Universitat Politcnica de Catalunya
Universitat de Barcelona
Universitat Rovira i Virgili

Master in Artificial Intelligence

Computing vision

Top-Down Image segmentation

Authors:
Alejandro Surez Hernndez
Johannes Heidecke

December 2016

Contents

List of Figures

1 Obtain Image Segmentation by Active Contour Models

Active Contour Models (also called "snakes") depend on adequately set parameters in order to work as desired. In this section we observed the effect of changing the parameters' values when trying to segment the outline of a heart from an image.

1.1 Parameter Experiments

The parameters to evaluate are the following:

- α: Controls the *elasticity* of the model, with higher values corresponding to less stretching. A high α leads to the model trying to keep the points as close to each other as possible.
- β: Controls the *rigidity* of the model, with higher values corresponding to less flexibility. A high β leads to a model with as little curves as possible.
- λ: Controls the proportion of the balloon force. The model expands for $\lambda > 0$ and contracts for $\lambda < 0$.
- κ: Controls the proportion of the external force that stops the model when arriving at edges. High values of κ make the model stop at weaker edges. In our case we set κ to $\frac{\kappa'}{\max(G_x,G_y)}$, with G_x and G_y being the components of the force field. We tweak κ' instead of κ itself.
- *maxstep*: Defines the maximum step size in pixel, aimed at minimizing unwanted crossing-over.

In the following we examine the effects of changing the value of the different parameters. We try to examine optimal and critical values. We keep a base configuration set of $(\alpha, \beta, \lambda, \kappa', maxstep) = (0.1, 0.01, 0.05, 0.2, 0.4)$ and then change individual parameter values while keeping the others invariant.

1.1.1 Alpha

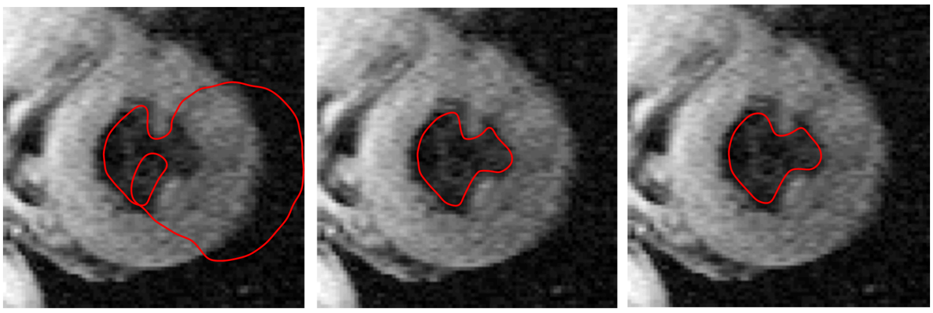
The parameter α controls the elasticity of the model. The given value of 0.1 yields relatively good results. As can be seen in figure 1, our experiments showed that decreasing α to 0.05 augments the results slightly, since the model can "grow" more tightly into the corners of the heart. When decreasing the value further, the model fails to stop at the edge of the heart segment and grows too far. The optimal value is thus somewhere near 0.05 and the first critical value that leads to failure lies between 0.02 and 0.05.

Our experiments show almost no change when increasing α to 0.15. A value of 0.2 already seems to be a critical value - the model does not expand at all and stops in the initial shape of a circle, see figure 2.

1.1.2 Beta

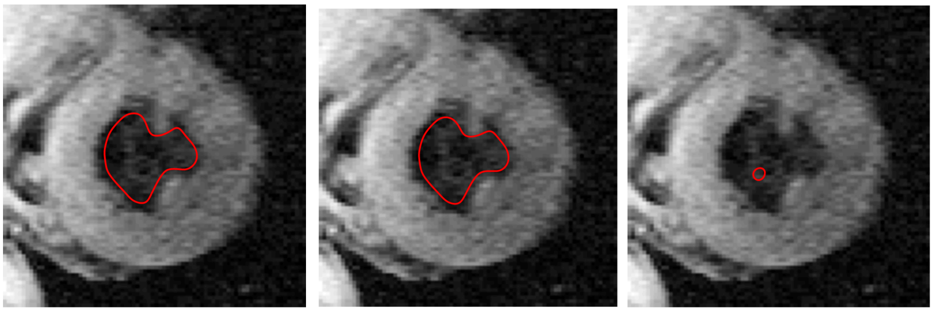
The parameter β controls the rigidity of the model. As figure 3 shows, the initial value of 0.01 leads to the same results as any smaller value, including 0. This indicates that 0.01 is sufficiently small enough to allow curves that fill the shape adequately.

Increasing β to 0.55 shows a more rigid shape with less defined curves. This change leads to worse results than the initial configuration. A critical value is reached for 0.65, when the model fails to grow beyond the initial shape, see figure 4. 0.01 seems to be the optimal value for this parameter.



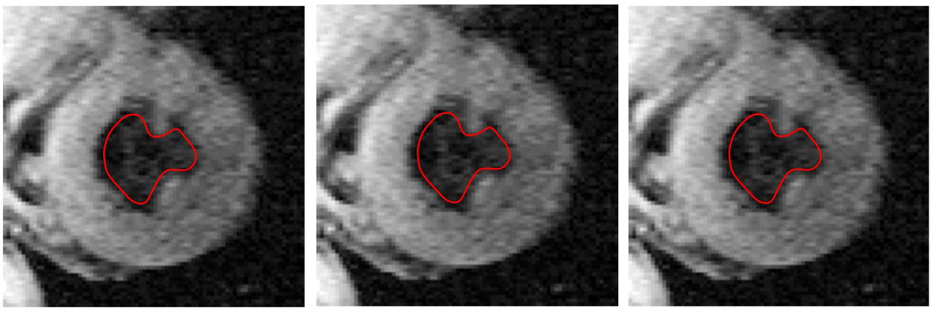
(a) $\alpha = 0.02$ (b) $\alpha = 0.05$ (c) $\alpha = 0.1$

