Medical Image Analysis - Image segmentation with One Shape Prior - A template-Based Formulation

Mhamed Hajaiej, Alaa Eddine Mahi and Akrem Bahri

March 5, 2020

1 Introduction

We, Mhamed Hajaiej, Alaa Eddine Mahi and Akrem Bahri, worked on the article titled "Image Segmentation with One Shape Prior - A Template-Based Formlation" [1]. This article has been written by Siqi Chen, Daniel Cremers and Richard J. Radke in 2012.

Article and work done: The core of the article is to reformulate the regularization-based single-shape-prior segmentation problem and develop a template-based approach to produce a segmentation which is a more systematic and a more efficient approach. Our work is based on some articles cited in the bibliography, but mainly on the article [1] mentioned above.

Repartition of work: There is no truly-delimited repartition of the work between us. To some extend, we may say that Alaa Eddine Mahi and Akrem BAHRI worked more on the theoretical aspects while Mhamed Hajaiej focused on practical and experimental aspects, but the present report was written by all of us, as well as the defense.

2 Presentation of the article

2.1 Review of the regularization-based approaches

The energy minimization is considered to be an efficient approach to the problem of image segmentation with one shape prior. We define the energy function as :

$$E(\mathbf{C},\mathbf{T}) = E_{data}(\mathbf{C},\mathbf{I}) \ + \ \lambda.D(\mathbf{C},\mathbf{T}(C_{ref})) \tag{1}$$

Where $\mathbf{I}$ represents an input image $\Omega \longrightarrow \mathbb{R}$. $\mathbf{C}$ is the shape presentation : typically, those used are parametric shape representations, signed distance functions (SDFs) or binary charasterisic functions u : $\Omega \longrightarrow \{0, 1\}$. T is the shape transformation, limited to parametric global transformation for the shape-prior segmentation (rigid, similarity, ...) and D is the distance shape measure.

The typical optimization strategy employed consists in : First, fixing T and updating $\mathbf{C}$ which becomes a standard image segmentation problem that could be resolved by optimization methods such as level-set-based gradient descent (suffers from low convergence), discrete graph cuts (works only for specific class of energies). Second, Fix $\mathbf{C}$ and update T which is solved typically using Gradient-based optimization methods such as gradient descent. Then, these two steps are iterated until convergence.

These approaches suffer from some drawbacks : First, the energy minimization is not guaranteed, since we have a local optimality for the second step here below. Second, adding to the inherent local optimality, the convergence can be very slow which doesn't suit for medical imaging where there is an urgent need to get fast, robust and ideally optimal solutions. These shortcomings for the shape-prior-based segmentation approaches can be illustrated by the Fig. **??** , presenting the segmentation results of ultrasound cardiac test images where the algorithm converges to a local optimum in nearly 30 seconds with 100 iterations, while the method that will be presented later is able to converge to a near-global optimum in as little as 3 seconds.

2.2 Principle of the new template-based formulation

The shape representation used in this article is the binary characteristic partition $u : \Omega \longrightarrow \{0, 1\}$. The data energy term in equation (1) uses the Maximum-Likelihood model :

$$E_{data}(u) = -\int_{\Omega} \log \mathbf{P}_{int}.u \; d\mathbf{x} - \int_{\Omega} \log \mathbf{P}_{out}.(1-u) \; d\mathbf{x} = \int_{\Omega} Q.u \; d\mathbf{x} + cst \tag{2}$$

where $\mathbf{P}_{int}(\mathbf{x})$ and $\mathbf{P}_{out}(\mathbf{x})$ are the probabilities that pixel $\mathbf{x}$ belongs to the object and the background respectively, $\mathrm{Q}(\mathbf{x})$ is the log-likelihood map describing the confidence that a certain pixel belongs to the object or the background.

Then, the λ -balanced regularization term added to the data term in the equation (1) is replaced by using the reference shape u_{ref} as a template in the formulation. We use then $u = T(u_{ref})$ to obtain :

$$E(u) = E_{data}(T(u_{ref})) = \int_{\Omega} Q.u_{ref}(T(\mathbf{x}))d\mathbf{x} \tag{3}$$

Hence, the image segmentation problem concerns the minimization of the energy function defined by the Equation (2) over the geometric transformation parameters T. This template-based formulation offers many advantages compared with the regularization-based form presented in part 2.2, such as the non dependence to parameter λ.

