SORBONNE UNIVERSITÈ
Computer Science

Advanced Image Processing - TADI

Practical work on: Deformable Models

Students:

Maria Cristina Centorame
Andreea Elena Bodea

Academic year 2023/2024

1 Parameters

Interpretation of each parameter, its role and its influence on the segmentation result when varying its value

CODE SECTION 3a - Segmentation using active contours (snakes) - closed contours

- Step 1: Parametric Curve Initialization
  Firstly, the parametric curve is initialized using the following arrays: s, r, and c. The curve represents the initial guess for the contour of the object that needs to be segmented in the image and the arrays describe the coordinates of points along the curve. s is an array of angles that spans from 0 to 2π, creating a set of points distributed evenly in a circular pattern. r and c are arrays representing the radial and angular coordinates of the points on the curve. They are defined in such a way that the curve forms a circular shape with a center at $(140, 130)$ and a radius of 15.

- Step 2: Active Contour Segmentation
  After the initialization, the actual segmentation is done by applying the active contour (snake) algorithm to the input image. The function *active_contour*() from the skimage.segmentation is used and has the following parameters:

  - $gaussian(im, 0.1)$: The input image is first filtered using a Gaussian filter with a standard deviation of 0.1. This filtering aims to reduce noise and make the image smoother. If the value of the standard deviation increases then the resulting image will appear more blurred, because fine details and noise in the image are reduced. While noise reduction is a benefit, a standard deviation which is too may lead to a loss of fine details in the image. If the goal is to detect fine structures or edges in the image, overly aggressive smoothing can make it more difficult to segment features accurately.
  - *init*: The initial parametric curve defined in the previous step is used as the initial contour.
  - *alpha*: This parameter controls the elasticity of the curve, it represents the snake length. Increasing its value makes the curve stiffer resulting in the snake contracting faster.
  - *beta*: This value is the snake smoothness shape parameter and governs how much the curve resists changes in curvature. Increasing the value makes the curve smoother
  - *w_edge*: The weight of the edge adjusts the influence of the image gradient by controlling attraction to edges. It helps the curve to align with the edges in the image.
  - *gamma*: The last parameter determines the step size for moving the contour. Decreasing the values results in smaller steps and finer contour adjustments.

- Step 3: Visualization
  The result of the active contour segmentation is plotted as the original input image with both the initial contour (red dashed line) and the final segmented contour (blue solid line) drawn on top of it.

  The active contour algorithm iteratively refines the initial curve to align with object boundaries in the image. It does this by minimizing an energy function that combines the effects of internal forces (defined by *alpha* and *beta*) and external forces (defined by the image gradient and *w_edge*). The final contour represents the estimated boundary of the object in the image. (see Figure 1)

CODE SECTION 3b - Segmentation using active contours (snakes) - open contours

- Step 1: Parametric Curve Initialization
  Similarly to the first step of the segmentation using closed active contours, for the segmentation using open active contours the parametric curve also needs to be initialized. This time, the

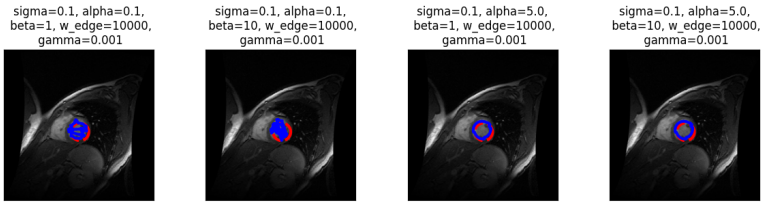


Figure 1: Active contours segmentation applied to "coeurIRM.bmp" image with different values of parameters

curve is only defined by the two arrays r and c, which describe the coordinates of points along the curve. They create a set of points forming a straight line from (20, 20) to (100, 100). Segmentation using an open loop, allows the detection of objects with open or non-enclosed boundaries.

- Step 2: Active Contour Segmentation
  For the actual segmentation step, the active contour (snake) algorithm is applied to the image using the *active_contour* function, in a very similar way to the previous section except for two differences. First, the values chosen for the parameters *alpha*, *beta*, *gamma*, *w_edge* and the gaussian function differ slightly. Second, a new parameter appears in the function, namely *bc*, which is specified as 'fixed', because the endpoints of the curve are anchored.

- Step 3: Visualization
  The visualization of the results is almost identical to the one for the closed active segmentation.

