Department of Electrical and Computer Engineering

Part IV Research Projects

Final Report

[Multi-Processor System on Chip for Real-time Face Detection System]

Prepared by Long-Sing Wong

Project 109

Project partner Lin Yang

Supervised by Dr. Morteza Biglari-Abhari

Co-supervised by Dr. Nitish Patel

Declaration of Originality

This report is my own unaided work and was not copied from anyone or anywhere nor written in collaboration with any other person.

Long-Sing Wong

11th Sep 2017

Multi-Processor System on Chip for Real-time Face Detection

Long-Sing Wong,
Department of Electrical and Computer Engineering,
The University of Auckland, New Zealand,
lwon229@aucklanduni.ac.nz

***Abstract*— There are many applications that require face detection technology especially on an embedded system. The goal of this project is to investigate the hardware-software co-design technique and the performance difference between hardware and software component. There are many algorithms available for face detection. In this paper, we propose an algorithm to detect faces by combining pixel-based and feature-based face detection algorithm. We will also discuss the advantages and limitations of our proposed algorithm and the solution to increase accuracy rate and reduce the processing time of our purposed face detection system.**

I. Introduction

FACE detection is the fundamental building block of many modern day technologies and applications such as face recognition technology and eye tracker application. The goal of this project is to design a real-time face detection system on an embedded platform. Our aim for the real-time face detection system is to process at least one 320 by 240 pixels image within a second. The purpose of this research project is to investigate the performance difference between hardware and software component. We have applied hardware software co-design technique to design the real-time face detection system by separating the system into hardware and software component. The basic structure of the face detection system is implemented on the software component while the hardware component is implemented to speed up the bottlenecks that are identified in the software component. My main contribution in this project is to design and implement the face detection algorithm on software component and integrate the system onto the embedded platform. The purposed algorithm described in this paper combines the advantage of existing face detection algorithms and increase the accuracy rate of face detection. This paper focuses on the software component of the real-time face detection system. For the hardware implementation and methodology, please refer to my project partner's report.

This paper is presented in the following sections. Section II explains past research that has been done on real-time face detection system on an embedded platform. Section III describes the component in the embedded platform and explains the methodology of the face detection modules including system structure. Section IV explains the implementation of each face detection modules in the software component. Section V shows the experimental result of testing in the face detection system. Section VI discusses the limitations and advantages of each face detection algorithms. Section VII explains the future work of this project. Section VIII concludes the findings and result of the face detection system.

II. Related Work

A. Partially Occluded Face Detection (POFD) [1]

POFD is a fusion of feature-based method and part-based method. This algorithm could detect partially occluded face. This algorithm is divided into 2 parts:

- Locate possible face component regions.
- Conclude partially occluded face for annotation.

1) Face skin detection:

Face skin detection can be achieved by the RCT Color based segmentation algorithm (discussed in the following subsection). It detects the face region based on the skin color.

2) Face annotation:

The underlying assumption of this part of the algorithm is that the nose is present in the probe-image and few other face components might be missing. First, it searches possible nose features in the image within the face bounding area. Then it creates arrays for all possible major face components base on the color and the location of the face component. The next step is to create a threshold bound array of the possible face component. Then the algorithm can annotate a partially occluded detected face based on the threshold bound array.

B. RCT Color based segmentation [2]

RCT color based segmentation is a face region detection algorithm with a very low computational requirement. Various research shows the human facial region has a unique color distribution which is very different from the background objects. This algorithm determines if the pixels in the image are in the range of chrominance value that corresponds to the chrominance level of human skin. It uses multiple color space such as HSV color space (Hue, Saturation, Value), RGB color space and YCbCr color space. These color space are used to filter the human skin color range. This algorithm assumes the face region is always in the top part of the image so the lower part of the image is masked and ignored for the faster computational time. According to the research paper[], the research has proven the result is quite accurate and reliable.

However, this algorithm has a slight drawback. Since this algorithm detects the human face using human skin color, it won't work if the feeding image is grayscale or the color of the face is not natural (i.e. paint on the face).

C. *Boosted Cascade of Simple Features of object detection* [3]

It is a machine learning approach that can process image very accurately and fast. It uses an integral image to train the classifiers. Their proposed algorithm is based on AdaBoost which identify a small number of critical facial features from a large set of training images and yield an extremely efficient classifier. Then by combining these classifiers into a complex cascade which allows the background image to be discarded rapidly. Using cascade classifiers guarantee that the discarded regions are statistically unlikely to contain facial features. It is a very efficient algorithm and used in many real-time object detection applications.

