Journal of Multimedia Information System VOL. 6, NO. 1, March 2019 (pp. 7-14): ISSN 2383-7632(Online)
https://doi.org/10.33851/JMIS.2019.6.1.7

Virtual Prototyping of Area-Based Fast Image Stitching Algorithm

Lakshmi Kalyani Mudragada[1]*, Kye-Shin Lee[1], Byung-Gyu Kim[2]

Abstract

This work presents a virtual prototyping design approach for an area-based image stitching hardware. The virtual hardware obtained from virtual prototyping is equivalent to the conceptual algorithm, yet the conceptual blocks are linked to the actual circuit components including the memory, logic gates, and arithmetic units. Through the proposed method, the overall structure, size, and computation speed of the actual hardware can be estimated in the early design stage. As a result, the optimized virtual hardware facilitates the hardware implementation by eliminating trail design and redundant simulation steps to optimize the hardware performance. In order to verify the feasibility of the proposed method, the virtual hardware of an image stitching platform has been realized, where it required 10,522,368 clock cycles to stitch two 1280 × 1024 sized images. Furthermore, with a clock frequency of 250MHz, the estimated computation time of the proposed virtual hardware is 0.877sec, which is 10x faster than the software-based image stitch platform using MATLAB.

Key Words: Area-Based Fast Image Stitching, Hardware Platform, Virtual Hardware, Virtual Prototyping.

I. INTRODUCTION

Image stitching is a technique for combining multiple images into one single image captured from different cameras (sources) to generate a panoramic image. Image stitching can reduce redundant information in multiple sets of images, increase the image storage capability, and enable more effective views of the real world [1]. Due to the numerous advantages, image stitching is widely used for self-driving cars/drones, 3D mapping, defense systems, and medical imaging [2] – [4]. However, due to the intense computing most of the image stitching hardware has been implemented with extremely high cost servers or high performance, multi-GPU/DSP platforms which require enormous computing and power usage [5] – [8].

On the other hand, the demands on customized image processing hardware that can perform more specific tasks are increasing due to emerging applications such as unmanned vehicles, remote sensing, and mobile computing, where the memory size, computational resources, and power are all limited [9]. That is, specific hardware implementation is needed when general purpose computers are not suitable due to constraints on speed, size, and energy consumption. In this case, even the general purpose GPUs cannot be used due to the power consumption that can easily exceed 100W. However, while image stitching platforms can be easily implemented on general purpose computers, hardware implantation imposes many challenges due to the contradiction between limited hardware resources and the requirements for high performance. As a result, it is still challenging to realize image stitching, hardware for embedded real-time applications. Another difficulty that limits the hardware implementation is that accurate performance – computation speed, power, and area estimation is not easy in the early design stage. This is only possible after the gate or transistor level implementation has been completed. Although there are open source hardware description language that support image processing algorithms such as the VHDL, it is a time consuming job to convert the VHDL code into the circuit level and run a series of simulations to figure out the hardware performance. To make matters worse, this iteration should to be repeated several times to end up with an optimized hardware performance. Therefore, in order to extend the applicability of the hardware-based image stitching platforms, there is an

Manuscript received March 08, 2019; Revised March 15, 2019; Accepted March 17, 2019. (ID No. JMIS-19M-03-006)
Corresponding Author (*): Lakshmi Kalyani Mudragada, ASEC260, Akron OH 44304, USA, 2344177717, lm135@zips.uakron.edu
[1]Dept. of Electrical and Computer Engineering, The University of Akron, Akron, Ohio, USA, klee3@uakron.edu
[2]Dept. of IT Engineering, Sookmyung Women's University, Seoul, Korea, bg.kim@sm.ac.kr

urgent demand for finding new design approaches that can facilitate the hardware design.

In this paper, an algorithm for area-based image stitching is proposed, which does the comparison of two images, find the overlap and stitch two images. Comparing two images takes more computations when compared to other two steps, which can be reduced by image binarization. In addition to that we reduced the image size by scaling. Further, we propose a virtual prototyping design approach for an area-based image stitching hardware. With the proposed approach, the overall structure, size, and computation speed of the actual hardware can be estimated in the early design stage. As a result, the optimized virtual hardware facilitates the hardware implementation by eliminating trail design and redundant simulation steps. Furthermore, the performance of the proposed virtual hardware has been compared with software-based image stitching algorithms.

