DEEP-PIPELINED FPGA IMPLEMENTATION OF ELLIPSE ESTIMATION FOR EYE TRACKING

Keisuke Dohi, Yuma Hatanaka, Kazuhiro Negi, Yuichiro Shibata, Kiyoshi Oguri

Graduate School of Engineering
Nagasaki University / Japan
1-14, Bunkyo, Nagasaki, Nagasaki
email: {dohi,riaru,negi}@pca.cis.nagasaki-u.ac.jp,{shibata,oguri}@cis.nagasaki-u.ac.jp

ABSTRACT

This paper presents a deep-pipelined FPGA implementation of real-time ellipse estimation for eye tracking. The system is constructed by the Starburst algorithm on a stream-oriented architecture and the RANSAC algorithm without any external memories. In particular, the paper presents comparative results between three different hypothesis generators for the RANSAC algorithm based on Cramer's rule, Gauss-Jordan elimination and LU decomposition. The evaluation results showed that the Gauss-Jordan elimination achieved the highest throughput while the solver with Cramer's rule was the most compact and that our proposed architecture achieved a real-time throughput of 62.5 fps with a single FPGA chip without using any external memories.

1. INTRODUCTION

In this paper, we present FPGA implementation of image-based ellipse estimation for an embedded eye tracking system based on Starburst algorithm [1]. While the Starburst is known to be a robust algorithm, it requires high computational performance, making it difficult to be implemented as a compact and portable system. Our goal here is to demonstrate highly efficient implementation of the Starburst algorithm on a compact FPGA platform without using external memories.

The Starburst algorithm mainly consists of three process steps; (1) pre-processing for camera images, (2) extraction of feature points that represent a pupil contour, and (3) estimation of the best fit ellipse for the feature points. For the former two steps, a deep-pipelined stream-oriented image processing architecture is promising, which can achieve a real-time throughput at a relatively low clock frequency without requiring any external memories. We have already shown such an approach is of benefit for a variety of image applications in terms of performance and power consumption [2],[3],[4]. In this paper, we firstly show how the former two steps of the Starburst algorithm can be restructured to be fitted with this framework.

The last process step in which an ellipse is estimated with the RANdom SAmple Consensus (RANSAC) algorithm[5], offers different computational properties. The RANSAC algorithm can robustly estimate an ellipse from a set of extracted feature points including some outliers. This robustness is achieved by a hypothesis-and-verify matching approach that consists of three steps; (1) randomly selecting a fixed number of feature points from the set of points including outliers, (2) generating hypothesis (ellipse parameters) from the selected points by solving a system of five simultaneous equations, and (3) verifying the generated hypothesis. The method repeats these three steps and finally returns the best hypothesis as a result. The RANSAC algorithm needs to iteratively estimate as many ellipses as possible for different point selection during a single camera frame, and the matrix solving with floating point arithmetic process easily becomes a performance bottleneck.

In contrast to large matrix solvers, few attention has been paid so far to small matrix manipulation on FPGAs. However, this issue is not obvious. For example, [6] reported the efficient algorithm to calculate an inverse of a 4x4 matrix with SIMD instructions was Cramer's rule, which is generally never used because of its high order of computational complexity. As is well known, especially for a small data set, execution performance does not necessarily reflect computational complexity and becomes more sensitive to architectures. Hence, in the latter half of this paper, we compare three kinds of FPGA implementation of equation solvers and discuss which approach is appropriate for small matrix manipulation on an FPGA.

One of the highest developed FPGA implementation of ellipse estimation has been reported by Martelli et al., aiming for detection of circular road signs [7]. In their implementation, feature points are extracted using histogram stretching, intensity gradients, and the edge extraction and thinning method. In order to avoid hardware complication, they imposed a limitation on ellipses that they can detect; ellipses with major axis 0° and 90° from the x axis can only be detected. However, our implementation does not have any limitations on ellipse to be detected, since we often need to detect inclined ellipses in eye tracking. In addition, there have been hardly any literature that focuses on efficient solver implementation for a small simultaneous equation system on an FPGA.

The rest of the paper is organized as follows. In Section 2, we present the Starburst algorithm which includes pre-processing and the RANSAC. In Section 3, we present the implementation details. In Section 4, we show evaluation results and discussion. And finally, in Section 5, we show the scope for future work and conclude the paper.