D. *High-Performance Face Detection with CPU-FPGA Acceleration* [4]

This research paper presents an acceleration technique on the CPU-FPGA platforms using Cascade Classifier algorithm. They first map the algorithm using the integrated OpenCL environment in FPGA. They have proposed a nested structure to speed up both memory access and computing iteration. This multi-layer architecture allows parallel programming using multiple cores on the FPGA board. Performance critical classifiers are performed on the FPGA fabric for hardware acceleration while the non-critical classifiers are implemented on the host CPU. Their proposed solution achieves a speed 30 times faster compared to the CPU implementation.

E. *Hardware acceleration of Face Detection System on FPGA* [5]

This research paper presents an implementation of a face detection system accelerated on FPGA by implementing skin color segmentation algorithm using NIOS II soft-core processor and implementing filter algorithms such as the median and morphological filter in the FPGA fabric. Skin color segmentation algorithm (discussed in the subsection A) is implemented to extract the possible location of the human skin in the input image. The median filter is used as a noise filter. It discards outlier pixels from the output image of skin color segmentation hence removing salt and pepper noise. The morphological filter is an image processing operation that process images based on shapes. It extracts the human face region by outlining the shape of the face. The profiling result shows the bottleneck is in the morphological processing due to computing intensive window-based algorithm hence the morphological filter is implemented in the hardware accelerator. They concluded that hardware design is able to achieve a speedup of 250 times faster than software implementation while processing an image of 800×600 pixel size.

F. *Face Detection System using grayscale image* [6]

This research paper presents a face recognition algorithm that is implemented using Verilog and Matlab. Image test benches are read into the system and stored in memories using Matlab code. The image is then converted to grayscale and extract the lip region from the image and generate histogram equivalence base on the input image. The generate histogram equivalence are converted into binary data and feed into the Verilog program. If the binary data of the extracted region is the same, then the system detects the face. From their experimental result, their program is capable of detecting a slight expressional change of the test image and differentiate two persons by comparing their lip images.

III. Methodology

To investigate the performance difference between hardware and software implementation, the hardware-software co-design methodology is applied to design the real-time face detection system on an embedded platform. The real-time face detection system is divided into 2 different components: hardware component and software component. The software component is implemented to provide the basic structure of the real-time face detection system while the hardware component is implemented to be the accelerator of the bottleneck that is identified in the software component. We are using Altera DE1-SoC FPGA development board as the embedded platform to test our real-time face detection system.

Altera DE1-SoC FPGA device has a hard processor system and a FPGA fabric. The software component is implemented on the hard processor system while the hardware component is implemented on the FPGA fabric. Altera DE1-SoC FPGA device has an ARM Cortex-A9, dual-core processors. A Linux Ubuntu operating system is booted from the micro SD card since the software component of the system is implemented based on the Linux kernel. We have also considered using AXI bridge protocol to intercommunicate between the hard processor system and the FPGA fabric. In this section, we would focus on the software component of the real-time face detection system and the algorithms implemented on the hard processor system. For hard component implemented on the FPGA fabric, please refer to the report of my project partner. [7]

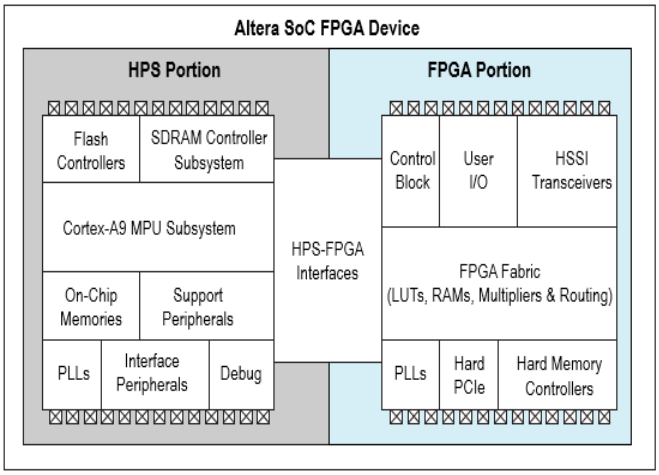

Figure 1: DE1-SoC system block diagram [8]

