COMPUTER LAB1: GRADIENT VECTOR FLOW (GVF) SNAKES

Ethelbert Uzodinma (S3886026)
Viljami Linna (S4169921)

ABSTRACT

GVF Snakes is an acronym for Gradient Vector Flow. It is a framework in image processing/computer vision tasks particularly used for image segmentation, motion tracking, shape recognition.

Index Terms— GVF, image segmentation, snakes, Active contour models, feature extraction, edge detection, gradient vector flow, shape representation.

1. INTRODUCTION

GVF snakes is a contour based framework whose performance depends on the properties of the curve and the various energies or forces acting on it during a particular usage. When a snake active contour is initialized the snake deforms and moves towards the object of interest in such a way as to minimize the energy and stops moving when the energy is minimum. At this point the snake should surround the object and segment it to differentiate it from the other regions.
The Sum of Forces or energy acting on the snakes can be given below as:

$$E_{snakes} = E_{internal} + E_{external} + E_{constraint} \quad (1)$$

The internal energy given by $E_{internal}$ consists of: the **elasticity forces** - designed to hold the curve together to keep it from stretching and the **bending forces** - that keep it from bending too much [1].
The external energy pulls the snake towards the desired image edges. it is given by $E_{external}$ consists of: the **pressure forces** and the **potential forces**[1].

Where $E_{int} = E_{elastic} + E_{bending}$

$\mathrm{E}_{elastic} = \frac{1}{2}\int_s^b \alpha(s)|V_s|^2 ds$

$\mathrm{E}_{bending} = \frac{1}{2}\int_s^b \beta(s)|V_{ss}|^2 ds$

$$E_{snakes} = \int_0^1 \frac{1}{2}(\alpha(s)|V_s|^2 + \beta(s)|V_{ss}|^2) ds \quad (2)$$

Given a particular object in an image, regions of interest can be segmented into various parts to give an enhanced description of its features of the object using the GVF snakes algorithm.

The Image Data provided for this experiment is compressed in a zipped folder named: **snakepract.zip**. Using this Image data we are to test how good the GVF snakes algorithm is in segmenting various images with different contrast and we compare its performance with the standard Vector Framework otherwise called the classical snakes.

2. METHOD

For the practical we first uploaded the file "snakepract.zip" into the working directory of MatLab.
This is done by entering the sequence commands below in the MatLab command line:

```
unzip snakepract.zip
cd snakepract
```

The images contained in the `Snakepract` folder for this experiment includes
`heart1.pgm, myChest1.pgm, heart.pgm, room.pgm, U64.pgm` etc

The GVF snake algorithim and the classical snakes performance are now evaluated using the `sdemo` and the various parameters. The main variated parameters and their meanings is listed in Figure 1.

Parameter	Meaning	Default value
sigma	level of gaussian blurring	0
kappa	external force weight	0.6
beta	ridigility of the snake	0
alpha	tension of the snake	0.05
mu	regularization parameter	0.1

Fig. 1: The main variated parameter meanings.

In the main menu, `"Open"` command is clicked to select the Image, `sigma` is also set in this menu.. In the next menu, the type of snake is selected in the radio buttons and `mu` value is set (Figure 2). Finally, in the next window we set other parameters, initialize snake and start iteration.

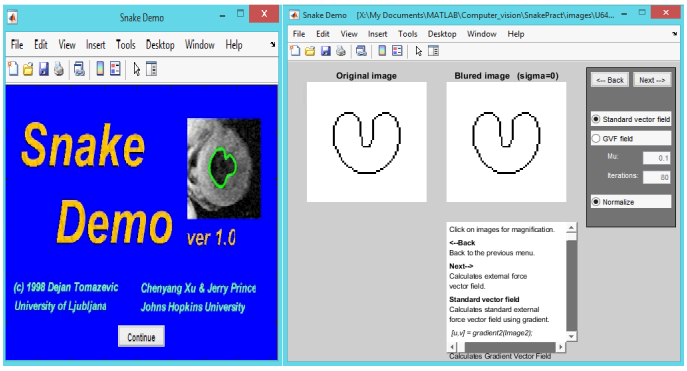


Fig. 2: A figure showing the sdemo first Menu

2.1. Exercise1

In this exercise we compared Gradient Vector Flow (GVF) with Standard Snakes (SVF). We tried to get snakes converge with borders of the `sdemo` build in image `'U Shape'` using both methods SVF and GFV. First we initialized snake around the shape and iterate with default parameters. From Figure 3 we can see SVF snake not converging to the dip as well as GVF.

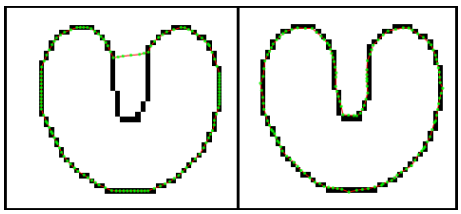


Fig. 3: Comparison of SVF and GVF snakes on U-shaped

Next, we initialized snake inside the shape. In this case GVF performed as well as before. Instead, SVF did not converge on shape due to its internal forces being stronger than external force far from shape borders. The result of this comparison is represented in Figure 4.

