Multiplatform GPGPU implementation of the Active Contours Without Edges algorithm

Olmo Zavala-Romero, Anke Meyer-Baese and Uwe Meyer-Baese

Abstract

An OpenCL implementation of the Active Contours Without Edges (ACWE) algorithm is presented. The proposed algorithm uses the General Purpose Computing on Graphics Processing Units (GPGPU) to accelerate the original model by parallelizing the two main steps of the segmentation process, the computation of the Signed Distance Function (SDF) and the evolution of the segmented curve. This work is being done as an open source software that, being programmed in OpenCL, can be used in different platforms allowing a broad number of final users and can be applied in different areas of computer vision, like medical imaging, tracking, robotics, etc. This work uses OpenGL to visualize the algorithm results in real time.

Active Contours without Edges

The ACWE algorithm, introduced by Chan and Vese[2], is a segmentation method that evolves an initial curve until the image is partitioned into two regions with assumed piece-wise constant intensities. The segmentation process is carried out by minimizing the functional:

$$\begin{aligned} F(c_1,c_2,C) &= u \cdot \text{length}(C) + v \cdot \text{area}(C) \\ &+ \lambda_1 \int_{inside(C)} |u_0(x,y)-c_1|^2 dx\,dy \\ &+ \lambda_2 \int_{outside(C)} |u_0(x,y)-c_2|^2 dx\,dy \end{aligned} \quad (1)$$

Where C is the boundary of the segmentation, u_0 is the image and the functions **length** and **area** are regularizing terms of the functional. The constants c_1 and c_2 are the average values of the image inside and outside the contour and λ_1, λ_2, u and v are parameters used to weight each of the functional terms. Equation 1 is solved by the level set method [5] and the solution is given by:

$$\begin{aligned} \frac{\partial\phi}{\partial t} =& \delta(\phi)[v \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \\ &- (u_0(x,y)-c_1)^2 - (u_0(x,y)-c_2)^2] \end{aligned} \quad (2)$$

Where ϕ is the level set and $\operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$ is the image curvature for the 2D case.

References

[1] J. Calvin R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:265 – 270, 2003.

[2] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10, 2001.

[3] B. H., G. J., K. D., and W. M. Linear time euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1995.

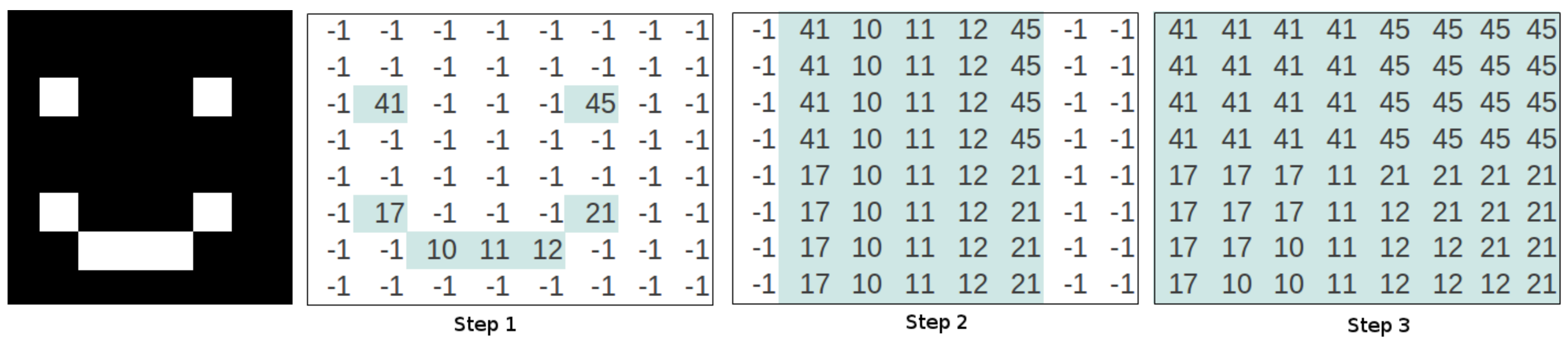
[4] S. Lankton. Sparse field methods. Technical report, Georgia Institute of Technology, 2009.

[5] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Computational Physics*, pages 12 –49, 1988.

[6] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International Journal of Computer Vision*, 50(3), 2002.

