Heterogeneous Parallel Architecture for Face Detection Enhancement

Aishah Albssami [1†] and Sanaa Sharaf[2††],
Aalbassami0005@stu.kau.edu.sa Ssharaf@kau.edu.sa
King Abdulaziz University, Jeddah, Saudi Arabia

Summary
Face Detection is one of the most important aspects of image processing, it considers a time-consuming problem in real-time applications such as surveillance systems, face recognition systems, attendance system and many. At present, commodity hardware is getting more and more heterogeneity in terms of architectures such as GPU and MIC co-processors. Utilizing those co-processors along with the existing traditional CPUs gives the algorithm a better chance to make use of both architectures to achieve faster implementations. This paper presents a hybrid implementation of the face detection based on the local binary pattern (LBP) algorithm that is deployed on both traditional CPU and MIC co-processor to enhance the speed of the LBP algorithm. The experimental results show that the proposed implementation achieved improvement in speed by 3X when compared to a single architecture individually.

Keywords:
Face Detection, LBP, MIC, Scheduling.

1. Introduction

During the past years, the problem of face detection has received a lot of attention due to its range of applications in many fields. Moreover, various methods of detecting human faces in photos and videos have been proposed in recent years because it is considered the first phase in most systems that process and analyse faces, such as face recognition systems, face verification systems, surveillance systems, social media, and more. These systems totally depend on face detection, if there is a delay in face detection time this will affect systems' progress. Detecting faces is a complicated process due to some conditions such as image position and orientation, occlusion, lighting conditions, facial skin colour, glasses or facial hair, etc. Also due to rapid development in computer hardware in performance and speed, and as a result of the increase in the evolution of devices that capture images and videos, the computational cost increases to process these images. High- performance computing HPC is extremely valuable in reducing these computational costs because it provides a parallel architecture to obtain high performance and accurate results. Intel many integrated core (MIC) also known as Xeon Phi is one of the heterogonies high-performance computing architectures along beside Graphics Processing Unit (GPU) and field-programmable gate array (FPGA). Intel provided MIC coprocessors in 2012. It improves the parallel performance and energy efficiency for many applications supports all programming models that are available for traditional intel architecture processors such as OpenMP and MPI. MIC also support C/C++ and Fortran languages unlike GPU which need an SDK such as CUDA or OpenCL to run any application. In this paper we enhanced the performance of face detection problem using local binary pattern LBP face detector and processed by using a heterogeneous platform involving MIC architecture and CPU since many studies used GPU and FPGA widely to enhanced and accelerate the face detection algorithms. MIC architecture is the main contribution of this study and it's not used before to accelerate or enhance any face detection algorithms based on the best of our knowledge. The rest of this paper is organized as follows: the related work will be discussed in section2. Section 3 will discuss the LBP based face detection. Methodology will be explained in section 4. The experiment results will be present in section 5 while section 6 will have the conclusion.

2. Related Work

Face detection consists of several steps that requires the image to be scanned to find the rectangles (windows) that contain faces. Those rectangles are size invariant and may appear in any arbitrary location in the image. This requires dividing the image into many overlapped rectangles and apply certain classifier to find out either those rectangles contain face or not. This process is computationally intensive and thus the need to deploy parallel architectures and co-processors aroused. Several trials to speed up Local Binary Pattern (LBP); one of the most common face detection techniques were made. Previous attempts were made to parallelize LBP algorithm to reach real time face detection as proposed in [1], the authors using OpenCL reaching computation time of 20ms for a 640*480 image resolution. Also, an attempt on non-standard image resolutions was made in [2]. which we couldn't consider as a benchmark to compare with. Attempts to evaluate the different performance on multiple architectures was introduced in [3]. the authors introduced the different performance on general purpose processors, SIMD units, multi-core architectures and GPUs reaching 30fps and measuring the energy needed along with the performance. In [4] authors proposed viola-jones enhancement on GPU reaching 37fps for HD Images.

Manuscript received February 5, 2022
Manuscript revised February 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.2.25

While in [5] their study was based on introducing an implementation for LBP algorithm by both scaling feature scaling and image scaling and a comparison between both in terms of accuracy and performance time reaching up to 50 frame per second (fps) for feature scaling and 45 fps for image scaling. GPGPU was used in [6] to accelerate the computation of the LBP-based face detection algorithm using feature scaling, the authors achieved processing speed up to 287fps for 640*480 image resolution.