A. System Overview

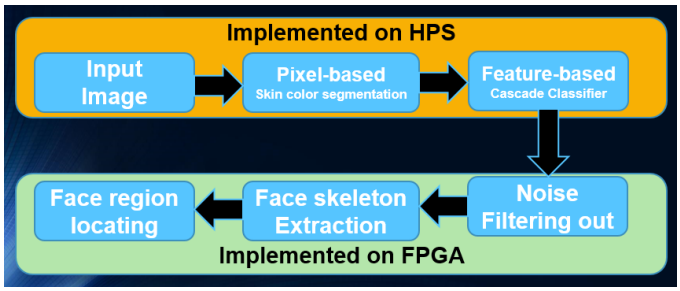

Figure 2: Face Detection System block diagram

The software component implemented on the Hard Processor System has 3 modules to provide the skeleton of the real-time face detection system: input module, pixel-based face detection module, and feature-based face detection module. An image is fed into the face detection system and be processed by the pixel-based and feature-based face detection module. The output image from the software component is transferred to the hardware component for further processing. The result processed image from the hardware component is then transferred back to the software component for display.

B. Input Module

The input module imports images from a directory to our real-time face detection system. For the purpose of testing the real-time face detection system, we have used a face detection data set from "Faces in the Wild" [9] to test the face detection system. This face detection data set includes images that vary in sizes, skin tones, position of faces and background in order to test the performance and accuracy of our face detection system under different conditions. The input module uses <dirent.h> to feed the input images into the face detection system. It is a header in the C POSIX library that contains constructs to facilitate directory traversing and it is portable between different platforms.

C. Pixel-based face detection module

Pixel-based face detection algorithm searches the entire image to detect the position of the faces. If the pixel in the image fits the definition of human faces, then that pixel is treated as part of the human face. The pixel-based face detection module is implemented based on the skin color segmentation algorithm which is one of the pixel-based face detection algorithms.

Because human skin tone has a very specific color range in various color spaces and usually very different from the background color, the pixel-based face detection module detects the possible region of human faces in an image by finding the color value of each pixel within an image. The pixel-based face detection module is implemented based on 3 color space filters to loop through every pixel in an image and filter out the pixels that do not satisfy the definition of human skin pixels. The resulting output of this module extracts all possible human skin region in the input image and masks out of range pixel.

D. Feature-based face detection module

Feature-based face detection algorithm uses a search window to extract the position of all possible facial features within an image. A search window is a small region that contains a small part of the image. The feature-based face detection module is implemented base on the Haar's Cascade Classifier algorithm to extract the possible location of facial features and eyes features.

A cascade classifier is a machine learning approach such that each classifier is trained with hundreds of positive and negative examples that are scaled to the same size. After the classifier is trained, it can be applied to the search window and move the search window across the input image to check every location using the resultant classifier. The resultant classifier consists of multiple stages that are applied to the search window until a stage is rejected or all stages are passed. If every stage in the classifier pass, then the classifier outputs the position of the search window to indicate the region of the possible facial features.

IV. Implementation

The real-time face detection system is implemented using Open Source Computer Vision Library (OpenCV). It has a C++ interface and supports Linux kernel. The OpenCV library takes advantage of multi-core processing and hardware acceleration of using Open Computing Language (OpenCL). Both pixel-based and feature-based face detection modules are implemented using OpenCV framework.

Skin Color Segmentation algorithm is implemented based on 3 color space filters. For each color space filters, it converts the color space of the input image to the specific color space based on the filter. Then for each pixel within an image, compute the color value of the pixel. If the pixel value does not satisfy the filter constraint (discussed in the following section), then the pixel is masked otherwise keep the input pixel.

```
Mat ColorSpaceFilter(Mat inputImage, Mat outputImage)
{
        convertColorSpace(inputImage, outputImage, Converted_color_space);
        for (int i = 0; i < inputImage.rows; i++)
        {
                for (int j = 0; j < inputImage.cols; j++)
                {
                        Vec3b ColorValue = inputImage.at<Vec3b>(Point(j,i));

                        //if the color value of a pixel does not satisfy the constraint
                        if (!condition)
                        {
                                //turn the pixel black
                                outputImage.at<Vec3b>(Point(j,i)) = Vec3b(0,0,0);
                        }
                }
        }
        return outputImage;
}
```

Figure 3: Pseudo code of color space filter

A. RGB color space filter

RGB is the most commonly used color space. This color space filter can remove some color pixels which are out of range of the general skin color. However, the main drawback of using RGB color space filter is it cannot differentiate the effect of luminance. Hence the resultant output is not very accurate and requires further processing from other color space filters.

