HIGH SPEED VISUAL SALIENCY COMPUTATION ON GPU

Bo Han, Bingfeng Zhou[†]

Institute of Computer Science and Technology
Peking University, Beijing, P.R.China

ABSTRACT

Visual saliency analysis provides a power tool for many applications. In this paper, we propose a practical and high performance GPU-based visual saliency computational model. Several novel ideas are introduced for saliency computations, such as the feature extraction as signed difference values in the Lab color space and the feature fusion based on the information theory. For our implementation on the GPU, besides the programmable shaders we fully exploit other computational resources on the GPU to accelerate the computations. Our experimental results demonstrate the effectiveness of our saliency maps and indicate an order of magnitude speedup over the CPU-based implementation.

Index Terms— Attention model, Saliency map, GPU

1. INTRODUCTION

Visual attention (VA) is an efficient mechanism which filters out redundant parts and selects only important visual information for further identification and understanding. It greatly reduces the quantity of input data and gives our ability for rapid analysis and reaction to the complex environment. No doubt, such a mechanism is much superior to any available machine based methods. Therefore, some computational models have been proposed to mimic the biological behavior and the neuronal architecture. The most famous one has been proposed by Itti and Koch [1]. His work is based on the concept of saliency maps. Due to the effectiveness of attention models, they have found their wide range of applications including image/video representation and compression, information retrieval [2], realistic rendering [3], object detection and robotic control [4], video adaptation on small-display devices [5].

However, most of these applications adopt much simplified versions of the biologically plausible model, due to the fact that high computational complexity for imitating biological functions requires massively parallel implementations to obtain fast responses. Although these simplified methods can efficiently solve problems in their corresponding fields, they are lack of generality and usually not suitable for sophisticate tasks, such as robotic control and target search.

*Thanks to the NSFC for funding (Grant No.60573149).

†cczbf@pku.edu.cn

Fortunately, such a massively parallel computational model fits well with today's graphics processor units (GPU). GPUs are built on highly parallel architectures and are evolving rapidly in recent years. Its huge bandwidth, amazing computational power and increasing programmability have attracted lots of researchers to solve general-purposed computations on GPUs, known as GPGPU [6]. Based on the facts, we are inspired to utilize the GPU to implement our saliency model.

The main contributions of this paper are described as followings. First of all, we propose a high performance GPU-based visual saliency model. Our work demonstrates the efficiency of utilizing GPUs for such a computation intensive task, which is due to their similar parallel processing models. We expect our work can find its applications for time-consuming multimedia analysis, robotic control,etc. Besides, some novel ideas are introduced in our saliency model, such as the feature extraction as signed difference values in the Lab color space and the feature fusion based on the information theory.

2. ATTENTION MODEL AND GPU

In this section we quickly review several basic concepts of visual attention models and some GPGPU technologies.

Most recent visual saliency computational models are built around two important concepts: the Feature Integration Theory (FIT) and a neurally plausible architecture [7]. They suggest that the bottom-up biological systems response to the visual stimulus with low-level feature extraction, center-surround mechanisms and lateral inhibition effects. All these happen in massively parallel. Following these basic principles, the famous Itti's model [1] mainly consists of three parts: early visual features extraction, feature maps building, and feature map fusion. This model utilizes the image pyramid to build the feature maps: intensity contrast, R-G/B-Y color opponents and orientations. In addition, a global nonlinear amplification function $N(.)$ and a across-scale combination method are proposed to combine these separate feature maps into a single saliency map. We notice that such computations are dominated by per-pixel operations, across-scale interpolations and image convolutions, which are indeed the advantages of today's GPU.

The GPU's computational power mainly stems from its

highly parallel architecture, which has dozens of fragment and vertex processors working simultaneously and has deep pipelined stages to yield high throughput. For general purpose computation, the application's computation kernels are written as a series of vertex and fragment shaders, while the data are stored as the geometries and textures. Drawing graphics primitives activates the computation kernel to be executed for each vertex or pixel. All are performed in a highly parallel manner. In recent years, some GPGPU techniques and data structures have been developed, such as reduction, sort and linear algebra [6]. Moreover, for visual saliency computations additional resources on the GPU can be utilized. The automatic Mipmap texture generation hardware can act as a good candidate for building the image pyramid; the automatic pixel interpolation of texture mapping makes the cross-scale operations nearly computation free; and the blending units can accumulate results as addition operations. Besides, the occlusion query, which counts the number of pixels drawn by a rendering pass, provides us an efficient tool to analyze the statistical information of an image. The details of utilizing these hardware features are presented in section 4.

3. OUR COMPUTATIONAL MODEL

Our visual saliency model is based on Itti's [1] and Frintrop's VOCUS [4]. In this section, we present an overview of our model and emphasize several new ideas presented in our work, that is, feature extraction in Lab color space and map-fusion by means of the information quantity.