2. ALGORITHMS

2.1. Reflection removal

The first pre-processing step of the Starburst algorithm is removal of reflections in a pupil image, which often cause extraction of undesired feature points. To relieve this undesirable situation, we used a bilinear interpolation which was derived by simplifying a method proposed in [8]. Let $\mathrm{I}(x, y)$ denote luminance of the pixel at (x, y). Let $\mathrm{R}(x, y)$ denote a binary reflection map, which is easily obtained based on a threshold luminance. $\mathrm{R}(x, y) = 1$ means that the pixel at (x, y) is in a reflection region and to be interpolated. Two points, (x_r, y) and (x_l, y), are required for the interpolation and their coordinates are calculated as follows:

$$x_r = \min\left\{x' : \sum_{i=0}^{L-1} \mathrm{R}(x' - i, y) = 0, x' > x\right\} \tag{1a}$$

$$x_l = \max\left\{x' : \sum_{i=0}^{L-1} \mathrm{R}(x' + i, y) = 0, x' < x\right\} \tag{1b}$$

where L is a parameter on the filter size. Using these two points, interpolated luminance $\mathrm{I}'(x, y)$ is calculated as:

$$\mathrm{I}'(x, y) = \frac{\mathrm{I}(x_r, y) \cdot (x_r - x) + \mathrm{I}(x_l, y) \cdot (x - x_l)}{x_r - x_l} \tag{2}$$

Although this method executes interpolation only in a horizontal direction, the majority of reflections can be effectively removed in a practical environment as shown in Fig. 1 (a) and (b).

2.2. Extraction of feature points of a pupil contour

The Starburst feature point extraction method starts to radially find feature points from a base point Ps, and returns a set of the nearest points which have larger intensity derivative than a threshold on each ray as shown in Fig. 1 (c). In addition, another extraction process starts from firstly extracted feature points towards the base point to improve the robustness. In this paper, we only used the single step Starburst feature extraction for ease of compact hardware implementation.

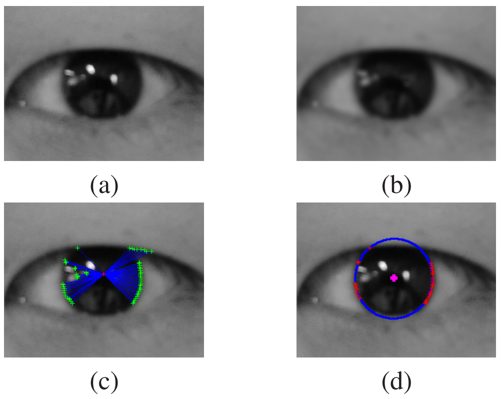

Fig. 1. Overview of the Starburst algorithm. (a) Original image. (b) Reflection removal and box blur. (c) Starburst feature extraction. Green point denotes a feature point and red point denote base point for the extraction. (d) Estimated ellipse(blue line) and center point(purple). Red points denote inliers.

2.3. RANSAC

The RANSAC is an iterative method that finds a model from a data set of points including outliers. At first RANSAC randomly samples a subset from the extracted feature points. The minimum size of subset depends on a target model, and an ellipse needs at least five points.

After sampling a subset, a hypothesis is generated from the subset. We used the following ellipse equation,

$$x^2 + Ax_iy + By^2 + Cx + Dy + E = 0 \tag{3}$$

where A, B, C, D and E are parameters to be estimated. Using the method of least squares, a system of simultaneous equations for estimating an ellipse is obtained as Eq. (4).

Hypothesis is generated by solving Eq. (4) and is verified by successively substituting all the feature points and the obtained parameters into Eq. (3). The values of left-hand side are considered as an error and the number of inliers is counted based on a threshold error value. The above three steps, random sampling, hypothesis generating and verifying, are repeated until a new dataset of the next frame image arrives. Finally the best hypothesis, which has the maximum number of inliers, is returned as estimated parameters.

3. IMPLEMENTATION

3.1. Design overview

Fig. 2 illustrates our proposed system. Pixel data input from the camera interface are streamed into Starburst module through cascaded pre-processing filters. The Starburst module detects up to 128 feature points and the Trimming module eliminates invalid feature points every one frame. The Random sampling module samples five points from the

$$\begin{pmatrix} \sum x_i^2 y_i^2 & \sum x_i y_i^3 & \sum x_i^2 y_i & \sum x_i y_i^2 & \sum x_i y_i \\ \sum x_i y_i^3 & \sum y_i^4 & \sum x_i y_i^2 & \sum y_i^3 & \sum y_i^2 \\ \sum x_i^2 y_i & \sum x_i y_i^2 & \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i^2 & \sum y_i^3 & \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i y_i & \sum y_i^2 & \sum x_i & \sum y_i & \sum 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix} = \begin{pmatrix} -\sum x_i^3 y_i \\ -\sum x_i^2 y_i^2 \\ -\sum x_i^3 \\ -\sum x_i^2 y_i \\ -\sum x_i^2 \end{pmatrix} \tag{4}$$

