Template Matching-Based Eye Tracking Technique With FPGA for Foveated Rendering

Gabriel Ayres de Oliveira
Faculdade de Engenharia
Universidade Federal de Juiz de Fora
Juiz de Fora, Brasil
gabriel.oliveira@engenharia.ufjf.br

Estêvão Coelho Teixeira
Faculdade de Engenharia
Universidade Federal de Juiz de Fora
Juiz de Fora, Brasil
estevao.teixeira@ufjf.edu.br

***Abstract*—Eye tracking using foveated rendering is needed to reduce the overhead of graphics processing on host computers. In an attempt to accelerate the process of pupil finding, we focus on using a template matching technique to estimate the user's gaze position. The implementation is done on specialized hardware apart from the host machine, in this case an FPGA in an effort to offload these calculations to a dedicated hardware. The results show a slow but accurate process which uses a very small number of logic units.**

***Index Terms*—eye tracking, template matching, foveated rendering, FPGA**

I. Introduction

Eye tracking has many applications, such as studying how the human gaze reads a wall of text or an image, or how users navigate a computer program interface [1]. Lately, however, eye-tracking has been used in virtual reality headsets, which have been advancing rapidly in the past few years. One of the most prominent advances in the field has been on increased integrated display's pixel density. This is in order to make the output as indistinguishable as possible to what the human eye would see. Given how graphic processing unit (GPU) advancements have been lagging behind, those devices are hindered by a hardware that cannot render frames as fast as needed. Foveated rendering then comes into play by generating those frames with areas of higher and lower definition, mimicking the physiology of the human eye which increases the process' efficiency. Eye tracking goes hand-in-hand with foveated rendering as the eyes rarely have their gaze fixed in one point only.

There are different ways to track a user's gaze. Two examples are Electro-OculoGraphy, which needs physical contact to the user's skin, and Video-OculoGraphy which captures the eye movement by using a camera [2].The Video-OculoGraphy technique was chosen for the present investigation, implementing it as an FPGA paired with a camera, and using a template matching [3] technique to locate the pupil. The method present here differs from previous publications, as those use different ways to track the user gaze (e.g. segmenting the pupil using binary images and ellipse fitting) [4]. The method also differs from other published works who also apply several mathematical transformations as to make the pupil easier distinguishable on the expense of a considerable amount of logic units [5].

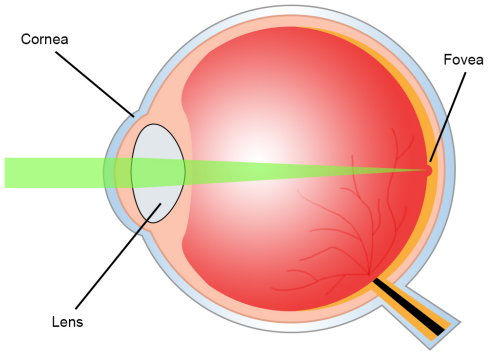

Fig. 1. A cross section of the human eye showing the fovea, a small groove on the back of the eye.

In Section II is explained what is foveated rendering and how it works. In section III the image correlation process is detailed. In section IV the implementation is laid out explaining how the system was put together followed by how the parameters were chosen given the development board's imposed limitations. This section also explains how the template was stored in subsection B, while on subsection C it is explained, with the support of a flowchart, how the final system works. On the results section, the findings are briefly discussed. Finally, the conclusion is presented.

II. Foveated Rendering

The human eye has particular properties that make rendering images in their full resolution a waste of computational resources. The fovea is a small region in the back of the eye where images are projected as illustrated by Fig. 1 [6]. This area is filled with photoreceptors named "cones". Those appear in three different types which respond to the wavelengths of the three primary colours. The cone count in the fovea decreases dramatically from the center towards its edges, which consequently for the human vision means the images formed away from the center have decreased fidelity [7].

The high fidelity area of the human vision spans five degrees from its center, which is quite small given the total area

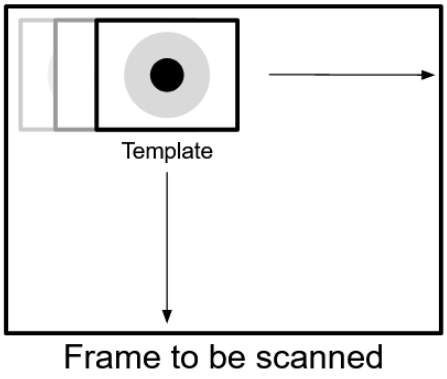

Fig. 2. The template is overlaid and then moved over all possible positions of the scanned image.

spanned of around 135 degrees vertically and 160 degrees horizontally. Rendering images with different fidelity levels based on the user gaze lifts some of the computational strain put on the GPU, which allows for increasing the frame rate. This type of approach has shown significant increase of five to six times the original frame rate [8].

