UDC 519.68:681.513.7:591.169

Gaze Tracking Acceleration using CUDA Technology

E. A. Sibirtseva, I. M. Gostev

National Research University Higher School of Economic
Department of Computer Science
20, Myasnitskaya, Moscow, Russia, 101000

Low-cost gaze tracking systems are in great demand due to their wide range of application. Commonly, extra devices are needed (for instance, head mounted cameras); however, in this investigation gaze tracking is performed in real-time based on the video stream from an infrared video camera. A comparative analysis of the existing analogues was executed and the main features of gaze tracking systems were highlighted and prioritized. These features are price, tracking accuracy, angle error, flexibility, and usability.

A methodology was developed which allows to calculate a gaze direction vector according to the relative position of eye center and corneal reflection from an infrared diode. The centers of an eye and reflection are estimated using the vector field of image gradients and additional weighting. CUDA technology is used to accelerate the developed algorithms.

The main advantage of the developed algorithm is the ability to detect and continuously track pupils' centers, regardless of the head position, which significantly extends the scope of the gaze tracking system under consideration.

Key words and phrases: image processing, gaze-tracking, human-computer interaction, infrared illumination, CUDA, parallel computing; GPU; AHP.

1. Introduction

The first gaze tracking systems were built in the 1800s and investigated eye movement by using direct observation or special contact lens with a hole for pupil as in Edmund Huey research [1]. Non-intrusive eye trackers appeared later in 1922, when Guy Thomas Buswell presented his approach to this issue, which involved using beams of light that were reflected on an eye and then recording them on film [2]. Current studies into this field are mainly based on video recorded information and real-time analysis of gaze direction. Thus, eye tracking technology has become more accessible to a wide use in various spheres of life, such as advertising, marketing, product design, psychology, healthcare, and education.

At present, gaze tracking systems are closely related to the human-computer interaction process. Primarily, it is used to study user's attention on certain components of computer interface to increase its usability and make interaction more intuitive. Yet another recent area of research focuses on Web development, where the heat maps represent the statistics of user's visual behaviour for the purpose of more advantageous advertisement placement. However, the use of natural eye movements in gaze-aware and attentive systems is beyond the scope of the current research.

Other practical applications of eye trackers may be as an alternative way of computer user interface interaction to optimize specific tasks performance; for instance, text typing, words translation, web-surfing, or 3D modelling. Furthermore, it may help people with locomotor apparatus disabilities to communicate and interact with a computer; hence the social significance of the present research.

However, tracking in real-time has a great limitation due to the nature of eye movements. According to the significant amount of studies, such as Fischer and Ramsperger's in 1984 [3], eye movements are composed of rapid "jumps" from one object to another and fixation on them, long enough for the human brain to perceive the nature of the object. Such fixations last approximately 200–600 ms; while the saccadic movements between them last approximately 30–120 ms [4]. Saccades are ballistic movements intended for rapid visual scene scanning; in other words, once the saccade movement has been started, it cannot be interrupted, until it reaches its original destination. Moreover, an eye can perform smooth pursuit movements which

Received 18th September, 2014.

are responsible for smoothly following a moving target. Correspondingly, normal eye movements are composed of rapid saccades and occasional smooth pursuit of moving objects, alternated with fixations on the object of interest; which is typically the darkest areas of the image and its edges, as demonstrated on the fig. 1. Therefore, to track the eye movement it is needed to at least twice in 30 ms detects the gaze direction, while the saccade occurs. In other words, the calculation should be held at about 120 fps. And it is extremely fast. Even faster than you blink.

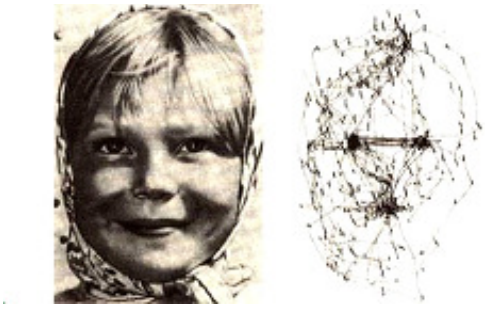


Figure 1. Trace of saccades of the human eye on a face while scanning [5]

The second limitation is related to the amount of data being processed. Every 8.3 ms a frame is read and should be processed (assuming its size 640 × 480 pixels). Almost all image processing operations need to pass at least once through all the pixels of the image, i.e. to make 307200 operations for each image processing, which naturally leads to the significant loss in performance. Fortunately, this problem has a solution — processing not only on CPU (Central Processing Unit) but also on GPU (Graphics Processing Unit).

GPU consists of thousands of cores for efficient handling of concurrent tasks. Thus each pixel is processed in a separate thread and the number of operations is reduced to three: transmit the memory to GPU, process and transmit it back to the CPU. However there are several drawbacks in GPU computing: such as long time to data transfer from CPU to GPU and back, arbitrary order of threads execution and threads simultaneous access to the same location of memory. More detailed information about how we solved these issues can be found in section 5.

The main aim of this investigation is to present a comprehensive study of gaze tracking process and accelerate it with CUDA [6] technology. The bulk of this investigation concentrates on studying video-based tracking of corneal reflection in multispectral illumination. Common analogues of the current gaze tracking systems are introduced and analyzed using AHP (Analytical Hierarchical Process) approach [7] in comparison with the developed tracker in section 2. Section 3 is devoted to the main methodology, where the applied devices are listed, the validity of using infrared illumination is demonstrated, and the conversion of pupil and corneal reflection relative position to the gaze direction vector is explained. After that algorithm of gaze tracking is described in section 4. Special attention is paid to the eye center localization, which is based on a very robust method of image gradients. Section 5 is about the developed system acceleration combining different approaches of CPU parallelization and CUDA computing. The results of the investigation are shown in the section 6, namely: tracker accuracy and robustness, comparison of sequential and parallel version of tracker, tracker accuracy with different input devices. In conclusion, future development is discussed and the ways of speed up and accuracy increase are provided.