Signed Distance Function

The ACWE algorithm requires an initialization contour that is the zero value of a level set. The start function of the level set is commonly a SDF. In this work the algorithm introduced by Maurer et al. [1] is used to compute the SDF; this algorithm obtains the exact Euclidean distance of binary images in any dimension. Maurer et al. proposed that, rather than computing the distance directly, they treat every feature pixel (pixel with value of 1) as a Voronoi site and find the discrete Voronoi diagram of the image. The next figure shows the results of every step in Maurer et al. algorithm.



-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	41	-1	-1	-1	45	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	17	-1	-1	-1	21	-1	-1
-1	-1	10	11	12	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

Step 1

-1	41	10	11	12	45	-1	-1
-1	41	10	11	12	45	-1	-1
-1	41	10	11	12	45	-1	-1
-1	41	10	11	12	45	-1	-1
-1	17	10	11	12	21	-1	-1
-1	17	10	11	12	21	-1	-1
-1	17	10	11	12	21	-1	-1
-1	17	10	11	12	21	-1	-1

Step 2

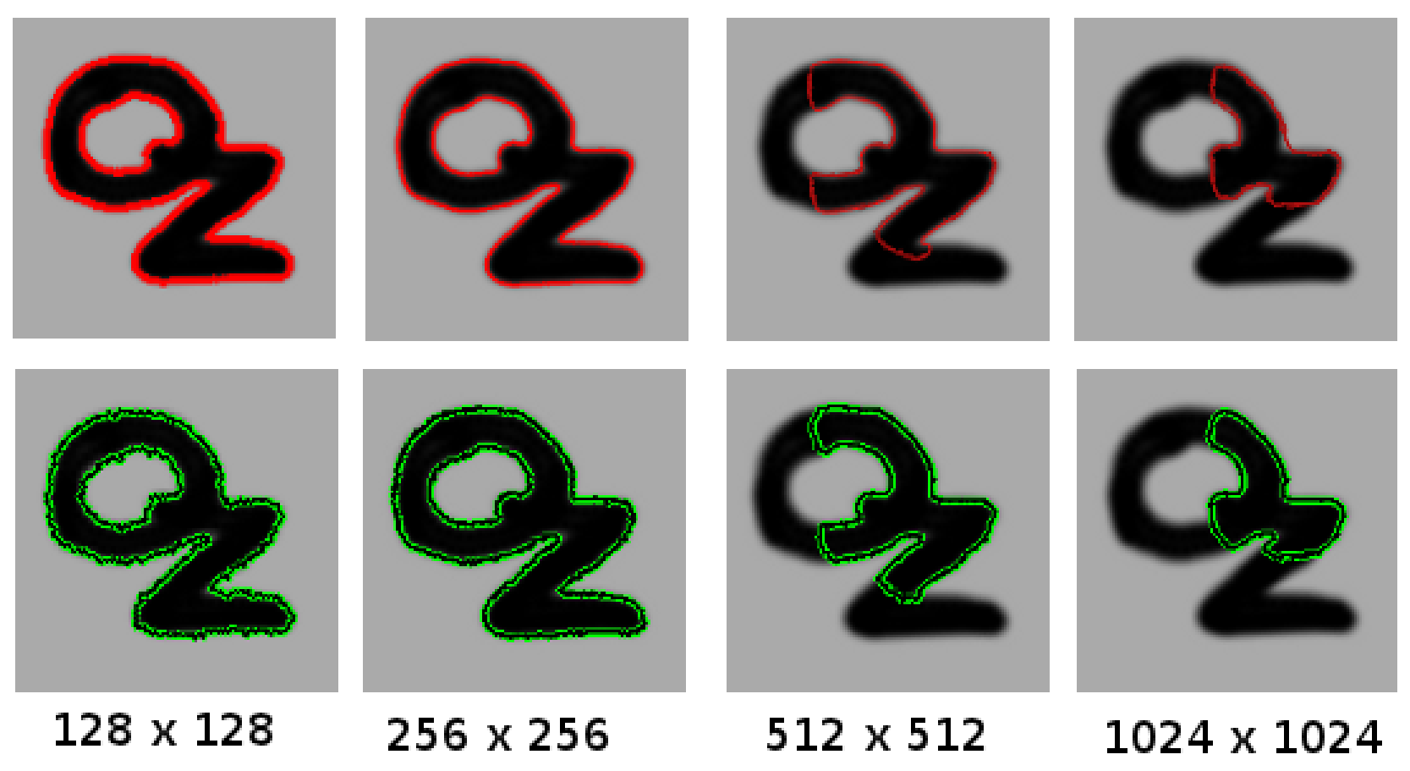
41	41	41	41	45	45	45	45
41	41	41	41	45	45	45	45
41	41	41	41	45	45	45	45
41	41	41	41	45	45	45	45
17	17	17	11	21	21	21	21
17	17	17	11	12	21	21	21
17	17	10	11	12	12	21	21
17	10	10	11	12	12	12	21

