COMPUTER VISION LAB ASSIGNMENT 1

February 26, 2021

Sjoerd Bruin (s2563304) & Jasper van Thuijl (s2503204)

University of Groningen

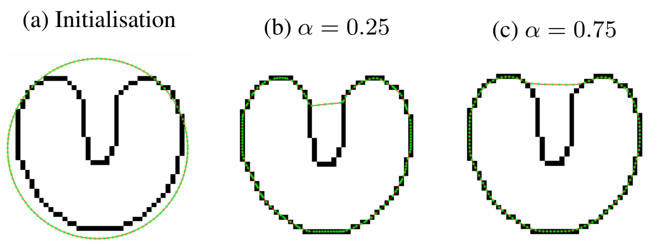
1. INTRODUCTION

In this report we aim to describe differences between GVF and classic snake frameworks that can be used to create object contours in (noisy) images. The program Snake Demo by Dejan Tomazevic was used in order to fit several images with varying parameters.

2. EXERCISE 1

First of all, we tried to fit the classic snake to the U-shape. We started off with the initial parameter settings that are used in the program ($\alpha = 0.05, \beta = 0, \gamma = 1, \kappa = 0.6, D_{min} = 0.5, D_{max} = 2$ and number of iterations 150). The alpha parameter represents the tension of the snake, representing a tension of the snake to search for a shortest path. The beta parameter represents the curvature weight, which favours sharp edges for lower values of beta and more smoothly curved contours for higher values of beta. The kappa parameter represents the strength of external forces, such as edges. A higher kappa makes it so that the contour is more attracted to edge structure compared to other contour considerations (curvature, tension). The gamma parameter defines the step size, and the Dmin and Dmax parameters define the resolution of the snake. We do not change gamma, Dmin and Dmax in the course of the the experimentation for this lab.

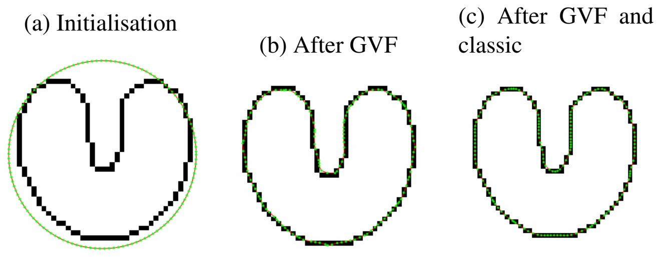
Fig. 1: Classic snake using $\alpha = 0.25$ and $\alpha = 0.75$



The result was not very convincing: the snake managed to approximate the sides of the U-shape, but the crevasse forms a problem. The snake cannot descend into the crevasse, instead choosing to connect to itself well above the bottom of this crevasse. As a result, the contour of the U-shape is not really approximated well at all. Changing the parameters did not impact this: increasing alpha made descent into the crevasse even more shallow (see figure 1). Changing kappa did not really impact this behaviour, while increasing beta seems to make the descend more shallow. The same behaviour was evident regardless of whether we initialised the snake across the shape ($r = 0.5$) or fully outside of it ($r = 0.6$). Convergence also takes quite a while: it took about 400 iterations to get to a steady shape for the snake using radius $r = 0.6$ and alpha = 0.25.

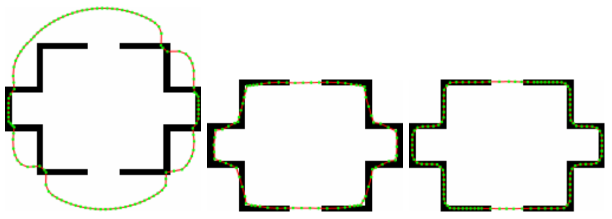
The GVF snake, on the other hand, convergely quickly, in less than 100 iterations. Additionally, it can approximate the crevasse much better, as shown in figure 2b. The result of the GVF snake has a drawback: it does not quite approximate the sharp edges going into and out of the crevasse quite as well. We can improve this by applying a classic snake, which is better with such sharp corners, after the GVF. We basically use the GVF snake as an initialisation for the classic snake, so that the classic snake does not have to find its way into the crevasse by itself but can instead focus on approximating sharp corners inside the crevasse. The result is shown in figure 2c.

Fig. 2: Combined GVF and classic snake contour on U64.pgm shape



3. EXERCISE 2

Similar as observed in Exercise 1, using the initial parameter settings seems to obtain a much faster convergence for GVF than for the classic variation, and almost seems to fit directly. However, it does have a bit of a problem with the corners, which seem too rounded (Figure 3b). The classic variation seems to need about 700 iterations in order to reach a perfect fit (Figure 3c). The corners are also much better approximated in this version. In order to increase the convergence speed for the classic snake version, we will need to change the parameters. After increasing β to 1 we only need about 200 iterations to reach a similar result. For the GVF version, we can now obtain a similar fit as before in only 20 iterations.