Based on the literature on skin color segmentation, the RGB color space filter is implemented based on the following rules:

$$0.84G + 44 > B > 0.84G - 14 => Skin$$
$$0.78G + 42 > B > 0.79G - 67 => Skin$$

If both rules are satisfied, then the pixel is considered as a human skin pixel otherwise the pixel is masked.

Figure 4: RGB color space filter output

B. YCrCb color space filter

YCrCb color space is a type of color space that is used as a component of the color image in digital photography and video system. Y is the luma component which represents the brightness of an image while Cr and Cb are the red and blue difference chroma components respectively. The YCrCb color space depends on the RGB color space since the YCrCb color space is a way to represent RGB color information after encoding.

Based on the literature review on skin color segmentation, the YCrCb color space filter is implemented based on the following rules:

$$128 > \text{Cb} > 102 => \text{Skin}$$
$$160 > Cr > 125 => Skin$$

If the pixel value satisfies both rules, then the pixel is considered a human skin pixel otherwise the pixel turns black.

Figure 5: YCrCb color space filter output

C. HSV color space filter

HSV color space provides a better perceptive of human skin tone. H is hue which shows the basic color in an image. S is saturation which shows how the hue is diluted by the white light. V is a value that shows the intensity of the color.

Based on the literature review on skin color segmentation, H value of human skin tone occupies a very narrow range near the maximum and minimum possible values:

$$255 > H > 240 => Skin$$
$$19 > H > 0 => Skin$$

If both rules are satisfied, then the pixel is considered as a human skin pixel otherwise the pixel is masked.

Figure 6: HSV color space filter output

D. Haar's cascade classifier

The feature-based face detection module is implemented based on the Haar's Cascade Classifier. OpenCV already contains many pre-trained classifiers for face, eyes, mouths, etc. These classifiers are stored as XML files and can be used to identify facial features. Face Cascade Classifier can detect the whole input face but it cannot detect partial faces since it requires every facial feature to pass all stages. In order to detect the partial face as well, we have decided to use both face and eye cascade classifiers to test input faces. Although using 2 cascade classifiers increase the processing time of each image, the feature-based face detection module can detect partial face assuming at least one eye feature is in the input image hence the accuracy of the face detection system increases.

```
Algorithm LocateFeatureBox is:
        input: input image, cascadeClassifier
        output: coordinate of the detected feature box

        vector<Rect> LocateFeatureBox(Mat inputImage,string cascadeClassifier)
        {
                CascadeClassifier cascade;
                vector<Rect> features;

                //load the cascade
                cascade.load(cascadeClassifier);

                //Detect the cascade features
                cascade.detectMultiScale(imgOriginal, features, 1.2 , 3);
                return features;
        }
```

Figure 7: Pseudo code of cascade classifier

E. Multi-Processing

After the input image is processed by both face detection modules, the outputs from these modules are combined to

produce a 2D array representing the output image. If the pixel is masked, the corresponding element in the 2D array output "0" otherwise output "1". If one of the facial feature boxes from the feature-based face detection module is within the face region from the pixel-based face detection module, the system concludes there is a face in the input image. The output 2D array is transferred to the hardware accelerator via the AXI bridge protocol for further processing. Please refer to my partner's report for the hardware implementation.

Figure 8: Output Image of Software component (with eye feature box)

To reduce the processing time of each input image, the parallel processing technique is applied. Since the input image of both pixel-based and feature-based detection modules is the same, child processes are forked from the parent process to handle both pixel-based and feature-based face detection modules at the same time. However, creating a child process is computationally expensive hence some overhead of creating a child process is expected.