Figure 1: Heart: tweaking the values of α. $\alpha \leq 0.1$



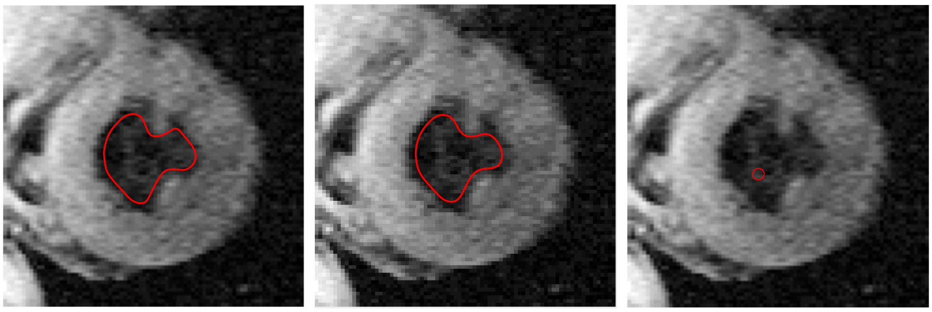
(a) $\alpha = 0.1$ (b) $\alpha = 0.15$ (c) $\alpha = 0.2$

Figure 2: Heart: tweaking the values of α. $\alpha \geq 0.1$



(a) $\beta = 0$ (b) $\beta = 0.0001$ (c) $\beta = 0.01$

Figure 3: Heart: tweaking the values of β. $\beta \leq 0.01$



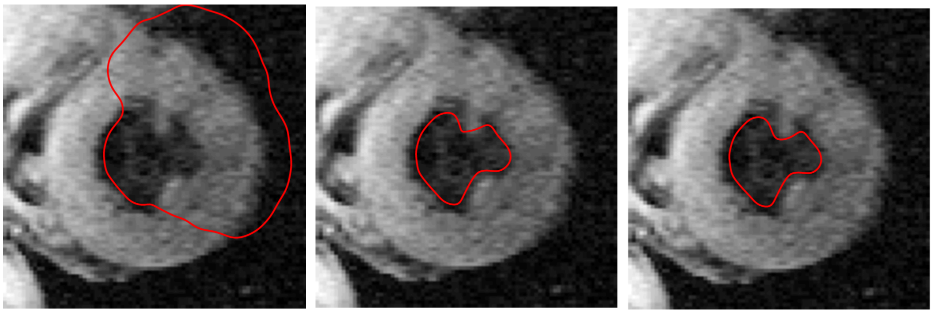
(a) $\beta = 0.01$ (b) $\beta = 0.55$ (c) $\beta = 0.65$

Figure 4: Heart: tweaking the values of β. $\beta \geq 0.01$

1.1.3 Kappa

(**NOTE**: Any numerical value mentioned here actually refers to κ', see section 1.1).

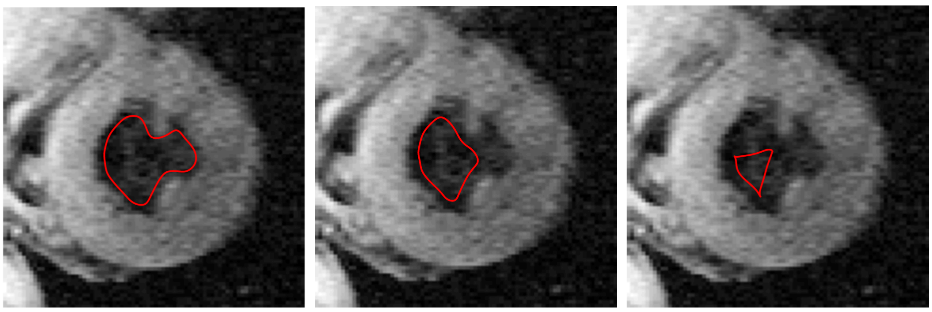
Kappa controls the proportion of the external force which is based on the image's gradient. The higher the absolute value of kappa, the stronger the influence of the external force on the shaping of the model (as opposed to the balloon force, see section 1.3). For high values of kappa, the model will stop at very weak edges. Our experiments show that the initial value of 0.2 is near optimal. When decreasing kappa to 0.18, the shape over-extends a little on the top part where the gradient of the surrounding shape is not yet strong enough. Further decreasing to 0.16 leads to a critical value where the model fails to stop at the relevant edges and outgrows the entire area. See figure 5.



(a) $\kappa = 0.16$ (b) $\kappa = 0.18$ (c) $\kappa = 0.2$

Figure 5: Heart: tweaking the values of κ. $\kappa \leq 0.2$

As figure 6 shows, increasing kappa above the initial value of 0.2 decreases segmentation quality. The external force is too strong, so small gradients that are related to noise and not the actual edge of the heart get weighted enough to stop the model from growing. When setting kappa to 1000 the shape fails entirely to segment the shape of the heart and follows noisy gradients towards a triangular shape, see figure 6 (c).