Step 3

Step 1. Assign to each pixel that represents a Voronoi site its corresponding index on the image.
Step 2. Compute a 1D partial Voronoi diagram; each pixel is assigned the index of its closest Voronoi site considering only one dimension.
Step 3. Compute the 2D partial Voronoi diagram.

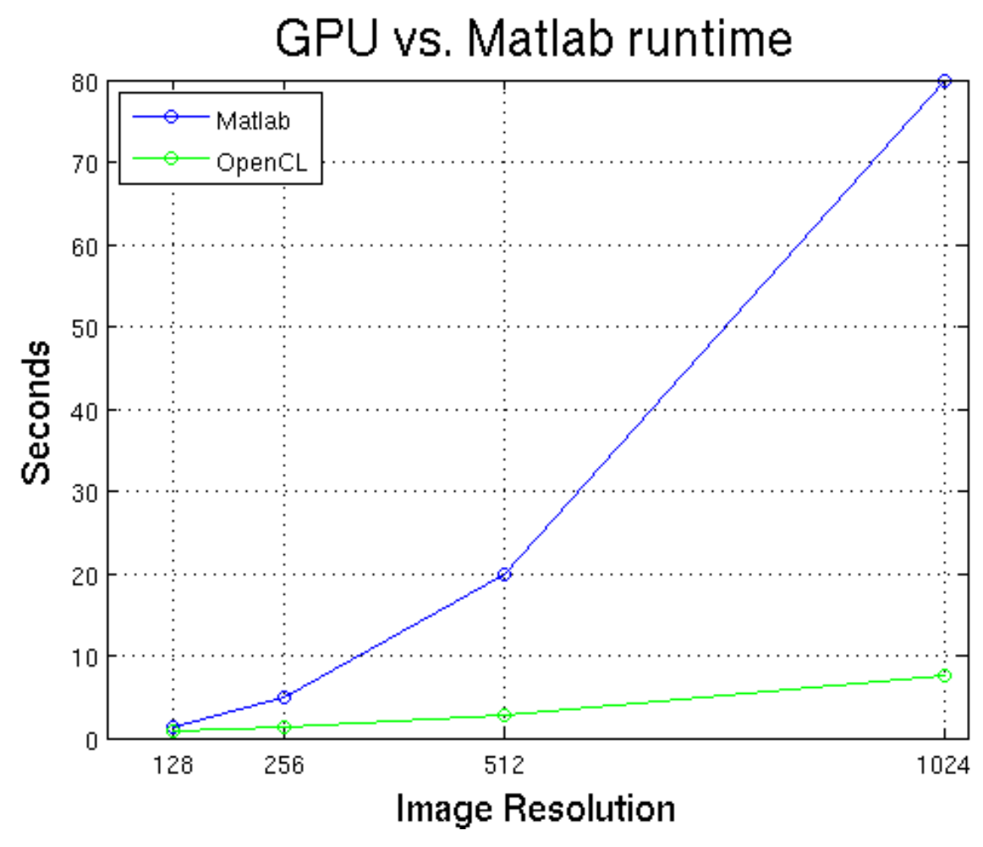
Results

The OpenCL implementation of the ACWE was compared with the Matlab version developed by Shawn Lankton [4], which is an equivalent sequential version of the work currently presented. The proposed GPGPU implementation of the Chan and Vese algorithm can segment images 10 times faster than its Matlab equivalent using images sizes of 1024 x 1024.