Connection with the intensity-based image registration problem : The template-based reformulation of image segmentation with one shape prior is related to image registration. This is, the segmentation problem is equivalent to registering the binary shape template u_{ref} and the image log-likelihood map Q when the registration metric is standard correlation. For instance, to efficiently optimize (2), most intensity-based registration algorithms tend to apply various forms of gradient-based strategies such as gradient descent or Levenberg-Marquardt to a cost metric. While this approach works well for non-rigid or deformable transformations, it is difficult to locate the optimum solution with gradient-based local optimization techniques when the transformation is limited to global similarity transform since the registration cost function is non convex. Fig. 2 illustrates an attemptof segmentation based on registering the log-likelihood map to the binary shape template with the correlation metric under similarity transform : the algorithm converges to a local optimum in approximatively 15 seconds (compared to 3 seconds for the method presented in the next part).

2.3 Adaptation : more efficient and powerful approach

2.3.1 Principle

With the template-based formulation, we start with an initial binary template image and we modify it using the information from the log-likelihood map to do the segmentation. In practice, we

iteratively alternate between finding the best translation for the template and between finding the best rotation and scaling. When we reach convergence, we apply a deformable transformation to make the model softer as all the previous transformations are rigid transformations. In order to find the best translation, rotation and scaling, we do an exhaustive search. This way, we avoid the gradient method which can easily get stuck in local minimum. Even if this seems to be a bad idea because of the complexity of the full search problem, the computations can still be done in a matter of seconds thanks to an easy trick : Solving the dual problem in the space of Fourier. In fact, for an image of size n×n we pass from a complexity of O(n^4) to O(n^2 $log(n)$). First we will present the present the method for finding the translation on the Fourier domain and we will compare its speed to the naive method. Afterwards, we will show how we can adapt this method to find the rotation and scaling using the log-polar transformation and we will show the results on one example.

2.3.2 Finding translation using Fourier transform

If we denotes by Q the log-likelihood map and u_{ref} the template image. The problem of finding the best translation is equivalent to finding the translation t such that :

$$E(t) = \sum_x Q(x).u_{ref}(x-t)$$

The$\sum_x$ is O(n^2) and we have to do it for the n^2 possible translation t. This results in a O(n^4) problem, which is computationally slow. However, this problem has a dual in the Fourier space given by :

$$E(t) = F^{-1}(F_Q.F^*_{u_{ref}})$$

Using the Fast Fourier Transform (FFT), we can compute F_Q , $F_{u_{ref}}$ and then apply F^{-1} to obtain E(t) in O(n^2 $log(n)$).
We have implemented the two methods and We compared the speed of the search using the Fourier dual and using the naive method as a function of the size of the image in Fig 3. We can remark that for a small image of size 53×71, the naive algorithm takes 0.26s comparing to 0.024s for the fast one, while for an image of size 482×644, the naive algorithm takes 14min and 30s comparing to 0.28s for the fast one. Note that 14min is too slow for finding a translation, especially that we have to do it multiple times until we reach convergence.
Finally, we show an example of a template and an image before and after searching for the translation as well as the obtained E(t) in Figure 4,5 and 6.

2.3.3 Adaptation for finding the rotation and scaling

Finding the rotation and the scaling is more complicated and can't be done directly with the same method as finding the translation. However we can transform it to a similar problem by applying a certain transformation to the data : the log-polar transformation.
We consider the log-polar coordinate system (ρ,θ) where ρ is the log of the distance between the Cartesian point (x,y) and θ is the angle. If we chose the point (0,0) as the center, we have :

$$\rho = log(\sqrt{(x^2+y^2)}) \text{ and } \theta = tan^{-1}(\tfrac{y}{x})$$

Changing the scale of (x,y) to (sx,sy) in the Cartesian space is equivalent to a translation with (log(s),0) in the log-polar space. In the same way, the rotation of (x,y) to (x cos(r)+y sin(r),-x sin(r)+y cos(r)) is equivalent to a translation with (0,r) in the log-polar space. And thus, finding the rotation and scaling in the log-polar space is equivalent to finding the translation $t^{rs} = (r, s)$

minimizing the energy $E(t^{rs})$ and now we can apply the same method presented in the previous paragraph, after transforming the images to the log-polar space.
We show an example of the log-polar transformation in Figure 9

2.3.4 Iterative algorithm

We use iteratively the estimation of the translation and of the rotation-scaling for the segmentation problem until we reach convergence. This corresponds to algorithm 1 in the annex.
We show the effect of the implemented algorithm in one example in Figure **??**

2.4 Deformable transformation

2.4.1 Principle

Despite its efficiency, the current method uses only translation, rotation and scaling of the shape prior in order to do the segmentation, which is limited in practice as most of the times, the desired segmentation can't be obtained as a similarity transformation of the available shape prior. In order to solve this, we have to introduce a deformable transformation. Lucky, this is still possible in our case. In fact we can start by finding similarity transformation and we add at the end a small deformable transformation operating local changes on the transformed shape prior.

