularizing term in the same way the parametric snakes have their internal energy, hence its importance on the final curve is easily controlled by means of the weighting parameter $\alpha$. Furthermore, since its scope is bounded to the last steps of the snake deformation it does not trouble convergence to the contour concave regions. Finally, by restricting curvature to a band around contours, the integration step in an Euler numeric scheme, related to second order terms [83], is not a critical value. It follows that, except for the very last refinement steps, it can be arbitrarily high, so that speed of convergence increases. The above comments makes the scheme given by (4.19) conform to the following naive idea: obtaining a rough representation of objects should be computationally efficient, only requiring regularity is computationally expensive.

An important remark on the regularity of the final curve should be made. It is well known that the curvature term alone only guaranties continuity of the curve, preventing the snake from leaking into small holes. However, our scheme takes advantage of a synergy between the "stop and go" formulation and the curvature term to introduce a higher smoothness in the final shape. If we reformulate (4.19) as:

$$\frac{\partial \Gamma}{\partial t} = \underbrace{(V_0 + I\langle \nabla g, \overrightarrow{n} \rangle)\overrightarrow{n}}_{\text{Convergence}} + \underbrace{I \cdot (\alpha\kappa - V_0)\overrightarrow{n}}_{\text{Regularizing term}}$$

we find that, if a small $V_0$ ensures convergence, the snake regularity follows from the competition between curvature and $V_0$ near the object boundary. This competition favors the regularization during that later steps of the convergence and produces a smoothing effect on the curve. Our experiments show that the effect is extremely close to the thin-plate (curvature minimizing) term of parametric snakes. In this way, our scheme overcomes the problems that arise when the internal behavior of parametric snakes is formulated in geometric terms. Although the implicit version of parametric snakes internal energy is straightforward [94], the curvature minimizing term introduces a 4th order differential operator in the level sets formulation. This fact makes an efficient numeric implementation non feasible. Our approach allows similar effects in a simpler way.

### 4.2.3 Stop and Go Snakes design

**The roles of stop potential and the characteristic function. A design criterion.**

When trying to put the scheme to work some questions arise: Which is the stop potential function $g$? How can I produce the mask, $I$, to decouple the different effects provided by the scheme?

Analyzing the proposed scheme, we realize that although the stop potential and the decoupling term $I$ can be as general as we like, we can reduce the complexity of the design by making $g = 1 - I$. This holds true because $\nabla(1 - I)$ locally defines an outward vector field, in the same way the STOP term is desired to work. With this simplification in mind we will show how our scheme performs on typical object characterization spaces such as the contour-based space and the region-based space. Notice that any segmentation scheme can be expressed as a contour-based or region-based scheme. Remind that classical snake potentials are defined on the contour space, while region-based snakes potential is defined on the characteristic function/mask space. In this section we show the usage of the *Stop and Go* snake in both spaces. Besides, we propose the likelihood map space as a decoupling function as well as the stop potential function $g$.

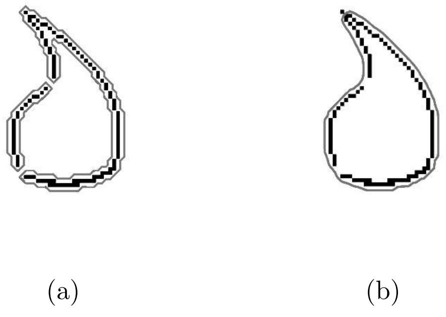


**Figure 4.4:** Modelling of an open drop. (a) Snake without regularity term and (b) Stop and Go snake.

**Contour space** The contour space is defined by the contours of the object of interest. In the contour space the snake converges to the desired contours and tries to close open edges. Although our approach is derived from the mask space, one can consider contours as a degenerate approach to regions in which each contour is a mask we want to approximate. This consideration allows our scheme to work on contour maps.