III. Image Correlation

Template matching consists on overlaying parts of the original image with a template as to evaluate if it is present on this image. A similarity score is calculated for each possible position of the template. In Fig. 2 it is shown how the template is moved over the original image.

This technique was chosen for its simplicity of implementation on hardware by using Verilog modules and for yielding consistently good results. For any pixel with indexes (x, y) the correlation score $C_{(x,y)}$ between the template and the original image pixels overlaid by it, is given by

$$C_{(x,y)} = \sum_{x=0}^{x_{max}} \sum_{y=0}^{y_{max}} (1023 - |P_{image}(x,y) - P_{template}(x,y)|) \quad (1)$$

where $P_{image}(x, y)$ and $P_{template}(x, y)$ are the pixels from the image and template respectively that are analyzed two by two. Here, 1024 is an arbitrary value, which is the maximum value a pixel captured by the camera can have. If these two pixels are identical, their comparison equals 1024, otherwise if these two pixels are on complete different ends of their possible values, the result is zero.

The correlation score is the sum of all the pixels likeness. The maximum value of the correlation depends on the size of the template. A picture that is totally different from the template will receive the minimum score possible which is zero, on the other hand a picture that is exactly the same as the template will receive a score denoted by

$$C_{max(x,y)} = 1023 * x_{max} * y_{max} \quad (2)$$

where x_{max} and y_{max} are the maximum coordinate values, which is the template's resolution. In this case with the template's resolution set to 256x256 pixels, the maximum correlation score is 67,043,328 while the minimum is zero.

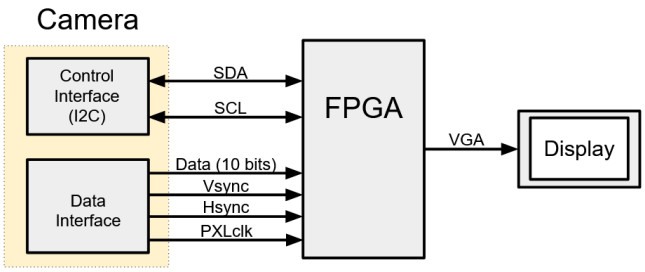

Fig. 3. Control and data connections between the camera and the FPGA.

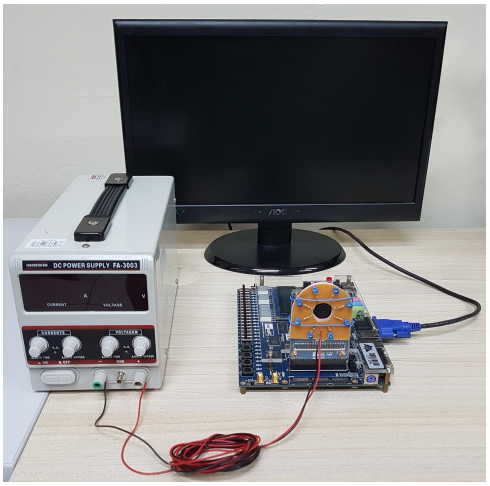

Fig. 4. The camera setup with the attached IR LED ring, connected to the FPGA.

IV. Implementation

The project was implemented using a Terasic DE2-115. This development board contains an Altera Cyclone IV FPGA [9]. The camera used is a TRDB-D5M, which supports resolutions up to 1024x768 pixels at 30 frames per second (fps) [10]. For this investigation a resolution of 640x480 pixels was chosen. The final calculated pupil position is shown on the VGA output overlaid onto the original image.

All of the camera parameters are set up by the FPGA using the I2C bus. The camera feeds the FPGA with the pixel values of the captured image using a 10-bit parallel data bus in sync with the horizontal (Hsync), vertical (Vsync) and pixel clock (PXLclk) lines as illustrated in Fig. 3. There is an infrared LED ring positioned around the camera that makes the whole picture brighter except the pupil, without blinding the user. The setup for the test is shown in Fig. 4.