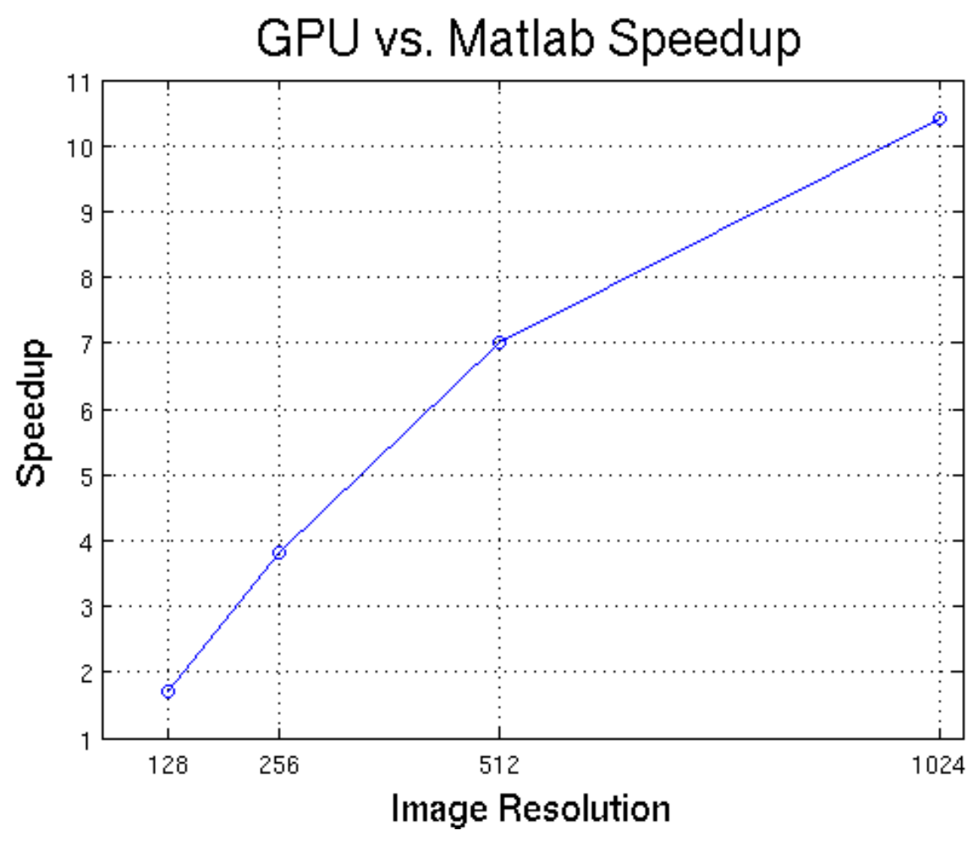


128 x 128 256 x 256 512 x 512 1024 x 1024

ACWE Example

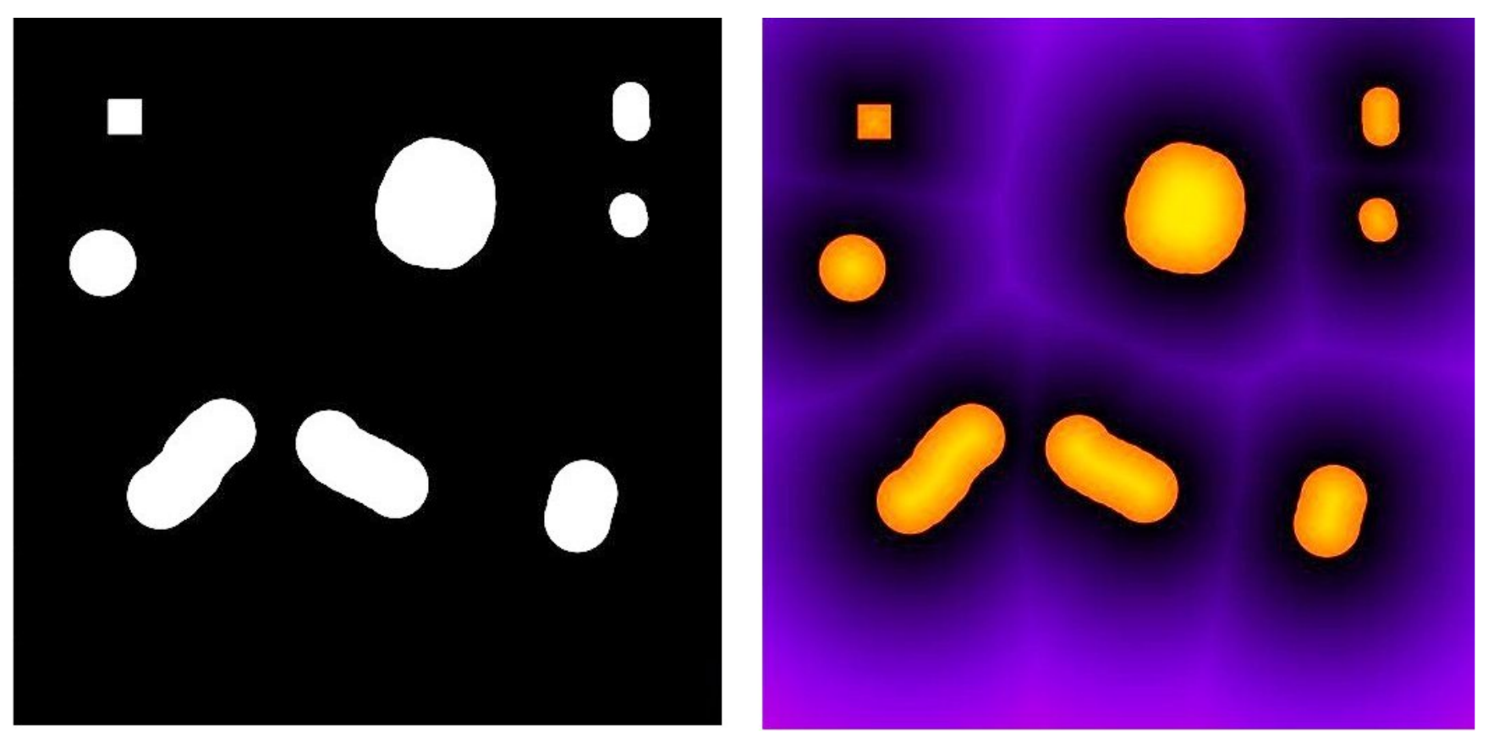


ACWE Performance test