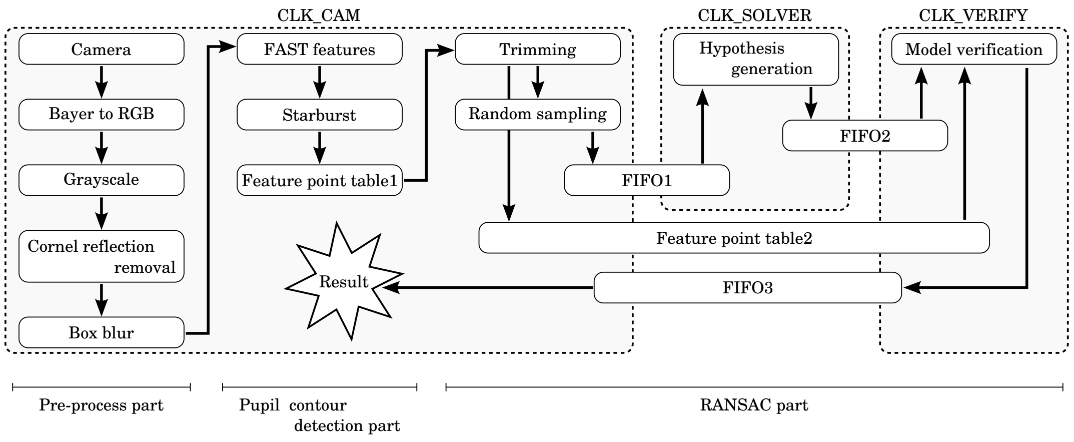

Fig. 2. Overview of our eye-tracking system.

valid points set and stores them into FIFO1. The Hypothesis generation module executes floating-point arithmetic operations for estimating elliptic parameters. The Model verification module counts the number of inliers for the generated hypothesis and updates the temporal best hypothesis when better hypothesis is found. Note that all the modules only access to on-chip memories, e.g., FFs, BRAM and Distributed RAM.

3.2. Pre-processing

Fig. 3 shows a basic architecture framework for stream-oriented image processing, which we call *streamed structure*. The architecture consists of a register array for a window region of interests, FIFOs for line buffers, and pipelined arithmetic for particular operations. This architecture outputs operation results at the same throughput as data input. Although most of the pre-processing can be straightforwardly implemented on this architecture framework, the reflection removal process is relatively complex. We split the process into two steps; the determination of envelop pixels and the interpolation. As a result, dynamic control flows were mitigated and all the pre-processing modules were implemented on the streamed structure.

3.3. Feature point extraction for pupil contour detection

The Starburst feature extraction process was also implemented on the streamed structure. We split the process into three parts; (1) calculation of intensity derivatives for all the pixels, (2) calculation of distances and angles of coordinate points from the center point, and (3) update of the feature points table.

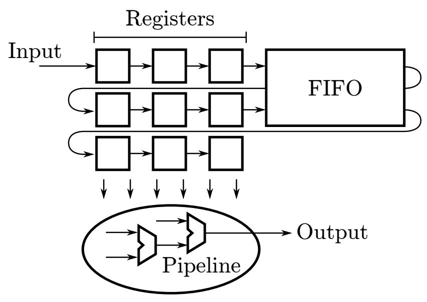

Fig. 3. Overview of our streamed structure.

The intensity derivatives are calculated by a segmentation test that is inspired by FAST corner detection[9]. By comparing the intensity of a candidate pixel with each point on a 16-point ring that surrounds it, we can know possible directions in which the candidate pixel is able to be recognized as a feature point.

The distance and angle between the candidate point and center point are calculated using add, multiply and $\arctan$ operations. We used a table whose i-th entry holds the coordinate of the feature point that was most recently found on the i-th direction and its distance from the center point. The table is updated when a new feature point that is nearer to the center for each direction. After scanning all the pixels, the table holds a set of the Starburst feature points for the corresponding frame. In our implementation, the number of entries of the feature points table is 128.

3.4. RANSAC

As shown in Fig. 2 , the RANSAC part includes three clock domains and asynchronous FIFOs and tables are provided for passing data between different clock domains. Using a double buffering technique with the dual-port RAMs, the Hypothesis generation module and the Model verification module work in parallel. The Random sampling module randomly generates addresses for the feature point table using a 32-stage liner feedback shift register.

