FPGA Implementation of RANSAC Algorithm for Real-Time Image Geometry Estimation

Jia Wei Tang [*] Nasir Shaikh-Husin [†] Usman Ullah Sheikh [‡]
VeCAD Research Laboratory, Faculty of Electrical Engineering,
Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Malaysia
E-mail: jwtang2@live.utm.my[*], nasirsh@fke.utm.my[†], usman@fke.utm.my[‡]

***Abstract*—Random Sample Consensus (RANSAC) is commonly used in many estimation tasks especially in computer vision applications due to its simplicity. This paper presents a hardware/software co-design implementation of RANSAC algorithm for real-time affine geometry estimation on a field programmable gate array (FPGA) platform. Double buffering technique is used to store and process data in pipeline. Experimental result shows that the proposed system managed to speed up the software process by about 11.4 times for 100 data points. The proposed architecture was also tested on Altera DE2-115 with 100 MHz NiosII Processor running to handle a video stream of 30 frames per second.**

Keywords*—*RANSAC; FPGA; embedded hardware system; image geometry estimation

I. Introduction

RANSAC is a robust estimation algorithm that was first proposed by Fischler and Bolles [1] in 1981. It is an iterative method to find parameters for a particular mathematical model based on a set of observed data. This non-deterministic algorithm increases the probability to find a reasonable result with a finite number of iterations. It is assumed that the set of data to be processed contains inliers and outliers. The former is data which can be explained by a mathematical model while the latter is data which does not fit well with the model, such as noise. As a robust estimator, RANSAC is able to remove outliers and estimate the model parameters with high degree of accuracy even with substantial level of contamination.

RANSAC is widely used in various computer vision applications such as image stitching [2], [3], motion estimation [4] and object detection and tracking [5], [6]. In most of the applications, RANSAC is utilized for image geometry estimation task [7], [8], [9]. Its usage is extended to real-time computer vision problem [4], [5], [10] where an immediate response of a system is required especially in online detection and tracking system. Moreover, this algorithm is also implemented in some embedded systems such as Unmanned Aerial Vehicle (UAV) application [10] whereby only limited processing power and memory resources are available. Therefore, hardware accelerated RANSAC module will greatly improve the performance of embedded real-time image processing.

Image geometry estimation is the process of determining the fundamental matrix or parameters which describes the transformation between two images. Several steps are needed in order to perform the estimation including feature detection, feature matching and fundamental matrix estimation. Firstly, feature detection extracts feature points such as corners from two input images. The feature matching process then correlates the features from one image to the other to find their corresponding pairs between two images. RANSAC algorithm plays the role of estimating the transformation parameters based on these matched points.

This paper presents a hardware/software co-design implementation of RANSAC algorithm for real-time affine geometry estimation on an FPGA platform. The rest of this paper is organized as follows. Section II discusses the literature review and related work on RANSAC. Section III describes the RANSAC algorithm and how suitable part of the algorithm is chosen to be hardware accelerated. Section IV describes the proposed system of RANSAC on FPGA. Section V discusses the result and performance analysis of the proposed system and last but not least, Section VI gives the conclusion of this work.

II. Related Work

Various modifications to RANSAC algorithm have been introduced to improve its efficiency in terms of accuracy, robustness and computation speed. Torr and Zisserman introduced MLESAC (Maximum Likelihood Estimation Sample Consensus) [8] to estimate image geometry. They proposed fitness scoring technique in hypothesis evaluation to increase the accuracy, unlike the basic algorithm which uses only the number of inliers. PROSAC (Progressive Sample Consensus) was introduced in [3] on the image matching problem using guided sampling method to speed up the standard RANSAC algorithm. It sorts data based on matching scores in previous feature matching process, and generates hypothesis from top-ranked data, progressively moving to less ranked data, where the worst case is whole data. However, this method requires extra step of data sorting which increases the complexity of implementation. In addition, R-RANSAC (Randomized RANSAC) with $\mathrm{T_{d,d}}$ test proposed in [11] sped up the RANSAC through partial evaluation which quits the current iteration of hypothesis evaluation if the hypothesis is far from the truth . The termination occurs if the generated hypothesis fails $\mathrm{T_{d,d}}$ test, where all d data out of randomly selected d data are not consistent with the hypothesis. On top of that, there are also other modifications proposed such as MAPSAC (Maximum A Posteriori Sample Consensus) [12], LO-RANSAC [13], and preemptive RANSAC [4] with the aim of improving the speed and accuracy of the basic algorithm.