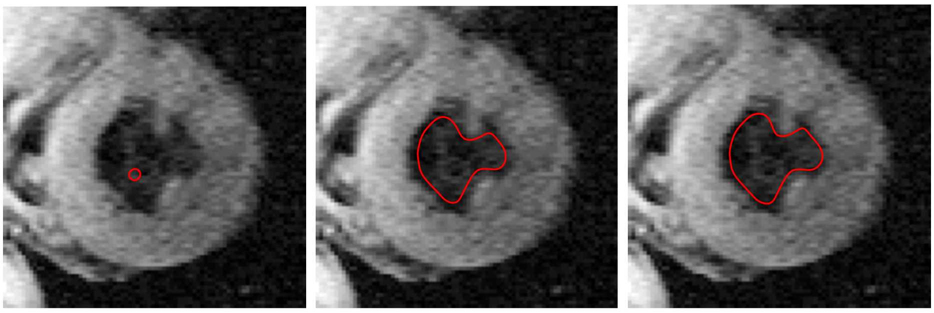


(a) $\kappa = 0.2$ (b) $\kappa = 0.4$ (c) $\kappa = 1000$

Figure 6: Heart: tweaking the values of κ. $\kappa \geq 0.2$

1.1.4 Lambda

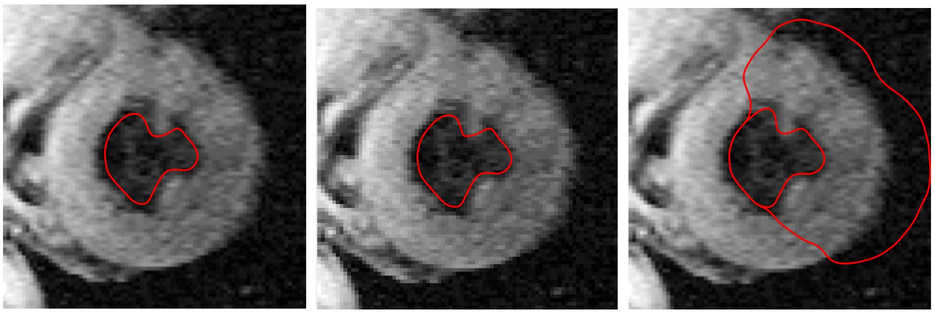
Lambda defines the weight of the balloon force, as opposed to the external force being controlled by the kappa parameter. A high value of lambda corresponds to a strong balloon force. Our experiments show that when decreasing lambda to 0.02, a critical value is reached where the model fails to grow, since the internal energy trying to keep the points close together is stronger. This can be observed in figure 7. The values of lambda are positive since our initial shape lies within the shape we want to segment and thus our model needs to expand. For negative lambdas, the model would contract instead.



(a) $\lambda = 0.02$ (b) $\lambda = 0.04$ (c) $\lambda = 0.05$

Figure 7: Heart: tweaking the values of λ. $\lambda \leq 0.05$

As figure 8 shows, even just slightly increasing lambda to 0.055 already worsens the result - the balloon force is weighted too strongly. For a value of 0.06, the model expands to much and outgrows the wanted shape. Thus, the optimal value seems to be close to 0.05.

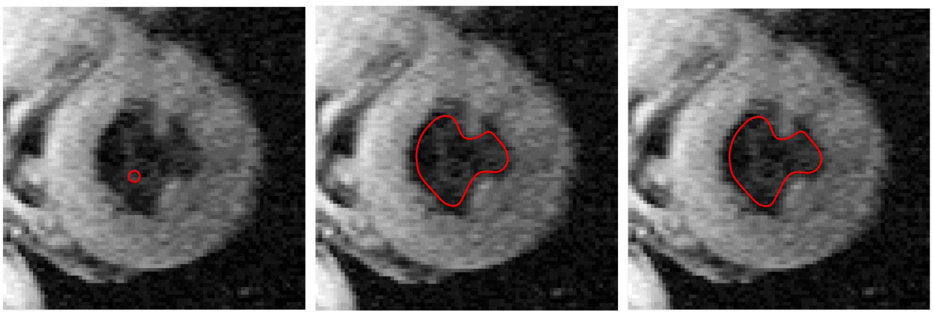


(a) $\lambda = 0.05$ (b) $\lambda = 0.055$ (c) $\lambda = 0.06$

Figure 8: Heart: tweaking the values of λ. $\lambda \geq 0.05$

1.2 Maxstep

Increasing maxstep did not lead to any changes of the segmentation's results in our experiments. However, as figure 9 shows, when choosing a very small value such as 0.001, a critical value is reached where the model fails to grow.



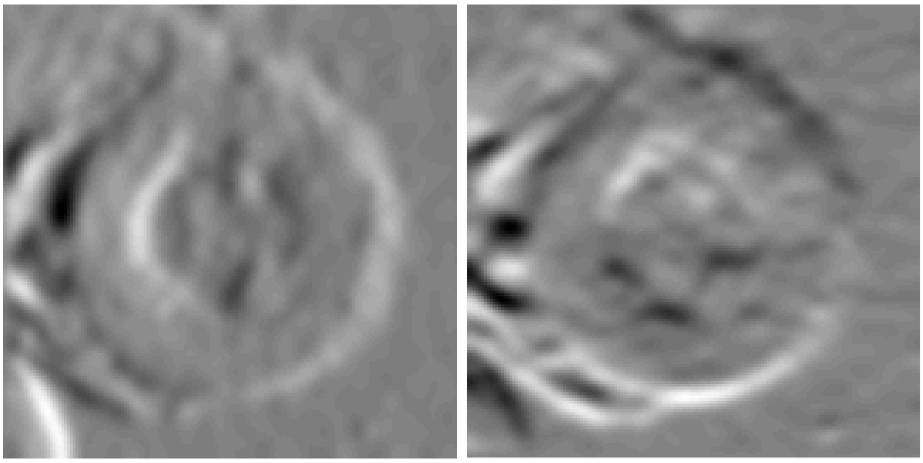
(a) $maxstep = 0.001$ (b) $maxstep = 0.01$ (c) $maxstep = 0.4$

Figure 9: Heart: tweaking the values of maxstep. $maxstep \leq 0.4$

1.3 Forces of the Segmentation

There are different forces at work that control the growing of the model. The curves are drawn toward edges by the potential force, defined as the negative gradient of a potential function, in this case the gray scale image smoothed with a Gaussian kernel. Together with the balloon force, that pushes the model to expand (for positive λ) or contract (for negative λ), the potential force contributes to the external force. At the same time, internal forces define the properties of the model: trying to keep the points of the model close together (elasticity) and avoiding bending too much (rigidity).

