OpenCL Implementation of a Color Based Object Tracking

Marko Jocić, Đorđe Obradović, Zora Konjović, Daniel Tertei
{m.jocic, obrad, ftn_zora}@uns.ac.rs, danieltertei@gmail.com
University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia

Abstract – in this paper we present an algorithm for real-time object tracking based on color. Firstly, a two-layer perceptron is trained aimed at coping with scene illumination changes. Based on this training, a piece of OpenCL code is generated for the purpose of harnessing the power of GPU computing. Then, color based object tracking is done in four steps, and finally a fast connected component labeling algorithm is applied for determining distinct regions, with only the largest one selected and tracked on the image. Proposed algorithm is executed both on CPU and on GPU for comparative analysis.

1. INTRODUCTION

Real-time object tracking is the critical task in many computer vision applications such as surveillance perceptual user interfaces, augmented reality, smart rooms, object-based video compression, and driver assistance [1].

The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms [2].

Tracking objects can be complex due to [2]:

- loss of information caused by projection of the 3D world on a 2D image,
- noise in images,
- complex object motion,
- non-rigid or articulated nature of objects,
- partial and full object occlusions,
- complex object shapes,
- scene illumination changes, and
- real-time processing requirements

Selecting the right features plays a critical role in tracking. In general, the most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space. Feature selection is closely related to the object representation. For example, color is used as a feature for histogram-based appearance representations, while for contour-based representation, object edges are usually used as features [2].

Color provides powerful information for object recognition and tracking moving objects based on color information is more robust than systems utilizing motion cues [3] [4].

Among all features, color is one of the most widely used for tracking. Despite its popularity, most color bands are sensitive to illumination variation. The apparent color of an object is influenced primarily by two physical factors: the spectral power distribution of the illuminant, and the surface reflectance properties of the object. In image processing, the RGB color space is commonly used to represent color. However, the RGB space is not a perceptually uniform color space, that is, the differences between the colors in the RGB space do not correspond to the color differences perceived by humans [5]. Additionally, the RGB dimensions are highly correlated. In contrast, $L^*u^*v^*$ and $L^*a^*b^*$ are perceptually uniform color spaces, while HSV is an approximately uniform color space. However, these color spaces are sensitive to noise [6].

Due to noise in images, and in order to be able to cope with scene illumination changes, we used a simple artificial neural network (ANN) – a two-layer perceptron [7] to calculate the value of membership function for each pixel on the image, which represent a degree of similarity between the pixel's color and the desired color of object that should be tracked.

Real-time image segmentation requires fast and effective way of computing certain data for each pixel of the image. To achieve this, we created automatic generation of code for OpenCL framework, which then runs on graphics processing unit (GPU).

This paper consists of seven sections. Following this introductory section and OpenCL technical overview section, the training of the proposed artificial neural network is shown in section three. Color based object tracking algorithm is described in fourth section. Section five shows comparative results of presented algorithm running both on CPU and on GPU. Sixth section contains one example of practical application of the proposed algorithm. The final section contains concluding remarks and future research directions.

2. OpenCL

OpenCL™ is the first open, royalty-free standard for cross-platform, parallel programming of modern processors found in personal computers, servers and handheld/embedded devices. OpenCL (Open Computing Language) greatly improves speed and responsiveness for a wide spectrum of applications in numerous market categories from gaming and entertainment to scientific and medical software [8].

OpenCL code platform modes consist of one host and one or more compute devices, where each compute device is composed of one or more compute units, and each compute unit is further divided into one or more processing elements. The main idea for speeding up program execution and computing with OpenCL is by replacing traditional loops with data and task parallel OpenCL code. So, serial code still executes in Host (CPU) thread, while parallel code executes in many Device (GPU) threads across multiple processing elements.