2.4.2 Proposed method

Giving the high variety of the possible local deformation that we can encounter, it's very difficult to make a parametric description of the deformable transformation. That's why we opt for a free-form deformable (FFD) model, based on the B-splines. This method consist on defining a mesh on the template, and to deform it by moving the control points. We show the example seen in the course to demonstrate the principle of this method in Figure 12 where the red dots are the original points of the mesh and the green dots are the moved points. We remark also that the larger is our mesh, the more control points we have we can approximate more complicated deformation (at the cost of longer computation of course).
More technical details of this method are shown in the annex. We show the results obtained in the paper using this method in Figure 13.

2.4.3 Implemented method

We had problems when trying to implement the method proposed by the paper, that's why we have decided to apply a simpler algorithm, based on the same principle, in order to show the importance of the deformable transformation.
We have defined a mesh and control points like the method proposed by the paper, but we didn't use the B-splines functions. Instead, we define around every control point a small window and we search for its optimal position inside this window (this works because we have already found the similarity transformation and the true position of the control point shouldn't be far away). Thus, we search for the best translation that we can apply to move the control points P in order to have a good deformation. This method is much simpler and it probably won't work in complicated cases, but it is still useful to show how we can adapt our algorithm to deformation using the FFD model.
We have applied this method on some easy examples that we show in Figure 17 and 20

3 Intensity based segmentation using Gaussian Mixture Models

Until now we have been using the log-likelihood map Q as a the typical two-phase Gaussian model in (1), which is fixed during each iteration.Nonetheless, the foreground probability $P_{in}(x)$ and the background probability $P_{out}(x)$ can have more general, hence more accurate forms, giving the log-likelihood map $Q(x)$ different representations which can be adapted considering the image segmentation task at hand.

$$\begin{aligned} Q(x) &= -\log\left\{e^{\frac{-(I(x)-M_{In})^2}{2\sigma^2}}\right\} - \log\left\{e^{\frac{-(I(x)-M_{Out})^2}{2\sigma^2}}\right\} \\ &= (I(x)-M_{Out})^2 - (I(x)-M_{In})^2 \end{aligned} \tag{1}$$

Where M_{In} and M_{Out} are the mean intensities and σ the variance of the two Gaussian distributions representing respectively the foreground and the background

3.1 Interactive segmentation

The framework we will be using is the Interactive image segmentation which is a commonly used method which can be summarized as following. A user provides strokes and bounding box which represent not only hard constraints that the segmentation should satisfy, but also a method to approximate the object which we denote as O and the background denoted as B intensity distributions. In the case of the bounding box, the area outside the bounding box represents the background. Hence, we can define the log-likelihood map Q which can be defined -using the indicator function $\mathbb{1}(A)$ where A is a given set- for the three regions which form a partition of the observed image. First, the background then the object and finally the complementary of the union between the object and the background $(O \cup B)^c$.

3.2 Gaussian Mixture Model

Until now we have used a simple model using parametric Gaussian distributions to represent the background and foreground assuming both have the same variance. However, we can use Gaussian Mixture Model (GMM) to represent complex intensity models. We represent each of the foreground and background using a full variance Gaussian mixture to keep the generality of our approach, with K components each, which amounts to 2K components. Thus we obtain the following probability density equations.

$$\begin{aligned} P_{in}(x) &= \sum\nolimits_{k_{in}} \pi_{in}(k_{in}) P(x|\theta_{in}(k_{in})) \\ P_{out}(x) &= \sum\nolimits_{k_{out}} \pi_{out}(k_{out}) P(x|\theta_{out}(k_{out})) \end{aligned}$$

Where θ represents the parameters of our Gaussian distribution, and $k_{in}, k_{out} \in 1, ..., K$ the component of either the background or the background respectively. Finally, π_{in}, π_{out} are the weights of the k^{th} foreground and background component respectively.

3.3 Approach using GMM

Given u_{ref}, we create K components of the GMM for both the background and foreground. The most common approach is to use an Expectation-Maximization (EM) algorithm, which assigns probabilities for each component in the E-step and updates the different parameters of the GMM

in the M-step, given an intensity observation. Nonetheless, this approach is very costly computationally without significant practical benefits. That's why, we first divide both the foreground and background to K pixel clusters, then we generate a Gaussian component for each cluster. However, this approach assumes that we can find a well separated and low-variance clusters. This assumption can be verified using the color quantization technique which uses the eigenvector of the color variance to determine how to split the clusters as described in detail by the work of Orchard and Bouman [2].
As we update the similarity transform, we obtain a different intermediate segmentation. To update the GMM, a straightforward approach will be to generate a GMM from the ground-up during each iteration. However, this may be very costly computationally. Instead, we adapt the EM type update, where each pixel in the new segmentation is assigned to one of the K components, by evaluating its likelihood of belonging to each component with the GMM obtained previously. Besides, user provided information such as strokes and bounding box can be utilized be setting $Q(x)$ to large positive or negative values depending on whether x belongs to the predefined foreground or background areas.