(a) Initial settings Classic (b) Initial settings GVF (c) Classic 700 iterations

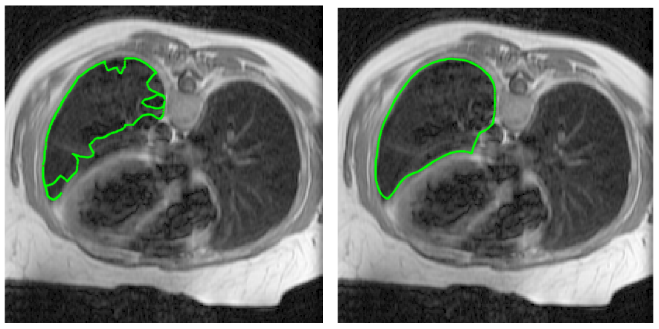
Fig. 3: GVF and classic snake contour on room.pgm shape

4. EXERCISE 3

It is quite difficult to get a good approximation of the left lung using the classic snake. It has a tendency to get stuck on local minima using the default settings, which are present in the form of edges inside the lung structure (which presumably correspond to the larger bronchi. Changing the default settings changes the contour somewhat (alpha from 0.05 to 0.25), but it doesn't find a close match in the left lung. If we increase the gradient parameter sigma from 1.0 to 2.0, we more closely approximate the border of the left lung, but the snake still gets stuck in local minima away from the actual border and produces some almost cyclical structures as a result (figure 4a). When using the GVF snake (with default settings and sigma = 2), we get a much closer approximation of the left lung, but we see a few spike-like elements in the boundary, which follow narrow edges in the left lung (and one edge going away from the lung). We can solve this by decreasing kappa from 0.6 to 0.1, such that the strength of individual edges is slightly less significant in the creation of the contour (figure 4b). This gets rid of the spikes and produces a smooth and closely-fitted contour to the left lung. We additionally had to set beta to 0.3. The convergence was slower than with default settings, so we had to increase the number of iterations to 400.

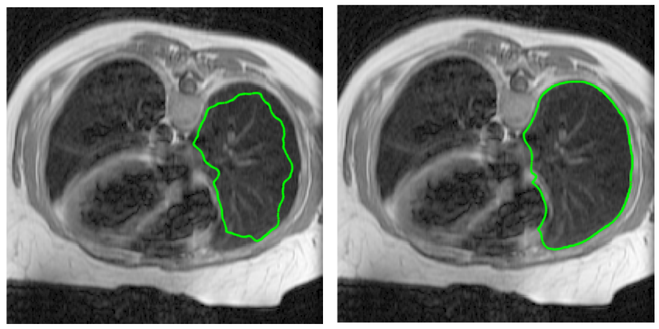
For the right lung, the classic snake has issues as well (see figure 5a). The bronchi are more pronounced in the right lung, so the snake has even more edges that could pull it away from the actual contour of the lung. We get a quite decent result for the GVF snake when we set sigma = 2 (figure 5b), but we have some problems with the regions between the lung and the heart. The GVF snake crosses this region and stops only when it reaches the boundary of the heart. We have not been able to prevent this by using different settings. Ultimately, the GVF snake produces better results, but it has its flaws. The resulting contours can be found in figure 5.

Fig. 4: GVF and classic snake contour on left lung of chest.pgm image



(a) Classic snake (b) GVF snake

Fig. 5: GVF and classic snake contour on right lung of chest.pgm image



(a) Classic snake (b) GVF snake

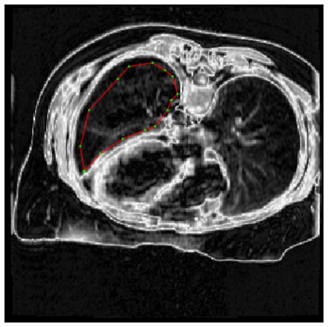
5. EXERCISE 4

Even with the improved contrast, it proves to be quite difficult to create a perfect alignment of the snake with the left and right lungs using the classic method. Due to misalignment of the initial positions found in `myheart.mat` (left lung) and `myheart2.mat` (right lung) with the `new.pgm` image, we needed to create new initial positions. The new initial position for the right lung can be observed in Figure 6a. Using the initial settings the approximation does not converge to the lung edge. Increasing the number of iterations to 400 does not improve this problem, meaning that the contour is stuck in local minima. In order to smoothen out the wrinkles, β was increased to 4, since increasing this parameter penalizes oscillations in the contour. The resulting contour is a lot smoother, however it still does not seem to align with all of the borders. Increasing β also seems to have the side effect of needing less iterations to reach the same stable condition. In an attempt to improve the fit, we tried to reduce α to 0.01, and also tried to reduce κ to 0.4, however it is still not possible to obtain a good fit.