II. AREA-BASED IMAGE STITCHING WITH BINARY IMAGES

This section describes the area-based image stitching algorithm using binary images (1-bit pixels) which can speed up the computation by simplifying the operations. In this case, pixel comparison between the two images can be realized with a simple logical operation instead of subtraction and multiplication. The area -based image stitching algorithm is converted into a virtual hardware, where the hardware size and computation time can be estimated.

2.1 Algorithm

Figure 1 shows area-based image stitching algorithm. First two input images (24bit color) are read which have overlapping area. A pixel color in an image is a combination of three colors Red, Green, and Blue (RGB). The RGB color values are represented in three dimensions XYZ, illustrated by the attributes of lightness, chroma, and hue. The quality of a color image depends on the color represented by the number of bits the digital device can support. If each Red, Green, and Blue occupy 8 bits, then the combination of RGB occupies 24 bits and supports 16,777,216 different colors. The 24 bits represent the color of a pixel in the color image. The grayscale image has represented by luminance using 8 bit value. The luminance of a pixel value of a grayscale image ranges from 0 to 255. The conversion of a color image into a grayscale image is converting the RGB values (24 bit) into grayscale value (8 bit).

Furthermore, the gray scale (8 bit) input image is converted into binary images. The binary image can be obtained by comparing each pixel with a threshold level – usually the mid-level of the minimum and maximum pixel value, and representing the pixel value with either 1 (pixel intensity greater than the threshold) or 0 (pixel intensity less than the threshold), where 1 and 0 stands for the white and black level, respectively. In this case, the threshold, 128 is suitable for our application since it still preserves useful information. In addition, by using binary images, the amount of computations to compare every pixel of two images is significantly reduced, which enables a very fast image stitching.

In addition, the size of the binary image area reduced by using a scaling factor K. To rescale the image, consider a KxK pixel data in sequence, such that all the information of the image is maintained. Each KxK data is replaced with a single bit (1 or 0). If 1 is occurring more pixel than 0 in KxK pixel data, replace the KxK pixels with 1. Similarly, if 0 is occurring more pixels, replace KxK pixels with 0. In this way we can reduce the original binary image (size MxN) to $M/K \times N/K$.

Next, we start comparing the two $M/K \times N/K$ sized images. Each pixel in the image is being compared to all the pixels in this case. An XOR operation is used to compare the pixels. How to compare the two images is fully discussed in section 2.2 and 2.3.

After comparing two images we can find the optimum overlap between the two images. When two pixels that are compared are same the XOR operation output is 0. If the two pixels that are compared are not same the XOR operation output is 1. All the pixels that are compared are added and divided by the number of comparisons (i.e., average of XOR outputs). Since XOR operations states that same inputs 0 output, the overlapped area gives the 0 output or the maximum overlapped area gives the minimum average value. Therefore, the minimum average value is used to determine the optimum overlapped area.

After finding the optimum overlapped area of two images, we use grayscale images for stitching. Using grayscale data takes more time for finding the optimum overlap area. Thus we only use binary images for comparing the pixels and finding the overlapped area. Finally, the stitch image is obtained by removing the overlap area in the image-2, and concatenating the two images

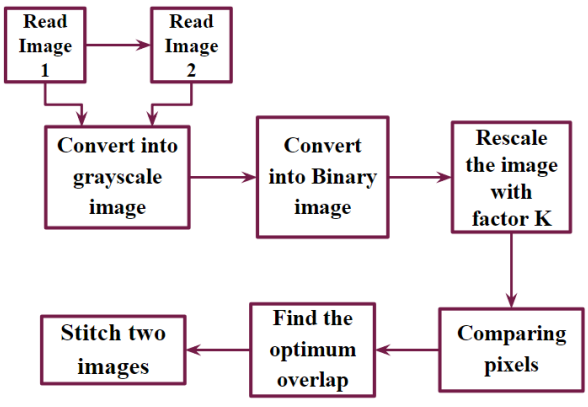


Fig. 1. Area-based Image Stitching Algorithm.