3.4 Experiments

The figure below Fig. 21 represents an example where we use the GMM intensity model with $K = 5$. The first figure, Fig. 21a shows the input image as well as the reference shape template colored in red. The second figure, Fig. 21b shows the segmentation result after the similarity transformation parameters are updated and the algorithm converges. The last figure, Fig. 21c shows the final segmentation result after deformation registration which enables us in this case to recover the stem accurately.

References

[1] S. Chen, D. Cremers, R. J. Radke, *Image segmentation with one shape prior A template-based formulation, In Image and Vision Computing*, Volume 30, Issue 12, 2012, Pages 1032-1042, ISSN 0262-8856

[2] M.T. Orchard and C.A. Bouman, *Color quantization of images. IEEE Trans. Signal Processing*, 39(12): 2677-2690, 199.

4 Annex

4.1 Improvements

4.1.1 Partial optimality

Using the Shape prior image segmentation algorithm while neglecting the window-ltering effect, each individual update of t and (r, s) can obtain the global optimum. However, we have no guarantee to attain a global optimum for all four parameters (t, r, s, u). For instance A noise-corrupted image with two scissors (one big and one small) since the maximum likelihood energy model is unnormalized Eq. 1 , we can easily verify that the energy of the big scissor segmentation is higher than the energy of the small scissor segmentation. However, given the initial shape template, our algorithm only converges to the local optimal result.
This issue can be addressed by normalizing our energy function so as to be invariant regarding a change of scale of the foreground image.However, this solution isn't optimal since the registration of the foreground image isn't always accurate. Thus, adding a regularization term which diverges to infinity where the energy is bounded by a certain threshold that we can determine experimentally can constrain our algorithm to converge by changing the set of feasible solutions.

4.1.2 Topological approach : Persistent Homotopy

The Shape prior image segmentation algorithm relies on clustering using Generalized Gaussian Mixture Models with full covariance matrices which achieves better results compared to other clustering methods such as Fuzzy C-means or K-means++. However, this type of clustering doesn't exploit the geometrical properties of the observed objects which can be efficiently exploited by topological clustering approaches who can give optimal results when the given data set has inherent geometrical properties that can be represented as homology groups. Which is exactly the case in the medical image analysis field where organs or medical instruments can be represented as a homeomorphic mappings of a simple manifold.

4.1.3 Adapting Convolutional Neural Networks

While in this paper we used mainly standard segmentation methods using a combined template based registration and segmentation approach. We haven't discussed the possible use of Deep Neural Networks to perform these tasks. In fact, Deep Learning has achieved remarkable results performing many computer vision tasks such as object detection, recognition and image segmentation relying on now popular Convolutional Neural Networks architectures such as Fast R-CNN or Yolo. These methods rely on regional feature detection using bounding boxes, which is a similar approach we used to perform image segmentation using user-provided information as in section 3. That's why, using transfer learning to perform image segmentation on already pre-trained networks which can be fine tuned using the images of interest we want to segment can lead to a more accurate segmentation while with easier algorithms which can be adapted to the task at hand.

4.2 Code remark

The function giving the optimal translation using the FFT method was giving always the result that we hoped for. However, the function doing this for rotation and scaling was more complicated to manage. In fact, it's harder to find the allowed set of acceptable set of transformation : the transformations that don't drastically change the shape prior as we can for example map a part of it out of the image or split it in two pieces. Also, the quality of the rotation and scaling found

depends on the center that we chose for the log-polar transformation which was an additional difficulty. Finally, we could find ourselves stuck in situations where the shape prior is not updated anymore even if we didn't reach the transformation that we are looking for. In fact, not all of the implementation details are clearly cited in the paper and that's why we couldn't obtain the same results.
For the function computing the deformation, the manipulation of the "rectangle" and " quadrilateral" objects are inspired by an open source code that we have found explaining the quadrilateral to rectangle transformation of the package PIL. It is not as effective as the method proposed by the paper, but it still shows how the method can adapt to deformable transformations.

