Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 13, Number 1, pages 57–58, January 2024

Hardware Implementation for Real-Time Homography Transformations

Ikko Uehara and Chikatoshi Yamada
Dept. Electronic and Communication Systems
National Institute of Technology, Okinawa College
Okinawa, Japan
{ac224602@edu., cyamada@}okinawa-ct.ac.jp

Shuichi Ichikawa
Dept. Electrical and Electronic Information Eng.
Toyohashi University of Technology
Aichi, Japan
ichikawa@tut.jp

***Abstract*—This study aims to determine the optimal method for implementing real-time Homography transformation using FPGA. We performed simulations of Homography transformation for each pixel and image region.**

***Index Terms*—Homography, pixel, area, FPGA**

I. Introduction

Homography transformation is one of the most basic image transformation techniques. In recent years, the demand for the multi-function cameras and applications have been increasing for the use of remote classes and remote work. In remote classes, a blackboard may be projected and distributed by a camera. However, depending on the positions of the camera and the teacher, it may be difficult to read the blackboard. One of the solutions to this problem is to set up the camera so that it is not covered by the teacher and to perform Homography transformation. In order to achieve this during remote classes, Homography transformation must be performed in a time shorter than the input interval for videos [1].

II. Principle of Homography Transformations

Homography transformation is the transformation of an arbitrary quadrilateral into an arbitrary quadrilateral. The shape of the quadrilateral can be transformed by computing the coordinates corresponding to each point. In general, when the coordinates (x, y) of a quadrilateral are transformed to the corresponding coordinates (x', y') by Homography transformation, using the following equation.

$$x' = \frac{ax + by + c}{gx + hy + 1} \quad (1)$$

$$y' = \frac{dx + ey + f}{gx + hy + 1} \quad (2)$$

In this study, as shown in Fig. 1, lines E, F, E', F' are drawn to divide rectangle ABCD before deformation and rectangle A'B'C'D' after deformation in the same ratio, and the pixel values of the rectangle before deformation are put into the coordinates of the corresponding rectangle after deformation.

In the Homography transformation of this study, the cross-ratio method was used. The cross-ratio is the ratio of AC:CB and AD:DB when there are four points A, B, C, and D on a line. As shown in Fig. 2, there are four straight lines from point O, and two overlapping lines n and m are drawn for those lines, and the points of contact A, B, C, D, A', B', C', D' are given. The following equation is expressed as a compound ratio.

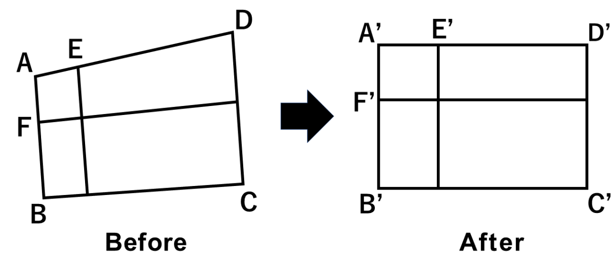

Fig. 1. Homography Transformation diagram (coordinates).

$$\frac{AC \cdot BD}{AD \cdot BC} = \frac{A'C' \cdot B'D'}{A'D' \cdot B'C'} \quad (3)$$

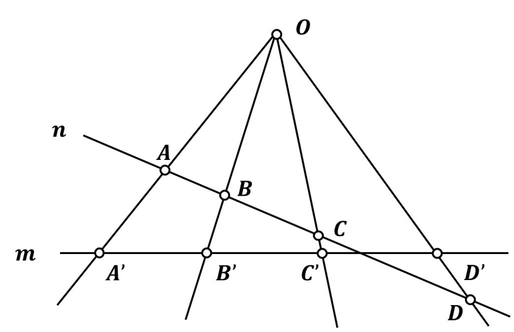

Fig. 2. Multiple ratio of distance [2].

III. Proposed method

In order to investigate a suitable program for real-time Homography transformation, a program for Homography transformation [3] is implemented on FPGA and the results of transformation speed are compared. Two methods are examined for comparison. The size of the image used is 1440 × 810.

Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 13, Number 1, pages 57–58, January 2024

$$AF : AB = A'F' : A'B' \quad (4)$$
$$AE : AD = A'E' : A'D' \quad (5)$$

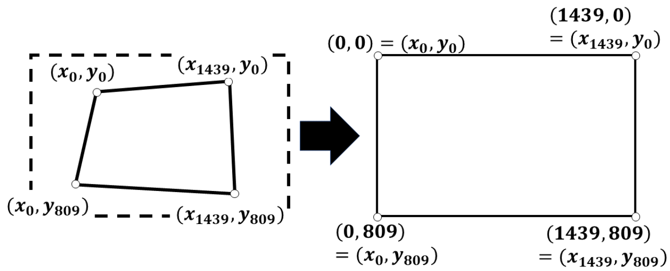

Fig. 3. Homography Transformation diagram.