3. LBP-based Face Detection Algorithm

Nowadays according to the exponential growth of computer hardware in speed and performance we try to solve the face detection problem which is a classical and time-consuming problem. One of the most important classifiers in face detection is the Local binary Pattern (LBP). The LBP- based face detection process starts by converting the input colour image to grayscale then using integral image to speed up the calculation of LBP feature extraction. Sliding window technique is used where the input image is scanned by sliding a rectangular sub window of fixed size (The LBP detector) over image from left to right and top to bottom. Then process every sub window to recognize if there a humane face or no. This process repeated many times by rescale and shift the sub windows until entire image is scanned, and faces are detected. We will apply specific scheduling strategy to process and mapping the LBP detector to the heterogeneous architectures to achieve better utilization and more speedup.

3.1 Local Binary Pattern (LBP)

Local binary patterns (LBP) is one of visual descriptor type in computer vision used for classification. It is a texture operator that is robust, extremely fast, highly effective and computationally simple with powerful discrimination introduced by Ojala et al [7].It used in texture analysis initially but later used in many fields of computer vision and image processing such as image face detection. LBP-based face detection applications are one of the active research topics gave excellent results that outperformed many techniques in face detection tasks. LBP algorithm come in a variety of forms. The basic LBP investigated the relationship between the center pixel in a 3*3 grid and its eight neighboring pixels. Later, it expanded to apply to any region of the image. The center pixel is compared with its surrounding neighbors. If the value of the center pixel is equal to or larger than the neighbour's values LBP will threshold the neighbor's values by zero; otherwise, the neighbor values will be one. The generated code is represented as eight bits in binary format. Each 1 in binary bits will convert to decimal in a clockwise direction. The pixels that have less than eight neighbours will be ignored because it will generate incorrect information. Generally, the LBP operator is applied to a grayscale image because it has just 256 potential values because the 3*3 grid consists of eight pixels where 28=256 values as shown in Fig.1

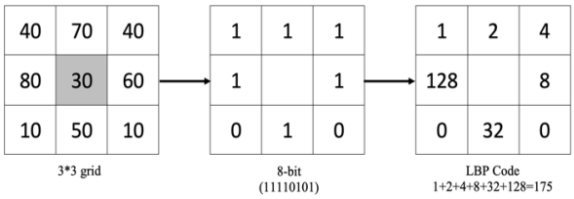

Fig. 1 Local binary pattern.

3.2 Cascade Classifier

A cascade classifier is a machine learning technique that consists of consecutive stages each stage represents a weak classifier these weak classifiers are composed together to be a strong classifier. It determines whether a given sub-window contains a face or not by discarding non-face regions, which helps in reducing the computation time. After image is scanned the generated sub windows are flow to the cascade classifier. Each stage consists of a threshold and three to ten of LBP features. Each feature has two weights one is a negative and one is positive and a 256-bit lookup table (LUT). The 8-bit LBP number that generated from comparing the central region with its neighbours' regions (Fig. 1) is used as an index to the LUT to get either 0 or 1 bit. The 0 bit are indicating the negative weight of the feature while the 1 indicates the positive weight. These weights summed together and compared to the stage's threshold. If the sum is larger than the stage threshold this means the current sub window probably have face and then the window will pass to next stage and so on. otherwise, if the sum is less than the stage' threshold this means the stage is failed and classifier will reject all the following stages. Then the next sub window will be passed to first stage to be processed. This process is repeated until the window passes all stages and the face is detected as shown in Fig 2. The features at the latest stages are more restricted than the earliest ones.

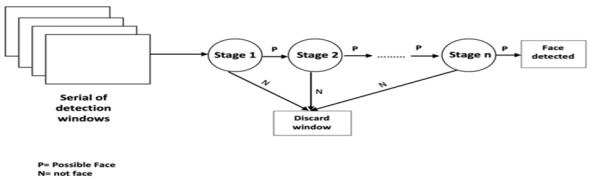

Fig 2 Cascade Classifier.

4. Methodology

Overview of our work is described in Fig.3. To process one frame of given image we scan the image and divide it into multiple sub windows using sliding window technique. The generated sub windows from input image is a challenge, because to detect faces in image of size 640 x 480 almost 1051993 sub windows will be produced. All these sub windows must be process in millisecond of time. Can we process these sub windows in parallel? Can we split these sub windows to be run over CPU and MIC as hybrid process? Yes, and this is what we are did in this study. According to the scheduling strategy described in [8]. we will be assigning these sub windows to different architectures (CPU-MIC) this will be described in section 5.