4.3 Details of the B-spline FFD model

We can formulate this by defining the mesh domain X×Y, a grid over the N×N template domain. We define the control points $P = (P^x_{m,n}, P^y_{m,n})$ for $(m, n) \in [1, X] \times [1, Y]$ and $(x, y) \in [1, N] \times [1, N]$. We chose to use the cubic B-spline functions because of their continuity properties (C^1 at control points and C^2 everywhere else). Thus, the deformed position of a pixel (x,y) is defined by the cubic B-splines and the 16 neighboring control points :

$$T(x, P) = \sum_{k=0}^{3} \sum_{l=0}^{3} B_k(u) B_l(v) P_{i+k,j+l}$$

With $i = \lfloor \frac{x}{X} \rfloor - 1$,$j = \lfloor \frac{y}{Y} \rfloor - 1$,$u = \frac{x}{X} - \lfloor \frac{x}{X} \rfloor$, $u = \frac{y}{Y} - \lfloor \frac{y}{Y} \rfloor$ and B_k are the basis function of the cubic B-spline.
Now that we have formalized the problem, all we need to do is to optimize the energy over P:

$$E(P) = \int_\Omega Q.u_{ref}(T(X, P))dX$$

This can be done using the following iterative algorithm that we apply after finding the similarity transformation :
-While not converged do
——Compute the gradient $\nabla E = \frac{\partial E(P)}{\partial P}$
——Update The control points P
-end while

4.4 Algorithm

4.4.1 Algorithm1 : iterative algorithm for finding translation, rotation and scaling

-While not converged do
——Compute translation t from u_ref and Q using FFT
——Update u_ref with t
——Compute rotation r ans scaling s from u_ref and Q using log-polar and FFT
——Update u_ref with r and s
-end while

4.5 Figures

Algorithm 1 Shape prior image segmentation with GMM

Input : Shape prior template u_0 and the image I.
Initialization: Compute initial GMM parameters and Q
-while Not converged **do**
—— 1.Compute translation t from u_i and Q
—— 2.Update u_i with t to u_{it}
—— 3.Compute s and r from u_{it} and Q
—— 4.Update u_{it} with s and r to u_{i+1}
—— 5.Update the GMM parameters and Q.
end while;
—— Deformable transformation estimation.