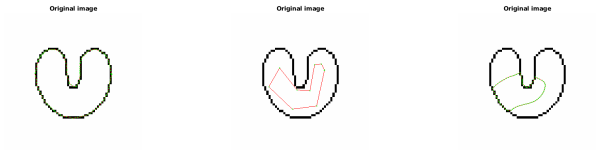


Fig. 4: Comparison of SVF and GVF snakes with inside initialization. From left to right: GVF snake, snake initialization, SVF snake.

Next we applied some blurring for the image setting `sigma=2`. As displayed in Figure 5, blurring helps classical snake to converge on dip in U-Shape but in this case makes GVF inaccurate.

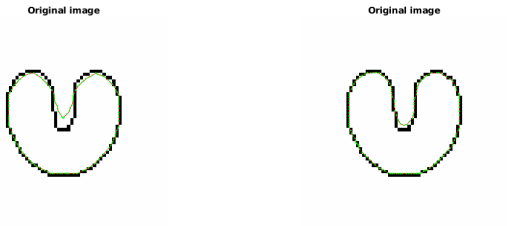


Fig. 5: Comparison of SVF and GVF snakes, `sigma=2`.

Finally we varied values of the `alpha, beta, mu, sigma` and `kappa`. We found that `alpha, beta` and `gamma` should be in balance to avoid to complex shapes still getting the snake converge well on the shape.

2.2. Exercise2

In this exercise we continued comparison of SVF and GVF methods now using `sdemo` build in image `room` as an input.

We initialized snake around the shape and apply both methods on it. We found GVF snake rounding the sharp corners of the shape and hence not fitting perfectly on the shape. We got SVF converging almost perfectly on the shape but requiring many times the number of iterations compared to the GVF (600 vs 20). Similarly with Exercise 1, SVF snake can not converge when initialize snake inside the shape. Our results is displayed in Figure 6.

Fig. 6: Comparison of SVF and GVF snakes on `room.pgm`. where the **left** is the output of GVF with resulting rounded corners, **right** is the output of the standard classical snake with sharp convergence at the edges

2.3. Exercise3

In this exercise we applied GVF and SVF methods for `'chest.png.'` Results is displayed in Table 1 below. We found out both methods working well with accurate initialization and right parameters. We used alpha=0.2, beta=0.2 for both methods. Mu was set to 0.08 when using GVF method. We varied sigma and found 4-5 working well with SVF and 2 with GVF. The right lung was harder because bottom and left border easily converged incorrectly.

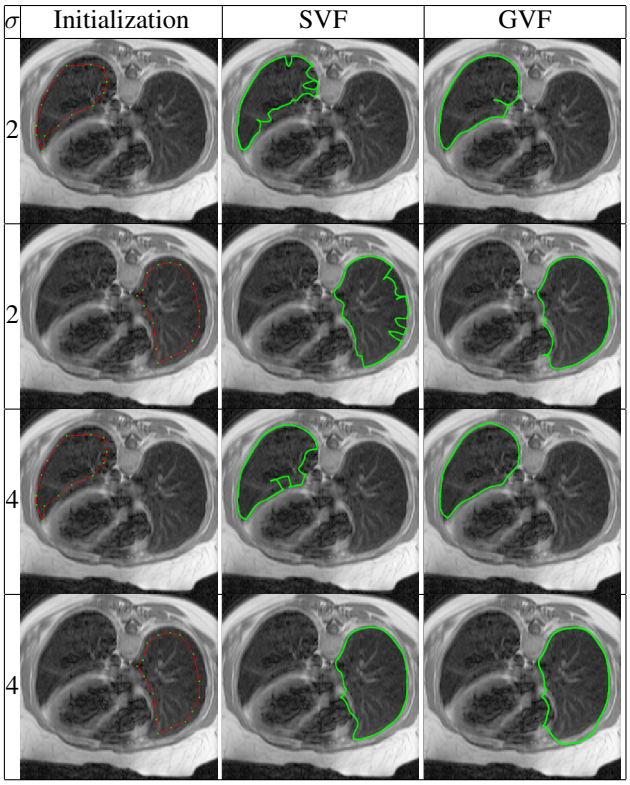


Table 1: Results of experiment with `'chest.pgm'` using different values of `sigma` with initialization `'mychest.mat'` for left and `'mychest2.mat'` for the right lunch. `mu=0.08`.

