GPU-Accelerated Key Frame Analysis for Face Detection in Video

Xuan Qi   Chen Liu

Department of Electrical and Computer Engineering
Clarkson University
Potsdam, New York
{qix, cliu}@clarkson.edu

Abstract - Since video monitoring cameras now are implemented widely, video analytics has drawn much attention as a new research area. Correspondingly, there is an emerging need to detect faces through a period of video or a great number of videos to make comparisons with the stored identities in the database for personal identification or other purposes. Thus, face detection in video has gained great attention. However, when there are lots of videos or when the video is very long, the workload for face detection becomes very huge. As a result, the detecting time is prolonged. Thus, there is a need to detect faces quickly with reduced computation time. In this paper, we propose to add key frame analysis into the face detection process and employ the state-of-the-art graphic processing unit (GPU) platform to improve the overall performance. Our experimental results show that speedup as high as 3 folds can be achieved in terms of frames per second processed when GPU platform is introduced, compared with high-performance general-purpose CPUs. This shows great potential in applying GPU towards this kind of applications.

Keywords: GPU; OpenCV; Face detection; Key frame analysis.

I. INTRODUCTION

Face is a very good feature not only for identification purpose, but also can be used for revealing other attributes like gender, age, ethnicity, and emotional state of a person. Thus, face is often regarded as a significant biometric identifier in the law enforcement and human-computer interaction (HCI) communities [5], and face recognition has been a research focus. On the other hand, due to the dramatic advancement in microprocessor performance, face recognition is no longer computationally prohibitive. This greatly contributed towards the wide adoption of face recognition technology, even on mobile devices.

Recently, face recognition in videos gains much attention because it can recognize targeted subject(s) from a sequence of video frames rather than from only one single-shot still image. In many of the application scenarios, however, direct control on the acquisition conditions is not always possible. Hence, we normally need to process a large amount of long videos in order to achieve the satisfactory level of identification accuracy. The traditional way of performing face detection on every frame will require much computation, thus a great deal of processing time is necessary. To improve the overall speed of face detection, in this paper we introduce

graphic processing unit (GPU) platform to accelerate this process. If every frame that contains face(s) is sent from the front-end capturing device to the backend server (in the cloud) for processing, it will pose a very huge computation demand on the backend. Therefore, key frame extraction is added after face detection, but before face recognition. By doing so, we can dramatically reduce the computation load on the backend. Another associated benefit is to reduce the communication overhead or bandwidth requirement between the frontend and backend. Because transferring every frame is now replaced by transferring the extracted key-frames only. To summarize, our target is to develop a real-time processing system to extract “best quality” frame, i.e., key frame, which can be sent to backend for matching and recognition.

The rest of the paper is organized as follows: Section II presents a brief overview of related works in key-frame analysis. Section III describes detailed methodology of designing and implementing our key-frame analysis system. Sections IV and V are the experiment environment and experimental results of our work. At last, conclusions are presented in Section VI.

II. RELATED WORKS

Breitenstein et al. implemented real-time GPU-based face pose detection in video sequences, which shows the impressive performance improvement brought by GPU platform [2]. Yakhu et al. proposed a method of applying face quality index to determine the key-frame extraction [7]. In their work, at the first stage, the HAAR-like face detection method is applied to detect the face. Next, the face images are all transformed into same size and the quality index is computed. The face quality index used is mainly determined depending on face skin characteristics, which is relatively global traits. What’s more, the edge change ratio is another useful global trait that can be used to determine key-frames. Mann et al. proposed to determine the key-frame by calculating the difference of edge change ratios of adjacent frames [12]. The edge in their work is generated by the edge detection of original video images. Another effective method is key-point description. Schoeffmann et al. used ORB (Oriented FAST and Rotated BRIEF) key-point descriptor to describe the endoscopic video frames and extract key-frame based on these key points [15]. This category of method was also applied by Du et al. [19].

Anantharajah et al. proposed quality-based frame selection for face clustering in news video [1]. They came up with the criteria about quality frame selection using four metrics: face symmetry, sharpness, contrast and brightness. These four metrics can also be used to choose key frame. In another work by Nasrollahi et al., they also introduced symmetry, sharpness, brightness and face size as metrics to evaluate a face image quality [6]. Besides, they proposed to multiply these four metrics by different weights and sum the results up together to generate a final quality score for the corresponding face image. The common ground of these two approaches is: they both applied face detection first; then evaluated the face quality metrics based on detected faces; finally, they used these metrics to form a quality score in order to determine which frame will be extracted as the key-frame.