2. Analogues Analysis Using AHP

The Analytical Hierarchy Process (AHP) is a structured technique for organizing and analyzing complex decisions, based on mathematics and psychology methods. At the first step the main criteria are highlighted according to which the comparison

would be made and the existing alternatives are presented. Thus the hierarchy of problem-criteria-alternatives is built. Then the pairwise comparison of alternative rated according to criteria is executed using techniques known as Overall Preference Matrix (OPM) and eigenvectors. The main purpose of this procedure is to estimate the most suitable option from the given alternatives.

2.1. Criteria

The following criteria were identified:

Price. As soon as gaze-tracking systems are designed for a narrow target audience, overprice is a common issue. And large-scale experiments may require a large number of such systems. Thus, the price is an important but not critical factor in the decision making.

Tracking accuracy. The tracking accuracy is percentage of correct eyes detection amongst all tracking results. Many gaze-tracking systems are based on infrared video tracking. However ambient light from lamps and the sun can interfere with the infrared diode and cause tracking errors. So tolerance to the ambient light is also considered (the higher tolerance the better). To track not only eye movement but also gaze direction the system should be able to track pupils. And for the given problem this criteria is crucial.

Angle error. Gaze direction is calculated as a vector from user's pupils to his point of interest. However, high accuracy in this criterion is based on calculation performance and video resolution. Angle errors can cause incorrect result, while having the best tracking accuracy. So the idea is that with such errors the system does not understand precisely where the user looks.

Flexibility. This tracking system is supposed to be used in investigation, so some addition functions may become needed during the research. This criterion includes API availability or additional applications existence. To carry out all the experiment some software extensions may be needed, thus flexibility is significant for the problem solving.

Usability. In this case usability includes set-up time and portability. Since the experiments may take long time some non-contact systems (cameras or screens) are highly preferable than contact ones (glasses of helmets). Portability is crucial for outdoor experiments.

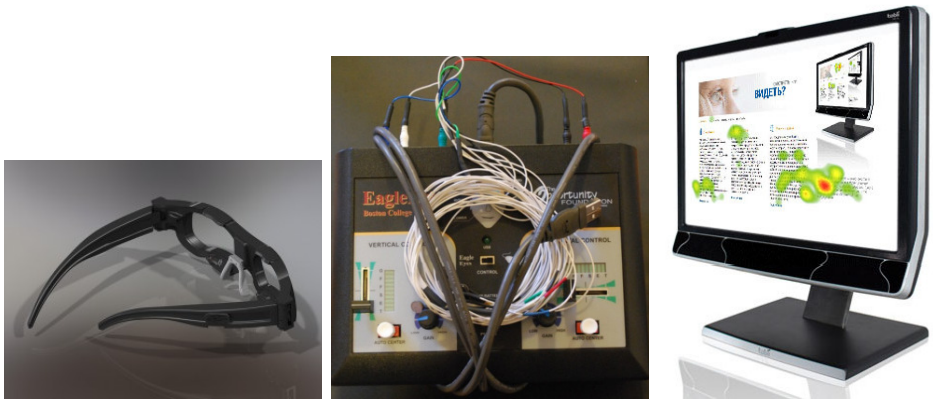
2.2. Alternatives

The following alternatives were selected as the most widely spread gaze-tracking systems.

The EC8™System. The market leader in the sphere of contact gaze tracking systems is The EC8™System (see Fig. 2(a)) which represents special glasses with frame-mounted micro cameras with infrared diodes [8]. These glasses are a great success in order to their portability and ability to be used not only to interact with a computer, but also to analyze drivers' behavior or customers' selection of a product in the store. The technique of bright pupil is applied to pupil detection; the advantage of this technique is its high tolerance to different lighting conditions, nonetheless it is not efficient for outdoors experiments, since extraneous infrared illumination interferes with monitoring.

EagleEyes. EagleEyes tracker was designed for people with locomotor disabilities [9]. Tracker consists of a main console, in which electrodes are plugged in, -volt batteries power unit, electrode cables and USB cable for PC connection (see Fig. 2(b)). Unlike all other gaze-trackers, this is not based on a video stream and infrared illumination, but on electrooculography. Electrodes, attached to the user's head measure an electric signal which is sent when the eyeball is rotated.

Distinct advantage of electrooculography approach over an infrared video is total independence from external lighting and the ability to use glasses or contact lenses during gaze-tracking. Nevertheless, this technology is considered outdated and is



a b c

Figure 2. The EC8™System tracker (a), EagleEyes tracker (b), Tobii T60XL System tracker (c)

not used anywhere except medical institutions. Particularly difficult is caused by initial device set-up and correct positioning of the electrodes. It is impossible to use EagleEyes without an assistant even after long training.

Tobii T60XL System. Tobii T60XL System (see Fig. 2(c)) is the monopolist of the non-contact trackers field and contains both hardware and software parts [10]. Tobii's hardware consists of a 24 inch monitor with infrared mounted cameras and diodes. Moreover, Tobii's software offers a wide variety of services, which embraces a comprehensive analysis of the user's attention and eye-based computer interaction. This product's popularity relates to the comparatively accurate tracking results, which became possible in order to the use of certain techniques. One of them is based on the dark pupil effect, which occurs when the lighting source is offset from the optical path, and the reflection is projected away from the camera, making the pupil appear black. The second technique resolves issues with the point of gaze direction drifting and instead of monocular tracking, relies on binocular tracking with averaging results from both eyes.

Inner Development. An alternative solution is suggested: a custom designed gaze-tracking system made as the result of the current investigation. Unlike other systems it does not require additional devices, except for infrared diode. It consists of tracking software, which can both — output heat map of user's gaze direction and allow a user to control a computer by gaze. But the main advantage of this system is its flexibility. It can be extended to output any gaze-tracking information the user needs.