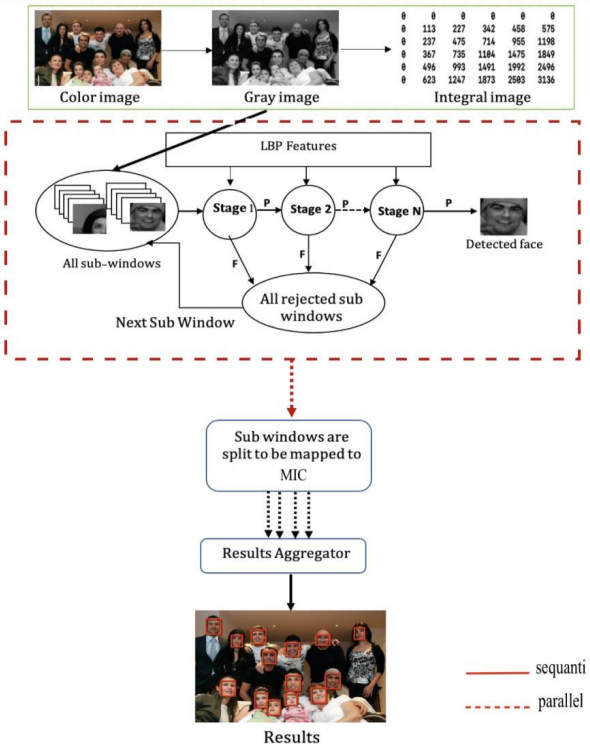

Fig. 3 Implementation Overview.

4.1 Image Processing

Before detecting a human face in an image, the input colour image is converted to a grayscale image where all pixel values become in range 0 to 255 to quickly calculate the integral image. The grayscale image was then converted to an integral image and stored in a 2D array structure.

4.2 Parsing classifier

The pre-trained LBP classifier set is obtained from OpenCV [9]. as an XML file with a fixed detection window size (24*24). We convert the XML file into a C structure. This XML file consists of 20 stages. Each stage contains three to ten weak classifiers (LBP feature) and one threshold. Each weak classifier has a lockup table consisting of eight integers and one indication to a rectangles list. These rectangles represent the dimensions of the features. Also, each weak classifier has two leaf values that represent the weight of the features. All this information from the XML file is converted into a 1D array structure to refer to each value by stage and feature indicators. Image processing and parsing classifier both are sequentially executed.

4.3 LBP detector parallelization

The LBP detector are implemented as a function with five parameters the original image, integralimage, scale factor, the x and y position, which represent the step size of the sub-window to scan the whole image. The sub-windows are scaled by scale factor =1.1 and shifted two pixel each time, then incremented until cover all image dimensions and faces are detected. This function then offloads to be run on MIC side. Intel MIC Programming is identical to CPU programming. Only one line of code is the main difference to declare that a specific part of the code should be executed on the Intel MIC architecture. OpenMP are used to parallelize scale and shift process of the sub- windows and Offload mode are used to run this parallel part on MIC as shown in pseudocode Fig.4. Then results are aggregate to display.

```
Program:Face detection algorithm
Input: original image
Input : Image Width
Input : Image Height
Input : scale =1.1
Input: CountofSubwindows=0
Output: Image with faces rounded by a rectangles
BEGIN
convert image to gray scale
compute integral image, integral
parsing the classifier from XML file to C struct

FOR scale=1 to maxScale incremented by scale factor
$OFFLOAD Directive
$openMP Directive
 {
  FOR  x_offset = 0 to Image gWidth - (24*scale) incremented by scale*2
    FOR y_offset = 0to Image gHeight -(24*scale)incremented by scale*2
      Calculate LBP features (image, integral, x_offset, y_offset, scale)
      CountofSubwindows++
}

      END
    END
  END
END
```

Fig. 4 Implementation pseudocode.

5. Experiment Results

The experiment has been conducted on AZIZ supercomputer. Each compute node has dual processor with total 24 core and Intel Xeon Phi Coprocessor with 60 core each core has 4 thread. Single compute node is used to run the implementation of proposed work. The code is written using C++ language while Intel compiler is used to compile it. We evaluate the work performance by testing three images with different resolutions (640x480, 1280 x 720, 1920x1080). Each image has distinctive number of human faces with different position. It can be seen from our discussions in sections 4 that the sub windows processing is the complex process in face detection. Therefore, any increase in the number of generated sub windows leads to significant increase in the total time of face detection process. We run the code on CPU and MIC separately and calculate the execution time to each architecture. Fig.5 shows the execution time of each architecture when running one frame for different sizes of images. Fig 6 show the speed of different architecture where we have achieved processing speed up to 9.81 fps from image of size 640 x 480 on CPU while on MIC, we have achieved processing speed up to 3.52 fps. And for image of size 1280 x 720 we have achieved processing speed up to 5.66 fps on CPU and 3.23 fps on MIC.