Table 1 shows implementation environment.

TABLE I
IMPLEMENTATION ENVIRONMENT

CPU	Intel Core i7-11800H 2.30 · 4.60GHz
FPGA	Zynq UltraScale+ MPSoC ZCU104
SW	Vitis 2021.1

A. Per-pixel calculation

For the per-pixel calculation, a program is implemented on FPGA to find the coordinates in the area to be transformed in the pre-transformed image corresponding to the coordinates of one pixel in the post-transformed image. We will also investigate the limits of parallel processing and optimize the program by combining it with sequential processing.

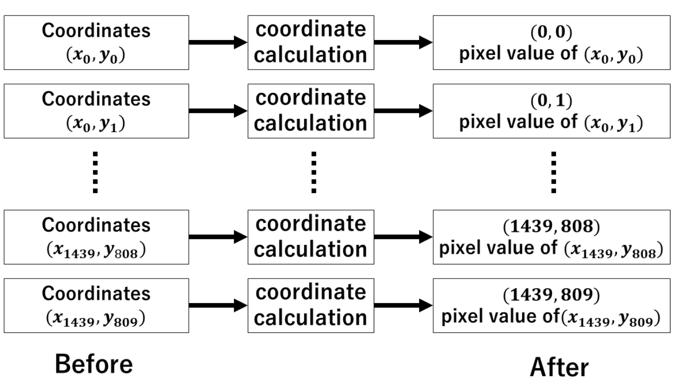

Fig. 4. Pixel calculation.

B. Per area Calculation

The image is divided into regions to be transformed, and each region is processed by a projective transformation. Optimization of the segmentation method is performed to see if there is any difference in computation speed by changing the method of segmentation. The projective transformation of the segmented regions is implemented in parallel processing to speed up the process, and the speed is measured.

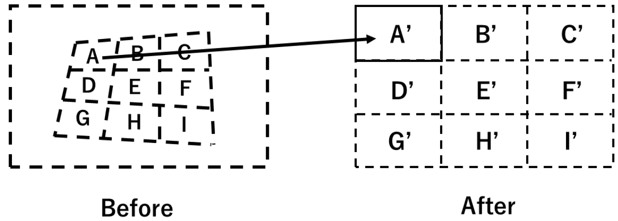

Fig. 5. Image of area calculation.

IV. Simulation Results

A simulation run of our method showed that the computation time of a single pixel was 484 ns. If all pixels were computed sequentially, it would take about 0.565 sec. In the region-by-region processing, the region was divided by x-coordinates. The coordinates (0,0) to (0,809) of the transformed image were calculated as a single region, and it was found to take 9427 ns. If all regions are calculated sequentially, it would take approximately 0.0136 sec.

TABLE II
PROGRAM COMPARISON RESULTS

Label	Clock cycle [ns]	Cycles	Latency [ns]
single pixel	11	44	484
area	11	857	9427

Table II shows that the per-pixel calculation is faster than the per-region calculation. In the case of sequential processing, the per-pixel calculation takes about 0.565 sec and the per-region calculation takes about 0.0136 sec, so the per-region calculation is faster. However, when parallel processing is implemented, it is expected to be faster than sequential processing. It is also expected that the per-region calculation will be faster due to parallel processing, and that the speed will change as the regions are changed.

V. Conclusions

In FPGA implementation, parallel processing is adopted to enhance the processing speed. since the processing speed for each region is considered to be dependent on the method of division, parallel processing is implemented based on how the speed changes when the region is divided as shown in Fig. 5. In addition, we will implement a program that can automatically detect the regions for Homography transformation using edge detection.

References

[1] Y.Tokiwa, Y.Kanazawa, "Comparison of acceleration methods for homography computation," IPSJ SIG, Vol.2014-CG-156, No.6, September 2014

[2] N.Chiba, H.Kano, M.Minoho, M.Yasuda, "Feature-Based Image Mosaicing," IEICE Trans. Information and Systems (Japanese Edition), Vol. J82-D2, No.10, pp.1581-1589, October 1999

[3] I.Uehara, H.Kano, H.Kamehama, C.Yamada, "Comparison of Processors for Real-time Homography Computation," IEEJ Transactions on Industry Applications, Vol.143, No.2, p.106-111, February 2023