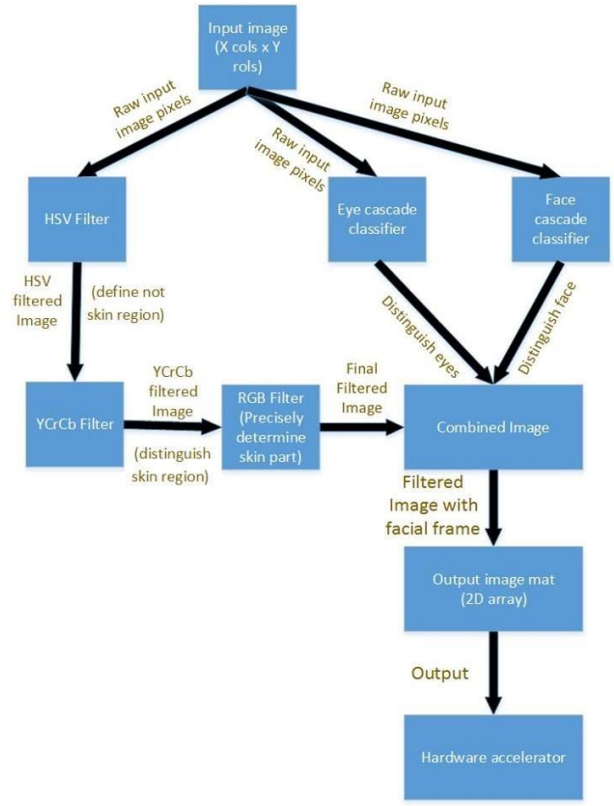

Figure 9: Software Implementation Work Flow

V. Experimental Result

The main objective of our real-time face detection system is to be able to detect whole or partial or multiple faces in an image and reduce the processing time of each input images. The experimental result would focus on accuracy and performance time of the real-time face detection system implemented on the software component. For the experimental result on the hardware component, please refer to my project partner's report.

A. Accuracy

Figure 10: Input (whole face)

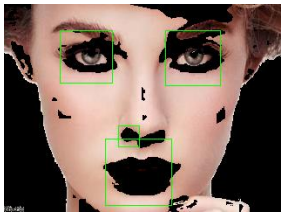

Figure 11: Output (whole face)

The face detection system works exceptionally well for single whole input face. During the accuracy testing, 62 images that have single whole human face are fed into the face detection system. The system correctly finds the face regions in every input images (no false negative). However, the accuracy suffers a little due to some false positive (identify a face region at the wrong location) within the image. 13 images out of 62 single whole human face images have some degree of false positive. The most extreme cases identify 7 false positives. Overall, the accuracy rate of detecting a single whole human face is 100% with 20.97% false positive rate. (Appendix 1)

Figure 12: Input (extreme)

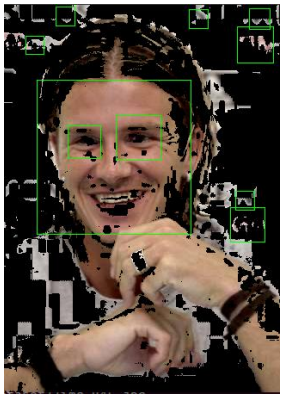

Figure 13: Output (extreme)

The other test case is multiple faces and partial face detection. The accuracy rate of partial face detection depends on the face coverage rate and camera focus. To successfully detect a partial face, the input face should have at least one eye feature hence if the face coverage rate is over 50%, the face detection system would not be able to detect the faces. If the partial face is in the background of the image (i.e. not the focus of the camera), the accuracy drops depending on the clarity of the partial face. To test the accuracy of multi-face and partial face detection, 18 images that have multiple and partial human faces are feed into the face detection system. The system correctly finds every face in 13 images. Most of these 13 images have 2 to 3 human faces. 4 out of 18 input multi-face images have some degree of false positive. From the experimental

result, we conclude the accuracy rate for multi-face and partial face detection is 72.2% with the false positive rate of 22.2%. If the input image contains more than 3 faces, the face detection system can detect some faces but not all faces due to camera focus and face coverage rate. (Appendix 2)

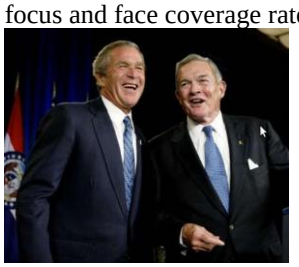

Figure 14: Input (multi faces)

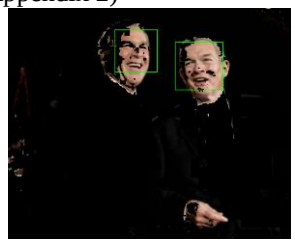

Figure 15: Output (multi faces)

B. *Performance*

x	y	Skin Color Segmentation	Eye Cascade Classifier	Face Cascade Classifier	total
320	240	0.21	0.41	0.49	1.11
640	480	1.02	0.78	1.48	3.28
960	720	1.87	2.33	3.41	7.61
1280	960	5.96	1.97	3.16	11.09