Journal of Multimedia Information System VOL. 6, NO. 1, March 2019 (pp. 7-14): ISSN 2383-7632(Online)
https://doi.org/10.33851/JMIS.2019.6.1.7

2.2 One Dimensional Image Stitching

Area-based image stitching can be performed in either one dimension or two dimensions. Figure 2 shows the one dimensional approach, where a 3X3 image is considered for simplicity. In one dimensional approach, the image is being shifted either horizontal or vertical direction only. In this example, the image-2 is shifter horizontally. During the first iteration, column-3 of image-1 and column-1 in the image-2 are compared. The shaded pixels indicate the pixels being compared. The two pixels in the shaded area (each from image-1 and image-2) are compared, which is realized by XOR operation. The result is 0 when the two pixels match otherwise the result is 1. Next, the XOR results are all added up and divided by the number of comparing pixels, (for first iteration 3). This value corresponds to the average of pixel comparison of each iteration. Similarly, in the second and third iteration, 6 and 9 pixels (each from image-1 and image-2) are compared. After comparing all the pixels, the averages of pixel comparison at each iteration are again compared, where the minimum average value leads to the optimum overlap. It is observed that from figure 2, first iteration average value is 0, second iteration average value is 0.66 and the third iteration average value is 0.55. Therefore area compared in first iteration has overlap. Figure 3 shows the one dimensional image stitching results based on the optimum overlap. The overlapped area highlighted in red color.

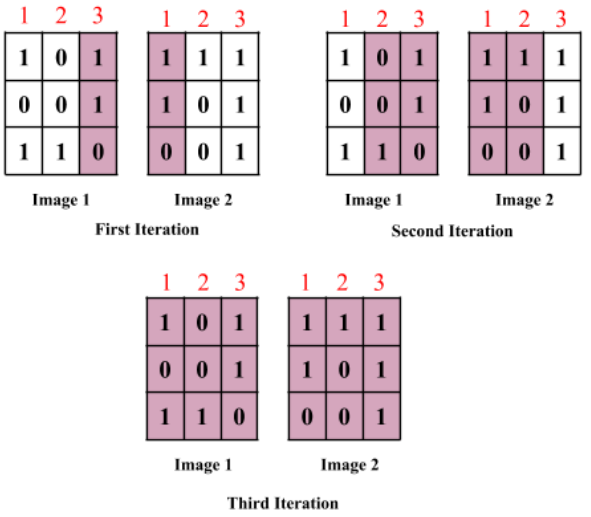


Fig. 2. One Dimensional Pixel Comparison.

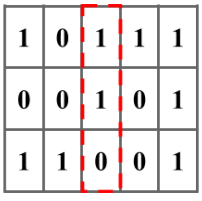


Fig. 3. One Dimensional Image Stitching Result..

2.3 Two Dimensional Comparison

For the one dimensional image stitching, image-1 and image-2 are shifted in the horizontal direction to vary the pixel positions to be compared. However, in the two dimensional image stitching, image-1 is shifted both in the horizontal and the vertical direction to vary the pixel position to be compared. As a result, it is more accurate than the one dimensional case, though this leads to more complicated computations. Figure 4 shows the two dimensional image stitching algorithm using two 3X3 images, where the shaded pixels indicate the pixels being compared. Similar to the one dimensional case, there are column 1, 2, 3 overlaps (obtained by just shifting image-1 horizontally). However, in addition to this for each column overlap, image-1 is shifted vertically that ends up with 5 different areas being compared. Therefore, there are total 15 iterations (5 iterations for each column overlap). The procedure for finding the average of pixel comparison for each area being compared is exactly same as the one dimensional case. Assuming the second iteration of column 2 overlap shows the minimum average of pixel comparison among the other overlap cases, the two dimensional image stitching result is shown in figure 5, where extra 0's are filled at the blank area that are generated after combing the two images (the two yellow pixels). The overlapped area (pixels named A, B, C, D) is highlighted in red color.