The Hypothesis generation module consists of a generator of a system of simultaneous equations (Eq. (4)) with the received five feature points and its solver. We implemented three kinds of solvers for simultaneous equations based on Cramer's rule, Gauss-Jordan elimination and Doolittle LU decomposition.

3.5. Hypothesis generation

3.5.1. Cramer's rule

Consider a system of n linear equations for n unknowns as follows:

$$A\boldsymbol{x} = \boldsymbol{b} \tag{5}$$

where A denotes an $n \times n$ matrix, $\boldsymbol{x}$ and $\boldsymbol{b}$ denote column vectors. According to Cramer's rule, the values for the unknowns are given by:

$$x_i = \frac{|A_i|}{|A|} \tag{6}$$

where A_i denotes a matrix formed by replacing the i-th column of A by the column vector $\boldsymbol{b}$. The determinant $|A|$ can be defined as:

$$|A| = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} A_{i,\sigma_i} \tag{7}$$

where σ denotes a permutation of the set $\{1, 2, ..., n\}$, σ_i denotes i-th number of σ and $\mathrm{sgn}(\sigma)$ denotes the signature of σ which is 1 or -1. Due to its high order of computational complexity, the Cramer's rule is generally never considered as a practical solution. However, its dataflow has quite simple and regular structure including rich parallelism.

We used a memory table called *order table*, whose j-th entry contains j-th permutation σ_j for column vectors of the matrix (3×5 bit) and the value of $\mathrm{sgn}(\sigma_j)$ (1 bit). five memories called *column tables* whose i-th memory contains i-th column vector of the matrix. These tables simplify the calculation of the determinant. At first, addresses for the column tables and a sign are fetched from the order table. Then, the corresponding elements of the matrix are fetched from the five column tables. After multiplying these five values and the sign, the result is accumulated. Repeating this for 120 times (the number of permutations for $n = 5$), determinant of the matrix is obtained. $|A_i|$ is calculated by using $b_{\sigma_{ji}}$ instead of $a_{i,\sigma_{ji}}$ from the i-th column table at the j-iteration, where σ_{ji} means i-th number of the permutation σ_j.

Since these multiplication steps are independent each other, pipelining can be fully applied. However, the final accumulation step imposes a pipeline stall due to the latency of the adder. We focused on the fact that ellipse estimation needs six determinant of matrices: $|A|, |A_1|, \ldots, |A_5|$, and we interleaved the calculation of the six determinants using an adder with 6-cycle latency, so that any pipeline stalls did not occur. Hence, after executing 720 sets of multiplication, six determinants are obtained every clock cycle. Finally, the ellipse parameters are calculated by dividing the last five determinants with the first determinant.

This hypothesis generator consists of two 42-bit integer multipliers and two double precision floating point (FP) multipliers, one double precision FP adder, and one double precision FP divider with some format converters. The estimated ellipse parameters are output as single precision FP values.

3.5.2. Gauss-Jordan elimination

Gauss-Jordan elimination, also known as the sweep-out method is one of the commonly used algorithms for solving a system of simultaneous equations. While its computational complexity is higher than that of the Gaussian elimination which is another popular method, Gauss-Jordan elimination does not need backward substitution which is an essentially sequential process.

Given an N-by-$(N+1)$ matrix $M^{(0)} = [A\ \boldsymbol{b}]$ and integer K $(\leq N)$, eliminated matrix $M^{(K)}$ in the K-th step is defined as follows:

$$m_{i,j}^{(K)} = \begin{cases} m_{i,j}^{(K-1)} & \text{if } j < K, \\ \frac{m_{i,j}^{(K-1)}}{m_{K,K}^{(K-1)}} & \text{else if } i = K, \\ m_{i,j}^{(K-1)} - \frac{m_{i,K}^{(K-1)}}{m_{K,K}^{(K-1)}} m_{K,j}^{(K-1)} & \text{otherwise. .} \end{cases} \tag{8}$$

In the RANSAC algorithm, the accuracy of the best solution in the repetitive trials is more important than that of each individual solution. Thus, we did not implement pivot exchanging, which makes the control flow sequential and complicated.

Our hypothesis generator based on the Gauss-Jordan elimination consists of cascaded five sub-modules, each of which corresponds to calculation of $M^{(K)}$ and consists of three single precision FP operators; adder, multiplier and divider.