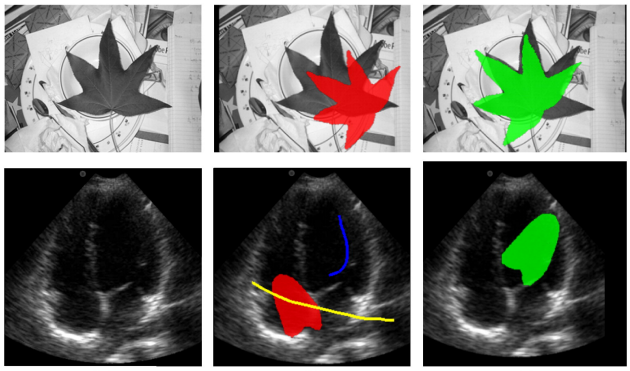

Figure 1: Segmentation result

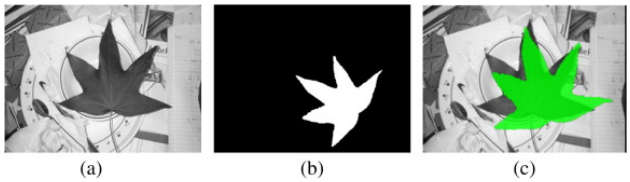

Figure 2: Segmentation result with multi-resoltion gradient descent method

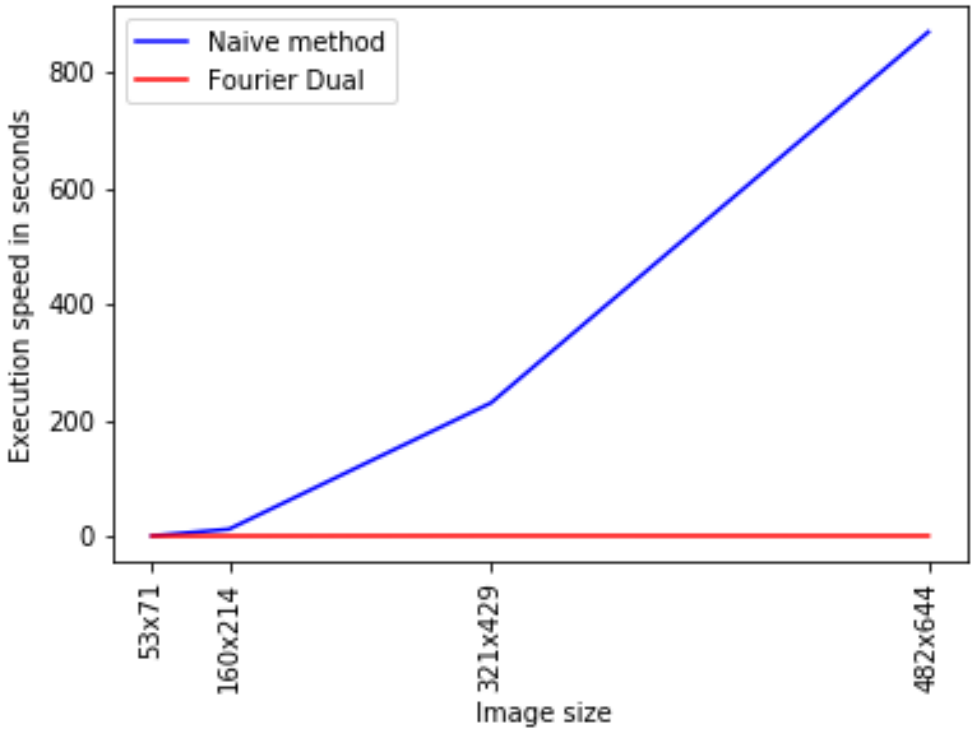

Figure 3: Execution speed as a function of the size of the image

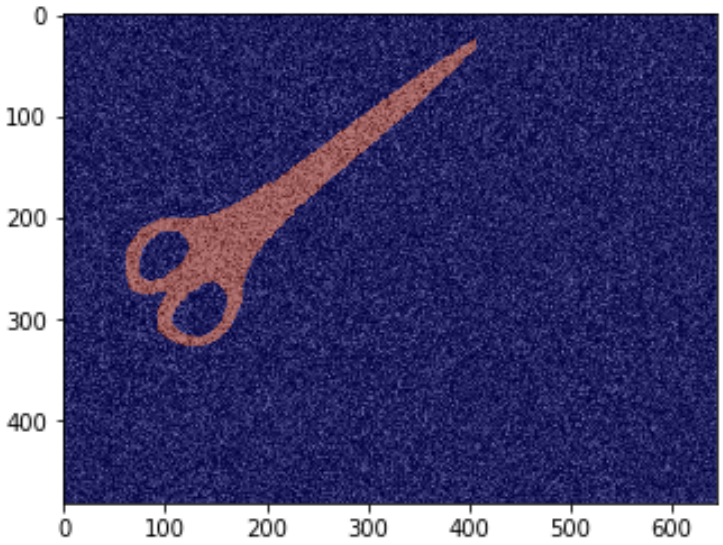

Figure 4: Before searching for the translation

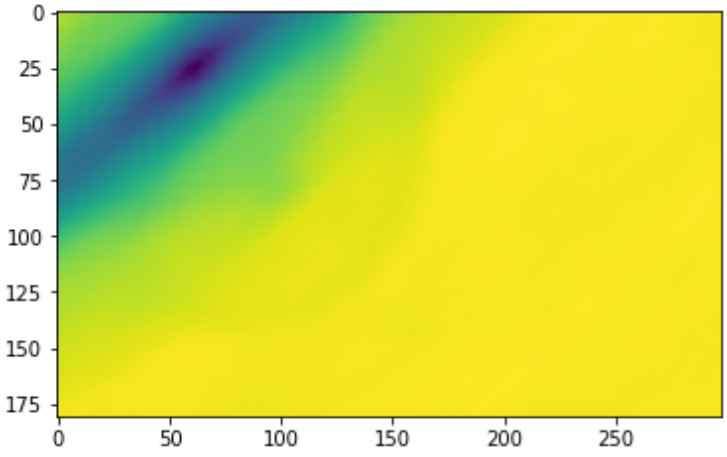

Figure 5: The energy E(t)

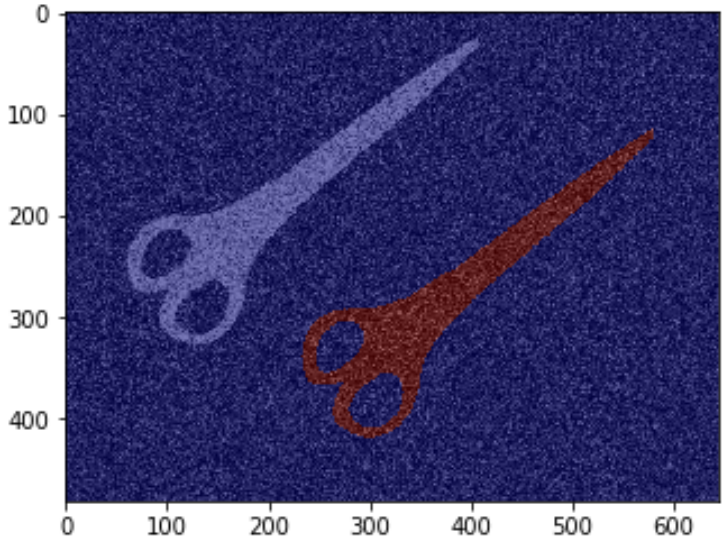

Figure 6: After finding for the translation

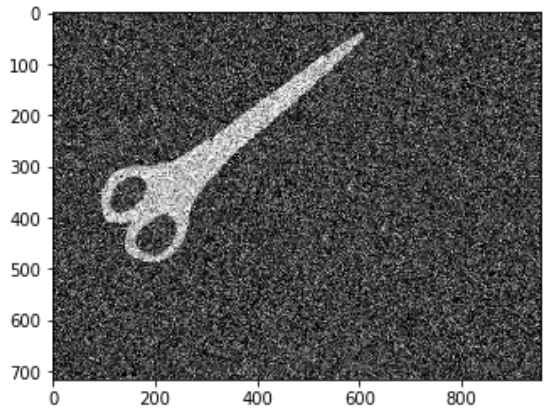

Figure 7: Cartesian coordinates of the original image.