2.3. Analytic Hierarchy Process

AHP begins with the definition of criteria weights in pairwise comparison. This weights show the importance of certain criteria for decision making in comparison to another. The value of weights is estimated according to Saaty scale [11] in the range from 1 to 9.

As the result of discussion was formulated the Overall Preference Matrix (see Tab. 1).

The matrix is normalized and then the eigenvector is calculated (see Tab. 2). It can be interpreted, that the Accuracy and Flexibility have the highest 36% influence on the whole decision on gaze-tracking system choice.

Before moving to the next stage, the Overall Preference Matrix should be validated on consistency. So the Consistency Rate is estimated and is equal to 4,3%. Since it is lower than 10%, the matrix is consistent.

Table 1

Criteria Overall Preference Matrix and Criteria Eigenvector

	Price	**Acc.**	**Error**	**Flex.**	**Usability**
Price	1	0.2	0.333	0.2	3
Acc.	5	1	3	1	7
Error	3	0.333	1	0.333	5
Flex.	5	1	3	1	7
Usability	0.333	0.143	0.2	0.143	1

Table 2

Criteria Eigenvector

	Eigenvector
Price	7.878 %
Accuracy	35.977 %
Error	16.167 %
Flexibility	35.977 %
Usability	4.003 %

At the second stage for each criterion a matrix of pairwise alternatives comparison is constructed. In the current case there are 5 preference matrices. All matrices are validated and are consistent. After that eigenvectors of every matrix is calculated and added to the final matrix with all eigenvectors (e.g. Tab. 3).

Table 3

Summary eigenvector matrix and

	Price	**Accuracy**	**Error**	**Flexibility**	**Usability**
EC8	0.122	0.122	0.106	0.078	0.143
EE	0.057	0.057	0.052	0.078	0.054
Tobii	0.263	0.558	0.421	0.196	0.401
Inner	0.558	0.263	0.421	0.647	0.401

Finally, the matrix (see Tab. 6) is multiplied by previously calculated criteria eigenvector. As a result we have a vector (see Tab. 7 with values that represent each alternative. So the maximum value of this vector is the most suitable option for the current problem. In our case it is House developed application (Inner) with the value 45.6%.

2.4. AHP Results

The obvious loser electrooculography EagleEye system requires a lot of efforts while set-up, calibration and user training and at the same time has the lowest tracking accuracy. It cannot be used in large-scale experiments, but it is still suitable for individual usage to people with locomotor disabilities as a way to interact with a computer.

The glasses EC8 could be a quite good choice, since they can be used not only with computer but also with every environment, for example, at class or while reading textbook. However its price and lack of flexibility keeps from choosing it.

Table 4

Result Eigenvector

	Eigenvector
Price	7.878 %
Accuracy	35.977 %
Error	16.167 %
Flexibility	35.977 %
Usability	4.003 %

Table 5

Eigenvector

	Eigenvector
EC8	0.105
EE	0.064
Tobii	0.376
Inner	0.456

So, according to the calculations made above, the most suitable solution is to use the currently developed gaze tracking system, which can be dynamically extended to the researchers' needs. This system slightly loses to Tobbii screen in accuracy and angle errors, but is more convenient to the current requirements. The main issue of Tobbii system is its lack of available API and additional software. While with inner development there is no such issue.

Taking everything into consideration, it is obvious that there is no system available for gaze tracking, which enables computer control using eye movements without the use of additional devices, which confirms the topicality of this system development.

3. Methodology. Overview

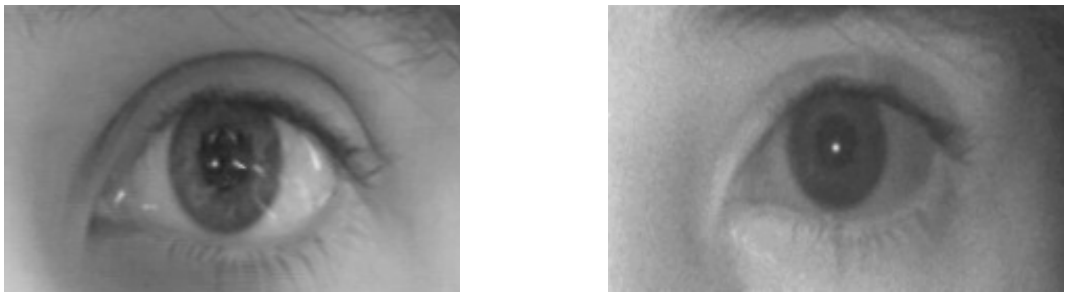
In order to calculate where the person looks, it is needed to have some sort of a static point on the image, which position is constant. It is performed to ensure that small head movements could not affect the tracking accuracy, because after head movement the gaze directed at the same point has a different offset vector than the previous one.

The corneal reflection is assumed as such a point and relatively to it eye movements are tracked. However, a great issue arises: depending on the lighting, reflections in the eye may change during tracking as snowing on Fig. 3(a), and this will lead to irrelevant tracking results. To avoid this, we use an infrared diode and camera, which filter daylight and lets only infrared light. Eye in infrared light, demonstrated on Fig. 3(b) has a clear reflection from the infrared diode, regardless of external lighting.

The main benefits of using infrared illumination are:

1) reduction of sensitivity to changes in illumination (radiation from sun or lamps are cut by an infrared filter) and, as a consequence, increase of the tracking quality;
2) unambiguous definition of a single corneal reflection from the infrared diode. Since the diode is fixed in space, then change of its location relative to the pupil allows us to calculate the gaze direction vector (see Fig. 4).