In this case, the code is segmenting an open contour, which means that the contour does not need to form a closed loop. It is particularly useful when segmenting objects that are not enclosed, like a line or a vessel in a medical image. The parameters can be adjusted to achieve the desired segmentation results based on the characteristics of the chosen image.
(see Figure 2)

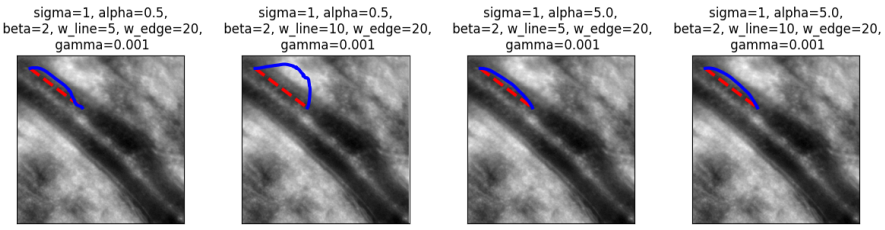


Figure 2: Active contours segmentation applied to "retineOA.bmp" image with different values of parameters

SECTION 4 - Segmentation using level sets (and region homogeneity)

- Step 1: Image Conversion
  Firstly, the input image needs to be converted to a floating-point representation to ensure that pixel values are in the range $[0, 1]$.

- Step 2: Initialization of Level Set Function
  Then, there are three initialization methods to choose from in order to define the initial regions of interest for segmentation. One option is to use a checkboard pattern using the function *checkerboard_level_set*, which takes the shape of the image and a parameter that determines the frequency of the checkerboard squares. Higher values of the parameter result in a finer pattern. This initialization method can be useful for certain segmentation tasks, such as segmenting objects with a clear alternating pattern or objects with specific texture characteristics. The second possibility is to initialize the level set function with a single circular region defined by

its center and radius using the *disk_level_set* function. This method is suitable for segmenting objects that can be approximated by a single circular shape, such as rounded objects or blobs. The last option is using multiple circles by calling the previous function multiple times. A loop is used to create and add multiple circular regions to the initial level set function. The number of circular regions is determined by the *circleNum* variable. The position and size of each circle are computed within the loop based on the dimensions of the image and the specified parameters (e.g., *circleRadius*, *circleStep*0, and *circleStep*1). Then, the *disk_level_set* function generates a level set function for a single disk with a specified center and radius. By adding up multiple circular regions, this initialization method is useful for segmenting objects composed of several circular or blob-like regions.

- Step 3: Chan-Vese Segmentation:
  The Chan-Vese segmentation algorithm is applied to the input image using the *chan_vese* function. The function requires the following parameters:
  - *image*: The floating-point representation of the input image.
  - *mu*, *lambda*1, *lambda*2: Parameters that control the behavior of the Chan-Vese segmentation. They influence the balance between the inside and outside regions of the level set. Typical values for *lambda*1 and *lambda*2 are 1. If the background is very different from the segmented object in terms of distribution (for example, a uniform black image with figures of varying intensity), then these values should be different from each other. Typical values for *mu* are between 0 and 1, though higher values can be used when dealing with shapes with very ill-defined contours.
  - *tol*: The value influencing the level set variation tolerance between iterations until convergence.
  - *max_iter*: The maximum number of iterations.
  - *dt*: This multiplication factor represents the time step for each iteration and serves in the acceleration of the algorithm. While higher values may speed up the algorithm, they may also lead to convergence problems.
  - *init_level_set*: The initial level set function created in the previous step.
  - *extended_output* = *True*: This flag indicates that the function should return additional information about the segmentation process, such as the evolution of energy over iterations.
- Step 3: Visualization
  After segmentation, the code creates a visualization to display the original image, the segmented result, the final level set function, and the evolution of energy over iterations.

Performing segmentation using level sets and initializing the level set function with multiple circular regions can be useful for segmenting objects with various regions of interest in the image. (see Figure 3)

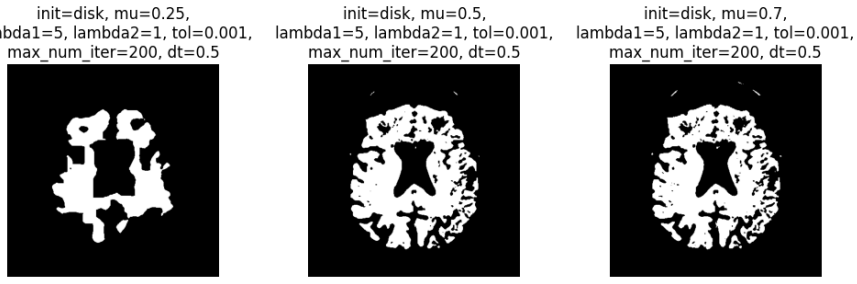


Figure 3: Level-set segmentation applied to "brain.bmp" image with different values of parameters

2 Segmentation

- Image 1:
  In this section, our primary objective has been to address the challenge of achieving a robust segmentation of the entire brain within the "brain2.bmp" image (see Figure 4).