Figure 16: Profile Result

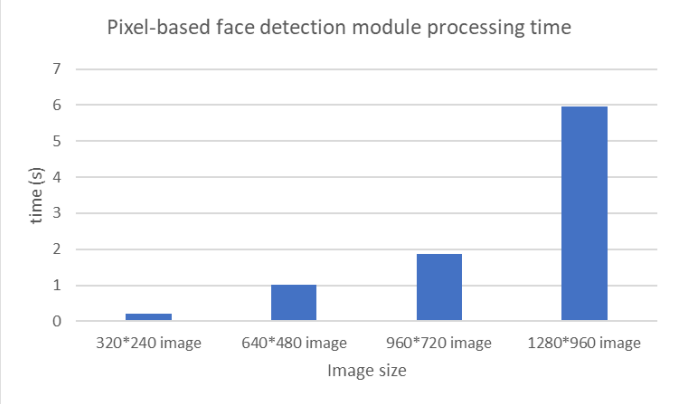

Figure 17: Pixel-based face detection processing time

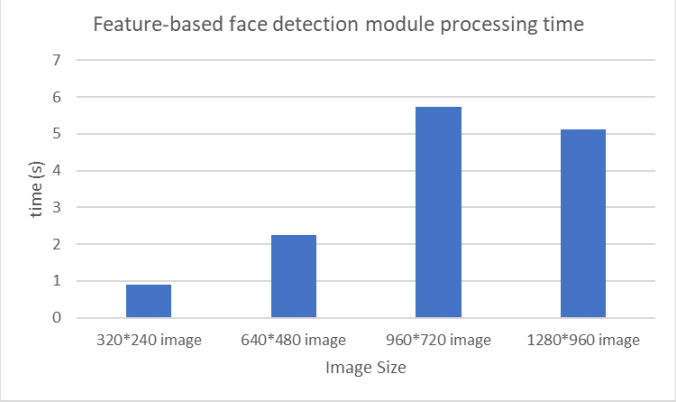

Figure 18: Feature-based face detection module processing time

The pixel-based face detection module searches every pixel within an input image. The computational complexity of the pixel-based face detection module is $O(n^2)$. Hence the higher the resolution of the input image, the longer the processing time. From the above profile result, we can conclude the bottleneck of the software component is in the pixel-based face detection module since the processing time in the pixel-based face detection module is significantly higher than the processing time in feature-based face detection module. Hence, we have implemented RGB color space and YCrCb color space filters using the hardware accelerator to investigate if the hardware accelerator provides a better performance rate. A set of skin color filter is implemented in the hardware component written in VHDL. The profiling result shows skin color segmentation algorithm runs in hardware component is 8 times faster compared to running in the software component. The detailed profile result of the hardware implementation would be discussed in my partner's report.

TABLE I. EXECUTION TIME FOR FULL SOFTWARE AND HARDWARE DESIGN

Performance	Software on ARM	FPGA Hardware
Execution time	210ms	27ms
Operating Frequency	N/A	50Mhz

Figure 19: Profile result of software component and hardware component

To optimize the performance of the face detection system, parallelization processing is implemented in the software component of the face detection system. The following profile result shows the difference of processing time between sequential tasks and concurrent tasks. X and Y are the resolutions of the input image.

x	y	sequential	concurrent
320	240	1.13	0.943
640	480	3.613	2.841
960	720	7.847	6.006
1280	720	8.631	7.024
1280	960	11.387	8.524

Figure 20: Profile result for sequential and concurrent tasks

Paired Samples T-Test

Paired Samples T-Test

			t	df	p	Mean Difference	SE Difference	Cohen's d
concurrent	-	sequential	−3.162	4	0.034	−1.454	0.460	−1.414

Note. Student's t-test.

Descriptives

Descriptives

	N	Mean	SD	SE
concurrent	5	5.068	3.107	1.390
sequential	5	6.522	4.105	1.836

Descriptives Plot

concurrent - sequential

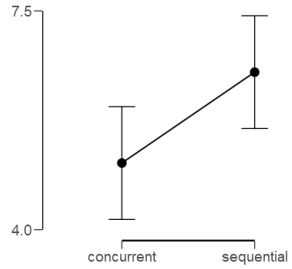

Figure 21: Statistical analysis

From the above profile result and statistical analysis, the mean processing time of sequential processing is higher than the mean processing time of multi-processing. A paired sample t-test is conducted to test the null hypothesis such that there is no difference between sequential processing time and concurrent processing time. The p-value is statistically significant (p-value = 0.034) which shows there is evidence against the null hypothesis. (i.e. There is evidence such that the performance time of multi-processing is better than the performance time of sequential processing.)