In the course of this study a built-in webcam, an HD video camera with IR filter, and Kinect v.2 were tested as input devices for gaze tracking.



a b

Figure 3. Eye in (a) daylight, (b) infrared illumination

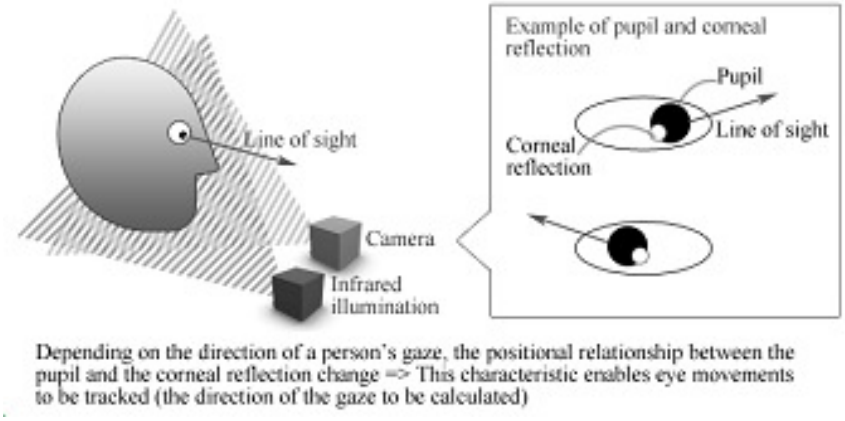


Figure 4. Graphical representation of infrared illumination approach

4. Gaze Tracking Algorithm

On account of the current research innovativeness, there is no foregone solution; thus variety of approaches is combined to achieve the most accurate results. Gaze tracking is an extremely complex issue, so it is indispensable to divide it into several steps (see Fig. 5(a)):

1) find eye region;
2) find eye and reflection centers;
3) calibrate and estimate gaze vector.

4.1. Eye Region Localization

First, real-time video capture is performed in MPEG unpacked format and converted to a single-frame representation. Initially the frame is represented in 8-bit 4-channel array of pixels, each channel associated with corresponding of red, green, blue or alpha value. In order to reduce the amount of computations, the original image is converted to 8-bit grayscale image, which contains only brightness values in the range [0, 255], where 0 corresponds to black and 255 — to white. In the same time the image is also mirrored. These two operations are done on GPU simultaneously.

Then the face region is detected using Haar Cascade [4] algorithm provided by OpenCV image processing library [12] accelerated on GPU and takes about 15 ms to execute. When the face location is known, we can detect eye region. And here is a trick. There are two ways to manage it: use Haar Cascade one more time but trained to detect eyes, or apply to the common knowledge of human anatomy.

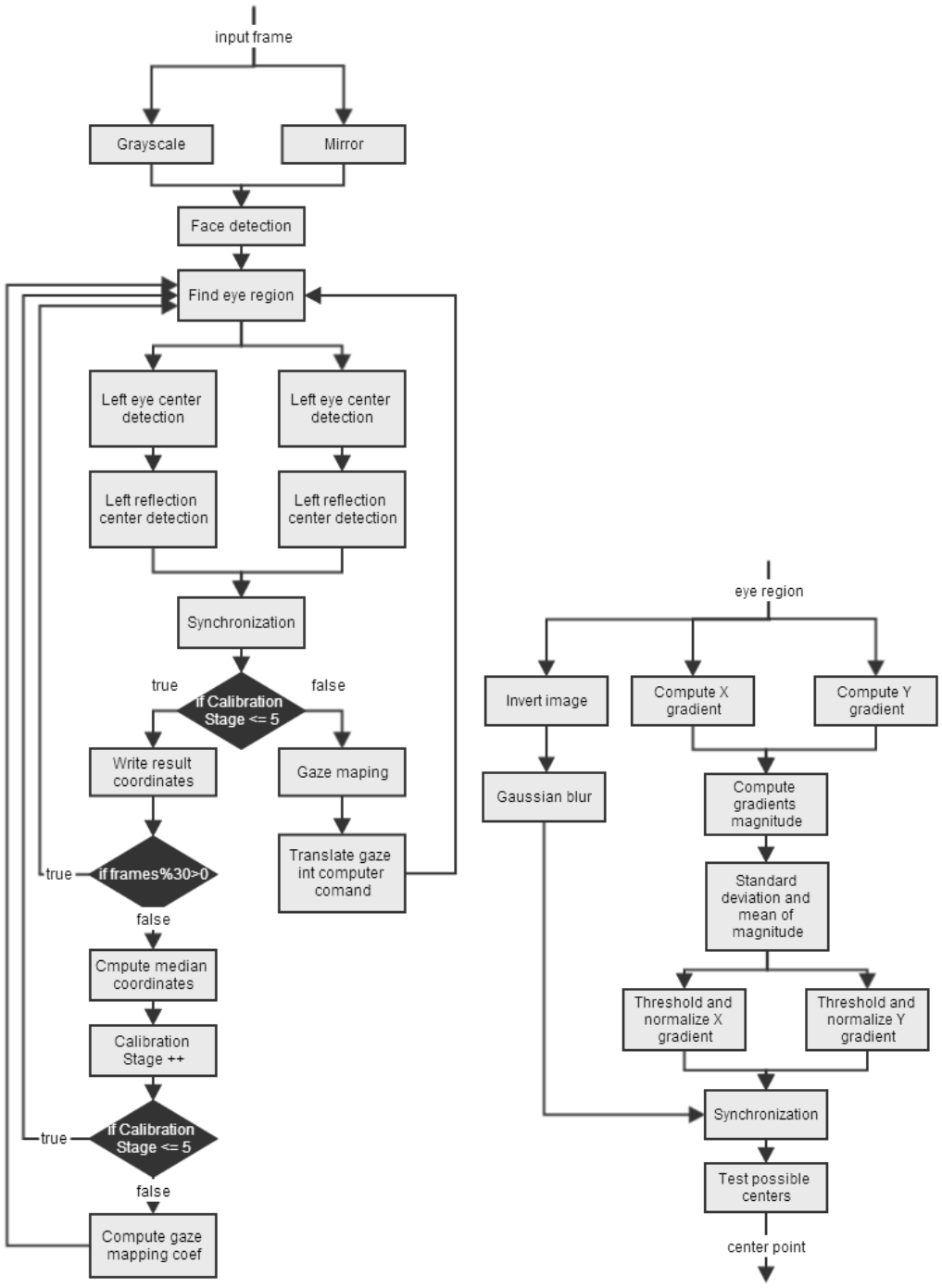


Figure 5. General algorithm flowchart (a) and Eye center localization algorithm flowchart (b)