3.1. Feature Computations

The input image is first converted from RGB to the perceptually uniform CIE Lab color space. The space is spanned by three axes: L represents luminance; a represent the color component between red and green; and b for the component between yellow and blue. We treat these three values of each pixel as approximations for the intensity $\mathcal{I}$, red-green color opponent $\mathcal{RG}$ and blue-yellow opponent $\mathcal{BY}$.

Then, the converted image is used to create a Gaussian Pyramid $\mathcal{P}_l(\sigma)$, where $l \in \{\mathcal{I}, \mathcal{RG}, \mathcal{BY}\}$ and $\sigma \in \{0..8\}$. Center-surround differences $\ominus$, which detects locations that locally stand out from their surroundings, is implemented as the difference between fine and coarse scales. The operation is obtained by interpolation to the finer scale and point-by-point subtraction.

$$\mathcal{M}_l(c, s) = \mathcal{P}_l(c) \ominus \mathcal{P}_l(s) \tag{1}$$

where $c \in \{2, 3, 4\}; s = c + \delta, \delta \in \{3, 4\}$. Noticed that we use the signed difference value to account for both on-center and off-center cells in receptive field, which respond to light at the center excitably or inhibitably, respectively. This fact is ignored in Itti's[1] by taking the absolute value. VOCUS[4] addressed this issue by adding additional maps. To analyze in a multi-scale manner, six maps of each feature l are yielded. Each map has separate positive and negative ranges to represent the on/off center responses.

Local orientation information is obtained by applying Gabor filters to the levels of luminance pyramid $\mathcal{P}_\mathcal{I}$. Since the center-surround differences are already determined implicitly by the gabor filters [4], we directly choose three levels $c \in \{2, 3, 4\}$ from $\mathcal{P}_\mathcal{I}$ to be convolved with four orientations filter $G(\theta); \theta \in \{0°, 45°, 90°, 135°\}$. Total twelve maps are generated for analysis of orientations. In this way, all the early visual features are obtained.

3.2. Saliencies Fusion

Since different multi-scale feature maps have different visual modalities and dynamic ranges, it is always a hard problem to combine them into a unique saliency map. Obviously all the maps have un-equal influences. So we need determine their corresponding importance. Four different combining strategies are compared in [8]. $N(.)$ [1] would fail when handling multi equal strong maxima, while the complex iterative scheme [8] as well as PCA method [2] suffer from the intensive computations.

We try to solve this problem from the perspective of information theory. Ideally, each feature map should be able to represent a sparse distribution of a few conspicuous locations over the visual field. These conspicuous spots are taken as informative events. According to Shannon's information theory, the information conveyed by an event $\mathbf{x}$ is inversely proportional to the likelihood of $\mathbf{x}$ being observed.

$$I(\mathbf{x}) = -\log(p(\mathbf{x})) \tag{2}$$

To get the probability $p(\mathbf{x})$, we count the pixels with their values above a certain conspicuous threshold τ, for example 60% of the global maxium. The ratio with the number of total pixels yields the approximation of $p(\mathcal{M})$. In this way, we obtain $I(\mathcal{M})$ to reflect the importance of a corresponding feature map, shown as Eq.3.

$$p(\mathcal{M}) = \frac{C_{pass}(\mathcal{M}(i,j) > \tau)}{C_{all}(\mathcal{M}(i,j))}; \tag{3}$$

$$I(\mathcal{M}) = -\log(p(\mathcal{M})) \tag{4}$$

In our model, different fusion methods are given for different fusion actions. Considering their same modality, six center-surrounding difference maps of each $\mathcal{I}$, $\mathcal{RG}$, $\mathcal{BY}$ are summed up directly by across-scale addition $\oplus$, same for the three multi-scale maps of each orientation.

$$\mathcal{M}_l = \bigoplus_{c,s} \mathcal{M}_l(c, s); \quad \mathcal{O}_\theta = \bigoplus_{c} \mathcal{M}_\theta(c) \tag{5}$$

where $l \in \{\mathcal{I}, \mathcal{RG}, \mathcal{BY}\}$; $\theta \in \{0°, 45°, 90°, 135°\}$ and $c \in \{2, 3, 4\}$. Then we obtain seven feature maps: one intensity, two colors and four orientations. Actually, each of the

intensity and color maps implicitly has two separate positive and negative maps, which for the on-center and off-center effects. Then, they are combined into three *conspicuity maps*: $\bar{\mathcal{I}}$ for intensity, $\bar{\mathcal{C}}$ for color and $\bar{\mathcal{O}}$ for orientation. Before that, we give a weight function of Eq.6. The global maximum gives the amplitude while $I(\mathcal{M})$ obtained by Eq.3 represent the uniqueness of a map. We use both of them to weight each sub-map. The positive and negative parts are considered separately. Afterwards, *conspicuity maps* are combined according to Eq.7–9, where $\phi \in \{+, -\}$, $K \in \{\mathcal{RG}, \mathcal{BY}\}$ and $\theta \in \{0°, 45°, 90°, 135°\}$.They are further normalized to [0,1] to eliminate modality differences.