A. Parameter definition

Since the image correlation method does not require a full colour image input, each pixel is saved as a 10-bit gray scale value. This choice reflects the pixel size outputted by the camera (10 bits for each colour) which is then converted to

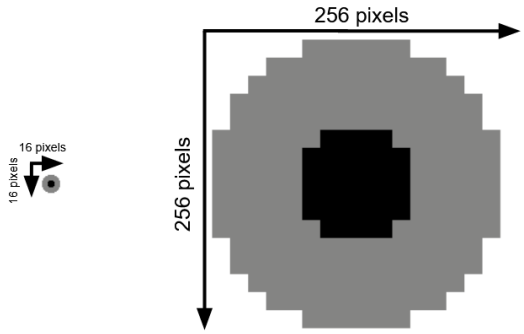

Fig. 5. The template as it is saved on SRAM and its scaled output.

gray scale using the International Telecommunications Union (ITU) guidelines [11]. The SDRAM is already being used as a frame buffer to show the images on the VGA interface. The calculations then are done with the entire frame being stored on the next available memory, the SRAM. Each one of the pixels is saved in one memory position. Although this may appear wasteful given the word size of this particular memory being 16 bits, mapping pixels this way requires one memory access per pixel. The total size of a gray scale frame stored is expressed by S_{frame} in Megabytes.

$$S_{frame} = \frac{640 \times 480 \times 16}{8 \times 2^{10} \times 2^{10}} = 0.58MB \tag{3}$$

Which is roughly 30% of the available SRAM memory for the DE2-115 board. This current implementation runs at a clock speed of 50 MHz.

B. The template

In a way to accelerate the process, the template to be compared with the original image is not stored in the SRAM. Doing so would mean that all the comparisons would take at least two SRAM accesses, thus doubling the time it takes for the process to complete. The alternative chosen here was to store the template as registers on a separate Verilog module. Saving a full image this way would take too many logic elements available on the FPGA used. The alternative was to store a smaller image and scale it to the desired size. The 16x16 pixels image chosen is then upscaled to a full 256x256 pixels image, which is then used as the template to correlate with different parts of the original image. The original template and its scaled version is shown in Fig. 5.

C. System Architechture

Two verilog modules are the center piece of the process. The first performs the correlation by comparing the pixels from the base image and the template, reading from the SRAM and from the template module. It starts the process with a position received from the second module and returns the correlation score for that position. The second module is tasked with controlling the positions, passing them to the first, and to record which one of those has the biggest score. The flowchart presented by Fig. 7 elaborates a bit more on this operation, where (X,Y) represent the position the correlation should start on and (cX,cY) the index of each pixel that is being compared.

Fig. 6. Different eye positions with their calculated center shown as a red dot.

Other modules present are the SRAM Controller, which handles the writings and readings of the searched frame and the Conversion module, which converts the full colour image to a grayscale representation.

V. Results

The proposed method was capable of successfully finding the pupil position. The red dots mark the calculated center on Fig. 6 for different eye positions. White dots are simply the reflection of the IR LEDs, which did not interfere with the measurements.

The time needed to parse one frame was 6 minutes and 43 seconds. One of the strengths of this method is a small usage of the FPGA's total logic units for the synthesized design which, including the modules for the camera set-up, SRAM controller and others, use 2292 logic elements corresponding to 7% of the total available. This is a very compact design when compared to other available methods, such as [12] who uses 39,914 logic units, corresponding to 76% of the available logic units. For embedded applications, the small usage of logical units means a more power-efficient system.