Table 1. Resource usage of each implementation.

	CRAMER	GAUSS	LU	Available
FF	20,667	22,267	24,062	28,800
LUT	18,709	19,130	19,725	28,800
BRAM	39	39	34	48
DSP48	47	44	39	48

These sub-modules can work in a macro pipelined manner, that is, a new hypothesis can be started to be generated after the first sub-module finish its calculation.

3.5.3. Doolittle LU decomposition

Our third solver is based on Doolittle LU decomposition, which is also a popular approach. This consists of the following three steps. Firstly, a given matrix A is decomposed into a lower triangular matrix L and a upper triangular matrix U. Then, the equation $L\boldsymbol{y} = \boldsymbol{b}$ is solved for $\boldsymbol{y}$. Finally, $\boldsymbol{x}$ is obtained by solving solving $U\boldsymbol{x} = \boldsymbol{y}$. Compared to the Gauss-Jordan elimination, the computational complexity of LU decomposition is reduced.

Given an N-by-N matrix $U^{(0)} = A$ and integer $K(\leq N)$, the K-th step of calculation for decomposed matrices $U^{(K)}$ and $L^{(K)}$ are calculated as follows:

$$u_{i,j}^{(K)} = \begin{cases} u_{i,j}^{(K-1)} & \text{if } i \leq K, \\ u_{i,j}^{(K-1)} - \frac{u_{i,K}^{(K-1)}}{u_{K,K}^{(K-1)}} u_{K,j}^{(K-1)} & \text{otherwise.} \end{cases} \tag{9}$$

$$l_{i,j}^{(K)} = \begin{cases} l_{i,j}^{(K-1)} & \text{if } j < K, \\ \frac{u_{i,j}^{(K-1)}}{u_{K,j}^{(K-1)}} & \text{else if } i \geq K, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The LU decomposition can be executed with a similar architecture to the Gauss-Jordan elimination. We used three sub-modules each of which consists of adder, multiplier and divider for single precision FP operations. After the decomposition, $\boldsymbol{y}$ and $\boldsymbol{x}$ are calculated by the forward substitution and backward substitution, respectively. Each substitution module also requires the same three operators for single precision FP values.

4. EVALUATION AND DISCUSSION

4.1. Evaluation environment

The evaluation system was implemented on an ML501 prototype board equipped with a Xilinx Virtex-5 XC5VLX50 FPGA and an OmniVision OV9620 CMOS camera device using ISE 13.4 tool sets. In the experiment, the clock frequencies for CLK_CAM, CLK_SOLVER, and CLK_VERIFY were constrained to 25 MHz, 100 MHz and 100 MHz, respectively, and these constraints were met for every evaluated implementation. Since our FPGA implementation employed the streamed structure, pre-processing and the Starburst feature extraction achieved the performance of 62.5 fps, which is the maximum throughput of VGA image generation for the camera device used. Note that any on-board memory devices were not used except for debug purpose.

Table 2. Slice usage of solvers and dividers.

	CRAMER	GAUSS	LU
solver	4,250	4,622	5,303
div	1,652	1,905	1,835
ratio	39%	41%	35%

4.2. Resource usage

Table 1 shows resource usages for each implementation with the available resource amount on the FPGA, and Table 2 shows how much portion of each solver is occupied by dividers. The solver with Cramer's rule (CRAMER) showed the lowest resource usage in FFs and LUTs despite of only this solver requires double precision FP operators due to accuracy requirements. This compact implementation is due to CRAMER needs the smallest number of operators among the three solvers and mainly consists of multipliers which can be efficiently built with DSP48E hard macro modules.

As Table 2 shows, FP dividers were dominant in terms of resource usage for each solvers. Although the Gauss-Jordan elimination (GAUSS) and the LU decomposition (LU) use single precision FP operators unlike CRAMER, the largest part is still occupied by dividers since they use multiple dividers. Since dividers are frequently used in these solvers, it seems difficult to reduce the resources without performance degradation. However, for the CRAMER, a fully pipelined FP divider is used only for five division. Thus, there should be room to reduce the required resources without a major performance drop by changing the structure of the divider.

4.3. Power consumption

Table 3 shows performance and power consumptions of each implementation. The power consumptions were measured by inserting a 1- ohm shunt register between the board and DC power supply. Note that the 'total' power value include the power consumed by not only the FPGA but also the camera device, HDMI display interface, debug circuits (including memories), and so forth. We also implemented and measured a system without any solvers and calculated the solver power values as a difference.