Figure 4.4 illustrates two effects of the stop and go scheme. The *Stop and Go* active model is showed deforming over a contour map, thus serving as an illustration of its usage in this kind of space. Figure 4.4.(a) shows the result of an evolution using *Stop and Go* without the regularizing term. Figure 4.4.(b) illustrates the result of the *Stop and Go* snake with the smoothing term. The figure shows a degenerated characteristic function. In this case the characteristic function are the edges of the "drop" figure. One can basically observe two main effects when comparing both figures: the smoothness of the curve is higher, and the term prevent the snake from leaking through small holes.

**Mask space** As mentioned in Section 3.2 region-based approaches usually rely on a characteristic function. It is obvious, that our approach works in this space provided that this was the theoretical scenery in which the active model has been designed to work.

### 4.2.4 Term Decoupling. The likelihood map space

Lacking of object masks in practical applications motivates searching for an alternate to perform the decoupling needed in the Stop and Go snakes. Likelihood maps arise as the perfect candidates for an approximation of object masks.

As it has been described in the former subsections, classical approaches use heuristically defined vector fields to create the velocity term of the snake. Usually, these approaches need previous classifications or contain implicit classification schemes, on which there is little control over the false positive and false negative regions. Therefore, an implicit classification, such as the region term of Paragios [71] or Zhu [82],

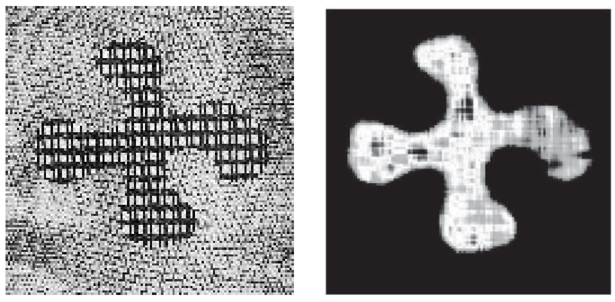


**Figure 4.5:** (a) Original image. (b) Likelihood map associated to the tetra-foil figure texture.

defines explicitly the regions we want the snake to converge to. The control on how accurate this classification is and the control on the false positive or false negative regions is introduced by adding terms (boundary term) to the region term. The boundary term acts as well as a smoothing and regularizing term.

To solve the deficiencies of the classic region approaches, we propose the use of a likelihood map. The likelihood map is defined as the likelihood value for each of the pixels of a given space to represent the target object. In our case, the likelihood map is the likelihood values for each of the pixels of a image. We can think of likelihood values as pseudo-probabilities which represents a continuous pseudo-density function. This approximation of the density function of the image features will serve as object masks and as external potential in the STOP term. Likelihood maps contain information about the accuracy of the classification to avoid false positive classified regions. Our approach exploits this fact to guide the deformation to the most probable regions. The main drawback of likelihood maps, as most region-based segmentation techniques, is its lack of accuracy at the real boundaries of the region of interest. In the particular case of likelihood maps, we obtain a set of regions with high likelihood value representing the objects of interest but are smaller in size than the real regions of interest. However, the boundary information of the likelihood map can be enhanced in multiple ways [87] as has been seen in the former section. We are not going to discuss now how to improve the likelihood and just stick to the matter in hand.

In figure 4.5(b) it is shown the likelihood map obtained for the textured tetra-foil of 4.5(a). Because bright regions (fig. 4.5(b)) yield an accurate representation of the target object, the likelihood map matches the behavior of a characteristic function. Therefore our estimation of the object mask will be a version of the likelihood map, $\check{L}$, normalized between 0 and 1. That is, replacing $I$ by $\check{L}$ in the *Stop and Go* formulation proposed in section 3, we obtain the following evolution equation:

$$\frac{\partial \Gamma}{\partial t} = \alpha\kappa\check{L}\cdot\vec{n} + \beta\check{L} < \nabla g, \vec{n} > \cdot\vec{n} + V_0(1-\check{L})\cdot\vec{n}$$

It only remains to define the STOP term, $\check{L}\cdot\nabla g$, that defines the object of interest.

**Using likelihood maps to define the Stop and Go field**

As pointed out in Section 3.2, any classic snake potentials (contour space, mask space) can be used as STOP term. The choice depends on the particular segmenting problem we handle. In the contour-based space setting, the object of interest is defined by image edges, and the snake converges to a closed model of the latter. Meanwhile, region-based spaces, are more suitable when operating in features characterizing the objects of interest (such as, statistical, texture based, motion based, color based, etc). We propose basing the STOP term on likelihood maps for feature spaces based segmentation using *Stop and Go*.