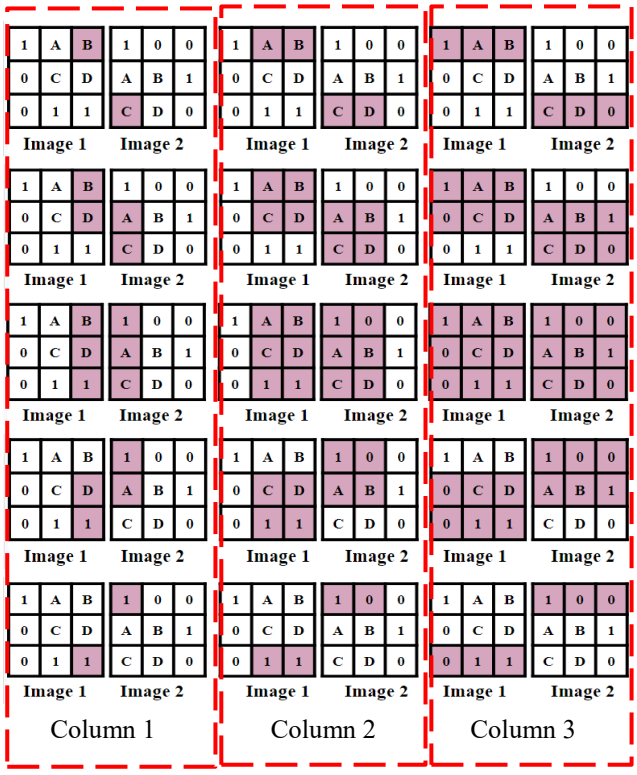


Fig. 4. Two Dimensional Pixel Comparison.

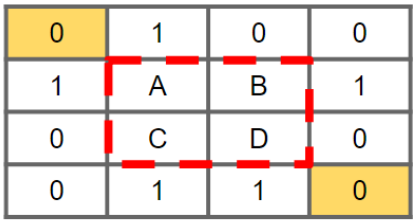


Fig. 5. Two Dimensional Image Stitching Result.

III. VIRTUAL PROTOTYPING OF AREA-BASED IMAGE STITCHING ALGORITHM

3.1 CONCEPT OF VIRTUAL PROTOTYPING

The basic concept of the proposed virtual prototyping is converting the conceptual or algorithm level image stitching platform into an actual hardware that consists of the memory, logic, or arithmetic unit. In addition, this allows estimating the memory size, number of adders, subtractors, and logic gates that are present in the image stitching hardware. Figure 6 shows the concept of the proposed virtual prototyping which is applied to an area-based image stitching algorithm, where Figure 6 (a) shows the conceptual representation of the algorithm and Figure 6 (b) shows the resulting virtual hardware obtained through virtual prototyping. The virtual hardware is equivalent to the area-based image stitching algorithm which performs the same operation, however the conceptual blocks are replaced with actual circuit components such as the memory, logic gates, and arithmetic unit. As a result, from the virtual hardware, the size and computation time of the actual hardware can be estimated. The virtual prototyping approach facilitates the hardware design by image processing circuits by enabling performance estimation at the early design stage.

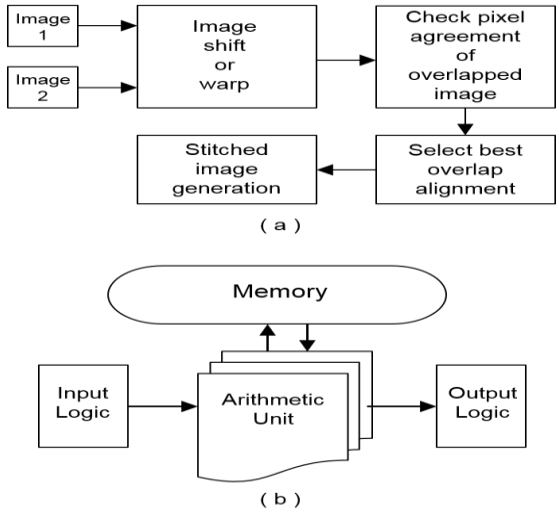


Fig. 6. Concept of virtual prototyping (a) Area-based mage stitching algorithm (b) Virtual hardware.

3.2 VIRTUAL HARDWARE IMPLEMENTAION

A virtual hardware platform is designed for the proposed area-based image stitching algorithm. Figure 7 shows the virtual hardware of area-based image stitching algorithm for the MXN image. Three different memories are used to store the image data. Memory-1 is used to store the image-1 data throughout the process. Similarly, memory-2 is used for image-2. The stitched image data is stored in Memory-3. Memory-1 and memory-2 are indicated in pink color. And memory-3 is indicated in yellow color at the bottom left corner. Writing and reading for memory-1 and memory2 is performed simultaneously. Since two memories are used, the computation time to read and write operations is halved. Control 1 block is used to write the grayscale image data (0 to 255) into memory-1 and memory-2. The converted grayscale image data are written into the memories. Next control 2 block is used to read the grayscale image data from memory-1 and memory-2 to convert grayscale image data into binary data (1 or 0).