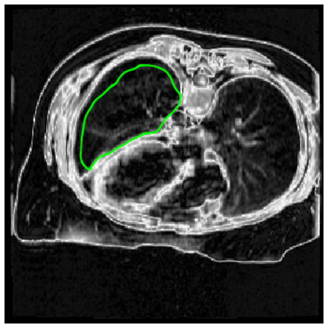
Similar as observed before, it seems that GVF has much better convergence to the lung edges than the classic method. The GVF method seems to obtain a good alignment using the newly specified initial position for the right lung, as can be observed from Figure 6c.

For the left lung, we also created a new initial position for the snake (Figure 7a). As with the right lung, the left lung also seems to be complicated to align. For the inital settings, the classic framework seems to not expand enough in order to align with the edges of the left lung (Figure 7b). The number of iterations was increased to 400, but the contour seems to reach a local minimum. In order to reduce the number of oscilations in the contour, we increased β to 10.

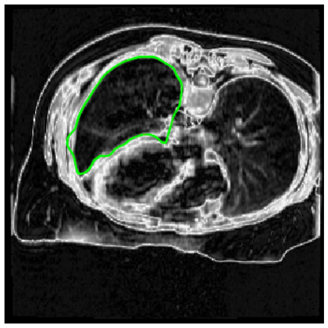
In comparison, the the GVF method seems to expand quite well. So well in fact, that it crosses some edges. We therefore reduced the number of iterations to 9, so that the iterations stop as soon as we reach the edges. We also increased β to 1, in order to favour a more smooth contour. The result can be observed in Figure 7c.



(a) Initial position



(b) Classic

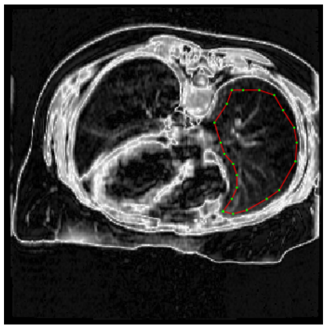


(c) GVF

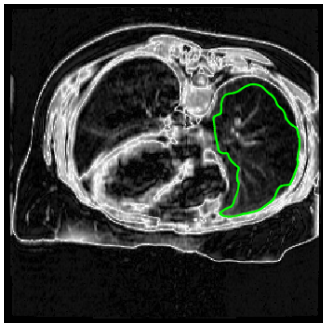
Fig. 6: Classic snake and GVF contour for the right lung in the new.pgm image

6. EXERCISE 5

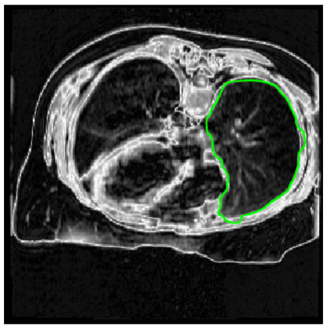
After loading `heart.mat` for the initial positions, we tried running the classic method using the default parameter settings for sigma = 0 (Figure 8b). The result contains a lot of oscillations in the approximation. Running the experiment again for sigma = 2 and sigma = 6, we observe a much



(a) Initial position



(b) Classic.
$\beta = 10$, 400 iterations



(c) GVF,
$\beta = 1$, 9 iterations

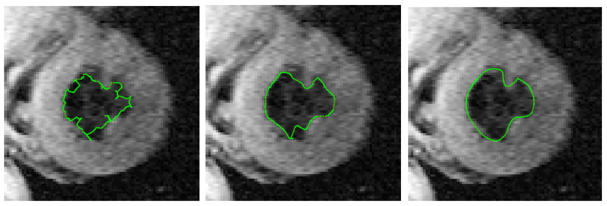
Fig. 7: Classic snake and GVF contour for the left lung in the new.pgm image

smoother approximations found in Figure 8b and Figure 8c. However, the last approximation does seem to cross a small part of the heart. The GVF method for `heart.mat` using the initial settings and sigma = 0 results in a good approximation, which only contains some oscillations. Repeating the experiment for sigma = 2 shows a better approximation. sigma = 6 seems to much as, the contour again crosses a part of the heart. Figures can be observed in Figure 8 d, e and f.

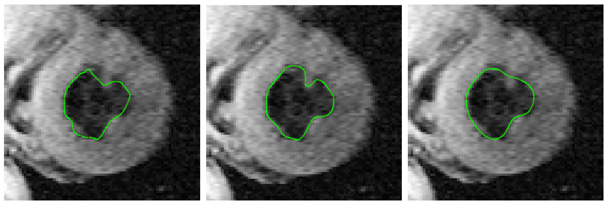
Repeating the experiments for the classical method on `heart1.mat` shows a similar approximation for sigma = 0. For sigma = 2 the result seems to be containing a lot of oscillations. For sigma = 6 we achieve a good approximation, similar to the previous approximations. Figures can be observed in Figure 9 a, b and c. The initial settings seem to perform quite well for the GVF method for sigma = 0 (Figure 9d). For sigma = 2 a similar result is obtained (Figure 9e). sigma = 6 again seems too high as a part of the heart is crossed again (Figure 9f).