Human anatomy is highly investigated since Leonardo da Vinci times and can be used to instantly eliminate eye region with great precision. Based on anatomical features of the human face, it is assumed that the eye is located on the top one-third of the face. Its width and height are 30% and 25% of the face's size correspondingly. Location relative to the side of the face is 15% of the face's width. All the proportions are shown on the Fig. 6.

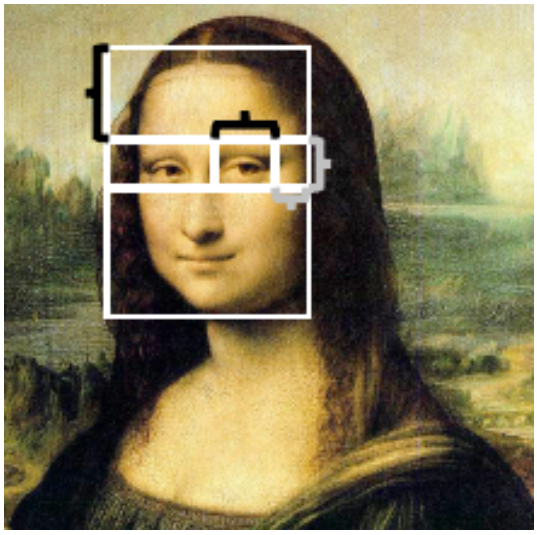


Figure 6. Face proportions

4.2. Eye Center Localization

The next step of our algorithm is to get the eye center from the detected eye region. For both eyes all the following steps execute in parallel, so only the process of left eye center detection is discussed (see Fig. 5(b)).

In productivity purposes the eye image is scaled to have width of 50 pixels.

Geometrically, the center of a circular object can be detected by analyzing the vector field of image gradients (see Fig. 7).

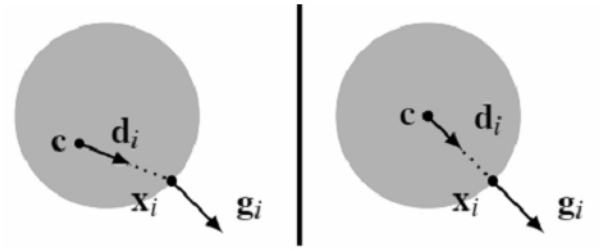


Figure 7. Artificial example with a dark circle on a light background, similar to the iris and the sclera. On the left the displacement vector d_i and the gradient vector g_i do not have the same orientation, whereas on the right both orientations are equal

In our case the pupil is a dark circle on a relatively bright sclera, so to find its center we need to analyze the vector field of image gradients. In order to obtain the image gradients, we compute in parallel the partial derivatives

$$g_i = \left(\partial I\left(x_i, y_i\right) / \partial x_i, \partial I\left(x_i, y_i\right) / \partial y_i \right)^T, \tag{1}$$

but other methods for computing image gradients will not change the behavior of the objective function significantly.

However, to decrease computational complexity only gradient vectors with a significant magnitude are considered. So gradients magnitude is computed and all remaining values are normalized, then a dynamic threshold is estimated, according to the formula:

```
threshold = 0.3 * std + mean
```

where `std` is the standard deviation of the magnitude matrix and `mean` is the mean value of it.

The relationship between a possible eye center and the orientation of all image gradients. Let point $\boldsymbol{c}$ be a possible center and g_i the gradient vector at position x_i. Then, the normalized displacement vector d_i should have the same orientation (except for the sign) as the gradient g_i (see Fig. 7). If we use the vector field of gradients, we can exploit this vector field by computing the dot products between the normalized displacement vectors (related to a fixed center) and the gradient vectors g_i. The optimal center c^* of a circular object in an image with pixel positions x_i $i \in \overline{1, N}$, is then given by

$$c^* = \arg\max \left\{ \frac{1}{N} \sum_{i=1}^{N} \left(d_i^T g_i \right)^2 \right\}, \tag{2}$$

$$d_i = \frac{x_i - c}{\sqrt{|x_i - c|}}. \tag{3}$$

The displacement vectors d_i are scaled to unit length in order to obtain an equal weight for all pixel positions. Dot products are negative if the vectors are pointing in opposite directions. The gradient function used creates vectors that always point towards the lighter region. Since the iris is darker than the sclera (white part) the vectors of the iris edge always point out. This means that at the center they will be facing in the same direction as the d vector. Anything pointing in the opposite direction is irrelevant. So if the dot product is negative it is set to zero.

Nevertheless, under some conditions, the maximum is not well defined, or there are local maxima that lead to wrong center estimates. For example, dominant eyelids and eyelashes or wrinkles in combination with a low contrast between iris and sclera can lead to wrong estimates. Therefore, prior knowledge about the eye is used in order to increase robustness. Since the pupil is usually dark compared to sclera and skin, a weight $\mathrm{w_c}$ for each possible center c such that dark centers are more likely than bright centers is applied. Integrating this into the objective function leads to:

$$\arg\max \left\{ \frac{1}{N} \mathrm{w}_c \sum_{i=1}^{N} \left(d_i^T g_i \right)^2 \right\}, \tag{4}$$

where $\mathrm{w}_c = I^*\left(c_x; c_y\right)$ is the grey value at $(c_x;\, x_y)$ of the smoothed and inverted input image I^*.