3.2.1 Converting the grayscale image into binary image

A comparator is used to compare each grayscale pixel in the image-1 and image-2 with 128. If the value of the grayscale image data is less than 128, output of the comparator is 0, if the value of the grayscale image data is greater than 128, and the output of the comparator is 1. Next, control 3 block is used to write the converted binary data into the respective memories, which is represented as the green color. Writing the binary data into the memories is realized in a pipelining process. Reading the grayscale data, converting the grayscale into binary takes 1 cycle. In the next cycle, reading the grayscale data and writing the previously converted binary data into the memories is performed. This process continues until all the grayscale image data is converted into the binary data.

3.2.2 Image size scaling

Now we need to scale the binary data with the scaling factor K (2, 4, 8, 16, etc.). Reducing the size of the image is helpful for reducing the computation time. We used different scaling factors and found that, the scaling factor K cannot be more than 8. If K is greater than 8 (i.e., 16 or 32 etc.), the image information is changed while rescaling, which may not lead to the accurate image stitching result. This conclusion is made based on MATLAB results. To rescale the image, we used a combinational circuit with XOR gate, two counters and a comparator. Control 4 block is used to read the binary data, which is in yellow color. To scale an image we read KxK (2X2 or 4X4 or 8X8 etc.) image data and compare each bit with 1, for comparing we used XOR gate. The XOR gate output can be 1 or 0, which is applied to the counters. There are two counters C in figure 7, which counts the number of 1's and the number of 0's. Next, we compare the total number of 1's and 0's using a comparator. The highest occurrence of data (1 or 0) is called the rescaled binary data, which is

written into the respective memories. Next, write the rescaled data simultaneously into the memory immediately after comparing the last bit. Control 5 block is used to write the rescaled binary data. Now we are fetching the data from memories to compare the images and find the overlap.

3.2.3 Two dimensional image comparison

Comparison to find the overlap between two images is being performed at this point. Read the binary data as explained in figure 4 (the shaded pixels). Control 6 block in figure 7 shows the control to read the binary data of image-1 and image-2 from the memories. Comparing the pixels in hardware implementation is realized by a combinational logic with XOR gate, adder and divider. Each area is compared with XOR gate and the results are summed using an adder. Next, a divider is used to divide the added value by the number of comparisons performed. For the example used (3X3 image) we get 15 divider outputs. There is no need to use another memory to store the average values of the compared areas. Figure 7 shows the logic to find the minimum, and further used to find the position of the overlapped area. The minimum value of the divider output is considered to be overlapped area. There is a possibility to be more than one minimum value. There is also the possibility of overlapping more than one arrangement that has the minimum pixel comparison average. In this case, the overlap with the maximum number of pixels is selected as the optimum overlap. Furthermore, the location and size of the optimum overlap area is obtained by the overlap location calculator block, which consists of combinational logic. After finding the optimum overlapped area, control 7 block is used to read the grayscale image data to perform the stitching operation. Since the binary image is not clear for naked eye, we use grayscale image data instead of binary data to stitch two images. Stitching operation is realized with a combinational logic. Control 8 block represents the control to write the stitched data into memory 3. The yellow color represents memory 3.