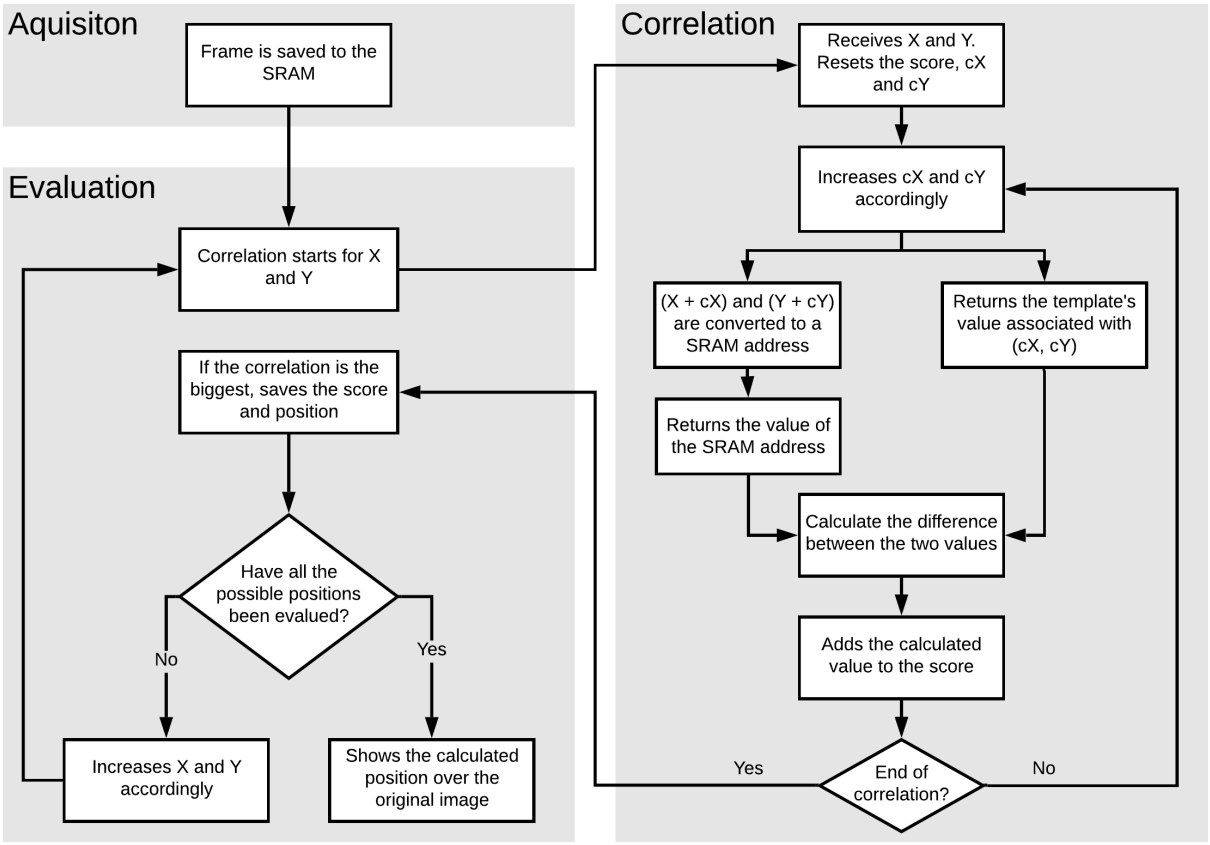

Fig. 7. The system's correlation flowchart.

VI. Conclusion

This paper presents a template matching-based eye tracking technique for foveated rendering using an FPGA. Although the process takes a considerable time, it is very efficient in terms of logic units usage, meaning that the speed can be improved by deriving a parallelized solution. As the camera is very close to the user's eye, improvement could also be achieved by halving the resolution of the image and the template which, should increase the overall speed by 16 times. For a real-time application, the image parsing should not last more than the time the system needs to generate a frame. For an output of 30 fps, for example, the entire process should not take more than 33.33 milliseconds. With the proposed modifications the results should approach the required time frame.

References

[1] W. Barfield, Virtual environments and advanced interface design. New York: Oxford University Press, 1995.

[2] Duchowski, Andrew T. Eye Tracking Methodology: Theory and Practice. Cham: Springer, 2017.

[3] Brunelli, Roberto. Template Matching in Computer Vision: Theory and Practice. Oxford: Wiley-Blackwell, 2009.

[4] Yan, Bei, Xigong Zhang, and Lifeng Gao. "Improvement on Pupil Positioning Algorithm in Eye Tracking Technique." 2009 International Conference on Information Engineering and Computer Science, 2009. https://doi.org/10.1109/iciecs.2009.5363013.

[5] Li, Xiaokun, and William G. Wee. "An Efficient Method for Eye Tracking and Eye-Gazed FOV Estimation." 2009 16th IEEE International Conference on Image Processing (ICIP), 2009. https://doi.org/10.1109/icip.2009.5413997.

[6] Kolb, Helga. "Simple Anatomy of the Retina." Simple Anatomy of the Retina. Accessed May 25, 2020. https://webvision.med.utah.edu/book/part-i-foundations/simple-anatomy-of-the-retina/.

[7] Strasburger, H., I. Rentschler, and M. Juttner. "Peripheral Vision and Pattern Recognition: A Review." Journal of Vision 11, no. 5 (January 2011): 13–13. https://doi.org/10.1167/11.5.13.

[8] Guenter, Brian, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. "Foveated 3D Graphics." ACM Transactions on Graphics 31, no. 6 (2012): 1–10. https://doi.org/10.1145/2366145.2366183.

[9] TERASIC.DE2-115 User Manual. [S.l.], 2010. Rev. 1.02.

[10] TERASIC. TRDB-D5M 5 Megapixel Digital Camera Development Kit User Manual.[S.l.], 6 2017. Rev. 3.

[11] ITU-R.Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. [S.l.], 3 2011.

[12] Kim, Dong-Kyun, Jun-Hee Jung, Thuy Tuong Nguyen, Dai-Jin Kim, Mun-Sang Kim, Key-Ho Kwon, and Jae-Wook Jeon. "An FPGA-Based Parallel Hardware Architecture for Real-Time Eye Detection." JSTS:Journal of Semiconductor Technology and Science 12, no. 2 (2012): 150–61. https://doi.org/10.5573/jsts.2012.12.2.150.