An OpenCL application runs on a host which submits work to the compute devices. OpenCL execution model consists of [9]:

- **Work-item**: the basic unit of work on an OpenCL device (work-items are grouped into local **work-groups**)
- **Kernel**: the code for a work item. Basically a C function (more specifically C99 (ISO/IEC 9899:1999), a past version of the C programming language)
- **Program**: Collection of kernels and other functions (Analogous to a dynamic library)
- **Context**: The environment within which work-items execute. Includes devices and their memories and command queues
- **Command Queue:** A queue used by the Host application to submit work to a Device. A sequence of commands scheduled for execution on a specific device

In order to efficiently harness the power of this parallel programming model, first the task has to be decomposed into work-items, meaning that N-dimensional computation domain has to be defined. Then, a kernel at each point in computation domain is executed. Programming kernels is done with OpenCL C language. This language is subset of C99, but without some features such as function pointers, recursion, variable length arrays and bit fields. In addition, it is a superset of C99 with additions for work-items and work-groups, vector types, synchronization and address space qualifiers. OpenCL language also includes a large set of built-in functions, such as image manipulation, work-item manipulation, specialized math routines, etc.

3. ANN TRAINING

In order to calculate membership function value for each pixel on the image, a simple two-layer perceptron is used. Proposed ANN consists of three input neurons, three neurons in the hidden layer and one output neuron (Figure 1). Sigmoid function is used as an activation function. *Hue*, *saturation* and *value* values are fed to input neurons.

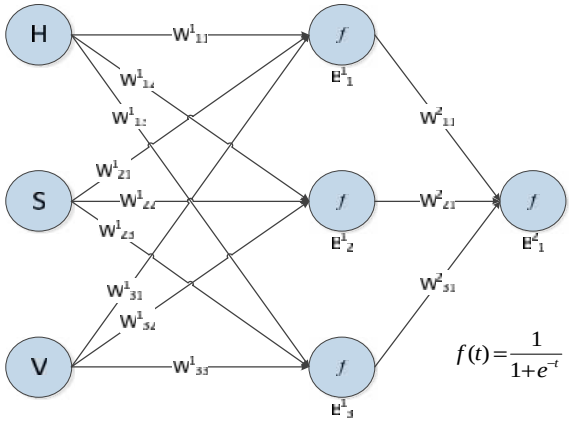

Figure 1 – artificial neural network structure

Training set for this ANN consists of two distinct parts. First part contains positive examples (object color that should be tracked), with output neuron value set to 1. The other part contains negative samples (color that should not be tracked), with output neuron value set to 0. Figure 2 shows an example where four points from the red marker are taken as positive examples, and all the rest are taken as negative examples.

Figure 2 – training set example

The network is trained with backpropagation algorithm [10]. After the network is trained, OpenCL kernel code for calculating value of output neuron is generated. This can be done because the proposed network is simple and the value of output neuron can easily be calculated by formula based on trained network's weights and neuron biases.

Value of output neuron can be calculated as:

$$R = f(f(H* W^1_{11} + S* W^1_{21} + V* W^1_{31} + B^1_1)* W^2_{11} + f(H* W^1_{12} + S* W^1_{22} + V* W^1_{32} + B^1_2)* W^2_{21} + f(H* W^1_{13} + S* W^1_{23} + V* W^1_{33} + B^1_3)* W^2_{31} + B^2_1).$$

Here, f is a sigmoid activation function, H, S and V are *Hue*, *Saturation* and *Value* values respectively, and W^k_{ij} are network weights.

For the ANN trained on the previously shown example of red marker, the resulting generated piece of OpenCL code looks like this:

```
float result = sigmoid(sigmoid(h * 0.2993147f +
s * 2.3967974f + v * 1.8910142f - 5.1325531f) *
7.5345923f + sigmoid(h * 0.9216228f + s *
2.5096687f + v * -0.5978528f - 0.4280259f) *
3.4976720f + sigmoid(h * 0.2387477f + s *
2.2689740f + v * 1.76323059f - 5.0153077f) *
7.1598966f - 10.5906315f)
```