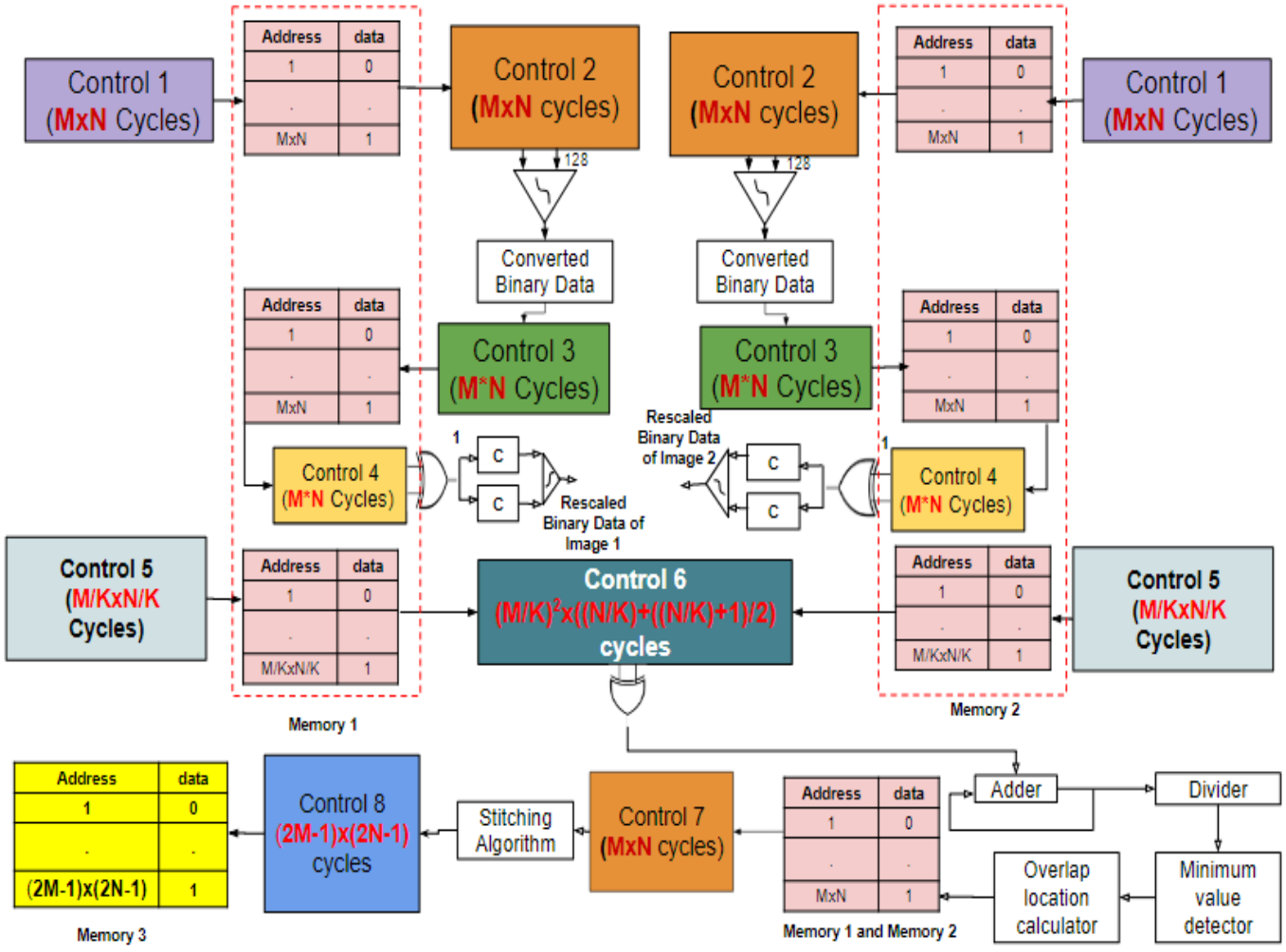


Fig. 7. Virtual Hardware of Area-based Image Stitching Platform.

IV. COMPUTATIONAL COST

The computational cost of virtual hardware can be calculated depends on the number of cycles taken to perform each operation that is shown in Figure 7. We considered MxN image to perform the analysis

Writing and reading operation in memory-1 and memory-2 is performed simultaneously. It takes $\mathbf{M \times N}$ cycles to write the grayscale image data into the memories, to read the grayscale image data from the memories to convert into binary data, reading the grayscale data (M × N), converting grayscale data to binary data and writing the binary data (+1) into the memories can be realized using pipelining. It takes $\mathbf{M \times N + 1}$ cycles to read and write the converted binary data into the memories. Reading KxK binary data bits to rescale the binary image also takes M × N cycles. Again, we can write the rescaled bit into the memory using pipelining. Writing the rescaled data also takes $\mathbf{M \times N + 1}$ cycles.

The number of cycles to read the binary data from the two images to perform comparison (XOR operation, addition and division) is given by.

$$\frac{\frac{M^2}{K} \times \frac{N}{K} \times \left(\frac{N}{K}+1\right)}{2} \quad (1)$$

After comparison and finding the optimum overlap, to stitch the images we need to read the grayscale image which takes **M*N** cycles. To write the final stitched image data (i.e., to be displayed) takes **(2M-1) * (2N-1)** cycles**.** This equation gives the worst case scenario of the overlapped area (i.e. if the overlapped area is only one pixel). The total number of cycles required to perform image stitching is given as.

$$\mathbf{4 \times (M \times N) + 2} + \left(\frac{\frac{M^2}{K} \times \frac{N}{K} \times \left(\frac{N}{K}+1\right)}{2}\right) + \mathbf{((2M-1) \times (2N-1))}. \quad (2)$$

Figure 8 shows the computational cost of stitching 1280x1024 sized image. We considered M=1280 and N=1024 in the above analysis and calculated the number of cycles required for each operation.