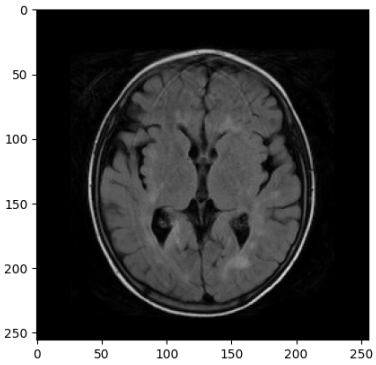


Figure 4: Original image of "brain2.bmp"

  – Segmentation method chosen: active contours. The choice of the segmentation method has been deliberated, with a preference for the active contours method due to its suitability for delineating simpler shapes, which accurately describes the characteristics of our brain boundary.

  – Results: The process consisted in adjusting the parameters of our method based on their impact to incrementally improve the segmentation results. Here follow some basic steps to explain what has been our way of reasoning.
  To begin, the snake has been initialized as a circular shape with its center positioned at the midpoint of the 2D image and its radius determined as a fraction of half the height of the image. This configuration has guaranteed the initial contour to lie outside the boundaries of the target object to be segmented.
  Random values have been set for the requested parameters ($def\_sigma, def\_alpha, def\_beta, def\_wedge$, ([0.1], [0.1], [0.5], [50], [0.001]). The continuous boundary of our object suggested the need of smoothing the image before proceeding to its analysis with the purpose of eliminating any insignificant, unwanted noise. Thus a higher value of *sigma* has been tested and the result shown in Figure 5 has confirmed our hypothesis.

Figure 5: Evolution of the active contour with $sigma = 0.1$(left) and $sigma = 1$(right)

  At this point, the increasing of *alpha* has imparted greater rigidity to the curve preventing it from self-folding due to its direct impact on the snake's length. (see Figure 6) In conclusion, the *beta* value has been increased to attain a smoother contour (refer to Figure 7) and, given the relatively smooth and continuous boundary of the brain, the *w_edge* has strategically been set to a smaller value to diminish contour deformations through a less

Figure 6: Evolution of the active contour with $alpha = 0.1$(left) and $alpha = 1$(right)

susceptibility to minor fluctuations in pixel values. Consequently, the impact of internal forces has been intensified, primarily regulated by the values of *alpha* and *beta*.(Figure 8)

Figure 7: Evolution of the active contour with $beta = 0.5$(left) and $beta = 5$(right)

Figure 8: Evolution of the active contour with $w\_edge = 20$(left) and $w\_edge = 50$(right)

– Improvements: The final result is shown in Figure 9.
While the current segmentation process has yielded impressive results, there is always room for refinement and optimization. One avenue for improvement lies in the fine-tuning of the algorithm's parameters, exploring a wider range of values to identify the most optimal combination for specific images. Furthermore, the integration of advanced methods, such as those based on deep learning, and the incorporation of more sophisticated pre-processing procedures have the potential to further elevate the precision of the segmentation.