$$\mathbf{w}(\mathcal{M}) = I(\mathcal{M}) \max_{v(i,j)\in\mathcal{M}} \mid v(i,j) \mid \tag{6}$$

$$\bar{\mathcal{I}} = \sum\nolimits_{\phi} \mathbf{w}(\mathcal{M}_{\mathcal{I}\phi}) \mid \mathcal{M}_{\mathcal{I}\phi} \mid \tag{7}$$

$$\bar{\mathcal{C}} = \sum\nolimits_{K} \sum\nolimits_{\phi} \mathbf{w}(\mathcal{M}_{\mathcal{K}\phi}) \mid \mathcal{M}_{\mathcal{K}\phi} \mid \tag{8}$$

$$\bar{\mathcal{O}} = \sum\nolimits_{\theta} \mathbf{w}(\mathcal{O}_{\theta}) \times \mathcal{M}_{\theta} \tag{9}$$

Finally, We weight each conspicuity map according to Eq.6 and summed up them to the global saliency map $\mathcal{S}$.

$$S = I(\bar{\mathcal{I}})\bar{\mathcal{I}} + I(\bar{\mathcal{C}})\bar{\mathcal{C}} + I(\bar{\mathcal{O}})\bar{\mathcal{O}} \tag{10}$$

To some extend, our fusion method acts as global nonlinear amplifications like $N(.)$[1] but it is relatively robust and avoids the time-consuming search for every local maxima. Especially, Eq.3 can be efficiently implemented by the occlusion query of current GPUs. Furthermore, by splitting the feature map into blocks and analyzing the local statistics, we expect to obtain more accuracy and spatial-context adaptive results, which is left for our further work.

4. REALTIME IMPLEMENTATION ON GPU

In this section we describe in details how to compute the saliency map with the GPU. After giving the data representation and the processing steps, we present the implementation details for several important operators mentioned in section 3. Our implementation is built on OpenGL and the Cg shader language.

The multi-channel RGBA texture format and the SIMD vector operations of the GPU are fully utilized to accelerate intensive computations. The intensity and two color opponents are stored in the separate channels of a single RGB texture, while four orientation maps reside in one RGBA texture. In this way, all the channels in one texture can be processed in parallel and benefit from the GPU's vector operations.

The original image is first uploaded to the GPU as a 2D RGB texture. To generate the final saliency map, it will pass through the following stages.

The input image is first resized with its aspect ratio unchanged by drawing a quad in a specific size, typical 640x480, which aims to reduce computations for large images and build a complete image pyramid. Then, a rendering pass is activated for color space conversion from RGB to Lab. The result is stored in a multi-channel texture with the 16bit floating point format to ensure the precision. Then, the automatic mipmap texture generation is further utilized to build the image pyramid, which is based on bilinear filtering. The Gaussian and other filters can also be utilized to build the pyramid by the ping-pong technique (alternately rendering to and reading from a pair of textures) [9]. Here we choose the mipmap method due to its simplicity.

Afterwards, we apply a simple fragment shader to compute center-surrounding differences (Eq.1). After assigning texture coordinates for each vertex of the quad, the shader samples texels from the fine and coarse levels and outputs their differences. Benefitting from the automatic texture interpolation, we are no need to care about the scale difference. The level of the pyramid can be specified by setting the derivative parameters for Cg texture functions. For the following stage of across-scale additions (Eq.5), we enable the blending function to accumulate all the differences.

The Gabor filters are always computation intensive. The filter kernel size directly impacts the performance. Fortunately, four Gabor filter kernels can be precomputed and stored as a RGBA texture. Filters with the size of 5x5 are adopted in our implementation.

Finding the maximum and minimum values of each feature map (Eq.6) is a typical GPGPU operation, called as reduction. On GPUs, reductions can be performed by the ping-pong techniques in O(logn) steps [6]. For a two-dimensional reduction, the fragment program reads four elements from four quadrants fo the input texture, and the output size is halved in both dimensions at each step. In this work, the maxima and minima correspond to the positive and negative parts, respectively.

The computation for Eq.3 is implemented by the occlusion query. This hardware mechanism can count the number of the pixels that finally pass through the rendering pipeline. Since we have obtained the global maxima for each feature maps, we can design shaders to only allow pixels with their values above a threshold to pass through. By means of the queried number of passed pixels, we can efficiently finish the computation for $I(\mathcal{M})$(Eq.3). Finally, by sampling multi-textures and setting weights as uniform parameters of the fusion shaders, we combine all the maps together to yield the desired saliency map (Eq.10).