We can visualize the potential force field of the heart image for both the x and y dimensions in figure 10



(a) x-dimension (b) y-dimension

Figure 10: Force Field of Heart Segmentation

As said before, the external force is a combination of the potential force visualized in figure 10 and the balloon force, and the weight of each component is controlled by means of the κ and λ parameters.

2 Image Segmentation by Active Contour Models in Noisy Images

Much like in the previous section we start by finding the critical values of each parameter and seeing how big is its influence in the segmentation. Then we speak about the potential and external forces. Finally we analyze the repercussion of noise in the segmentation.

2.1 Parameter tweaking

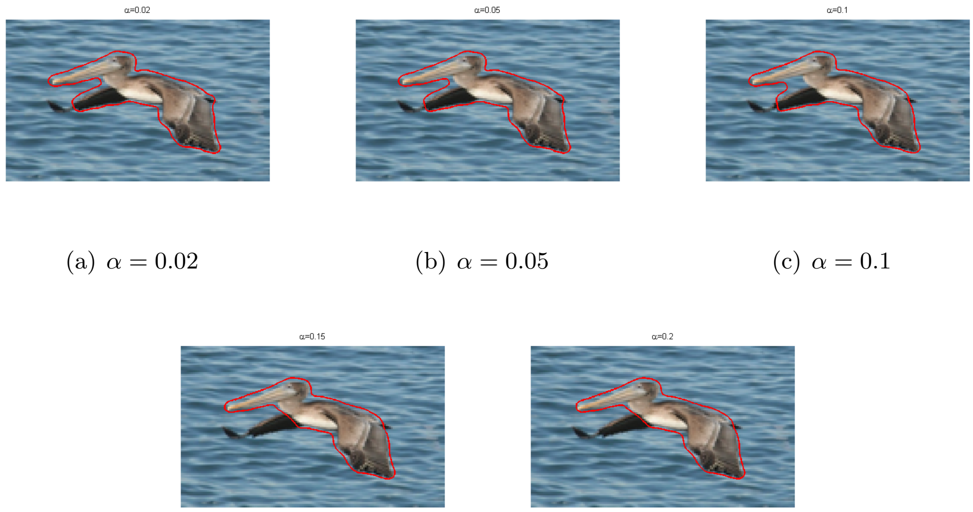
For the purpose tweaking the parameters we scale the image down to 0.25 its original dimensions in order for the algorithm to be executed in a reasonable time.

The same strategy as before is followed. This time, we vary one parameter while the others are fixed in their default values. The default values are, for this image:

$$(\alpha, \beta, \kappa', \lambda, \text{maxstep}) = (0.1, 0.1, 0.3, -0.05, 0.4)$$

One of the most important changes with respect to the previous case is that now the snake is shrinking instead of expanding (hence the negative λ value).

2.1.1 Alpha



(a) $\alpha = 0.02$ (b) $\alpha = 0.05$ (c) $\alpha = 0.1$

(d) $\alpha = 0.15$ (e) $\alpha = 0.2$

Figure 11: Bird: tweaking the value of α

In figure 11 we can observe the effect of varying the α parameter. We can see that at some point between $\alpha = 0.1$ and $\alpha = 0.15$ the right wing is left outside of the segmented region. Therefore we consider the parameter α decisive for the correct segmentation of the image, and we ascertain that its critical value is somewhat higher than 0.1. This is, of course, starting from the default parameters. Due to the tight relationship between the parameters the critical value can be altered when we modify the other parameters.

2.1.2 Beta

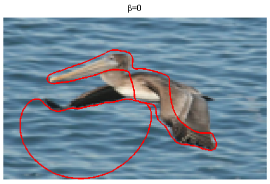
In figure 12 we can see that β is a very important parameter since it can make the segmentation fail catastrophically. It seems that when we start from the default values, the critical value is around 0.03. At this point the shape begins to twist on itself. High values of β can also yield a bad quality segmentation, producing a big gap between the edges of the bird and the snake.

2.1.3 Kappa

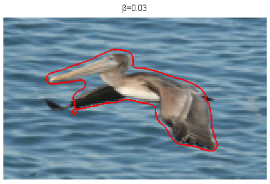
As seen in figure 13, tweaking the value of κ' is one of the possible strategies if we want the snake to fit around the right wing. The parameter is very critical. Setting it lower than 0.2 lets the algorithm very prone to crash (the snake shrinks to a singularity). On the other hand, very low values can produce that the right wing is completely missed. We need to set the value of κ' higher if we want to correctly segment the wing. However this requires to tweak the other parameters as well, since increasing the κ also increases the gap between the snake and the bird.

2.1.4 Lambda

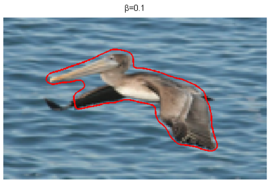
In figure 14 we can appreciate the influence of λ. We can see that its effect is very similar to that of κ, but inversed (which makes sense looking at the meaning of those parameters). A



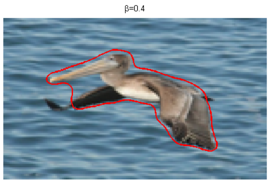
(a) $\beta = 0$



(b) $\beta = 0.03$

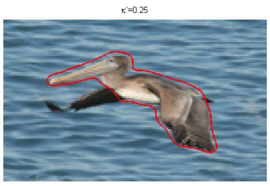


(c) $\beta = 0.1$

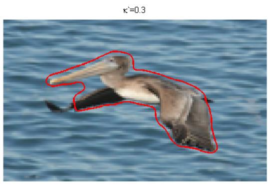


(d) $\beta = 0.4$

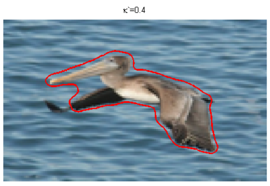
Figure 12: Bird: tweaking the value of β