VI. System Evaluation

The real-time face detection system consists of 2 face detection modules: pixel-based and feature-based face detection modules. Each module has some advantages and limitations regarding the performance and accuracy.

A. Pixel-based Face Detection Module

By combining outputs from all 3 color space filters (RGB, YCrCb, HSV color space filter), this module masks pixels that do not fit the definition of human skin tone. Skin color segmentation is a very effective algorithm that can extract multiple and partial human face regions hence it allows the face detection system to correctly identify partial face region. However, there is some limitations and drawback using skin color segmentation algorithm.

Skin color segmentation algorithm assumes human face is of natural color but it is not always true. If the input human face is wearing any un-natural color paint, skin color segmentation would mask the face region since it cannot identify skin pixel in the face region which results in a false negative. Skin color segmentation algorithm also assumes no other body part except face is in the input image. Because face skin tone is the same as the skin tone of other body parts (i.e. hand), skin color segmentation algorithm would identify the hand as part of the face region resulting in false positive. Skin color segmentation algorithm works poorly if the input image is of grayscale or poor lighting since the module cannot identify skin pixel from a grayscale picture.

From the performance profile result discussed in section V, the computation cost is high since the computational complexity is $O(n^2)$ hence the processing time of pixel-based face detection module is longer than the processing time of the feature-based module.

B. Feature-based Face Detection Module

Cascade Classifier algorithm is implemented to increase the accuracy rate of face detection. This module can identify eyes feature and overall facial features. It is quite accurate in terms of identifying facial features and also much faster than pixel-based face detection module but there are some limitations in the module.

Because each cascade classifier is trained to identify similar features, the face classifier would not be able to detect partial face features. Hence an eye cascade classifier is implemented to detect eye features as well. If a partial face with one eye feature is in the input image, the face cascade classifier cannot detect the face but the eye cascade classifier can identify the eye feature. Combining with the output from skin color segmentation, if the eye feature is inside the face region that is extracted from skin color segmentation, the system concludes that a partial face is in the image.

VII. Future Work

We have implemented the real-time face detection system on both hardware and software component. However, the inter-communication AXI bridge protocol between software component and hardware component was not finished due to limited time.

Face Detection is the basic component of many advanced applications such as face recognition. The next stage of this project should focus on the real-time face recognition on the embedded system.

We have achieved the real-time definition of the face detection system. However, there are some improvements that could be done to reduce the processing time. For example, using multiple DE1-SoC FPGA boards to handle the face detection system. It would reduce the processing time of each input image and enhance the performance.

VIII. Conclusion

A real-time face detection system is implemented on Altera DE1-SoC FPGA development board using hardware-software co-design technique. Software component provides the basic structure of the face detection system while the hardware component accelerates some operations in software. The software component is implemented on the Hard Processor System while the hardware component is implemented on the FPGA fabric and they are intercommunicated via AXI bridge protocol.

There are 3 modules in the software component: input module, pixel-based face detection module and feature-based face detection module. Input module handles input image and feeds the image to the system. Pixel-based face detection module is implemented based on Skin Color Segmentation algorithm which checks every pixel in the input image and identifies human skin pixels from it. Feature-based face detection module is implemented based on the Haar's Cascade Classifier algorithm which use search windows to identify similar facial features in the input image. By combining the resultant output from these two face detection modules, the system can detect whole faces, partial faces, and multiple faces within the input image.

The experimental result shows 100% accuracy of detecting a single whole face with 21% false positive rate. The accuracy rate drops to 72% with 22% false positive rate if multiple faces and partial faces are in the input image.

Pixel-based face detection module has a high computational cost hence the higher the resolution of the input image, the longer the processing time. It can detect partial face and

multiple faces but it depends on the lighting and the focus of the camera. Feature-based face detection module has lower computational cost but it cannot detect partial face features.

Multi-Processing technique is used to handle pixel-based face detection module and feature-based face detection module concurrently. From the profile result, there is some statistically significant evidence to suggest that multi-processing reduces the processing time of each input image.

In conclusion, we have answered our research question by investigating the performance difference between hardware component and software component. Skin Color Segmentation is the bottleneck in the software component. By implementing skin color segmentation using hardware accelerator, the processing time is 8 times faster compared to the system runs entirely on software component.