2.4. Exercise 4

We used the same parameter settings of the chest images above to test it on chest image of higher contrast, in this case `new.pgm`. We used `new1.pgm` image as our initial contour points. Surprisingly it turned out that with higher contrast using the same parameter settings used in `chest.pgm` for SVF and GVF snakes, they both gave poor results compared with the previous(see Figure 7 below)

2.5. Exercise 5

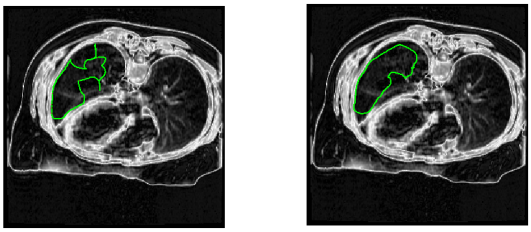
We now use `heart.pgm`, a heart ultrasound image with a far worse signal-to-noise ratio and experimented with different initialization and different values of sigma. The results are shown in the Table 2 below.
From the results we found out that

1. standard works well with high sigma. This is seen when we increased the sigma from 3 to 5 with a particular initialization, the result was a better convergence at the edges.

2. GVF works well with low sigma.This is seen when we increased the sigma from 3 to 5 with a particular initialization, the result was poorer convergence at the edges.

3. GVF converge better with bad initialization

4. standard is more accurate with right sigma and accurate initialization, but GVF does not depend much on the initialization, as seen from the table, GVF gave the same result for a given sigma with different initializations.



(a) SVF snakes on higher contrast chest image `new1.pgm`

(b) GVF snakes on higher contrast chest image `new1.pgm`

Fig. 7

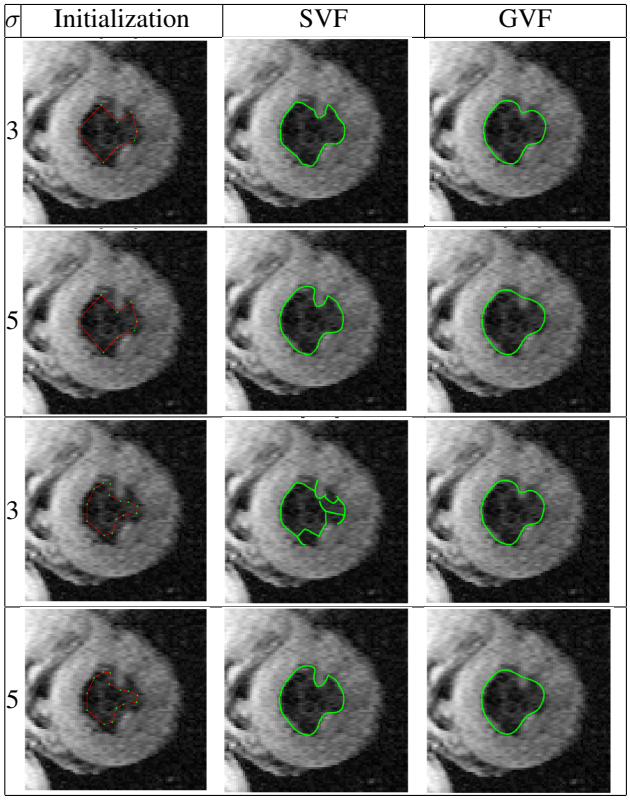


Table 2: Results of experiment with heart.pgm using different values of Sigma and different Initialization

2.6. Exercise 6

Snake-based image segmentation is not 100% accurate when used in scenarios involving the tracking an image involving multiple slices and are very sensitive to false local minima which leads to wrong convergence.
key problems include :

- variance of the quality of the results. Good results require right parameters (sigma, alpha, beta) and good initialization. It is very hard to get GVF or standard snakes working well in fully automated environment.
- requires lot of computation so not very good in real time applications.

From the experiments in exercise 1 and 2 above, it can be deduced that GVF snakes does not converge properly with perpendicular edges of images, while classical snakes did very well at this but failed to give good results at cavities.
This two methods could be combined using first GVF to make rough fit and after that use standard snakes to fit in details and avoid rounded corners. This can be shown in the Figure 8 below after running the test initially with GVF at 150 iterations, then SVF at the same number of iterations.

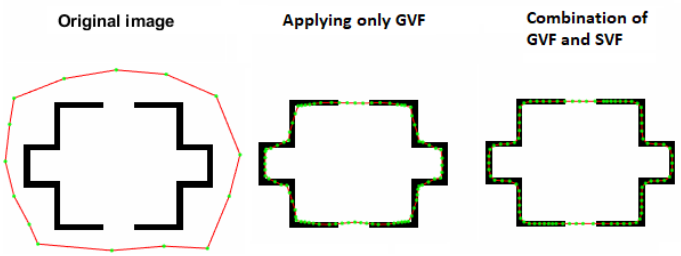


Fig. 8: results before combination (left), Applying first GVF (middle), combination of GVF followed by classical snakes (right)

3. CONCLUSION

The use of GVF snakes can be computationally expensive, it can be slow in some applications.One of the major problems is that it is sensitive to parameters.
Even though it has some advantages over the classical snakes (e.g detecting concavities), it should be used alongside the classical snakes to compensate for its shortcomings. Also the combination of the algorithms will could reduce computational cost.

4. REFERENCES

[1] Chenyang Xu and Jerry Ladd Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.