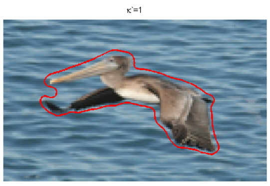
(a) $\kappa' = 0.25$



(b) $\kappa' = 0.3$



(c) $\kappa' = 0.4$



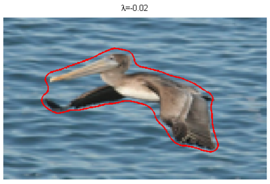
(d) $\kappa' = 1$

Figure 13: Bird: tweaking the value of κ'

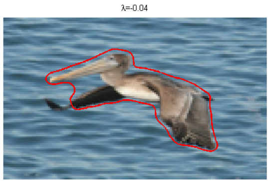
high value (in absolute module) like -0.055 can make the algorithm miss the wing completely while a low value makes the snake fit to the wing at the cost of increasing the gap. Should this parameter be altered, we would need to modify one or more of the other ones.

2.1.5 Maxsteps

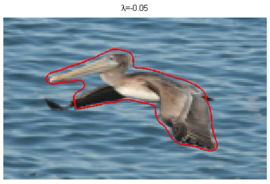
Figure 15 shows that the influence of maxsteps is not too high unless it is low enough. A value higher than 0.4 does not seem to have a big impact, while values lower than 0.01 seems to harm the quality of the result. Later we will see that this parameter can prevent the snake from crossing over, and therefore can be useful in the presence of noise.



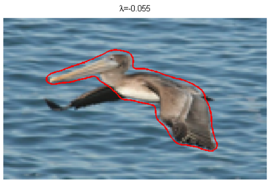
(a) $\lambda = -0.02$



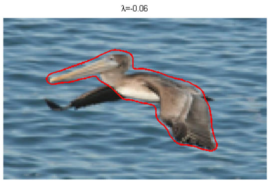
(b) $\lambda = -0.04$



(c) $\lambda = -0.05$

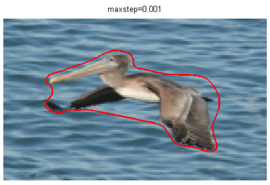


(d) $\lambda = -0.055$

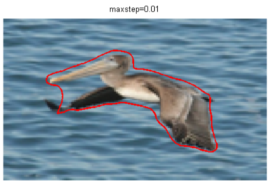


(e) $\lambda = -0.06$

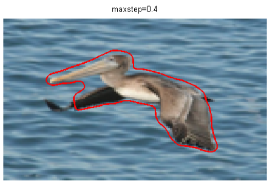
Figure 14: Bird: tweaking the value of λ



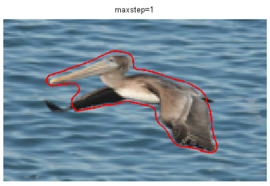
(a) maxsteps = 0.02



(b) maxsteps = 0.01



(c) maxsteps = 0.4



(d) maxsteps = 1

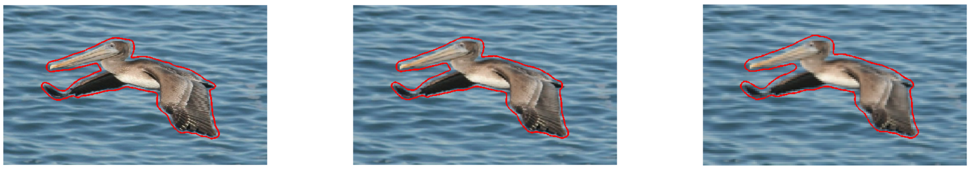
Figure 15: Bird: tweaking the value of maxsteps

2.1.6 "Optimum" values

As it can be seen in figure 11 for $\alpha = 0.1$ (or for any of the plots in which the tweaked variable takes the default value), the right wing is not correctly segmented when the image is scaled at 0.25 of its original dimensions. After taking note of the effect of each parameter and performing some manual tweaking we have come to a set of values that effectively segmentates the bird when the image is scaled to 0.25, 0.5 and 1.0 of its original dimensions. These values are as follows:

$$(\alpha, \beta, \kappa', \lambda, \text{maxstep}) = (0.1, 0.03, 0.8, -0.08, 0.4)$$

While these values are not optimum in a mathematical sense, empirically they have rendered good results. The segmented image is shown in figure 16.



(a) Scale: $\times 1$ (b) Scale: $\times 0.5$ (c) Scale: $\times 0.25$

Figure 16: Bird: segmentation with “optimum” parameters at different scales

2.2 External force

Much like in the previous case the external forces are a weighted combination of the force fields and the balloon force, being κ and λ the weights, respectively.

Also like before, the force field is obtained as the gradient of the smoothed grayscale image. This more or less the same concept than applying a Gaussian derivative filter to an image to extract its edges. The normalized force fields are shown in figure 17.

(a) x-dimension (b) y-dimension

Figure 17: Force field of bird segmentation

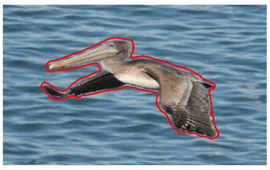
2.3 Influence of noise on the segmentation

We analyse two kinds of noise: additive white gaussian noise (AWGN) and salt & pepper noise. As we shall see, the previous selection of parameters is actually very vulnerable against noise. It is also worth mentioning that the tests were initially performed at 0.25 of the original dimensions, but operating with low resolution images renders to significantly worse results in the presence of noise. Therefore we retrieve the full size image.