RANSAC was used in the implementation of tracking for UAV applications proposed in [10]. In the paper, they presented a hardware/software co-design of real-time feature detection and tracking system on FPGA. RANSAC algorithm was performed in embedded software as only a small amount of data input was used in their system. RANSAC was also implemented for Robust Essential Matrix Estimation [9]. They proposed a multiprocessor system based on the Microblaze processors to provide parallelism. RANSAC implementation on FPGA was also applied in ellipse estimation for eye tracking [6] and road sign detection [14].

RANSAC algorithm might be one of the least computationally lengthy process in overall image processing problem if smaller data set with high inliers ratio is used. Hence, spending too much resource such as parallel processing or multiprocessors for RANSAC algorithm might result in insufficient resource available for other image processing tasks which are usually more complex. RANSAC computation using software processor might not give a good result for larger data set or data with lower inliers ratio in a short time. Limiting the number of iterations in a real-time system may yield non-optimal result and is worse with increasing amount of outliers. Thus, this paper proposes a real-time affine geometry estimation system for video applications where RANSAC algorithm is implemented in a hardware/software co-design. To speed up the algorithm efficiently with minimal resource, only the most compute intensive task in RANSAC iteration will be hardware accelerated. This provides a balanced solution for speed and resource usage for RANSAC algorithm.

III. RANSAC Algorithm

RANSAC algorithm for affine estimation can be divided into several steps. Three distinct matched points from two images after feature matching process are randomly chosen as samples. Hypothesis model for affine transformation is then generated from the selected data. $T_{d,d}$ test as proposed in [11] is applied to quit the current iteration if the hypothesis is far from the truth. Next, the hypothesis is evaluated with a fitness score by fitting its parameters to all data. The best hypothesis model is constantly updated in each iteration and emerges as result at the end of the algorithm.

However, the drawback is that RANSAC algorithm has no upper bound time. Conventional RANSAC algorithm terminates the loop adaptively based on the probability of getting better result on next iteration as proposed in [15]. Time taken for each loop varies every iteration and depends on the size of input data. However, in real-time applications, RANSAC computation will only have a limited period of time based on the speed constraints or the frame rate of the system. Limiting the number of iterations will not yield a constant computation period for RANSAC.

Time dependent termination criterion is proposed to ensure an upper bound time, regardless of the number of iterations or probability. Upper bound time for RANSAC is reserved within the frame period, depending on how much time the software processor is available for this process. The overall algorithm used in this paper is shown in Algorithm 1.

However, RANSAC results might not be optimized if the bounding time is too short. To provide a measure of the sufficiency of the upper bound time, the probability of at least one sample randomly selected is error free, p can be explored. The probability, p is formulated by the inliers ratio, r and

Algorithm 1 RANSAC

```
while time taken<upper bound time do
   1. Randomly select 3 distinct data as input samples.
   2. Generate hypothesis model.
   3. Apply $T_{d,d}$ test.
   4. Calculate the fitness score.
   5. Update and store best scored parameters.
end while
```

the number of iterations, k as in (1), and their relations is illustrated in Fig. 1. There is better chance of getting good result from RANSAC if the probability of selecting error free sample is higher. Based on the graph, lower inliers ratio requires more iterations to get a good result. In a fixed upper bound time, the more iterations that can be performed will increase the chance of getting a good result for a given inliers ratio. Hence, improving RANSAC computation speed using hardware accelerator will allow more iterations in a limited time, therefore increases the robustness of the system.

$$p = 1 - \left(1 - r^3\right)^k \tag{1}$$

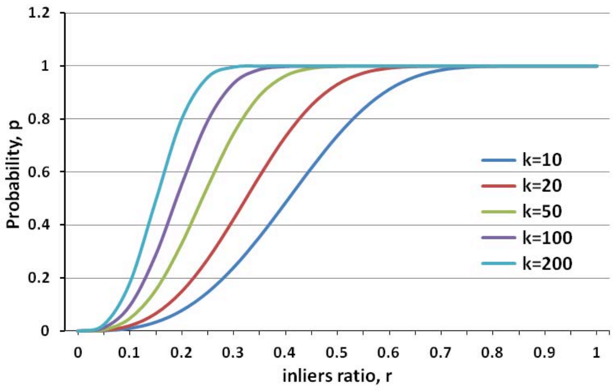


Fig. 1. The relation between probability of at least one sample selected is error free, inliers ratio, and number of iterations.