One major difference between our work and previous related works is our work emphasizes more on processing performance improvement. There are many application cases need real-time processing. For example, airport terminals or railway stations all need real-time face detection and recognition to identify the subject of interest in a timely manner. In this kind of real-time demanding application environment, computation capability occupies a more important role than algorithm's accuracy. In other word, with acceptable error rate, ensuring as low as possible delay is a must. To meeting this requirement, GPU acceleration is an effective method. By using GPU acceleration, we can achieve a higher computing performance compared to traditional CPU computing. Besides, the main memory of GPU, which is Graphics Double Data Rate Memory (GDDR), has higher bandwidth than the main memory in CPU system. This means we can have a higher data transfer speed while performing the calculation by using GPU acceleration, if we deal with the data transfer between the CPU and GPU wisely. Another difference is in our work we support more than one face or multiple face cases. It is easy to understand that in all surveillance video scenarios, there must be many people included. So the more-than-one-face capability makes our work one step further towards the application level than previous works.

III. METHODOLOGY

A. Design flow

In this work, face detection is implemented first, and then key frame extraction is implemented. The reason is that if face detection is performed first, key frame is then chosen based on face quality information. On the other hand, if key-frame is extracted first, since the criteria of key frame normally includes color histogram, edge change ratio, inter-frame distance [2], these criteria are all global traits and may not have much relationship to face recognition. Thus, we choose to have face detection performed first. The flowchart of our algorithm design is shown in Figure 1.

At the beginning, the face detection procedure is performed by using OpenCV face detection module [21]. This module has both CPU and GPU version. In the GPU version, the face detection procedure is fully parallelized and boosted by the GPU. Besides, the performance, like frame processing rate, is related to the video resolution and not affected by the

number of faces in the video. This performance behavior is defined in the algorithm used by the face detection module in both CPU and GPU versions. In the CPU version, the algorithm is same as the GPU version and it is also a multi-threading algorithm and can use all cores of CPUs.

Next, if the number of detected faces does not change, we perform the face quality evaluation on detected faces. After this procedure, we can get the four metrics employed to describe the image quality of each face: brightness, resolution, sharpness and symmetry. If the following face quality evaluation shows that current face image is better, then this frame will be used as key frame of the corresponding face. If the number of faces does change, it means the scene has changed. According to our flow chart, we would output the current key-frame to the backend and clear the key-frame buffer. Then, the face quality evaluation will be done on the remaining faces.

Fig. 1. Flow chart of our key frame analysis design

Another key point worth mentioning is if a detected face lasts not long enough, the system will not extract the key-frame image for that face. One reason is if a face just lasts for few frames, mathematically, it means the sample size for that face is too small. In this case, the confidence level of extracted key-frame will be low. Another reason is the face detector has error rate itself, which cannot be ignored. During our experiments, sometimes objects in the background such as books or the surface of clothes were also detected as “faces”. Fortunately, this kind of detection error always just lasts for very few frames. Hence, if we set a threshold that only let the key-frame of last-long-enough face to be extracted, we will not see the key-frame extraction on non-face objects. Lastly, there is another scenario that can be improved by this strategy, the fuzzy face image. For example, a person is moving across the scene, but his or her face is not clear enough to let the face detector keep locking on the face. It means when the face is relatively clear, it will be detected. When the face is not clear enough, the detector will lose track of this face. This process can be repeated many times and makes a single physical face to be detected as many different faces with short duration in

the system. So, with the strategy of not extracting short duration faces, this kind of key-frames of low quality faces can be avoided.

B. The face quality metrics

In this work, first of all, all color frames will be converted to 8-bit gray scale frames before further processing. During the processing stage, we use four metrics to describe the quality of a face: the brightness, resolution, sharpness and symmetry. The calculation methods for these metrics are described as follows.

a) *Brightness*: the brightness of the face image is calculated as in Equation 1:

$$\text{brightness} = \frac{\sum_{i=1}^{i=W} \sum_{j=1}^{j=H} B_{ij}}{W \times H} \quad (1)$$

where W is the width of the detected face image, H is the height, B_{ij} is the gray value of each pixel in the face image, the value range of which is $[0, 255]$. The brightness value of the face image equals to the summation of all pixels' value divided by the image size.

b) *Resolution*: the calculation of resolution is simple, which equals to the width W multiplies the height H of face image as in Equation 2:

$$\text{resolution} = W \times H \quad (2)$$

c) *Sharpness*: to compute the sharpness value, we first blur the face image by Gaussian operator. Then, we compute the absolute difference between original image and blurred image. Finally, the sharpness value equals to the absolute difference divided by the size of face image. The corresponding Equation 3 is shown below:

$$\text{sharpness} = \frac{\sum_{i=1}^{i=W} \sum_{j=1}^{j=H} |I - G(I)|}{W \times H} \quad (3)$$

where I is the original image, $G(I)$ is the image after Gaussian blur, W and H are the width and height of face image. The kernel of $G(I)$ is a 5-order 2-D Gaussian distribution matrix, with standard deviation of 1.0 in X and Y directions.

d) *Symmetry*: the symmetry value is computed as in Equation 4:

$$\text{symmetry} = 255 - \frac{\sum_{i=1}^{i=W} \sum_{j=1}^{j=H} |I - F_h(I)|}{W \times H} \quad (4)$$

where I is the original image and $F_h(I)$ represents horizontally flipped or flipped around y -axis of image I , W and H are the width and height of face image.