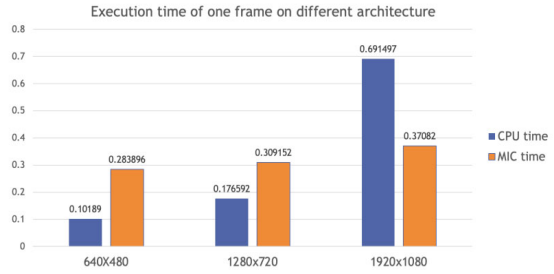

Fig. 5 Execution time of one frame on CPU and MIC.

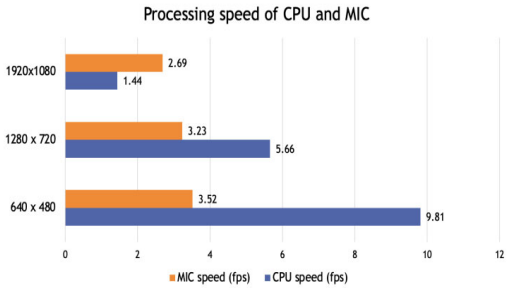

Fig 6 Speed of different architectures.

From Fig.5 and Fig.6 we can notice that the execution time of CPU is better than execution time of MIC which is not a promising result. However, we can enhance the performance of LBP algorithm by using both CPU and MIC together as heterogeneous platform instead of using any of them individually. Based on speed-based scheduling strategy described in [8] we distribute the image frames to be processed over MIC and CPU based on specific ratios. This ratio decides how many frame should be assigned to CPU and MIC as shown in Fig. 7. We are formalized the ratio as in Eq. 1, where R is the ratio of the tasks assigned to the architecture, p represent the architectures and N represent the total speed of architectures.

$$(p) = (speed(p)/N) * 100. \quad (1)$$

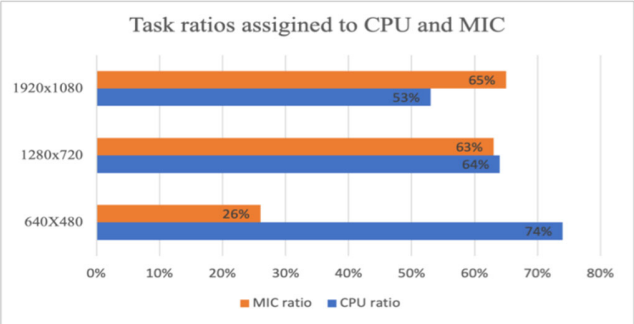

Fig 7 Ratio of tasks assigned to architectures based on speed

Therefore, to process 1000 frames for image of size (640 x 480) we divide these frames into (740 frames) to be processed by CPU which equals 74% of tasks while MIC will process (260 frames) which equals 26% of the tasks. After assigning frames to the different architectures we observed that the finish time of these two architectures are approximately equal which mean the MIC and CPU are processing the right amounts of frames, thus the frames were successfully assigned to the architectures as shown in Table 1.

Table 1: Finish time of CPU and MIC

#frames	1000		
Frame Size	640x480	1280 x 720	1920x1080
CPU #frame	740	640	350
MIC #frame	260	360	650
Finish Time (CPU)	75.433	113.07	243.06
Finish Time (MIC)	73.864	111.46	241.64

Comparing the run time of the CPU and MIC with a run time of a hybrid platform is also shown in Fig. 8 Which gives us an idea about the improvement in the total run time. For clarification, If CPU were used only the processing time will be 101.9368sec and if MIC were used only the processing time will be 284.0909sec But if the CPU and MIC used together as heterogeneous the processing time will be 75.43323 sec, this gives us an idea about the improvement in the total run time.

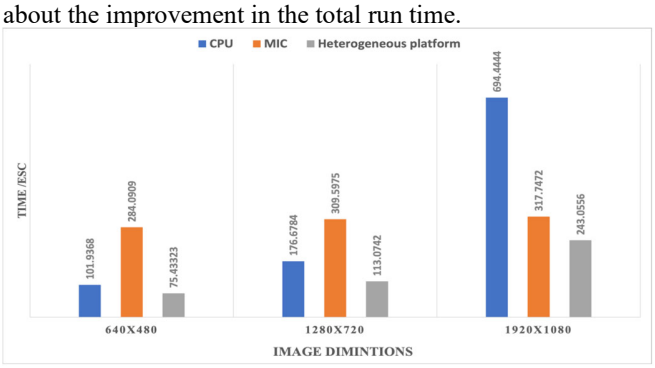

Fig 8 Experiment result comparison for different images size for CPU, MIC and heterogeneous (CPU-MIC).