APPENDIX

Picture_id	Faces in Picture	Number of Faces detected	False Positive	Total correct face detected	All Face Detected
1	1	3	2	1	Y
2	1	1	0	1	Y
3	1	4	3	1	Y
4	1	1	0	1	Y
5	1	1	0	1	Y
6	1	1	0	1	Y
7	1	1	0	1	Y
8	1	1	0	1	Y
9	1	1	0	1	Y
10	1	1	0	1	Y
11	1	2	1	1	Y
12	1	1	0	1	Y
13	1	2	1	1	Y
14	1	1	0	1	Y
15	1	2	1	1	Y
16	1	1	0	1	Y
17	1	1	0	1	Y
18	1	1	0	1	Y
19	1	1	0	1	Y
20	1	1	0	1	Y
21	1	1	0	1	Y
22	1	1	0	1	Y
23	1	1	0	1	Y
24	1	1	0	1	Y
25	1	1	0	1	Y
26	1	2	1	1	Y
27	1	1	0	1	Y
28	1	1	0	1	Y
29	1	1	0	1	Y
30	1	1	0	1	Y
31	1	1	0	1	Y
32	1	1	0	1	Y
33	1	8	7	1	Y
34	1	1	0	1	Y
35	1	1	0	1	Y
36	1	1	0	1	Y
37	1	1	0	1	Y
38	1	1	0	1	Y
39	1	1	0	1	Y
40	1	1	0	1	Y
41	1	1	0	1	Y
42	1	1	0	1	Y
43	1	1	0	1	Y
44	1	1	0	1	Y
45	1	2	1	1	Y
46	1	1	0	1	Y
47	1	2	1	1	Y
48	1	1	0	1	Y
49	1	1	0	1	Y
50	1	0	0	0	N
51	1	1	0	1	Y
52	1	1	0	1	Y
53	1	3	2	1	Y
54	1	1	0	1	Y
55	1	2	1	1	Y
56	1	1	0	1	Y
57	1	4	3	1	Y
58	1	1	0	1	Y
59	1	1	0	1	Y
60	1	2	1	1	Y
61	1	1	0	1	Y
62	1	1	0	1	Y

Appendix 1: Sample result for single whole face

Picture_id	Faces in Picture	Number of Faces detected	False Positive	Total correct face detected	All Face Detected
1	2	2	0	2	Y
2	2	2	0	2	Y
3	10	2	0	2	N
4	2	2	0	2	Y
5	10	5	1	4	N
6	8	3	1	2	N
7	2	2	0	2	Y
8	2	2	0	2	Y
9	2	2	0	2	Y
10	3	5	2	3	Y
11	4	3	0	3	N
12	2	2	0	2	Y
13	2	2	0	2	Y
14	3	4	1	3	Y
15	2	1	0	1	N
16	3	1	0	1	Y
17	3	3	0	3	Y
18	6	6	0	6	Y

Appendix 2: Sample result for multiple, partial face

ACKNOWLEDGMENT

We would like to thank our supervisor Dr. Moterza Biglari-Abhari and Dr. Nitish Patel for his help in this project.

REFERENCES

[1] P. Shanmugavadivu and A. Kumar, “Rapid face detection and annotation with loosely face geometry,” in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Dec 2016, pp. 594– 597.

[2] D. N. Arya, S. K. L. V., R. Reddy, S. S, and S. K, “A face detection system implemented on fpga based on rct colour segmentation,” in *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, Nov 2016, pp. 1–5.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, pp. I-511-I-518 vol.1.

[4] A. Mohanty, N. Suda, M. Kim, S. Vrudhula, J. s. Seo, and Y. Cao, “High performance face detection with cpu-fpga acceleration,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 117–120.

[5] S. V. Dharan, M. Khalil-Hani, and N. Shaikh-Husin, “Hardware acceleration of a face detection system on fpga,” in *2015 IEEE Student Conference on Research and Development (SCOReD)*, Dec 2015, pp. 283–288.

[6] S. Rana, M. P. Deepu, S. Sivanantham and K. Sivasankaran, "Face detection system using FPGA," *2015 Online International Conference on Green Engineering and Technologies (IC-GET)*, Coimbatore, 2015, pp. 1-4.

[7] J. M. Saul, “Hardware/software codesign for fpga-based systems,” in *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, vol. Track3, Jan 1999, pp. 10 pp.–.

[8] Altera (2014, Feb). *Introduction to Cyclone V Hard Processor System(HPS).* [Online].

[9] M. Kawulok, M. Emre Celebi, B. Smolka, “Advances in Face Detection and Facial Image Analysis” [Online], Available at: https://link.springer.com/book/10.1007/978-3-319-25958-1