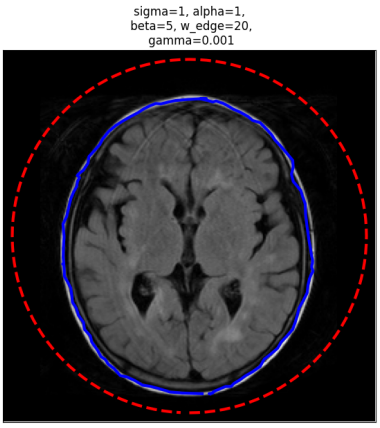
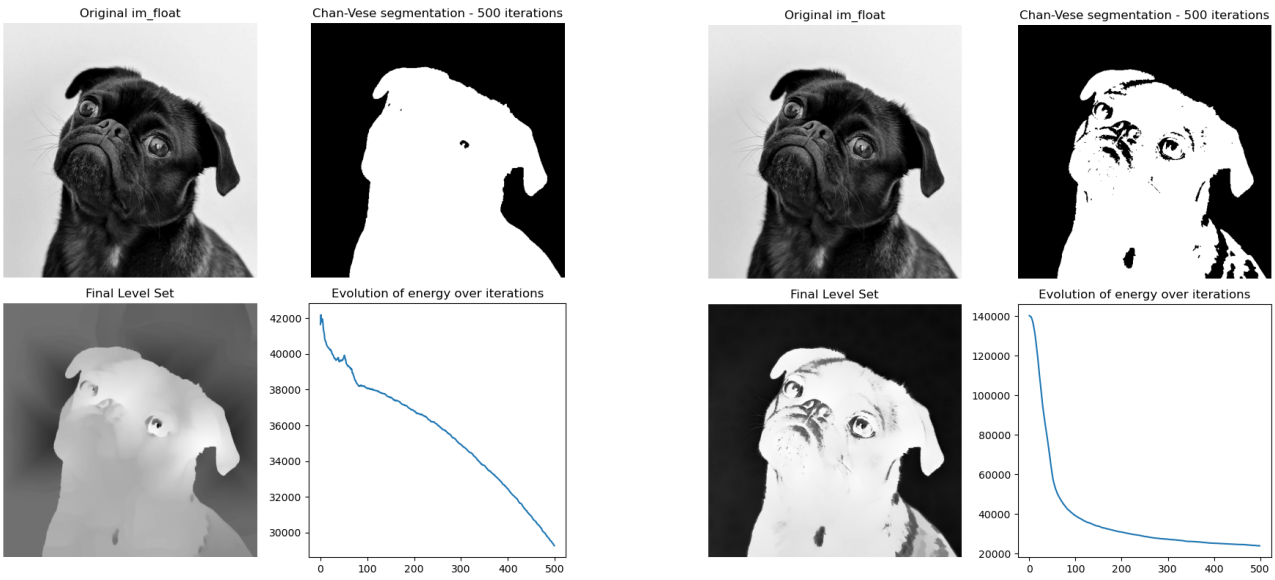


Figure 9: Final result for the segmentation of a brain in the "brain2.bmp" image

- Image 2: For our second experiments we choose the image shown in Figure 10a and displaying a dog, because it had both a distinguishable contour, but also enough details.
  - Segmentation method chosen: For this image the segmentation method we selected was level sets. The Chan-Vese Algorithm is designed to segment objects without clearly defined boundaries. In the image showing the dog, although the form of the dog is quite easily detectable, the different body parts of the dog like the eyes or chin are challenging to segment.



(a) Initial segmentation

(b) Final segmentation

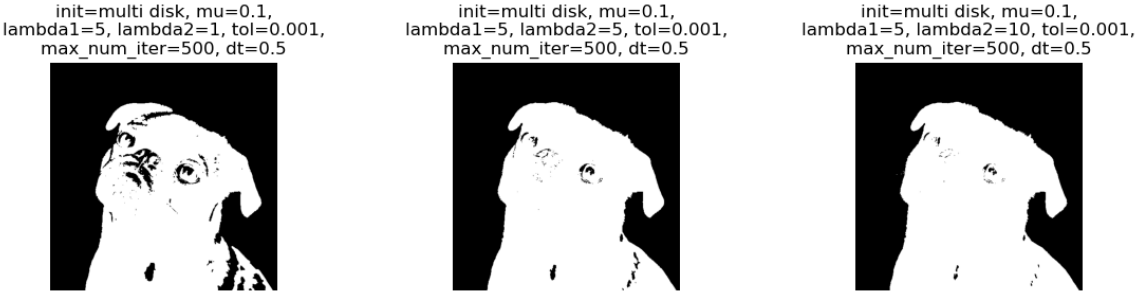
Figure 10: Segmentation using level sets

  - Results: As illustrated in Figure 10a we started our experiments from the parameter configuration found in the official documentation of the function, i.e. the following values: $mu = 0.25$, $lambda1 = 1.0$, $lambda2 = 1.0$, $tol = 0.001$, $max\_num\_iter = 500$, $dt = 0.5$. The selected method of initialization of the level set function was the third option described above, namely multiple circles as the initial region of interest. This choice was made because the object that we wanted to detect (the dog) did not have a clear alternating pattern nor a specific texture, and could neither be approximated by a a single circular region. After a substantial number of tests conducted with various parameter configurations, we found that a smaller value for mu, more exactly 0.1 and a higher value of $lambda1$, more precisely 5, result in a better segmentation (Figure 10b).

