A Robust Parallel Implementation of Active Contours

B. Saranya and S. Suthakar
Department of Computer Science, Faculty of Science, University of Jaffna
balansaranya99@gmail.com

Abstract

The main intention of this project is to speed up the performance of active contour process by parallelizing it. In traditional methods an algorithm namely 'Snake' has been applied for representing object contours. An energy minimizing technique is being used in this method. To speed up the convergence process I've parallelized the process by dividing the initial contour into sub-contours, converging them and combining them. As a result there will be a big change in the performance within a short period of time than in serial processing.

Keywords: Parallelize, contour, convergence

Introduction

- ❖ Active contours are computer-generated deformable curves which are being used for energy segmentation of objects or to locate object boundaries in the field of computer vision and image processing applications.
- ❖ These contours converge under the influence of energy minimizing technique.
- ❖ The energy is computed by minimizing a function of internal and external forces.
- ❖ The internal forces depend on the curve and the external forces are computed from the image.

Snake Energy = Internal energy + External energy

$$E_{snake} = \int_0^1 \left(E_{snake}(v(s))\right) ds$$

$$E_{snake} = \int_0^1 \left(E_{int}(v(s))\right) + \left(E_{img}(v(s))\right) + \left(E_{con}(v(s))\right) ds$$

- E_{int}: The internal elastic energy term.
  Continuity of the contour + smoothness of the contour.

$$E_{int} = {(\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)}/{2}$$

$\alpha(s)$ and β(s) are user defined weights.

$vs(s)$: First derivative term controlled by $\alpha(s)$.
$vss(s)$) : Second derivative term controlled by $\beta(s)$.

- E_{img}: Combination of the forces due to the image itself.

$$E_{img} = W_{line}\, E_{line} + W_{edge}\, E_{edge} + W_{term}\, E_{term}$$

E_{line} : The intensity of the image.
$E_{edge} = -|\nabla I(x,y)|$, It is based on the image gradient.

$$E_{term} = \frac{\partial\theta}{\partial n_\perp} = \frac{\partial^2 c / \partial n_\perp{}^2}{\partial c / \partial n}$$

- E_{con}: Constraint forces introduced by the user.

Methodology

As a parallel processing solution for active contour model, rather than dividing the image into sub-images, we divide the initial contours into sub-contours in radial direction and process them independently. To define the sub-contours, the bounding box of the initial contour is used (see Fig.1.)

Then each sub-contour is given to different processing elements, and the contour points simultaneously move towards the edge of the sub-objects iteratively. Even though the divider lines are used to form closed contours, the points on the divider lines are kept unchanged while the sub-contour points are being shifted.