So while gradients of image are calculated and normalized, the weighting matrix is computed. The weighting matrix is an inverted image of an eye, smoothed by Gaussian filter, in order to avoid problems that arise due to bright outliers such as reflections of glasses. And then all possible centers are tested, which returns us the final result – the eye center.

When the centers are found the completely similar algorithm is applied to evaluate the corneal reflection. Knowing eye centers, it is possible to detect the sclera region more precisely and decrease computational complexity once more. The only difference in reflection detection is that now we have to invert the image first and then proceed the same algorithm as for eye center.

4.3. Calibration and Gaze Vector Estimation

Given the relative position between pupil and glint, the screen actual coordinates of the gaze can be determined via a linear mapping procedure. The conventional approach for gaze mapping only uses coordinates displacement of pupil center and glint position [35, 36] as a pupil–glint vector. The main drawback with this method is that the subject must keep his or her head stationary, or the glint position in the image will change.

In practice, it is difficult to keep head still and the existing gaze tracking methods will produce incorrect result if the head moves, even slightly. Head movement must, therefore, be incorporated in the gaze estimation procedure. In this section, we introduce a new gaze estimation procedure that tolerates slight translational head movement.

According to our mapping procedure, the pupil–glint vector for X coordinate (for Y the mapping goes the same way) is represented by

$$\mathrm{map}_x = [\Delta x, g_x, 1],$$

where Δx is the pupil–glint displacement g_x is the glint image X coordinate. Unlike the existing methods which only uses Δx and Δy, this mapping procedure also includes the glint position. This effectively reduces the head movement influence. The coefficient vector c is represented by

$$\mathrm{coef}_x = [\alpha, \beta, \varepsilon]^T.$$

The pupil–glint vector measured during runtime can be mapped to the image screen locations through the following equation:

$$\mathrm{scr}_x = \mathrm{map}_x * \mathrm{coef}_x = \alpha \Delta x + \beta g_x + \varepsilon,$$

where scr_x is the calibrating point x coordinate. The coefficients α, β, ε are determined via a simple calibration procedure as discussed below.

The calibration procedure is very simple and brief. The user is sequentially given 5 pre-prepared points on the screen and should fixate his or her gaze at them for about 1 second. On each fixation, 30 sets of pupil-glint vectors are computed and stored for each coordinate. Only the median value passes to the pupil glint matrix 5×3 A_x. The transformation from pupil–glint matrix A to the target vector B_x, which contains x coordinates of pre-prepared calibration points, is given by

$$A_x * \mathrm{coef}_x = B_x. \tag{5}$$

Coefficient vector coef_x is acquired by solving equation (5) using the following formula

$$\mathrm{coef}_x = A_x^T * B_x. \tag{6}$$

5. Parallelization on CPU and GPU

According to the Table 8 in the results section, it is obvious that CPU parallelization and GPU computing made this tracker real-time and not a post-processing one. Such significant gain in efficiency is related to some parallelization techniques.

The first thing to parallelize on CPU are the processes that do not have access to the same data and run completely independent. In our case these are left and right eyes center localization, gradients x and y calculation, left and right reflections detection, and weighting matrix evaluation.

As for GPU, first of all, it is crucial to understand that GPU is not a panacea. There is a well-known joke that CPU is a sport car (just for two passengers, but very fast) and GPU is a school-bus (a lot of passengers, but slower). In other words, if you need to process small amount of data, use CPU. If the data amount is huge, use GPU. This happens due to time-consuming operation of read/write to GPU memory.

So only high loaded operations were transferred to GPU, the most costly of which is testing possible centers.

5.1. Testing possible centers on GPU

Precise estimation of eye center has complexity $O\left(n^2\right)$, which in our case means more than 6250000 operations for each eye. Centers testing takes 20 ms in sequential execution.

The idea of centers testing is the following: for each pixel its partial gradients x and y are compared to every other gradient x and y on the same image using dot product estimations. If the dot product is added to the new matrix of dot products. Then the maximum value from this matrix is the eye center. The most significant obstacle is that, since threads run this procedure simultaneously, each of them overwrites its result in the dot product matrix. The only solution is to make each thread write to its one new matrix and then sum all matrix into one final.

However, knowing that every write to the CPU operation costs around 0.6 ms, we could not afford to write a new matrix for each pixel (which would cost $2500 \cdot 0.6$ ms = 1500 ms). So we create an aggregative matrix with width and height set to pixels number of the image and execute the CUDA kernel according to this matrix and not the original one. In this kernel dot products of each pixel are written to the corresponding row of aggregation matrix. After that all rows should be summed into the result matrix. Nevertheless, this operation is not so trivial due to the threads race for resources, so the parallel reduction is executed for each column.

5.2. Reduction

As well known CUDA technology has 4 memory types:

1) read/write per-thread *registers*;
2) read/write per-thread *local memory*;
3) read/write per-block *shared memory*;
4) read/write per-grid *global memory*.

The first one is the fastest but has access only to one thread data, while the last one is shared data with CPU and is the slowest. In our case the most interesting type is shared memory. It is faster than global in 100 times and has access to resources of the whole block of threads.

The reduction operation is executed as follows: each block is divided into two parts on each step. Then the sum of corresponding values is written to the shared memory until only one thread remains. After that first values from each block are written to the shared memory and reduction operation executes one last time.

Assuming that each thread starts with one value, the approach of parallel reduction is to first add the values within each thread block, to form a partial sum array, which is written to the shared memory until only one thread remains. After that first values from each block are written to the shared memory and reduction operation executes one last time, in other words, the values from the partial sum array are added together from all of the blocks using the same parallel reduction algorithm. The same code is executed by each thread in parallel and CUDA kernels are invocated iteratively. Sequential addressing is used instead of a tree structure to accelerate the processing time of the last step – reduction of partial sums. This approach allows to boost speed up to 2.5 times and is completely conflicts free.