Before hardware implementation, benchmarking of RANSAC algorithm was performed to find out the most time consuming process. Table I shows on the speed profiling of RANSAC algorithm running on 100 MHz NiosII Processor. Based on speed profile, fitness scoring process, which uses 87% of the overall time was chosen for implementation on dedicated hardware.

TABLE I. Speed profiling of RANSAC algorithm in fully software implementation

Processes	Average time (μs)	Percentage
Random sample selection	40.56	4.0%
Hypothesis generation	58.05	5.8%
$T_{d,d}$ test	31.65	3.1%
Fitness scoring	878.17	87.0%
Update best model	0.68	0.1%

Fitness scoring is the process of calculating the fitness of the hypothesis model to all input data, as shown in Algorithm 2. The individual error calculation is based on [8].

If the data is an inlier, the error is its deviation distance with the hypothesis fitted point. If the data is an outlier, a predefined distance, *thdist* is used as the score, serving as a constant penalty for each outlier. Each data is considered as inlier if the absolute distance between hypothesis fitted point and data point is smaller than *thdist*. The final fitness score is accumulated from all individual errors for each data. A perfect fit will give a fitness score of 0. The number of computation for fitness scoring increases with size of data, as more data will require more computation and hence more time.

Algorithm 2 Fitness Scoring

```
fitness score=0
**for all** $data_i$ **do**
    $asubx = Abs(x_{2i} - (x_{1i}.H_0 + y_{1i}.H_1 + H_2))$
    $asuby = Abs(y_{2i} - (x_{1i}.H_3 + y_{1i}.H_4 + H_5))$
    $score = min((asubx^2 + asuby^2), thdist^2)$
    $fitnessscore = fitnessscore + score$
**end for**
Where:
Each $data_i$ contains a point pair ( $x_{1i}$, $x_{2i}$, $y_{1i}$, and $y_{2i}$ )
$H_0, H_1, H_2, H_3, H_4, H_5$ are affine parameters of hypothesis model.
$thdist^2$ is the square of predefined threshold distance.
```

IV. Proposed Hardware

The proposed hardware is designed for real-time video processing on 30 fps system with NiosII processor. In most applications, data inputs are coming from other modules such as feature detection and matching. Thus double buffering technique is used to enable process pipelining, which uses two on-chip memory as buffers. The hardware architecture is shown in Fig. 2. In each frame period, input data is directly streamed into one of the buffer while data from the other buffer is being processed by RANSAC algorithm. The functionality of both buffers is swapped at end of data every frame by switching the accessing bus between input data and RANSAC process. The bus switching is automatically done by the buffer switching controller in hardware, therefore RANSAC process must be terminated before the switching occurs in each frame period.

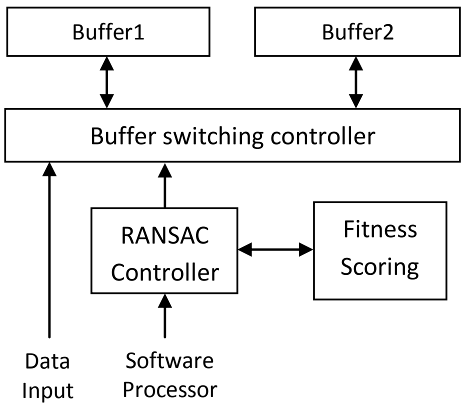


Fig. 2. Top level module of proposed RANSAC hardware/software co-design

To improve the data transmission speed, hardware accelerator of RANSAC must be able to read the data by itself from the on-chip memory without instruction from software processor.