7. EXERCISE 6

We have seen that the classic snake has issues when it is initialised partially inside and partially outside the shape we want it to approximate. What typically happens is that it shrinks further towards a local shape inside of the full shape,

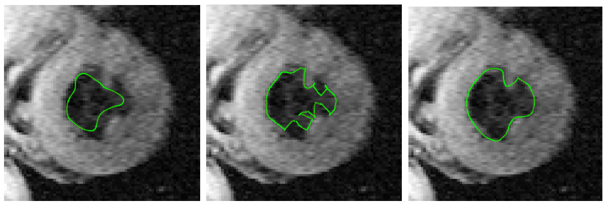


(a) sigma = 0 (b) sigma = 2 (c) sigma = 6

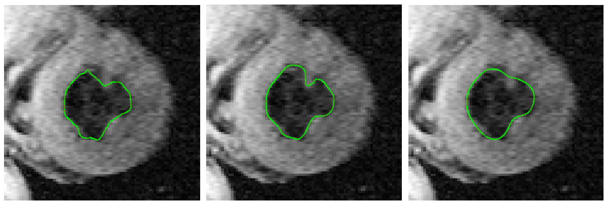


(d) sigma = 0 (e) sigma = 2 (f) sigma = 6

Fig. 8: Classic(a,b,c) and GVF (d,e,f) snake contour for heart.pgm using heart.mat



(a) sigma = 0 (b) sigma = 2 (c) sigma = 6



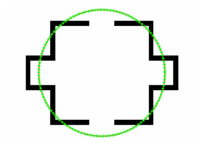
(d) sigma = 0 (e) sigma = 2 (f) sigma = 6

Fig. 9: Classic(a,b,c) and GVF (d,e,f) snake contour for heart.pgm using heart1.mat

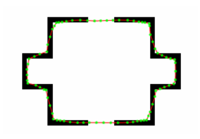
leaving out protruding features of the shape. This is undesirable: we want to approximate the entire shape regardless of whether our initial contour crosses the boundary of the shape. The GVF snake does not suffer from this problem: it can find the full shape even if its initial contour crosses the boundary. This makes the GVF snake a better snake for finding a coarse approximation of the boundary.

A disadvantage of the GVF snake is that it does not approximate corners quite as accurately as the classic snake can. As a result, if we try to approximate the boundary of the shape in 'Room.pgm', the GVF snake deviates a bit from the corners, opting instead for a smooth curve between the two sides and leaving a bit of space between the snake and the corner point.

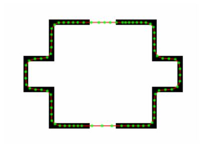
The above discussion presents a possible combination of the two methods. First, we use a GVF snake to approximate the shape without any of the pitfalls that are associated with the initialisation of the classic snake. This produces a fit that is close to the intended boundary contour, but showcases imperfect approximations in the vicinity of corners. Because the snake's position is now close to the boundary, we effectively have a very good initialisation for the classic snake to take over. The classic snake can now approximate the boundary contour easily (thanks to the initialisation that the GVF snake produced) and furthermore can produce a better approximation of the corners. The results of this are shown in figure 10. For the GVF snake step, we used mu = 0.1, numiter = 150, alpha = 0.05, beta = 0, radius = 0.4, and for the rest we maintain the default values. The classic snake step after that did not change any of these settings.



(a) Initialisation



(b) After GVF



(c) After GVF and classic

Fig. 10: Combined GVF and classic snake contour on room.pgm shape

It becomes clear when looking at figures 10b and 10c that the combination of the two algorithms produces a better result: figure 10c more closely approximates the corners of the shape than figure 10b.

8. CONCLUSIONS

1. It seems that GVF offers a much faster convergence compared to the classical variant. However, GVF does perform worse in corner approximation than the classical variant.

2. Increasing α leads to contour that stick closer to a tight contour. This means that it has more trouble with crevasses, but also that it is less likely to tunnel over an intermediate area between two disjoint regions.

3. Increasing β from the initial positions leads to faster convergence, and favors more smooth edges.

4. Combining the GVF and subsequent classic snakes seems to offer high convergence velocity, while also leading to a better corner approximation.

5. As can be observed from Exercise 5 and also from our other experiments, the initial snake position has a large effect in the end result. This is especially noticeable in the classic method for low sigma values. GVF seems to be less impacted by this.