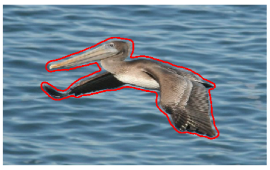
2.3.1 Additive White Gaussian Noise (AWGN)

We have observed that the algorithm is very vulnerable to this kind of noise. We start by adding very small amount of noise, and then we slowly start increasing it. At first we use our “optimum” values. Figure 18 shows how the image is segmented for 0-mean AWGN with different variances. It can be observed that the algorithm starts giving poor results when the variance of the noise is as low as 0.000025.

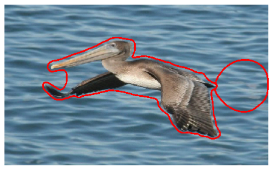
These problems can be countered up to certain extend by tweaking the parameters even further. In particular, we reduce the maxstep parameter and increase the α and β parameter



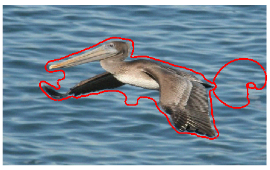
(a) AWGN with variance 0.000001



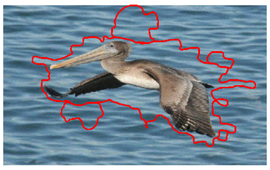
(b) AWGN with variance 0.000005



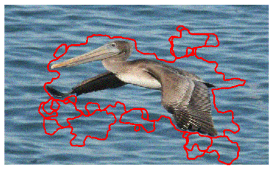
(c) AWGN with mean 0 and variance 0.000025



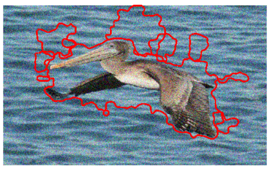
(d) AWGN with variance 0.000125



(e) AWGN with variance 0.000625



(f) AWGN with variance 0.003125



(g) AWGN with variance 0.0156

Figure 18: Segmentation with AWGN

to have a more stiff curve and avoid twists. We also reduce κ to avoid that the snake gets stuck too early in false edges. The selected values were at the end:

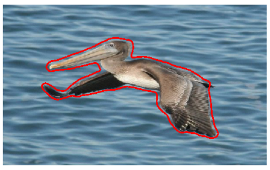
$$(\alpha, \beta, \kappa', \lambda, \text{maxstep}) = (0.3, 0.3, 0.6, -0.08, 0.15)$$

The segmentation with these parameters are shown in figure 19. Of course with big amounts of noise we obtain a suboptimal segmentation, but nonetheless better than in figure 18.

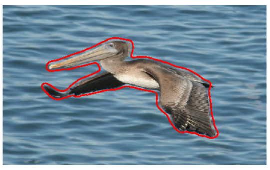
2.3.2 Salt and pepper

In this case the situation is more grim than with AWGN. Even with very low densities of salt & pepper noise the algorithm fails to converge to a solution consistently without the snake twisting over itself. The results (with our "optimal" values) are shown in figure 20.

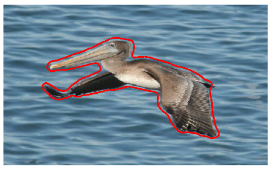
While tuning the parameters could lead to segment the image correctly under some circumstances, the results were not consistent enough to be of any utility. It turns out that salt & pepper noise is even more pervasive than AWGN for the snakes algorithm. However, if we suspect that there will be salt & pepper noise in the image we can deal with it easily with a median filter (in general, the median filter gives very good results in the presence of impulsive noise like in our case). To produce the result of figure 21 we have applied a



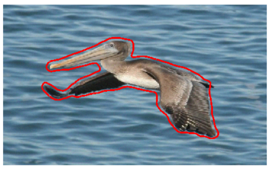
(a) AWGN with variance 0.000001



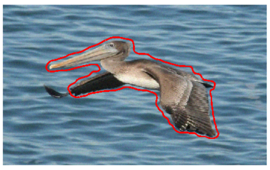
(b) AWGN with variance 0.000005



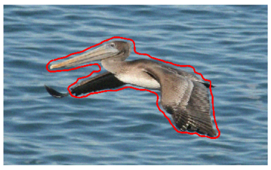
(c) AWGN with mean 0 and variance 0.000025



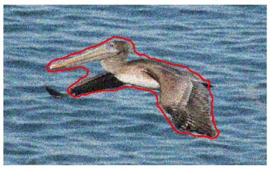
(d) AWGN with variance 0.000125



(e) AWGN with variance 0.000625



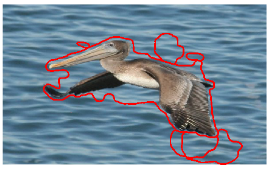
(f) AWGN with variance 0.003125

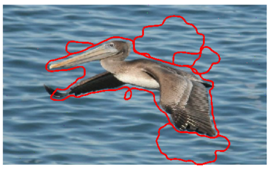


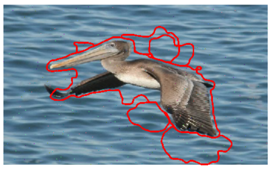
(g) AWGN with variance 0.0156

Figure 19: Segmentation with AWGN: countermeasures

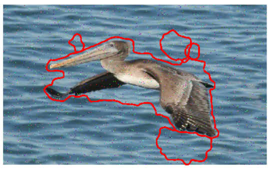
median filter with a 5×5 neighbourhood to a noisy image and then we have executed the snakes algorithm on the filtered image (with the same parameters we used for AWGN). Although the algorithm has missed one of the wings (probably we could get it back with some more fine tuning) the end result is clearly more satisfactory than in figure 20.