We also observed from the above ratios shown in Fig 7 that the MIC ratio increases when the image's size increases, and CPU ratio decreases because the time that MIC spent in transfer image data to the MIC memory was more significant than the time that MIC used to process. Therefore, when the processing requirement increased the time of transfer are become less than the computation time.

6. Conclusion

The hybrid program model which merges CPU with accelerators is not an optional choice anymore. Now in 2021 we must use hybrid model and using the theory that say merge performance of many architectures to enhance the speed. This paper has presented an enhancement implementation of the LBP based face detection algorithm on heterogenous architecture that consists of Intel Xeon processors (CPU) and Intel Xeon Phi accelerators (MIC). A thorough study of sliding window technique, integral image and LBP-based classifier to obtain a high detecting rate was conducted.. Parallel implementation details of the LBP algorithm were shown besides the parallel execution tested on hybrid platform(CPU-MIC) . The preliminary out- comes differed from what was anticipated then, after deployed work on hybrid model the obtained results prove that distributing the work between CPU and MIC as a heterogeneous platform gives us a promising improvement in comparison with CPU and MIC separately. Future work may include detecting the faces from video stream instead of solid images and use another scheduling strategies or deploying the work on many heterogeneous architectures instead of one.

Acknowledgments

This work is supported by the King Abdulaziz University , High-Performance Computing Center
(Aziz Supercomputer) (http://hpc.kau.edu.sa).

References

[1] N. NIKE and G. N. RATHNA, "Real time face detection on gpu using opencl," *Computer Science,* 2014.

[2] M. Chouchene, F. E. Sayadi and H. Bahr, "Optimized parallel implementation of face detection based on GPU component," *Microprocessors and Microsystems,* vol. 39, pp. 393--404, 2015.

[3] M. B. López , A. Nieto and J. Boute, "Evaluation of real-time LBP computing in multiple architectures," *Journal of Real-Time Image Processing,* vol. 13, no. 6, 2017.

[4] M. Fayez , Faheem, I. Katib and N. R. Aljohani, "Real-time image scanning framework using gpgpu face detection case study," *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV),* 2016.

[5] Y. M. Abdelaal , M. Fayez and H. Faheem, "Performance evaluation of image scanning:face detection case study," p. pp. 1–15, 2019.

[6] M. . R. Ikbal, F. Mahmoud, M. Fouad and I. Katib, "Fast implementation of face detection using lpb classifier on gpgpus," *Intelligent Computing- Proceedings of the Computing Conference,* p. pp. 1036–1047, 2019.

[7] T. Ojala , M. Pietikäinen and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recogni- tion,* vol. 29, no. 1, p. 51–59, 1996..

[8] B. König-Ries and H. Faheem, "A new scheduling strategy for solving the motif finding problem on heterogeneous architectures," *International Journal of Computer Applications,* vol. 101, p. 27–31, sep 2014.

[9] "pre-trained LBP classifier," [Online].Available: https://github.com/opencv/opencv/blob/master/data/lbpcascades/lbpcascade_frontalface.xml.

Aishah Albassami received the B.E. With second honor degree in Computer Science from Umm Al-Qura University, Makkah, Saudi Arabia, and she is currently a master's student in the department of cmputer science at FCIT, King Abdulaziz University (KAU), Jeddah, Saudi Arabia. Her research interests are in the fields of High-Performance Computing and Machine Learning.

Sanaa Sharaf received the B.E. with first honor degree in computer science from King Abdulaziz University, Jeddah, Saudi Arabia, and MSc with Distinction from University of Bradford, UK in Information Security in 2006. Sanaa finished her Ph.D. in Grid Computing from the University of Leeds, UK in 2012. In 1998, she joined the Computer Science Department, King Abdulaziz University, as a Teacher Assistant. She is currently an Assistant Professor in the Computer Science Department, Faculty of Computing and Information Technology, KAU. Her main areas of research interest are Information and System Security, Grid/Cloud Computing and High- Performance Computing. Since 2013 she started some administrative assignments includes: Supervisor of Information System department Sulaymaniyah branch, FCIT vice-dean in both Faisliyah branch and University of Jeddah and now she is the High- Performance Computing Center deputy director for Academic Affairs, King Abdulaziz University, Jeddah, Saudi Arabia.