5. EXPERIMENTAL RESULTS AND DISCUSSION

To demonstrate the potential and efficiency of our GPU-based visual saliency model, we use several images and video sequences to evaluate its performance and further compare its output saliency map with Itti's VA model[1]. All the experiments were run on a 2.8G Pentium 4 with an Nvidia Geforce

6800GT.

We give a quantitative performance comparison between our proposed GPU-based method and the C++ implementation of Itti's model. Images with three different resolutions were tested. We need to mention that the time costs for our GPU-based implementation including the costs of uploading the image to the GPU and downloading the saliency map after computations. Tab.1 gives the results in the frame rate per second (fps), which indicates an order of magnitude improvement in computational speed. The results also show the data transfer between the CPU and GPU can greatly impact the performance.

Image Size	352x288	640x480	1280x720
Itti(CPU)	12.2 (fps)	2.5 (fps)	0.8 (fps)
Ours(GPU)	186.3 (fps)	122.5 (fps)	53.4 (fps)

Table 1. Performance for different size images

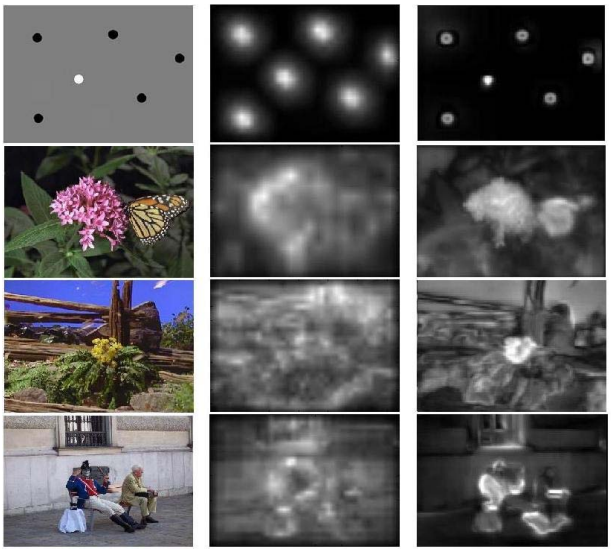

Fig. 1. Saliency maps. Left:original images; Middle:saliency maps given by Itti's model; Right: saliency maps generated by our proposed implementation.

Besides the performance, we present some examples of our saliency maps for subjective evaluations in Fig.1. The first image clearly demonstrates the effectiveness of handling the on-center/off-center responses separately. Itti's model can't determine the difference of white and block circles on a gray background. Other maps also prove the efficiency of our model.

6. CONCLUSION

In this paper, we propose a practical GPU-based visual saliency computational model.We extract early visual features in the *Lab* color space and keep the signed difference value to handle the separate on-center/off-center responses. Besides, a new feature map fusion method is presented based on the information theory. It can be implemented efficiently by occlusion query of the GPU. In our GPU-based implementation, besides the computation power of the programmable shaders, other resources on the GPU, such as interpolation, blending, mipmap generation, are fully utilized to accelerate our saliency model. Experimental results indicate an order of magnitude speedup over its CPU competitor. The output saliency maps further prove its effectiveness of catching salient regions in a image. We expect our work can help time-consuming applications based on saliency models. We also plan to add motion as an important saliency in our future work.

7. REFERENCES

[1] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.

[2] K. Rapantzikos and Y. Avrithis, "An enhanced spatiotemporal visual attention model for sports video analysis," in *International Workshop on content-based Multimedia indexing (CBMI)*, June 2005.

[3] Hector Yee, Sumanita Pattanaik, and Donald P. Greenberg, "Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments," *ACM Trans. Graph.*, vol. 20, no. 1, pp. 39–65, January 2001.

[4] Simone Frintrop, *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*, vol. 3899 of *Lecture Notes in Computer Science*, Springer, 2006.

[5] Cheng Wen-Huang, Wang Chia-Wei, and Wu Ja-jing, "Video adaptation for small display based on content recomposition," *IEEE Transaction on Circuits and Systems for Video Technology*, 2007.

[6] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krger, Aaron E. Lefohn, and Timothy J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, 2007.

[7] L. Itti and C. Koch, "Computational modeling of visual attention," *Nature Reviews Neuroscience*, vol. 2, no. 3, pp. 194–203, Mar 2001.

[8] Christof Koch and Laurent Itti, "Feature combination strategies for saliency-based visual attention systems," *Journal of electronic Imaging*, 2001.

[9] M. Strengert, M. Kraus, and T. Ertl, "Pyramid Methods in GPU-Based Image Processing," in *Workshop on Vision, Modelling, and Visualization VMV '06*, 2006, pp. 169–176.