4. COLOR BASED OBJECT TRACKING ALGORITHM

Proposed color based object tracking algorithm consists of four steps, as shown in Figure 3. Firstly, a raw image is blurred in order to reduce image noise. For this purpose, Gaussian blur with standard deviation equal 1 is used. The second step is based on our previous work [11]. A classical fuzzy set is created from the image, by calculating membership function value for each pixel on the image. This is done both on CPU (as an output from previously trained ANN) and on GPU (with previously generated OpenCL code) to compare performances later. Output for each pixel is real value in range [0,1]. Following this step, α-cut is applied to the resulting classical fuzzy set, which produces a classical set, or more precisely a binary image, where all pixels on the image either do satisfy certain minimal membership value (white pixels) or do not satisfy this criteria (black pixels). Concluding with these first three steps, the image segmentation is done. The fourth and the last step is applying fast connected component labeling algorithm [12] to detect regions, and once all the distinct regions are detected, the largest one is selected and tracked. Figure 4 illustrates application of the proposed algorithm.

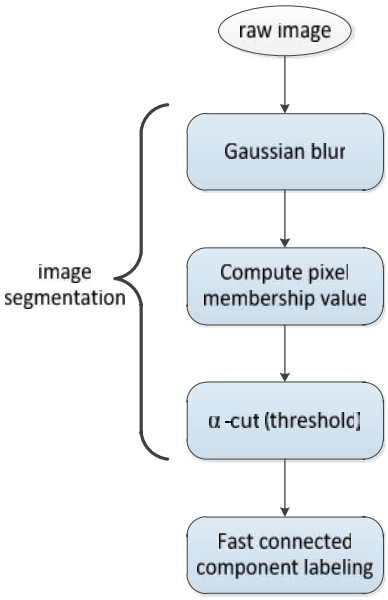

Figure 3 – color based object tracking algorithm diagram

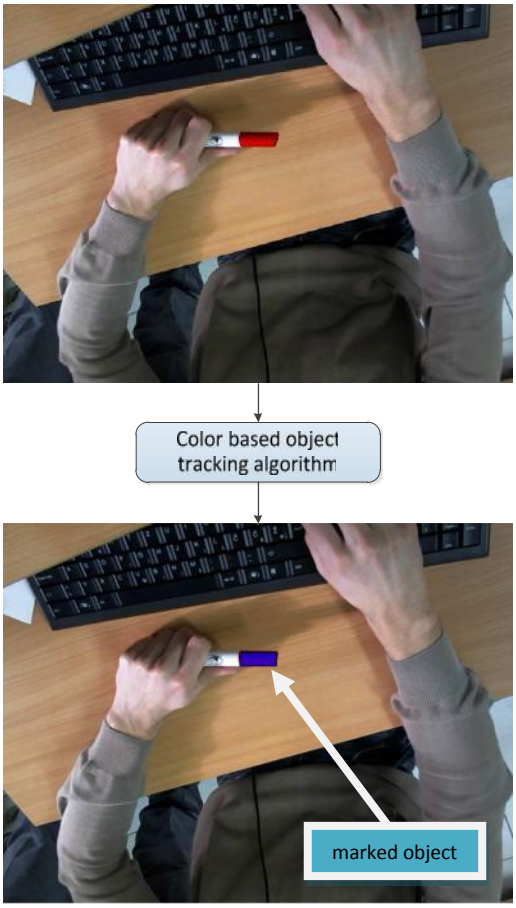

Figure 4 – an example result of the proposed algorithm

5. PERFORMANCE ANALYSIS

In order to show performance increase when using parallel OpenCL code that executes on GPU, opposed to serial code that executes on CPU, a series of performance measurements are done for various image resolutions. Images (or frames) are fed to the algorithm through the

web camera and then processed. Performance is measured in processed frames per second (FPS), where maximum value of FPS on the used web camera (Logitech C525 HD Webcam) is 30. Measurements were done on the system with Intel Core i5-2320 CPU, NVidia GeForce GT 420 GPU, and 8GB of RAM. Results are shown in Table 1.