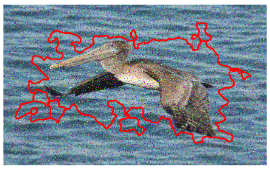






(a) Salt & pepper with density 0.00001 (b) Salt & pepper with density 0.0001 (c) Salt & pepper with density 0.001





(d) Salt & pepper with density 0.01 (e) Salt & pepper with density 0.1

Figure 20: Salt & pepper noise

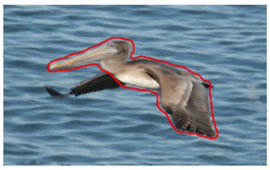


Figure 21: Segmentation with salt & pepper noise with density 0.1 (10% of the image) after applying median filter and snakes algorithm

3 Exploring the segmentation by level sets

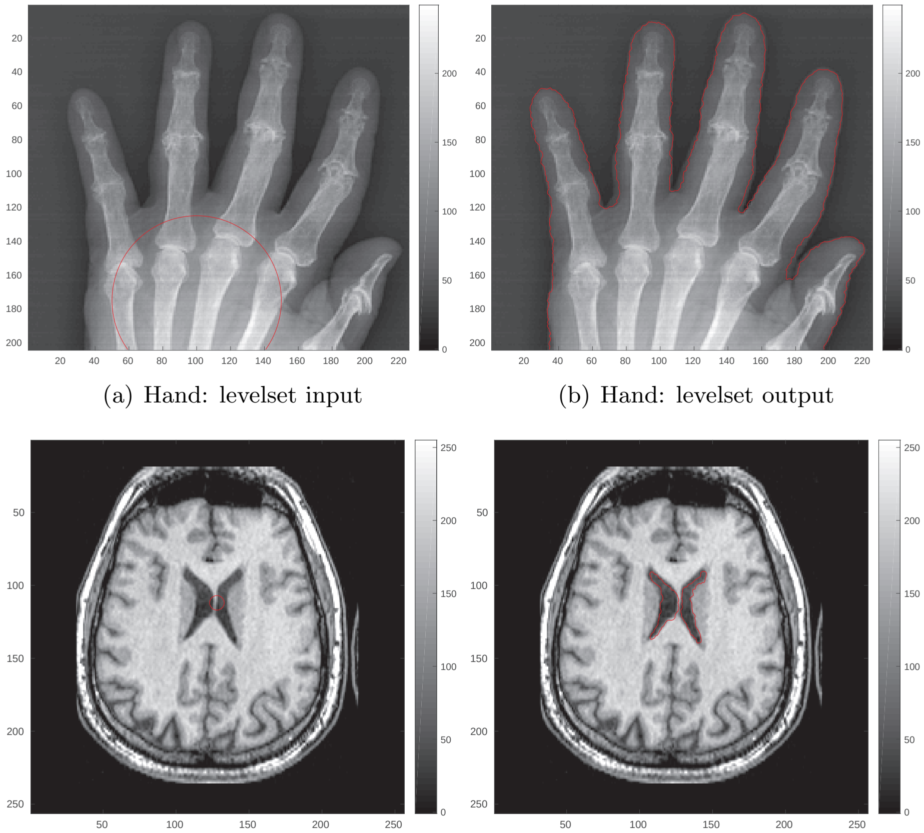
In both examples (hand and brain) of the `levelset_demo.m` the initial level set model is just a circumference that evolves depending on the gradient of the image and that finally adjusts to the desired anatomical shape: in the first case, the hand itself; in the second, the two small ventricular shapes in the center of the brain. In the second case the active contour is even able to modify its topology dividing into two contours. Although the source code for this example is already provided to us, we show, for the sake of illustration, the initial level set and the final contour for each of the two demonstrations in figure 22.

The provided `levelset` method is an implementation of the level set segmentation algorithm by Chan & Vese[1]. This algorithm seeks to minimize the following energy functional:

$$\min_{c_{in}, c_{out}, \Gamma} E(c_{in}, c_{out}, \Gamma$$

where $E(c_{in}, c_{out}, \Gamma) = \mu \cdot \mathrm{Length}(\Gamma) + \nu \cdot \mathrm{Area}(\omega) + \lambda_{in} E_{in}(c_{in}, \Gamma) + \lambda_{out} E_{out}(c_{out}, \Gamma)$ and

[1] **T.F. Chan, L.A. Vese** *An active contour model without edges*, Lecture Notes in Computer Science, vol. 1682, pp. 141-151, 1999



(a) Hand: levelset input (b) Hand: levelset output

(c) Brain: levelset input (d) Brain: levelset output

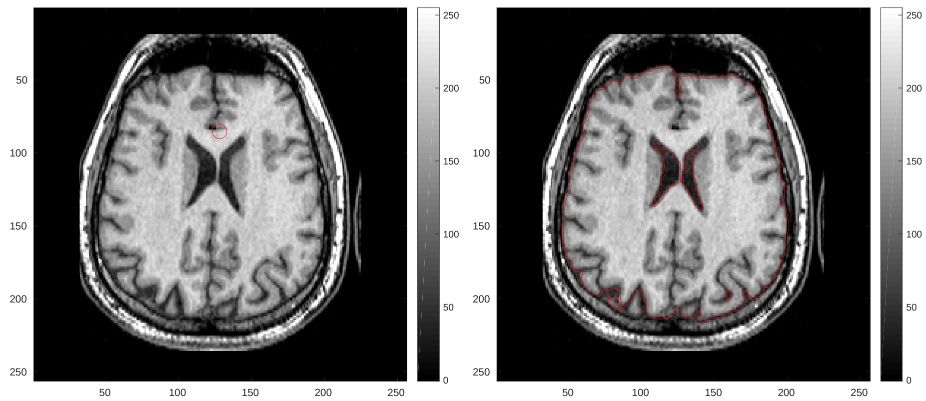
Figure 22: Initial and final contour for the first level set demo

energy functional (NOTE: ω is the region inside the contour). In this expression, μ, ν, λ_{in} and λ_{out} are constant and govern the weight of the contour perimeter, the contour area and the internal and external energies in the overall functional. For instance, increasing μ and reducing ν can lead to contours with smaller perimeters and bigger areas or, in other words, closer to a circumference. On the other hand the internal and external energies model, respectively, how different the outer gray level is from the current c_{out} and the inner gray level from the current c_{in}. Therefore λ_{in} and λ_{out} have to be tuned bearing this in mind. An initial contour and average intensities have to be given to the algorithm as well (the intensities can have an arbitrary value or can be initialized with the mean pixel intensity inside the contour boundary), so these appears as arguments of the `levelset` method as well. In the method the initial contour is inferred from the given *level set function*, or ϕ, which measures the distance from each of the image pixels to the contour (positive outside the contour, negative inside).