Notice that the gradient, $\nabla(1-I)$, properly defines the STOP direction, provided we convolve with a gaussian kernel, to overcome lack of differentiability of the characteristic function. Unfortunately, this step may close some concave areas to the snake. Now, as argued in former paragraphs, likelihood maps are continuous approximations of masks. Hence, the gradient $\nabla(1-\check{L})$, is a suitable STOP term. Besides since the former gradient is negligible outside a band around the contours of $L$, we can merge together the two factors, $\nabla g$ and $\check{L}$ of the STOP term. This leads to the following simplified formulation:

$$\frac{\partial\Gamma}{\partial t} = \alpha\kappa\check{L}\cdot\vec{n} + \beta < \nabla(1-\check{L}), \vec{n} > \cdot\vec{n} + V_0(1-\check{L})\cdot\vec{n} \qquad (4.20)$$

From the likelihood map point of view, our aim by using this formulation is to find the areas with higher probability of being the region of interest. The evolution by this equation will bypass the low likelihood regions. On the other hand, as the term $V_0(1-\check{L})\cdot\vec{n}$ can be interpreted to come from an area minimization, this effect will be present in the process. Therefore, by varying the $V_0$ parameter we will be able to "filter" the likelihood map by area. The regularizing term weighted by the curvature parameter $\alpha$ and the region parameter $V_0$ will impose a smooth behavior of the curve and, if desired, prevent the snake from leaking through little holes.

For the sake of the fastest speed of convergence possible, we use the following numeric scheme:

### 4.2.5 Stop and Go Numeric Formulation

Evolution of an initial snake $\Gamma_0$ under (4.20) is implemented using the Level Sets [86] formulation. That is, given any initial surface ($\phi_0$) properly defining the interior of $\Gamma_0$, the snake evolution at time $t$ coincides with the 0 level contour of the solution to:

$$\frac{\partial\phi}{\partial t} = (\alpha\check{L}\ \mathrm{div}(\frac{\nabla\phi}{|\nabla\phi|}) + V_0(1-\check{L}))|\nabla\phi| + \beta < \nabla(1-\check{L}), \nabla\phi >$$

The explicit Euler scheme we use in the numeric implementation of the former equation is given by:

$$\begin{aligned}\phi_{t+1} \qquad &= \phi_t + (\alpha\check{L}\tfrac{u_{xx}u_y^2-2u_{xy}u_xu_y+u_{yy}u_x^2}{|\nabla u|^2} + \\ &+ V_0(1-\check{L})|\nabla\phi_t| + \beta < \nabla(1-\check{L}), \nabla\phi_t >)\Delta t \qquad (4.21)\end{aligned}$$

where $\phi_t$ stands for the solution at time $t$ and derivatives are computed using centered finite differences. Notice that the speed of convergence hinges upon the magnitude of the time step $\Delta t$, the higher it is, the less iterations the algorithm needs. Accuracy is determined by $V_0$.

## 4.3 Likelihood map enhancements and Stop and Go

Stop and Go active models are suitable to absorb the generative approach used in the statistical deformable models (generative snakes), as showed in the former sections. In this section we want to deepen into the likelihood map enhancement process in order to define a more robust characteristic function for the Stop an Go term decoupling.

As it has been introduced in former sections, likelihood maps suffer from several drawbacks that should be solved, namely: its lack of accuracy at the real boundaries of the region, and the fact that it has high value on "safety" areas, and low likelihood elsewhere. The "safety" areas are usually sub-regions of the regions desired. Therefore, we need to improve the boundary information by enhancing the likelihood map.