Table 1 – comparative performance results

Resolution	Performance	
320x240 (76.8K pixels)	GPU	29.6 FPS
	CPU	17.2 FPS
	GPU vs. CPU increase	**72.1%**
640x480 (307.2K pixels)	GPU	14.7 FPS
	CPU	4.3 FPS
	GPU vs. CPU increase	**241.7%**
800x600 (480K pixels)	GPU	9.5 FPS
	CPU	2.6 FPS
	GPU vs. CPU increase	**265.4%**
1280x960 (1228.8K pixels)	GPU	3.7 FPS
	CPU	0.9 FPS
	GPU vs. CPU increase	**311.1%**

Performance analysis illustrates superiority of OpenCL GPU computing to regular CPU computing, with performance increasing from 70% on low resolutions, up to 300% on full HD resolution.

6. AN EXAMPLE OF PRACTICAL APPLICATION

Implemented algorithm for color based image tracking can be used to manually draw shapes with tracked object. This can enhance video teaching lessons or can be used for some kind of presentation purposes. An example of this application, with red marker as drawing object is shown in Figure 5.

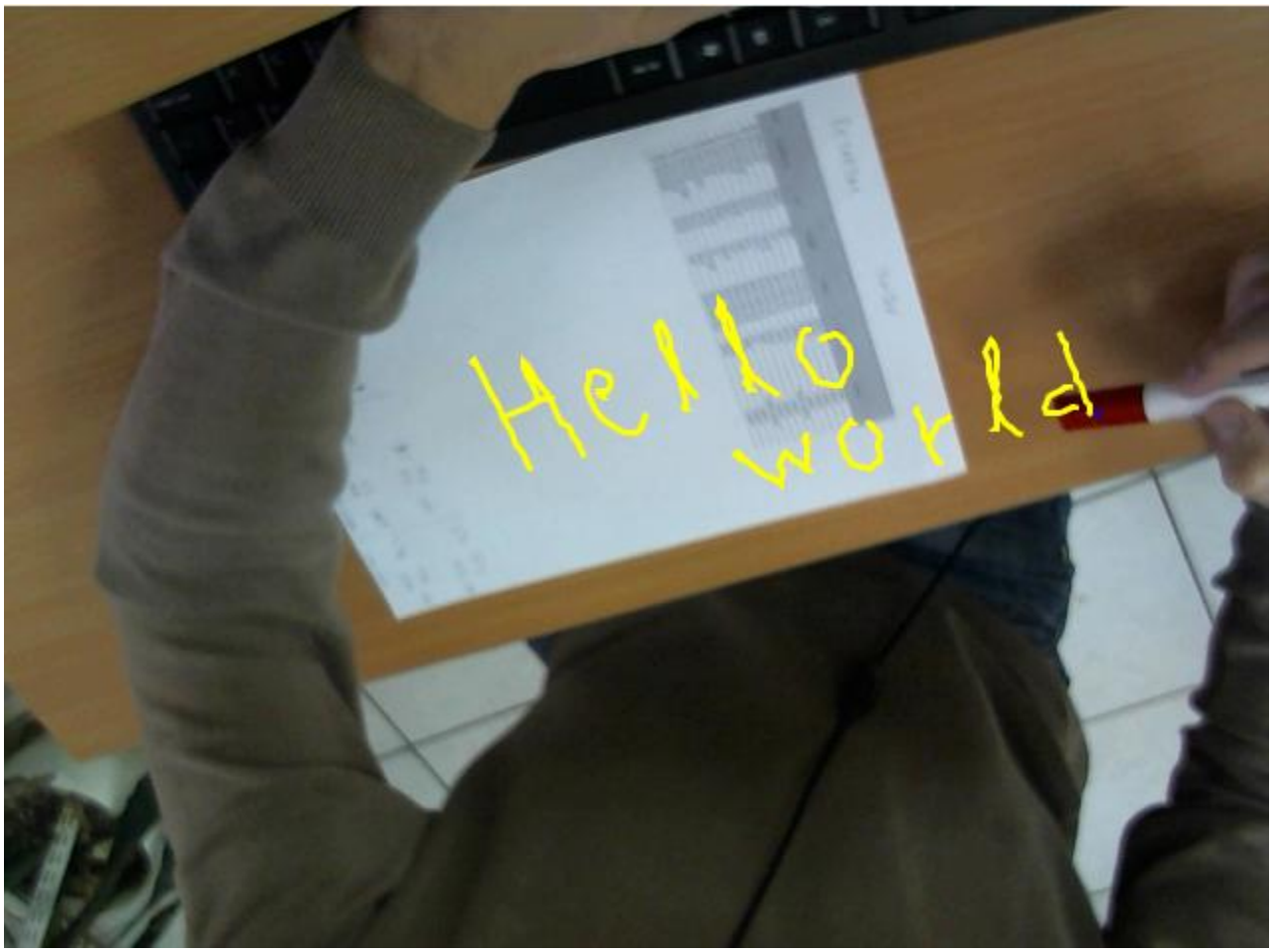

Figure 5 – an example of practical application, real-time drawing with red marker

7. CONCLUSION

In this paper we proposed an algorithm for color based object tracking. Proposed algorithm does image segmentation and fast connected component labeling in order to detect object that needs to be tracked based on its color. To support real-time algorithm execution, it has been implemented to run on GPU (with OpenCL). In addition, it has been implemented to run on CPU (with serial code) as well, for purposes of performance analysis, where GPU has shown as superior to CPU, with performance increase up to 300%. Future research includes improvement of the current algorithm, in a way to potentiate texture based object tracking. Also, as GPU computing has shown remarkable results in speeding up image segmentation, it can be used for fast and effective medical image analysis.

ACKNOWLEDGEMENT