The CRAMER consumed the highest power despite the

Table 3. Performance of each implementation.

	CRAMER	GAUSS	LU
$Power$ [W] (Solver/Total)	0.33/3.34	0.28/3.29	0.15/3.16
$Latency$ [us]	8.69	3.05	4.99
$Throughput$	1.1×10^6	10^7	3.8×10^6

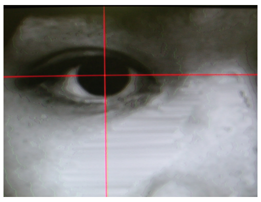

Fig. 4. Ellipse estimation result using the CRAMER solver. Crossing point shows the estimated center of the eye.

solver showed the lowest resource usage. The comparison results suggest that power consumptions are more sensitive for utilization of DSP48Es and BRAMs rather than FFs and LUTs. The total power consumption was 3.34 watts even for the CRAMER, which demonstrates the power effectiveness of FPGAs. Capability of a tight and efficient integration of dedicated arithmetic and I/O interface makes a huge contribution to the advantage of the FPGA implementation.

4.4. Throughput

We defined a *throughput* of a solver as the number of hypothesis generated per second. As in Table 3 , the GAUSS solver showed the lowest latency and the highest throughput and this value corresponds to approximately 9 times of that for the CRAMER solver. This is mainly due to the high order of computational complexity of the Cramer's rule. However, the performance difference was smaller than the difference in the computational complexity because of the higher usage rate of arithmetic pipelines in the CRAMER solver.

For the RANSAC algorithm, a higher throughput of the hypothesis generation improves the accuracy of ellipse fitting. As Fig. 4 shows, even the CRAMER solver successfully estimates reasonable ellipse parameters for practical images. In this sense, it can be considered that the throughputs obtained by the three solvers are enough for ellipse estimation for eye tracking, while detailed accuracy evaluation will be needed for future work.

5. CONCLUSION

In this paper, we presented that deep-pipelined FPGA implementation of the Starburst algorithm for eye tracking. The system consists of cascaded streamed architectures for image processing and an arithmetic architecture for generating and solving simultaneous equations for the RANSAC-based ellipse estimation. The implementation experiments showed that our architecture achieved a real-time throughput (62.5 fps) with low power consumption (3.34 W) without using any external memories. We also empirically investigated FPGA-oriented algorithms for solving a small system of simultaneous equations for ellipse estimation. While the Gauss-Jordan elimination achieved the highest throughput for solving the equations, the solver with Cramer's rule was the most compact and likely to get more smaller. This suggest that different approaches to the analysis of algorithms are important for FPGAs rather than traditional approaches based on computational complexity. Our future work includes detailed accuracy evaluation of estimated ellipses, optimization of architectures in terms of FPGA resource mapping, and use of a refined RANSAC algorithm with a more sophisticated sampling scheme for feature points.

6. REFERENCES

[1] D. Li, D. Winfield, and D. J. Parkhurst, "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," in *Proc. IEEE Vision for Human-Computer Interaction Wrokshop at CVPR*, June 2005, pp. 1–8.

[2] H. Matsubayashi, S. Nino, T. Aramaki, Y. Shibata, and K. Oguri, "Retrieving 3-d information with FPGA-based stream processing," in *Proc. 16th ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, Feb. 2008, pp. 261–261.

[3] K. Dohi, Y. Yorita, Y. Shibata, and K. Oguri, "Pattern compression of FAST corner detection for efficient hardware implementation," in *Proc. IEEE 21st Int. Conf. Field Programmable Logic and Applications*, Sept. 2011, pp. 478–481.

[4] K. Negi, K. Dohi, Y. Shibata, and K. Oguri, "Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm," in *Proc. Int. Conf. Field-Programmable Technology*, Dec. 2011, pp. 1 –8.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[6] "Streaming SIMD Extensions - Inverse of 4x4 Matrix," Intel Corp., Tech. Rep., 1999.

[7] S. Martelli, R. Marzotto, A. Colombari, and V. Murino, "FPGA-based robust ellipse estimation for circular road sign detection," in *Proc. IEEE 6th IEEE Workshop on Embedded Computer Vision*. IEEE, June 2010, pp. 53–60.

[8] Z. He, T. Tan, Z. Sun, and X. Qiu, "Toward accurate and fast iris segmentation for iris biometrics," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 31, no. 9, pp. 1670–1684, Sept. 2009.

[9] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking." in *IEEE Int. Conf. Computer Vision*, vol. 2, Oct. 2005, pp. 1508–1511.