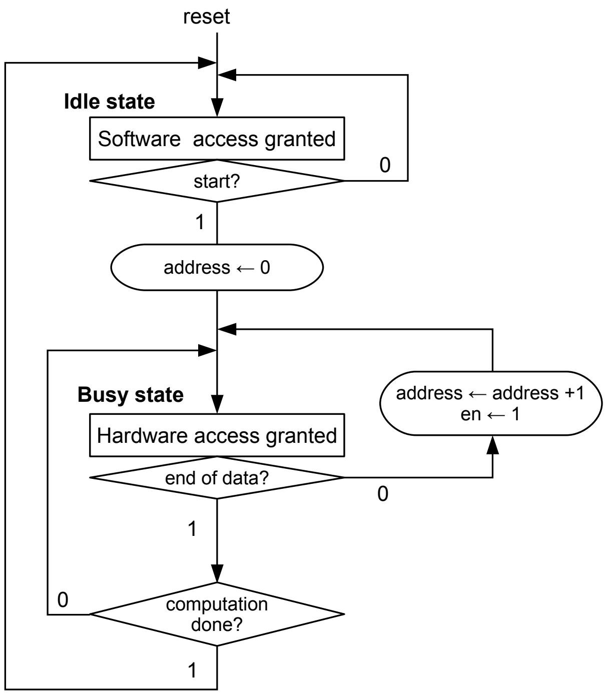


Fig. 3. Flow chart of RANSAC controller

A controller is inserted in between the software processor and buffers, to manage the data flow between memory, hardware accelerator and software processor. A direct memory access to the buffers from the hardware itself enables the data to be fetched for computation in fitness scoring module as fast as possible. As shown in Fig. 3, memory access for software processor is granted in idle state while hardware is granted in busy state. The memory access priority is given to hardware accelerator during fitness scoring computation while halting the transmission from the software processor until completion of the computation. As full control is granted for hardware accelerator, the throughput of hardware computation is able to reach one data per cycle. An address counter will point to the current processed data and will be increased after each cycle during computation. After reaching the end of data, the controller stays at busy state until the fitness scoring is done before returning to idle state as several cycles of latency is needed.

Fitness scoring module on the other hand performs the computation and accumulation of errors for each hypothesis. Fig. 4 shows the hardware accelerator utilizes three stages of pipeline with the aim to isolate multiplication processes, thus allowing faster clock rate. The first stage pipeline registers are located right after the first multiplication, while the other two stages of pipeline registers enclose the squaring processes. The overall process ends with an accumulator. There are four cycles of latencies before the fitness score begins to accumulate. The accumulator is reset on each new set of hypothesis. Thus, the total number of cycles required for fitness score computation is the number of overall data plus the four cycle latencies. Each data increase will only take one extra cycle.

Fitness scoring requires floating point computations as the processing data are not integer values. However, hardware implementation uses suitable fixed point precisions for each data input, parameters and output. Since this hardware is designed for 32-bit software processor, all affine parameters, H_0 to H_6 are properly scaled to different 16-bit precisions as shown in Table II. With this arrangement, two affine parameters can be assigned with a single transfer by the processor in each write instruction. Bit shifting and truncation are applied to the inputs and intermediate results for proper fixed point alignment in each mathematical operation.

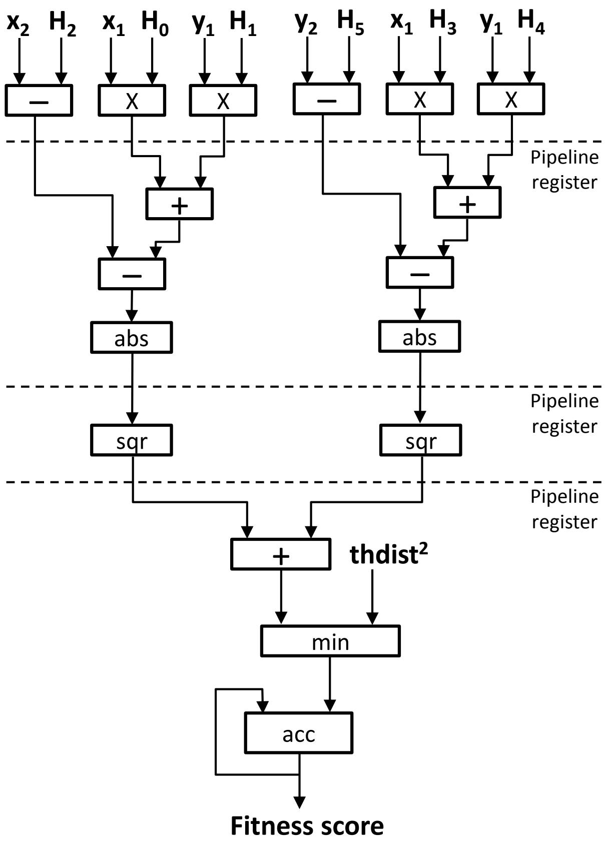


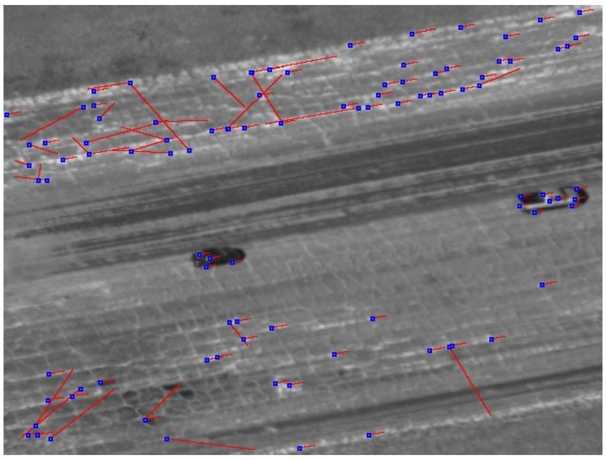
Fig. 4. Hardware module of fitness scoring