### 4.3.1 Geometry based enhancement of the likelihood map

The likelihood map can be enhanced according to its geometry in order to preserve the maximum information of the boundary possible. A plausible method for filtering or enhancing images was proposed in another context by Salembier et al.[96]. In their approach, Salembier et al. represent an image as a tree (**Max-Tree**) composed by flat regions and linking information among regions. Each flat region is a node $C_h^k$ in the tree. The process for creating the tree is divided in two steps:

- *Binarization step:* For each temporary node $TC_h^k$, the set of pixel belonging to the local background is defined and assigned to the max-tree node $C_h^k$.

- *Connected components definition step:* The set of pixels belonging to the complement of the local background ($TC_h^k \backslash C_h^k$, where $\backslash$ is the set difference defined on connected components) are analyzed and its connected components create the temporary child nodes $TC_{h+1}^k$.

The idea underlying the above formalization is to create a tree recursively by the analysis of the relationships among connected components of the thresholded versions of the image. Figure 4.6 illustrates the process for the *max-tree* creation. The following explanation of the creation of the max-tree uses the notation for the flat zones depicted in figure 4.6(b). In the first step, a threshold is fixed to the gray value 0, and all the pixels at level $h = 0$ are assigned to the root node $C_0^1 = \{A\}$. The pixels with value strictly superior to $h = 0$ form the temporal nodes (in our case $TC_1^1 = \{B, C, D, E, F, G, H, I, J, K\}$). Each temporal node is processed as it was the original image, and the new node will be the connected components associated to the next level of thresholding $h = 1$, $C_1^1 = \{B\}$. Let's illustrate a split. Suppose the process goes on till processing node $C_2^1 = \{C\}$. The temporal nodes at that point are the connected components strictly superior to $h = 2$, $TC_3^1 = \{E, G, I\}$ and $TC_3^2 = \{D, F, H, J, K\}$, and the associated nodes at $h = 3$ are $C_3^1 = \{E\}$ and $C_3^2 = \{D\}$.

Taking advantage of that concept, we use the *Max-Tree representation* to describe the likelihood map. This representation allows further processing to be done keeping topological issues unchanged. The likelihood map topological enhancement can

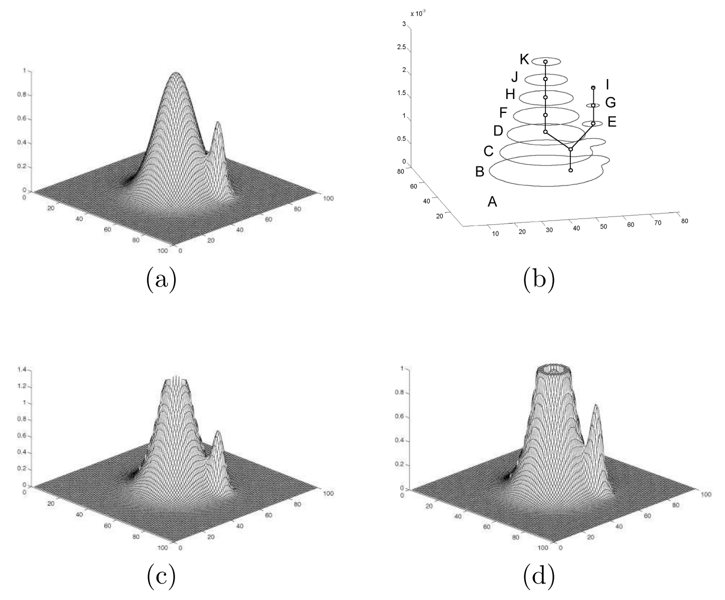


**Figure 4.6:** Creation of the topological enhanced likelihood map. (a) Original likelihood map. (b) Max-Tree representation. (c) Enhanced likelihood map without constraining maximum values. (d) Final enhanced likelihood map.