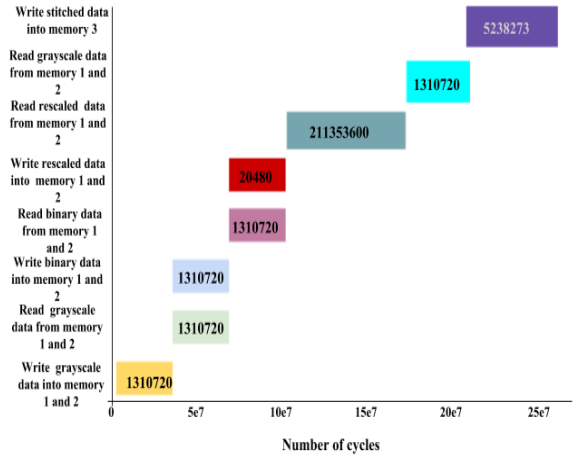


Fig. 8. Number of cycles required for each operation to stitch two 1280x1024 image.

V. RESULTS

The actual size and computation time of the area-based image stitching, hardware can be estimated from the virtual hardware. Table 1 shows the estimated computation time of the area-based image stitching, hardware compared with the software algorithm realized using MATLAB. The computation time of the virtual hardware is obtained by using equation 2 with different clock frequencies (250MHz, 500MHz, 750MHz and 1GHz) and image scaling factor K (2, 4, 8, and 16). The computation time decreases as the clock frequencies and scaling factor K increases. However, increasing K has the risk of losing the useful information contained in the image, thus can lead to poor stitching results.

Based on the simulation results, K=8 still gives good stitching quality with relatively fast stitching time. With K=8 and clock frequency of 250 MHz, the computation time of the virtual hardware shows 0.887 Sec which is 10x faster than MATLAB

Table 1. Computation time of MATLAB and virtual hardware platform for stitching two 1280X1024 images.

K value	MATLAB (sec)	Virtual Hardware (sec)			
		Clock Frequency			
		250 MHz	500MHz	750MHz	1GHz
2	599.06	215.2	107.6	71.736	53.802
4	51.77	13.516	6.758	4.503	3.379
8	8.2	**0.887**	0.443	0.295	0.221
16	3.351	0.0951	0.047	0.0317	0.0239

Figure 9 shows the result of two dimensional area-based image stitching. The results are obtained from MATLAB. The input images are shown in figure 9a, here grayscale images are considered as input images, i.e., after converting into a grayscale image to binary image, and the stitched image is shown in figure 9b. It is observed that the stitching operation is implemented precisely and there is no disruption near the overlapped area in the stitched image. After stitching the

leftover area is filled with 0 values to make it a complete image (left bottom and right top corners).

Input Image 1

Input Image 2

(a)

Output Image

(b)

Fig. 9. Image stitching result of 1280x1024 images for K=8. (a) input images (b) output image

VI. CONCLUSION

This work presented a virtual prototyping approach for a fast area-based image stitching hardware platform. If implemented with actual hardware (FPGA or Integrated Circuit). The area-based image stitching algorithm has been implemented in MATLAB and the result was shown. Further the virtual hardware was implemented and the computation cost of software and hardware implementation was discussed. With clock frequency of 250MHz, the estimated computation time of the proposed virtual hardware was 0.877sec, which was 10 times faster than the software-based image stitch platform using MATLAB.

REFERENCES

[1] J. H. Chen, and C. M. Huang, "Image stitching on the unmanned air vehicle in the indoor environment," in *Proc. SICE Annual Conf.*, Aug. 2012, pp. 402-406.

[2] S. Chen, "QuickTime VR – An image based approach to virtual environmental navigation," *SIGGRAPH 95*, vol. 29, pp. 29-38, 1995.

[3] R. Szeliski and H. Shum, "Creating full view panoramic image mosaics and environmental maps," *SIGGRAPH 97*, vol. 31, pp. 251-258, 1999.