TABLE II. Fixed point precisions for input and output of fitness scoring module

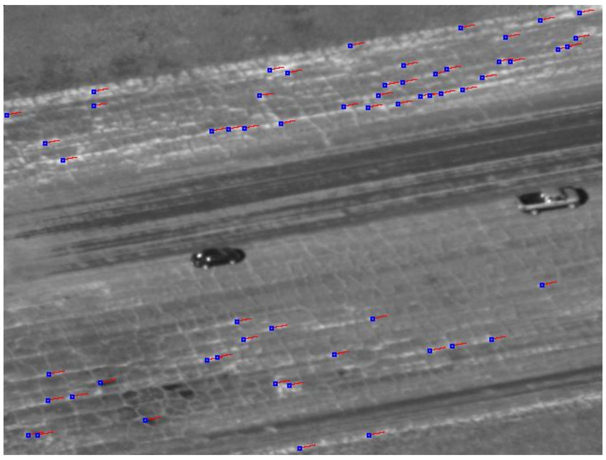
Parameters	number of bits	
	integer bits	fraction bits
x_1, y_1, x_2, y_2	11	0
H_0, H_1, H_3, H_4	4	12
H_2, H_5	11	5
Fitness score	9	12

V. Experimental Results and Analysis

The proposed RANSAC algorithm was verified using video frames from UAV database [16]. Images in Fig. 5 shows example of RANSAC result on affine estimation of two video frames. Feature points from one frame are indicated by blue dots while red lines link them to the position of their corresponding matched pairs in other frame after the feature matching, as shown in Fig. 5(a). Using different hypothesis generated from randomly selected points of these points, RANSAC will find the best affine parameters which are agreed by most of the points. Outliers are removed as illustrated in Fig. 5(b) and the best parameters are determined based on the fitness score of the hypothesis .



(a) Input points for RANSAC algorithm



(b) Inliers after RANSAC algorithm

Fig. 5. RANSAC result of finding the affine parameters for video number V3V1000003_004

The proposed system was tested using Altera DE2-115 development board with NiosII processor running at 100 MHz clock rate. Table III shows fitness scoring computation time and execution time of overall RANSAC algorithm for both software and hardware implementations. By increasing data sizes, software computation speed degrades significantly but time taken for hardware computation is barely affected. As mentioned in Section IV, each increase in data size will only cost one additional clock cycle for hardware to compute the fitness score. Time taken for overall RANSAC computation is almost constant in hardware because the other tasks are not data size dependent unlike fitness scoring. The speed up of the implemented hardware over normal software process for different data sizes is illustrated in Fig. 6.

In other perspective, the proposed architecture was also tested in a 30 fps real-time system. However, instead of using the whole 33 ms frame period as the upper bound time, 25 ms was chosen, leaving 8 ms gap for software processor to

TABLE III. Computation time comparison between hardware and software implementation of different data size

Data Size	Computation time (μs)			
	Fitness Scoring		RANSAC	
	Software	Hardware	Fully Software	Hardware Accelerated
12	161.99	23.57	294.32	153.81
24	321.22	24.44	456.77	162.90
36	513.99	23.32	640.08	160.93
48	692.92	25.51	823.01	169.35
60	856.24	26.68	988.54	155.88
72	1042.98	23.59	1172.21	154.55
84	1204.71	24.56	1333.07	165.08
96	1537.83	24.83	1670.74	150.82
108	1571.68	22.94	1703.23	149.45

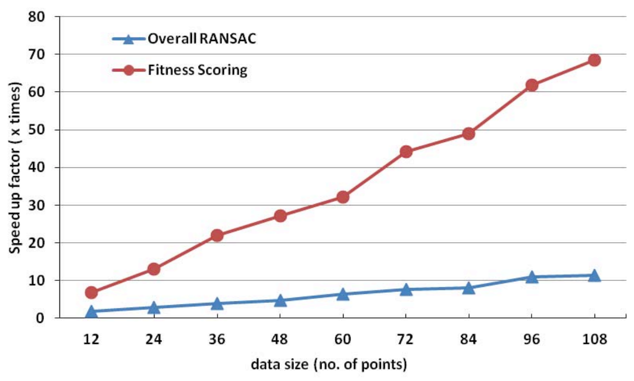


Fig. 6. Speed-up comparison between software and hardware computation

perform other tasks such as controlling and displaying. Fig. 7 shows the number of iterations, including the early termination by $T_{d,d}$ test, is much larger if the system is hardware accelerated using the same upper bound time. As described in Section III, the probability of getting good estimation increases with the number of iterations, it is a measure of increased robustness by the hardware accelerator in video processing.