be described as follows. Given the *max-tree representation* of the likelihood map $T$ and a parameter $\theta$ that represents the likelihood value and which we consider to be high enough to determine a set of seed connected components, a safety set is defined $S_T = \{C_h^k \mid h > \theta\}$. For instance, if the likelihood map represents a probability map, we can assume that regions with probability over 90% are reliable enough to represent safety areas that are used to initialize the method. The method enhances topologically all the connected components that contain any of the safety areas by weighting the value of the connected component by a function $f(h)$, where $h$ is the threshold parameter for the connected component. The function $f$ is monotonically increasing and is desired to be S-shaped. Let's define the set of connected components that have a node in $S_T$:

$$S_N = \{C_h^k \mid C_h^k \supset C_i^j, \quad \forall \quad C_i^j \in S_T\}$$

Then, the topologically enhanced likelihood map is defined as:

$$\check{L}(h) = (f(h) \cdot S_N(h)) \cup \overline{S_N}(h)$$

where $(f(h) \cdot S_N(h))$ is the product of the value of each of the connected components of $S_N$ at level $h$ with the value of a function at the threshold level. The result of the product is an "enhanced" set of components in which the value of each of the components has been increased. $\overline{S_N}$ is the complementary set of $S_N$. The resulting enhanced map $\check{L}$ is the union of the enhanced set and the complementary set. The effect of this enhancing process is to increase topologically the value of the neighboring connected components of the high likelihood value areas while preserving the rest of the likelihood map intact. A final step is performed so that the resulting values after enhancing never surpass the value of the connected component at level $\theta$. Therefore, each connected component the value of which is over $\theta$ is constrained to the value at level $\theta$. Recalling figure 4.6, fig 4.6(c) shows the non-constrained enhanced likelihood map, and fig 4.6(d) depicts the final result. As can be observed in fig 4.6(d) the topologically enhanced area has stepper slopes and therefore defines better the contours of the region. On the other hand, the low probability region is kept at the same level.

### 4.3.2 Two class enhancement of the likelihood map

This last topological enhancement method can be applied also in conjunction with the two class enhancement likelihood map providing much better results. Remember that the formulation of the two class enhancement is as follows:

$$\tilde{L}(I) = \left(\frac{\lambda}{\lambda + |L(I|M) - L(I|\overline{M})|}\right) \cdot L(I|M_i) \qquad (4.22)$$

where $L(I|M)$ is the likelihood map for image $I$.

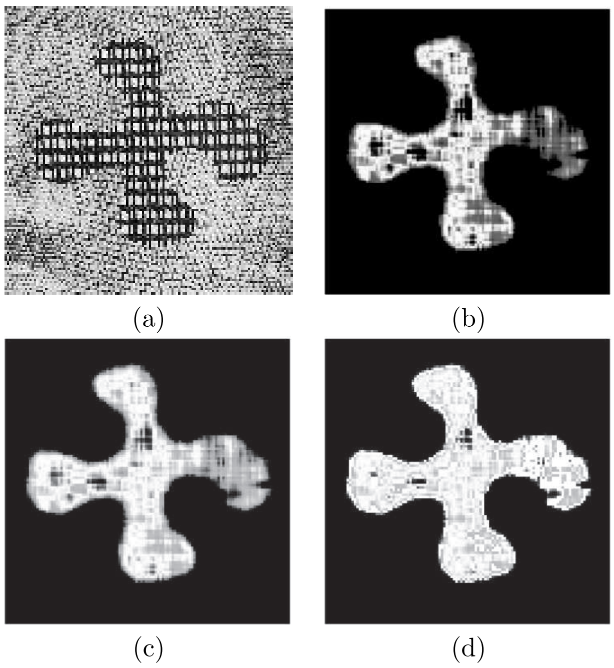

(a) (b) (c) (d)

**Figure 4.7:** Different likelihood map enhancements. (a) Original image. (b) Likelihood map associated to the tetra-foil figure texture. (c) Two class enhancing result of the likelihood map. (d) Topological enhanced likelihood map.

Figure 4.7 shows the method when applied to a synthetic image of a tetra-foil figure using a texture extraction process. Figure 4.7(a) shows the original tetra-foil image.

The texture of the tetra-foil area is used in the learning process and the likelihood map is built (fig. 4.7(b)). Figure 4.7(c) shows the enhanced likelihood map using the first method described. As one can see, the borders are better defined and some low likelihood areas which belong to the tetra-foil region are emphasized. Figure 4.7(d) shows the result after the topological enhancement. We can observe that the borders are clearly defined and the low likelihood regions are further emphasized keeping the general topology of the tetra-foil unchanged.