5.3. Processes Overlap

All CUDA operations consist of three parts: transfer data to GPU, execute CUDA kernel, and transfer data back to CPU. The CUDA technology provides an optimization technique known as Overlapping Kernel Execution and Data Transfers. So while one kernel is executed, the next can transfer data to GPU, and the next after next – transfer data to CPU. This ability is extremely helpful, since it makes possible to

track gaze direction at speed 120 fps, while the processing per frame is 17 ms, but processing of one frame can overlap with another, giving us desired speed of 120 fps and lag only in 1–2 frames.

6. Results

The result of the investigation is a robust gaze tracking system, accelerated by means of CUDA technology and parallel execution on 4-cores CPU. Processing speed is 120 fps, which satisfies the conditions for this technology application whether for controlling the computer, or for carrying out neuropsychological experiments.

Above (see Tab. 6) average execution time for each frame and average fps speed are measured in different versions of the system. In the first version sequential algorithm for gaze tracking was implemented. In the second version most complex calculations were transmitted to GPU. In the final version parallel execution of CPU and processes overlap on the GPU was implemented. As it can be seen from the table, the primary changes gave a huge boost in performance (x16 times). The following system improvements led to a threefold reduction in execution time, but more importantly, gained the desired 120 fps speed.

Table 6

Speed test of different versions of the system

	only CPU	**CPU + GPU**	**parallel CPU + overlap GPU**
Average time execution per frame	1147.86 ms	68.32 ms	17.03 ms
Average fps speed	0.9 fps	21 fps	120 fps

Further, in the final version of the system the average execution time of each function is measured to identify bottlenecks and understand what needs to be improved (see Tab. 7).

Table 7

Average functions execution time

Functions	**Average execution time**
Face detection with Haar Cascade	15 ms
Data transfer from GPU to CPU and back (x2)	1.2 ms
Test possible eye centers	0.9 ms
Mirror image and grayscale	0.002 ms
Calculate gradient X and Y	0.002 ms
Calculate magnitude	0.001 ms
Calculate standard deviation and mean value	0.001 ms
Calculate max value of the matrix	0.001 ms
Find eye region	0.00001 ms

According to the Table 7, the most expensive operation is face detection with Haar Cascade. So it makes sense to try another method, which can be easily parallelized on the GPU and is just as accurate as Haar Cascade.

The final carried out experiment was about the use of different input devices for tracking, namely, web-cam, HD camera with IR filter, and Kinect v.2 [13]. The Table 8 illustrates tracking accuracy measurements. Accuracy is measured during 10 minutes of tracking is the percentage of correctly detected gaze direction to the total number of frames.

Table 8

Accuracy test with different input devices

Input device	**web-cam.**	**HD cam. with filter**	**Kinect v.2**
Tracking accuracy	33.5%	59.2%	88.6%

Such a low accuracy of a webcam is associated with the fact that it detects a lot of reflections on the eye. Though, these reflections tend to vary depending on the ambient light, which causes such dramatic accuracy loss. Not so high accuracy of the HD camera derives from the fact that the filter adds a strong grain in the image and to determine anything, it is necessary to smooth the image with Gaussian filter. However, this blur operation virtually eliminates the corneal reflection.

The best results has Kinect v.2, as it has originally built-in infrared camera and two powerful infrared lamps, which give complete independence from external lighting, so you can work in total darkness (tested in practice). But Kinect v.2 has two drawbacks that do not allow to gain greater accuracy. The first of drawback is its low resolution (640×480), which gives only the approximate location of the relative position of the reflection to the pupil. Second drawback is its too powerful infrared lamps, which in extreme close distance from the face illuminate it in such a manner that only a white blob remains, thus the face cannot be detected.

7. Conclusions

Further development of this application can be led in different directions, namely, the recognition accuracy increase or the commands semantics extension. In the first case, an increase of computing power and the use of specialized infrared ophthalmological cameras can provide outstanding results. Developing more parallelizable algorithm for face detection is now the highest priority to accelerate the process execution. Moreover, use of more up-to-date hardware can provide wide opportunity to acceleration. The state of art Nvidia graphical processors makes it possible to run CUDA kernels inside other CUDA kernels also known as dynamic parallelism, which spare processors to perform so many transfer data operations.

In the second case the novel tendency in the human-computer interaction as non-command interfaces is assumed to be dramatically evolved, due to the fact of their more natural approach. For instance, gaze direction may be used to select an object on the screen, arm movements to manipulate 3D objects, furthermore, the user's face disappearance in front of the monitor to turn off the computer.

However, the gaze tracking data can be utilized not only to control the computer but also to interact with other technical devices. With the help of additional equipment it is possible to connect, for example, gaze tracking device and the wheelchair for people with locomotor disabilities in order to provide them with ability to control their wheelchair movement by gaze direction. Taking everything into consideration, certainly, the theme of gaze tracking is not confined only to the analysis of products preferences in a store or the decision-making where to place a profitable banner on the web page, but it can also solve urgent social issues; moreover, probably in the near future it may become an alternative to conventional input device like mouse and keyboard.