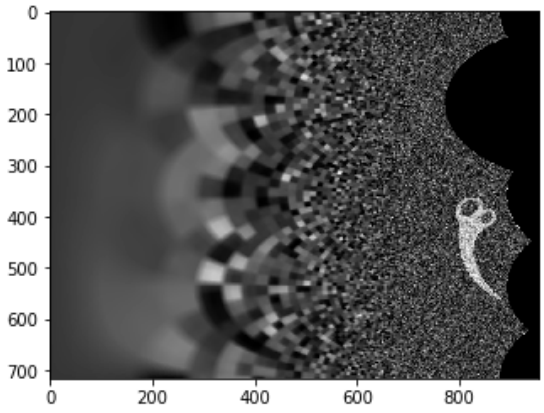

Figure 8: Log-polar coordinates with (x=500,y=500) as center

Figure 9: Example of the log-polar transformation

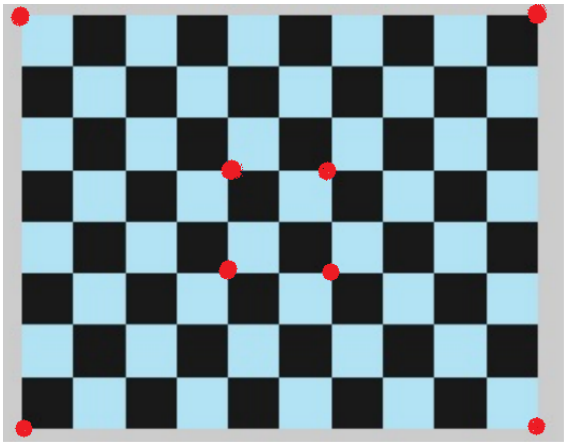

Figure 10: The original image.

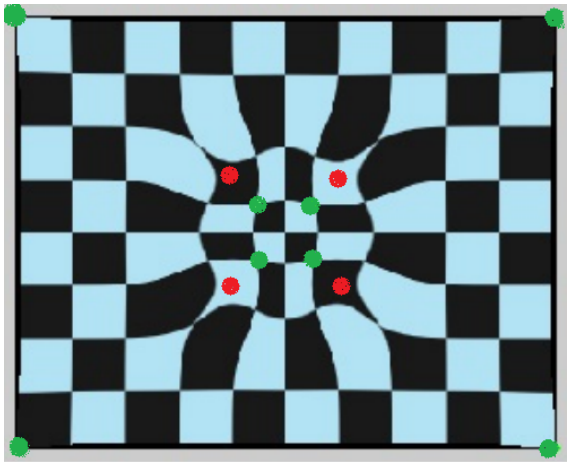

Figure 11: The transformed image

Figure 12: Principe for the FFD model.

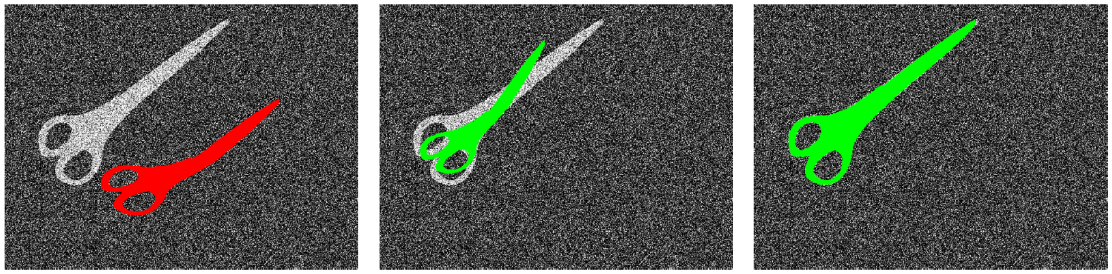

Figure 13: (a)Original image and template (b)Results after similarity transform (c)Results after applying the FFD deformation

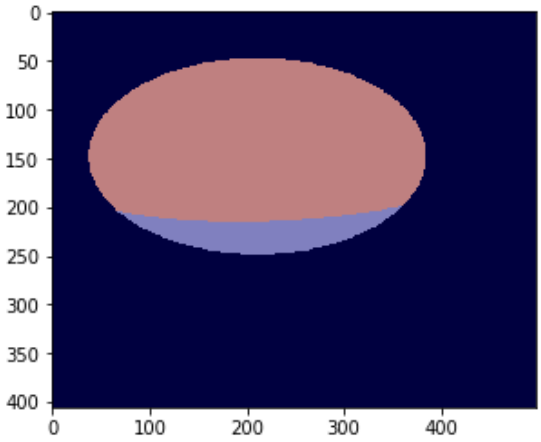

Figure 14: The original image.