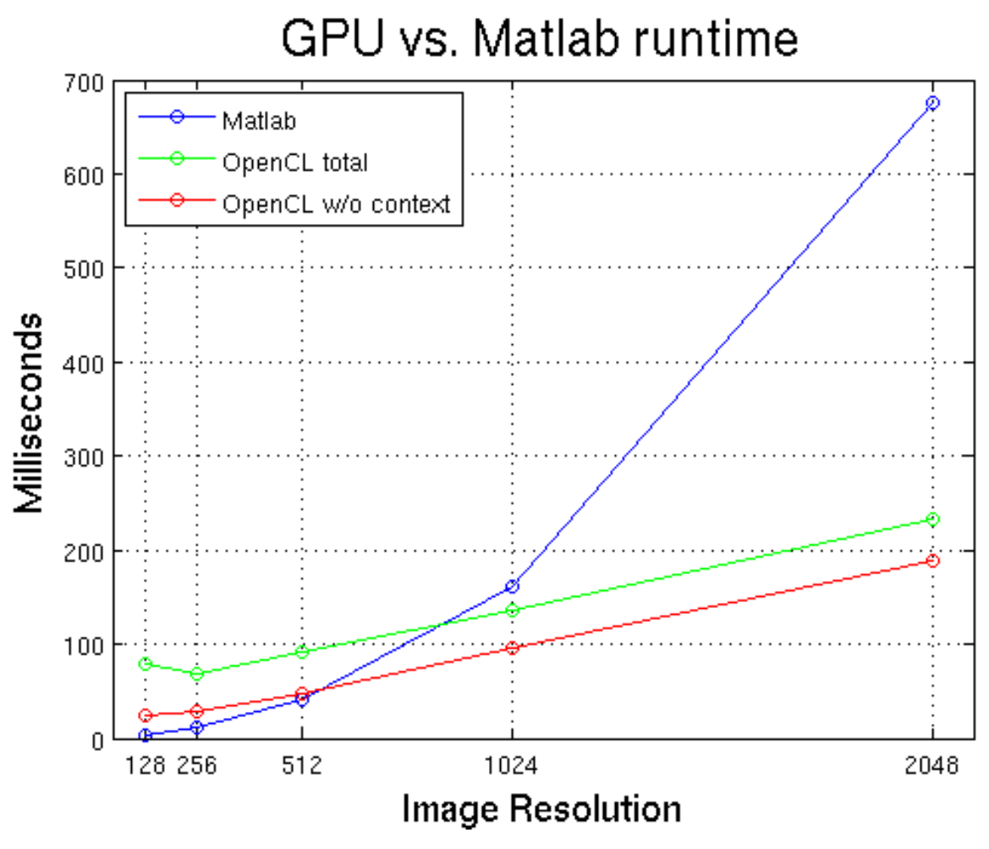


ACWE Speedup

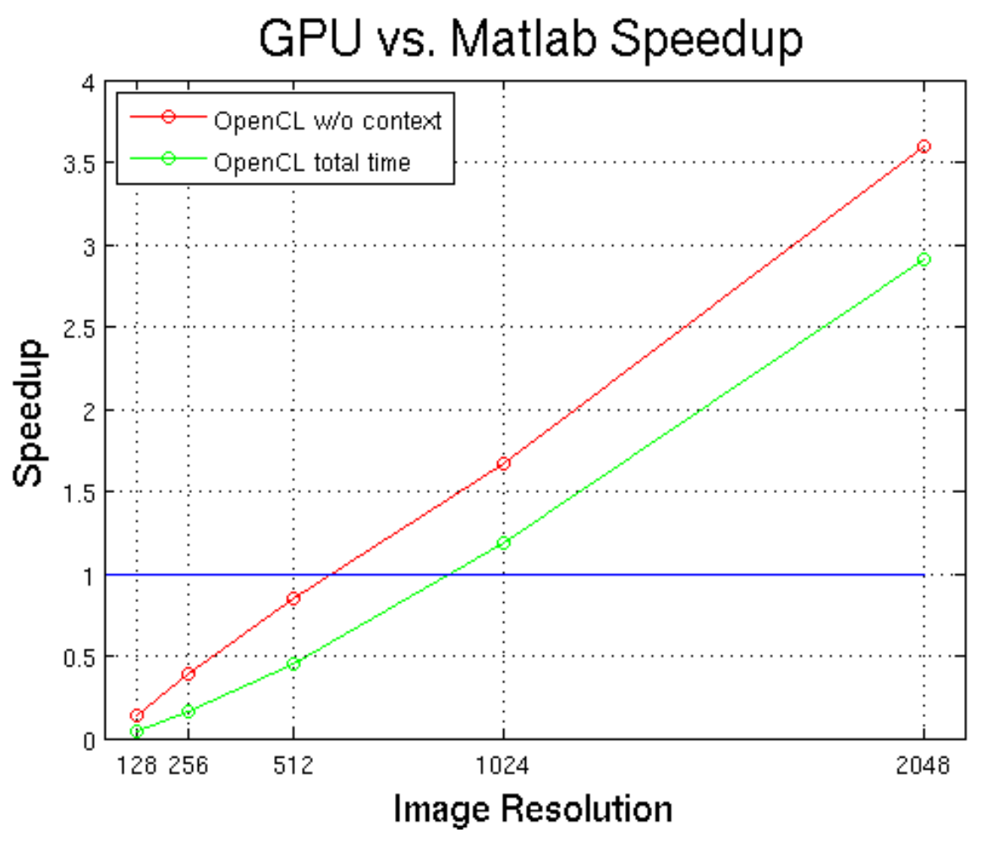
The SDF algorithm was compared with the Matlab function *bwdist* which uses the second algorithm proposed by Breu et al. in [3]. The SDF OpenCL program runs 3.5 times faster than the Matlab implementation of the Breu et al. algorithm for image resolutions of 2048 x 2048.