[4] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Computer Vision*, vol. 74, pp. 59-73, 2007.

[5] M. Qasaimeh and A. Sagahyroon, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," *IEEE Transactions on Computational Imaging*, vol. 1, no. 1, pp, 56-70, March 2015.

[6] J. H. Wang and S. Zhong, "An Embedded System-on-Chip Architecture for Real-time Visual Detection and Matching," *IEEE Transaction on circuits and systems for video technology*, vol.24, no.3, pp.525-538, march 2014.

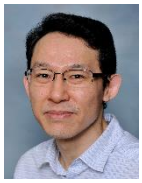
[7] S. Zhong and J. H. Wang, "A real-time embedded architecture for SIFT," *Journal of Systems Architecture*, vol.59, pp. 16–29, 2013.

[8] V. Bonato and E. Marques, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," *IEEE Transactions on Circuits And systems For Video Technology*, vol. 18, no. 12, December 2008.

[9] T. V. Huynh, "Deep neural network accelerator based on FPGA," in *Proc. IEEE Information and Computer Science*, Nov. 2017, pp. 254-257.

Authors

Lakshmi Kalyani Mudragada received the B-Tech degree from Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, India in Electronics and Communication Engineering in 2013, the M-Tech degree from Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, India in VLSI Design in 2015. Pursing MS degree at University of Akron, Akron, OH, in Electrical Engineering since 2017. Her research interests include, image processing applications, VLSI design and signal processing.



Kye-Shin Lee(S'02–M'06) received the B.S. degree from Korea University, Seoul, Korea, in 1992, the M.S. degree from Texas A&M University, College Station, TX, USA, in 2002, and the Ph.D. degree from the University of Texas at Dallas, Richardson, TX, USA, in 2005, all in electrical engineering. He was with Texas Instruments Inc., Dallas, TX, USA, from 2005 to 2008. In 2009, he was an Assistant Professor with the Department of Electronics, Sun Moon University, Asan-si, Chungnam, Korea. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, The University of Akron, Akron, OH, USA. His research interests include integrated circuits for sensors and image processing applications.

Byung-Gyu Kim has received his BS degree from Pusan National University, Korea, in 1996 and an MS degree from Korea Advanced Institute of Science and Technology (KAIST) in 1998. In 2004, he received a PhD degree in the Department of Electrical Engineering and Computer Science from Korea Advanced Institute of Science and Technology (KAIST). In March 2004, he joined in the real-time multimedia research team at the Electronics and Telecommunications Research Institute (ETRI), Korea where he was a senior researcher. In ETRI, he developed so many real-time video signal processing algorithms and patents and received the Best Paper Award in 2007. From February 2009 to February 2016, he was associate professor in the Division of Computer Science and Engineering at SunMoon University, Korea. In March 2016, he joined the Department of Information Technology (IT) Engineering at Sookmyung Women's University, Korea where he is currently an associate professor. In 2007, he served as an editorial board member of the International Journal of Soft Computing, Recent Patents on Signal Processing, Research Journal of Information Technology, Journal of Convergence Information Technology, and Journal of Engineering and Applied Sciences. Also, he is serving as an associate editor of Circuits, Systems and Signal Processing (Springer), The Journal of Supercomputing (Springer), The Journal of Real-Time Image Processing (Springer), and International Journal of Image Processing and Visual Communication (IJIPVC). From March 2018, he is serving as the Editor-in-Chief of The Journal of Multimedia Information System and associate editor of IEEE Access Journal. He also served or serves as Organizing Committee of CSIP 2011, a Co-organizer of CICCAT2016/2017, and Program Committee Members of many international conferences. He has received the Special Merit Award for Outstanding Paper from the IEEE Consumer Electronics Society, at IEEE ICCE 2012, Certification Appreciation Award from the SPIE Optical Engineering in 2013, and the Best Academic Award from the CIS in 2014. He has been honored as an IEEE Senior member in 2015. He is serving as a professional reviewer in many academic journals including IEEE, ACM, Elsevier, Springer, Oxford, SPIE, IET, MDPI, and so on.

He has published over 200 international journal and conference papers, patents in his field. His research interests include image and video signal processing for the content-based image coding, video coding techniques, 3D video signal processing, deep/reinforcement learning algorithm, embedded multimedia system, and intelligent information system for image signal processing. He is a senior member of IEEE and a professional member of ACM, and IEICE.