Table 9

Alternatives comparison table

	The EC8™System	**EagleEyes**	**Tobii T60XL System**	**Inner development**
Method	video-based	electrooculography	video-based	video-based
Pupil search	light pupil	-	dark pupil	dark pupil
Mono-/ binocular	binocular	-	monocular	binocular
Tolerance to ambient light	normal	-	low	high
Additional software	-	-	14	3
Set-up time	¡ 5 min.	10-50 min.	10 sec.	3-5 sec.
Session time	60 min.	20 min.	unlimited	unlimited
Computer control	+	+	-	+
Blink detection	+	-	+	+
Head movements freedom	unlimited	20 x 15 x 20 sm	30 x 15 x 20 sm	unlimited
Tracking accuracy	86%	70%	91%	89%
Angle error	1.2°	2.9°	0.7°	1.0°
Calibration	9 points	-	5 points	4, 8, 16 points
Portability	+	-	-	+
API	-	-	-	+
Price	$ 45,000	absent on the open market	$ 17,760	free

References

1. B. Fischer, E. Ramsperger, Human Express Saccades: Extremely Short Reaction Times of Goal Directed Eye Movements, Experimental Brain Research 57 (1) (1984) 191–195.
2. G. T. Buswell, Fundamental Reading Habits, Supplementary Educational Monographs (21).
3. K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, D. Weiskopf, Real-Time Volume Graphics, Ak Peters Natick, 2006.
4. T. N. Cornsweet, Visual Perception, Academic, New York, 1970.
5. R. C. Gonzalez, R. E. Woods, Digital Image Processing, Prentice Hall Upper Saddle River, New York, 2002.
6. Nvidia, cuda, accessed: 29.05.2014.
   URL `http://www.nvidia.com/object/cuda_home_new.html`
7. T. L. Saaty, K. Peniwati, Group Decision Making: Drawing Out and Reconciling Differences, RWS Publications, 2008.
8. Eye-Com Corporation, accessed: 16.02.2014.
   URL `http://eyecomcorp.com`
9. EagleEyes. The Opportunity Foundation of America, accessed: 16.02.2014.
   URL `http://www.opportunityfoundationofamerica.org/`
10. Tobii Eye Tracking Research, accessed: 16.02.2014.
    URL `http://www.tobii.com/en/eye-tracking-research/global/`
11. T. L. Saaty, Revista de la real academia de ciencias exactas, fisicas y naturales, Serie A: Matematicas 102 (2) (2008) 251–318.
12. Opencv, accessed: 29.05.2014.
    URL `http://opencv.org/`
13. Microsoft, accessed: 29.05.2014.
    URL `http://www.microsoft.com/en-us/kinectforwindows`

УДК 519.68:681.513.7:591.169

Повышение быстродействия системы слежения за взглядом на основе CUDA технологии

Е. А. Сибирцева, И. М. Гостев

Национальный исследовательский университет «Высшая школа экономики»
Департамент компьютерных наук
ул. Мясницкая, д. 20, Москва, Россия, 101000

В нынешнее время бюджетные системы слежения за взглядом пользуются большим спросом в связи с широким сектором их применения, как в промышленности, так и в персональном использовании. Как правило, для наблюдения за взглядом человека необходимы дополнительные устройства (например, носимые на голове камеры), однако в данном исследовании отслеживание взгляда происходит в реальном времени и основано на входном видеопотоке с инфракрасной камеры. Для выяснения актуальности данной разработки был проведён сравнительный анализ существующих аналогов и выделены основные характеристики систем слежения за взглядом. Данными характеристиками являются цена, точность слежения, угловая ошибка, гибкость системы и удобство использования.

Была разработана методика, которая позволяет производить расчёт вектора направления взгляда исходя из взаимного расположения центра зрачка и блика на роговице глаза от инфракрасного диода. Центры зрачков и бликов вычисляются, используя векторное поле градиентов исходного изображения и дополнительную матрицу весов. Технология CUDA применяется для ускорения работы данного алгоритма.

Основное преимущество разработанного алгоритма заключается в том, что изменение положения головы не влияет на обнаружение зрачка и слежение за взглядом, что значительно расширяет область применения данной системы слежения за взглядом.

Ключевые слова: обработка изображений, слежение за взглядом, человеко-машинное взаимодействие, инфракрасная подсветка, CUDA, параллельные вычисления, GPU; AHP.

Литература

1. *Fischer B., Ramsperger E.* Human Express Saccades: Extremely Short Reaction Times of Goal Directed Eye Movements // Experimental Brain Research. — 1984. — Vol. 57, No 1. — Pp. 191–195.
2. *Buswell G. T.* Fundamental Reading Habits // Supplementary Educational Monographs. — 1922. — No 21.
3. Real-Time Volume Graphics / K. Engel, M. Hadwiger, J. M. Kniss et al. — Ak Peters Natick, 2006. — Pp. 112–114.
4. *Cornsweet T. N.* Visual Perception. — New York: Academic, 1970.
5. *Gonzalez R. C., Woods R. E.* Digital Image Processing. — New York: Prentice Hall Upper Saddle River, 2002.
6. NVIDIA, CUDA. — Accessed: 29.05.2014. `http://www.nvidia.com/object/cuda_home_new.html`, accessed: 29.05.2014.
7. *Saaty T. L., Peniwati K.* Group Decision Making: Drawing Out and Reconciling Differences. — RWS Publications, 2008.
8. Eye-Com Corporation. — Accessed: 16.02.2014. `http://eyecomcorp.com`, accessed: 16.02.2014.
9. EagleEyes. The Opportunity Foundation of America. — Accessed: 16.02.2014. `http://www.opportunityfoundationofamerica.org/`, accessed: 16.02.2014.
10. Tobii Eye Tracking Research. — Accessed: 16.02.2014. `http://www.tobii.com/en/eye-tracking-research/global/`, accessed: 16.02.2014.
11. *Saaty T. L.* Revista de la real academia de ciencias exactas, fisicas y naturales // Serie A: Matematicas. — 2008. — Vol. 102, No 2. — Pp. 251–318.
12. OpenCV. — Accessed: 29.05.2014. `http://opencv.org/`, accessed: 29.05.2014.
13. Microsoft. — Accessed: 29.05.2014. `http://www.microsoft.com/en-us/kinectforwindows`, accessed: 29.05.2014.