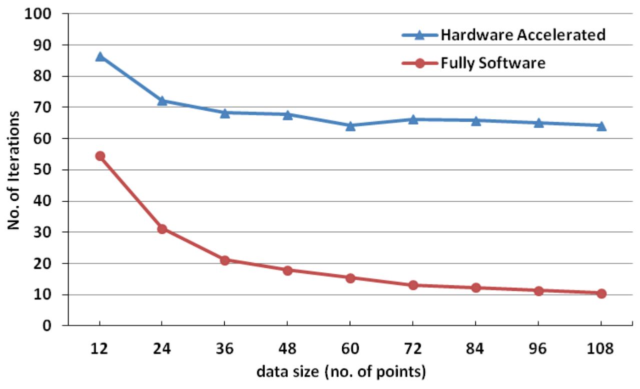


Fig. 7. A hardware/software comparison of number of RANSAC iterations performed in 25ms period

VI. Conclusion

A software/hardware co-design of RANSAC algorithm for real-time image geometry estimation is proposed and implemented in hardware using Altera DE2-115 board with NiosII Processor running on 30 fps. Fitness scoring task was chosen to be accelerated and successfully sped up the process, where the speed up factor increases with input data size. Besides, more iterations can be performed in the same amount of time using hardware accelerator over fully software process, therefore increasing the probability of getting good estimation result.

Acknowledgment

The authors would like to express their gratitude to Universiti Teknologi Malaysia (UTM) and the Ministry of Science, Technology and Innovation (MOSTI), Malaysia for supporting this research work under the Research Grant No. 01-01-06-SF1197 / R.J130000.7923.4S081.

References

[1] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[2] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[3] O. Chum and J. Matas, “Matching with PROSAC-progressive sample consensus,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005*, vol. 1. IEEE, 2005, pp. 220–226.

[4] D. Nistér, “Preemptive RANSAC for live structure and motion estimation,” *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005.

[5] M. Aly, “Real time detection of lane markers in urban streets,” in *IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 7–12.

[6] K. Dohi, Y. Hatanaka, K. Negi, Y. Shibata, and K. Oguri, “Deep-pipelined FPGA implementation of ellipse estimation for eye tracking,” in *International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 458–463.

[7] O. Chum, “Two-view geometry estimation by random sample and consensus,” Ph.D. dissertation, Czech Technical University, 2005.

[8] P. H. Torr and A. Zisserman, “Mlesac: A new robust estimator with application to estimating image geometry,” *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[9] M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, “FPGA implementation of the robust essential matrix estimation with RANSAC and the 8-point and the 5-point method,” in *Facing the Multicore-Challenge II*. Springer, 2012, pp. 60–71.

[10] B. Tippetts, S. Fowers, K. Lillywhite, D.-J. Lee, and J. Archibald, “FPGA implementation of a feature detection and tracking algorithm for real-time applications,” in *Advances in Visual Computing*. Springer, 2007, pp. 682–691.

[11] O. Chum and J. Matas, “Randomized RANSAC with T_d,d test,” in *Proc. British Machine Vision Conference*, vol. 2, 2002, pp. 448–457.

[12] P. H. S. Torr, “Bayesian model estimation and selection for epipolar geometry and generic manifold fitting,” *International Journal of Computer Vision*, vol. 50, no. 1, pp. 35–61, 2002.

[13] O. Chum, J. Matas, and S. Obdrzalek, “Enhancing RANSAC by generalized model optimization,” in *Proc. of the ACCV*, vol. 2, 2004, pp. 812–817.

[14] S. Martelli, R. Marzotto, A. Colombari, and V. Murino, “FPGA-based robust ellipse estimation for circular road sign detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2010, pp. 53–60.

[15] Z. Yaniv, “Random sample consensus (RANSAC) algorithm, a generic implementation,” *Imaging*, 2010.

[16] [Online]. Available: https://www.sdms.afrl.af.mil/