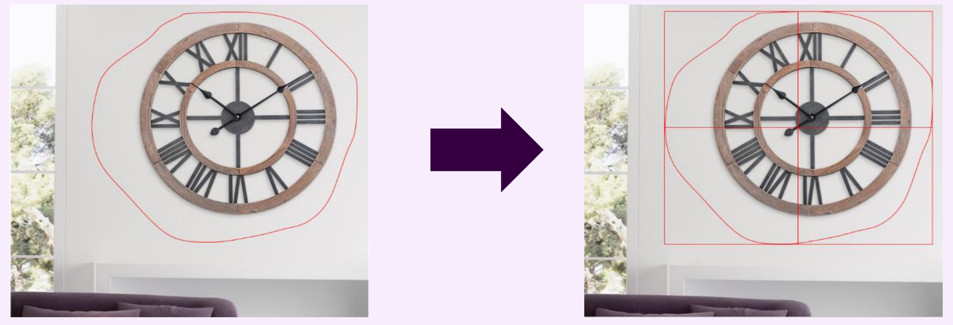

Fig. 1. Dividing initial contour into sub-contours in radial direction

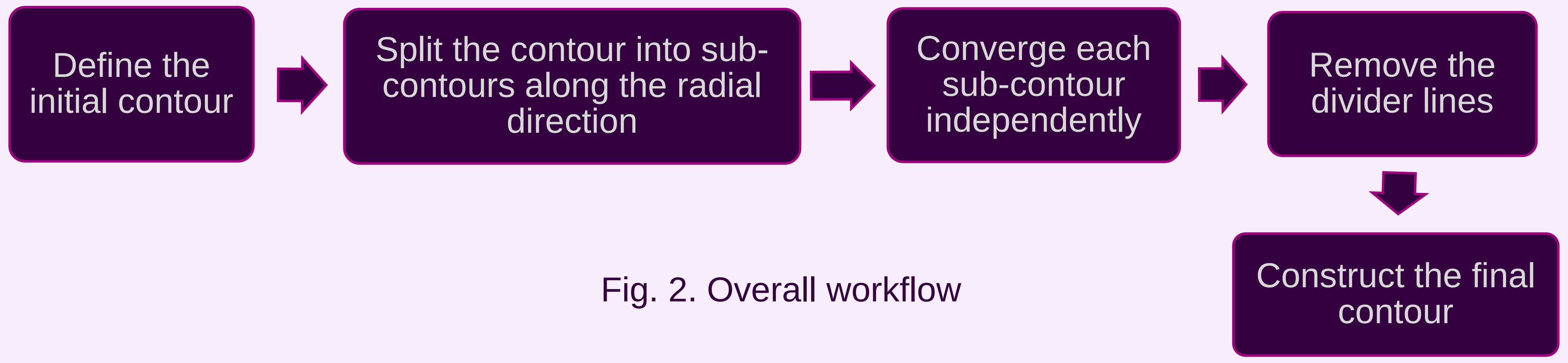

Fig. 2. Overall workflow

Merging Sub-Contours

When the sub-contour points are moving towards the boundary of the object simultaneously, the contour points on the divider lines are kept unchanged. Once the convergence is completed, the points on the opposite sides of each divider lines are connected to construct the final contour Fig. 3 illustrates this process.

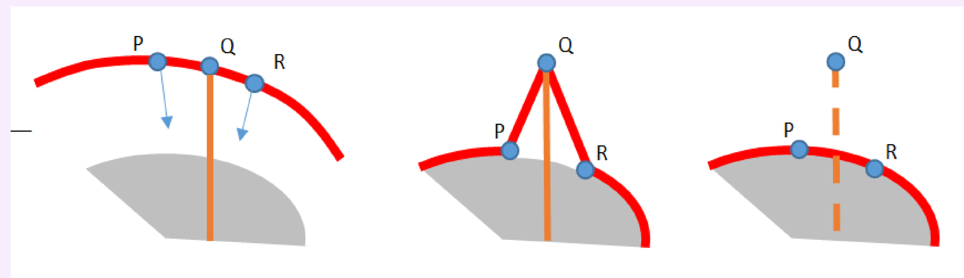

Fig. 3. Merging two sub-contours.

Sample Output

Some intermediate results from the start till the end of the proposed method are shown in Fig. 4.

Fig. 4. Intermediate results for an image.

Results and Discussion

For the evaluation, the proposed method was implemented with the help of Message Passing Interface (MPI) using two computers (Intel(R) Core(TM) i5-7200U, 8 GB RAM). Five different images of same dimension 3968x2976 were tested in serial and parallel environments. The execution times of serial and parallel implementation are given in Table 1. In order to compare the performances, same initial contours were used for both serial and parallel cases.

Table 1. Serial and parallel run times in seconds

	T_S	T_P	$S = T_s/T_P$
Image 1	4.523	3.112	1.453406
Image 2	3.425	2.314	1.480121
Image 3	5.281	3.998	1.32091
Image 4	4.910	3.428	1.432322
Image 5	3.911	2.855	1.369877

T_p - Parallel run time, T_s - Serial run time, S - Speed up

The proposed parallel method can be further improved by implementing using GPU.
In future there are possibilities to perform with concave objects.

Conclusion

- ➢ A parallel method is approached for implementing active contour model for making it more efficient.
- ➢ The given method segments in lesser time comparing to the serial active contour model.
- ➢ Using two PCs, the speed ups were roughly around 1.4. By using GPU, communication overheads can be reduced, and hence, the performance can be further improved

Reference

[1] P. L. Jerry and X. Chenyang, *Gradient Vector Flow Deformable Models, 2000.*
[2] A. Kass, A. Witkin and A. Terzopoulos, "Snakes: Active Contour Models," *International Journal of Computer Vision, 1988.*
[3] D. Cohen and Laurent, "On active contour models and balloons," in *CVGIP: Image Understanding, 1991.*
[4] S. Priya P Sajan, "GVF Snake Algorithm-a parallel approach," *International Journal of Engineering & Technology, 2018.*
[5] D. Oberhelman and Steven, "Active Contour Implementation," 2018.
[6] P. Jiang and Q. D. a. X. Hu, "A Parallel Realization of the Active Contour Model," *Applied Mathematics & Information Sciences, An International Journal, 2013.*