Research presented in this paper is partly funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia, Grant No. III 47003.

REFERENCES

[1] P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, 2003.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking," *ACM Computing Surveys*, vol. 38, no. 4, p. 13–es, Dec. 2006.

[3] M. M. Asif, P. Angelov, and H. Ahmed, "An Approach to Real-time Color-based Object Tracking," in *Evolving Fuzzy Systems, 2006 International Symposium on*, 2006, pp. 86 –91.

[4] T. Gevers and W. M. Smeulders, "Color based object recognition," *Pattern recognition*, vol. 32, no. 3, pp. 453–464, 1999.

[5] G. Paschos, "Perceptually uniform color spaces for color texture analysis: an empirical evaluation," *Image Processing, IEEE Transactions on*, vol. 10, no. 6, pp. 932–937, 2001.

[6] K. Y. Song, J. Kittler, and M. Petrou, "Defect detection in random colour textures," *Image and Vision Computing*, vol. 14, no. 9, pp. 667 – 683, 1996.

[7] J. Freeman and D. M. Skapura, *Neural networks: algorithms, applications, and programming techniques*. Reading, MA: Addison-Wesley, 1991.

[8] "OpenCL - The open standard for parallel programming of heterogeneous systems." [Online]. Available: http://www.khronos.org/opencl/. [Accessed: 19-Jan-2013].

[9] Khronos OpenCL Working Group, *The OpenCL Specification*. 2010.

[10] R. Rojas, *Neural networks: a systematic introduction*. Berlin; New York: Springer-Verlag, 1996.

[11] D. Obradovic, Z. Konjovic, E. Pap, and M. Jocic, "Linear fuzzy space polygon based image segmentation and feature extraction," in *Intelligent Systems and Informatics (SISY), 2012 IEEE 10th Jubilee International Symposium on*, 2012, pp. 543 – 548.

[12] J.-M. Park, C. G. Looney, and H.-C. Chen, "Fast Connected Component Labeling Algorihtm Using a Divide and Conquer Technique." University of Alabama, Tuscaloosa.