Moreover the provided method takes the following additional parameters:

- κ: penalizes the difference between the current ϕ and the distance function.
- τ: controls the displacement of the contour at each step. Can be increased for increased convergence speed or can be reduced if the iterations become erratic and or unstable.
- g: modulates the weight of each pixel in the distance metric. By default is a full-one matrix.

The main difference between the segmentation of the brain in `levelset_demo.m` and in `levelset_demo2.m` is the initial location of the active contour. In the first example the contour is located in the center of the brain and therefore adjusts to the shape of the two ventricular shapes that are located in the middle. However in `levelset_demo2.m` the contour is located outside and therefore it adjusts to the region with biggest area of the brain. Not surprisingly, the c_{in} and c_{out} are swapped accordingly while the rest are let as they are. We can see this in figure 22.



(a) Brain: levelset input

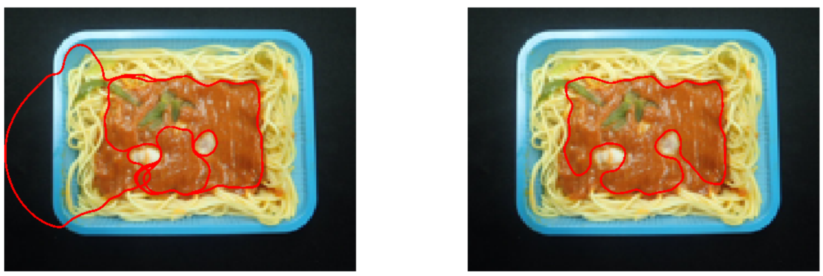
(b) Brain: levelset output

Figure 23: Initial and final contour for the second level set demo

4 Application of the Segmentation by Level Sets for Food Analysis

We applied both snakes and level sets to segment the ketchup area of the dish.

We employed snakes with positive balloon force (i.e. inflating the curve from inside of the ketchup area). The first major drawback of snakes became obvious in the very first execution of the algorithm with the recommended parameters: the contour cannot perform topological changes. Therefore it cannot surround the small ingredients ("islands", so to speak) in the middle of the dish without twisting over itself. This is evidenced in figure 24(a). The best we could do to avoid this is to reduce the λ value from 0.05 to 0.04. The result is in figure 24(b). Now the contour does not twist over itself, but the ketchup area is not completely segmented (still pretty good result taking into account that the contour cannot divide).

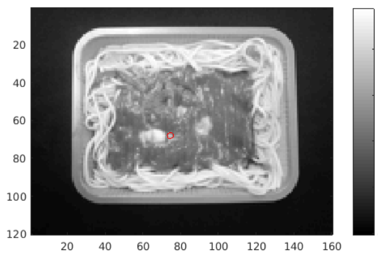


(a) Recommended parameters (with lambda positive)

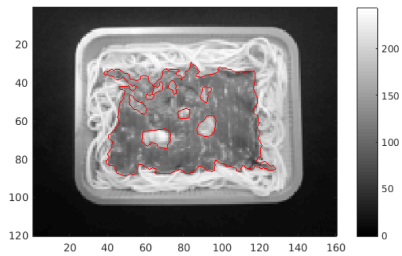
(b) $\lambda = 0.04$

Figure 24: Segmentation of the dish with snakes

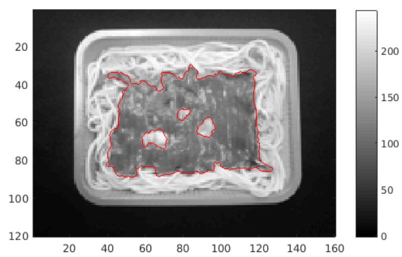
Without any doubt the best segmentations are obtained with level sets. In order to make it work correctly we need to set the initial c_{out} and c_{int} parameters correctly. Otherwise we would run into the issue presented in figure 25(a). Here, these parameters has been chosen poorly deliberately (high intensity value for c_{in} and low intensity value for c_{out}), and therefore the contour could not grow appropriately to fit the desired area. In our case the role of μ and ν can be appreciated in a very interesting way: for high μ and low ν, high perimeter is penalized more over high area, and therefore the segmentation is more uniform and the contour avoids forming holes as in figure 25(c), which is correctly segmented; on the other hand, high values of ν and low values of μ lead to the opposite effect: area is penalized over perimeter and therefore the contour is prone to form holes, as in figure 25(b). The rest of the parameters has been chosen similarly to those of `levelsets_demo2.m`.



(a) Wrong c_{in}, c_{out}



(b) High ν, low μ



(c) High μ, low ν. Best segmentation.

Figure 25: Segmentation of the dish with level sets

Appendices

A Annex

Here we list all the delivered script and function files delivered for this practice.

We include relevant observations. For full insight, we refer the reader to the source code.

- `ex1.m`: script employed to generate the plots for the first exercise
- `ex2.m`: script employed to generate the plots for the second exercise. Different sections of the script have been modified or commented in/out in-place to perform the exercise.
- `ex4.m`: in this script we read one of the dish images and apply both snakes and level sets to segment the ketchup area.