SDF Example



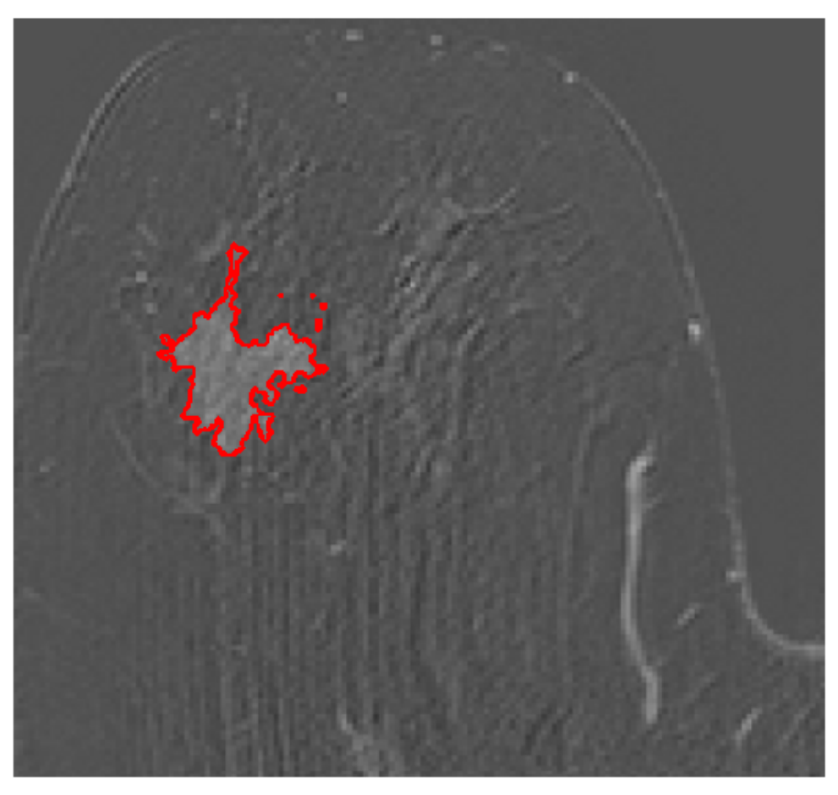
SDF Performance test



SDF Speedup

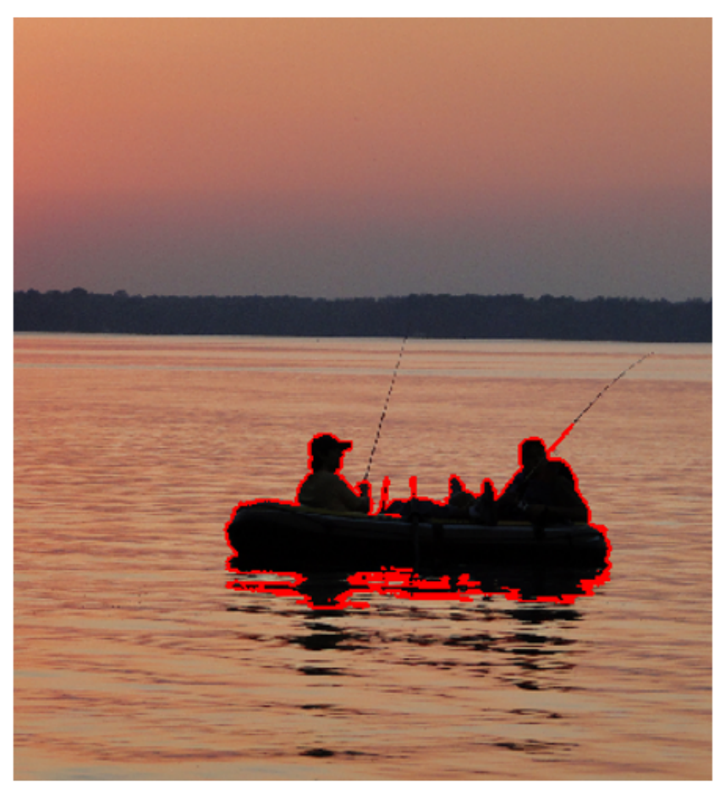
Conclusion and future work

The ACWE algorithm can be applied in different areas like medical images, tracking, and pattern recognition. The next figure shows some useful examples achieved with the previously described OpenCL implementation.



MRI Breast tumor

Jets



Inflatable boat

This work can be extended to perform 3D volume segmentation which would be very useful for many medical image applications and, due to the added dimension, a fast implementation is desirable. Another possible future work is to extend the ACWE algorithm to its multiphase version [6], this algorithm segments the image in more than two regions expanding its area of application and could be greatly benefited from the parallel architecture of the GPU.