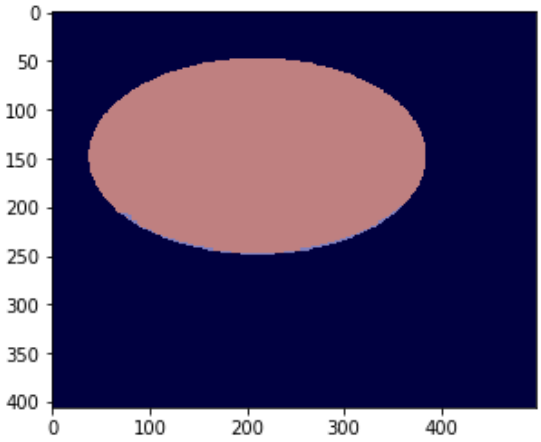

Figure 15: The deformed image

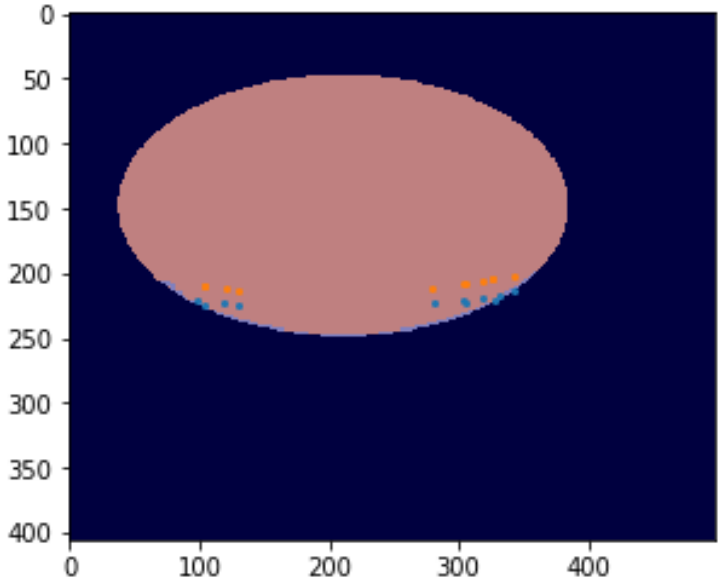

Figure 16: Mesh representation. Red dots are a subsample of the original control points. Blue points are their position after the transformation.

Figure 17: Finding simple deformations.

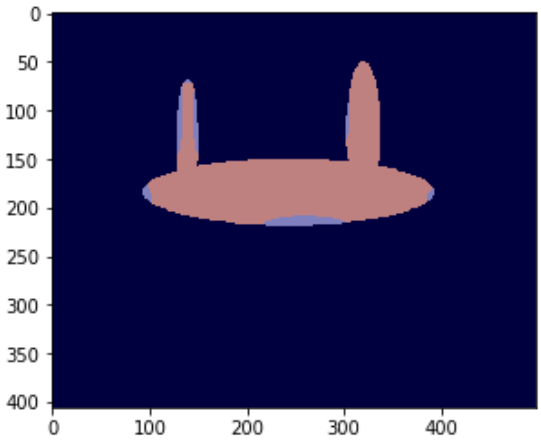

Figure 18: The original image.

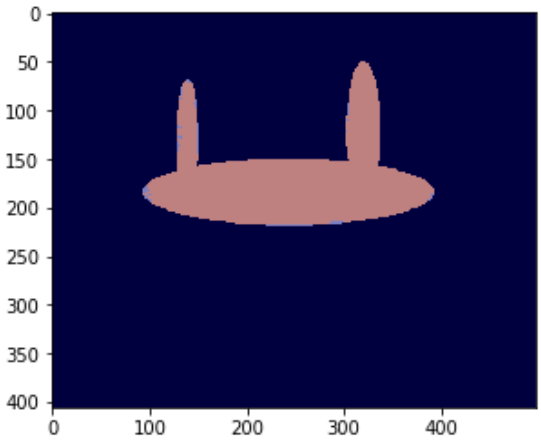

Figure 19: The deformed image

Figure 20: Finding simple deformations.

Figure 21: Shape-prior segmentation with GMM (a) Image with the reference shape in red. (b) The segmentation result after similarity transformation. (c) The segmentation result after deformable transformation. This gure is best viewed in color.