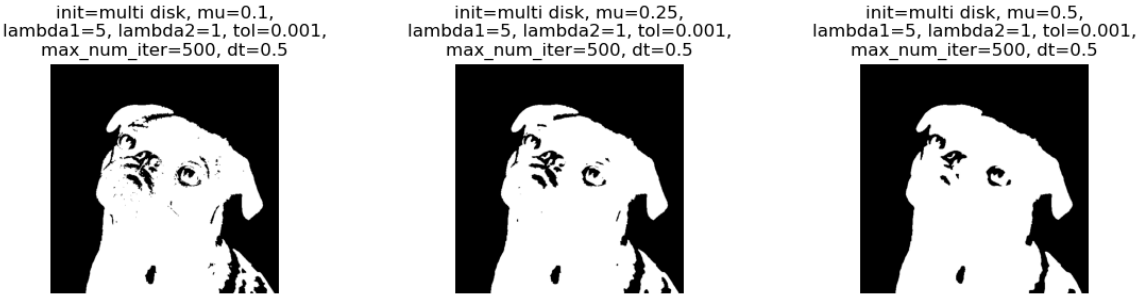
For each of the parameters required in the *chan_vese* function, we tried out different values and compared the segmentation result in order to make proper decisions. We started by trying out three values for *lambda*1: 1, 5 and 10, as shown in Figure 11a. Clearly, a higher value than 1 which is the typical reference value, helps in detecting not only the contour of the dog, but also parts of the body like the nose, eyes and mouth. This is due to the fact that the background is very different from the segmented objects in terms of distribution. However, choosing an extremely high value like 10 results in not detecting the contour of the fur of the dog anymore. The next parameter which we tried several variations of was *lambda*2. As displayed in Figure 11b, increasing its value from the original 1 is detrimental to the segmentation, as higher values result in a loss of details. With values for *lambda*1 and *lambda*2 set, we explored variations of the parameter *mu* and found out that a lower value, more exactly 0.1 instead of 0.25 enables the segmentation of more specific aspects of the dog's body and not only his contour Figure 11c. Additionally, we experimented with the parameter representing the number of iterations and concluded that a higher number results in better segmentation, which was in line with our expectations. However, the downside of the increased iterations is the time and computational efficiency. In the end, we varied *dt*, which represents the multiplication factor applied at calculations for each step. As shown in Figure 11e, although a higher *dt* may speed up the algorithm, it may also lead to convergence problems and therefore we choose to stick with the original balanced value of 0.5.

– Improvements: First, although we tried out a considerable amount of different parameters and of initial level set, a manual search for the ideal configuration is not possible. Instead, some automatic algorithms can be tested. Also, the comparison of the results was only conducted by qualitative means, but choosing some qualitative methods can result in a more objective assessment of the best segmentation. Also, prior to applying the Chan-Vese segmentation, preprocessing steps such as noise reduction (e.g., Gaussian filtering), contrast enhancement, or image thresholding to improve the quality of the input image. A cleaner input image can lead to more accurate segmentation.

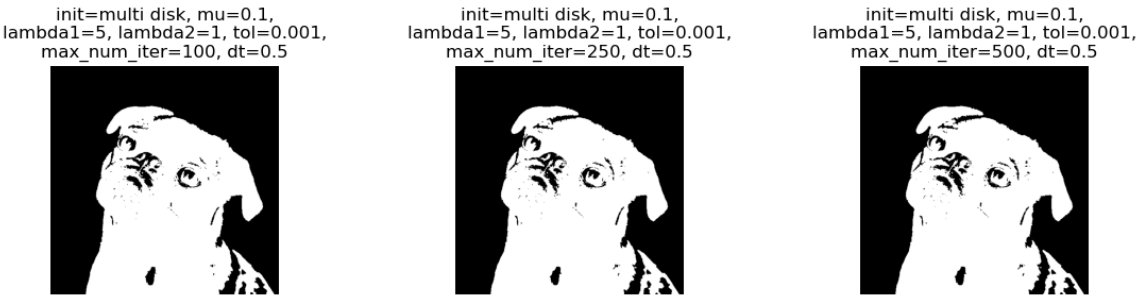
(a) Variations of parameter *lambda*1



(b) Variations of parameter *lambda*2



(c) Variations of parameter *mu*



(d) Variations of parameter *max_num_iter*

(e) Variations of parameter *dt*

Figure 11: Segmentation using level sets using different parameter configurations