For gray scale image, a totally symmetry image means the absolute difference of each pixel between original image and flipped image are all 0. On the contrary, a totally asymmetry image means the absolute difference of each pixel between original image and flipped image are all 255. Also, it is obvious that less absolute difference means higher symmetry degree. With the consideration of other three metrics in this work are all the higher the better, we use the highest possible pixel value 255 to minus average value of absolute difference of the whole face image and its flipped image. Then, the higher calculated symmetry value means higher symmetry degree of the face image.

C. Face quality

To select the best quality face image from all video frames, we use the square root of the distance between the metrics of current image and the metrics of the best frame to evaluate the quality of faces in current frame. The square root distance of each metric is also multiplied by different weights due to their different importance. The expression of the face quality S is shown as:

$$S = \sqrt{\sum_{i=1}^{i=N} [w_i \times (1 - \frac{m_i}{m_{\max}})^2]} \quad (5)$$

In Equation 5, m_i means the face quality metric, $m_{\max}$ means the best value of that metric, and w_i means the weight of that metric. All metrics are normalized according to $m_{\max}$. From this equation we can conclude that smaller S means the face image's quality is closer to ideal situation. N represents the number of metrics. In this work, $N = 4$. The weights of the four different metrics are shown in Table I, which are referenced from the experiment values used in [6].

TABLE I: THE VALUE OF THE WEIGHTS

Weight	w_1 brightness	w_2 resolution	w_3 sharpness	w_4 symmetry
Value	0.6	0.8	0.9	1.0

D. Face tracking

Another aspect needs to be considered is the face tracking. Because we generate different key frames for each different face, but the location of the face keeps moving continuously from one frame to next. So we need to track the movement of the same face to aid the key-frame extraction.

Algorithm 1 describes the top-level design of the face tracking. First, the detected face number is compared with the detected face number in previous frame. If these two numbers are equal, which means there is no change in the number of faces in the frame, we first invoke the CompareLocation algorithm. This algorithm can match current face's location to its nearest location, which supposed to be its location in previous frame. Then, the metrics and location information are renewed in the RenewMetrics function according to the new coordinate of each face. If the current detected face number in current frame is larger than that in previous frame, which means there are new face(s) come in at current frame, then we

create new face objects for each face image. What's more, the CompareLocation algorithm needs to be performed if previous detected face is not zero, because we also need to renew the metrics and other data of existing faces. Last case is the detected face number get smaller than previous frame. This situation means we should delete the face images that no longer exist. In this case, we also invoke CompareLocation and RenewMetrics first. Then, the CheckRenewStatus function checks the renew status of each face object. If a face object's status shows that it is not renewed, which means that face does not exist in current frame, it will be deleted from the face object library.

TABLE II: FACE TRACKING ALGORITHM

Algorithm 1 Face tracking
If (detected face num = previous detected face num) then CompareLocation(); RenewMetrics(); else if (detected face num > previous detected face num) then if (previous detected face num = 0) then CreateNewFaceObject(); else CreateNewFaceObject(); CompareLocation(); RenewMetrics(); endif endif else CompareLocation(); RenewMetrics(); CheckRenewStatus(); endif endif

TABLE III: COMPARE LOCATION ALGORITHM

Algorithm 2 CompareLocation
for all (objects in DetectedFaceLibrary) do Compute Distance To All Previous Face location(); endfor Choose Nearest Location(); Set Coordinate();

The CompareLocation is described as Algorithm 2. Based on the assumption that the movement of the same face is small between two frames, we consider that current face's location must be the nearest to the location which it appeared in previous frame. So, we traverse all the face objects in the detected face library, compute the distance to all previous face locations, and then choose the nearest one to match with the current face. This assumption of small movement is borrowed from the optical flow method [11]. Besides, the video streams of a movie or video monitoring are all at least 24/30 frames per second (fps), which means the time gap between two adjacent frames is really small (between 33 to 42 milliseconds). With the low speed characteristic of human motion, we feel this kind of small movement assumption is valid for the scope of this work.

IV. EXPERIMENT ENVIRONMENT

A. OpenCV

Then OpenCV 2.4 comes with the new Face Recognizer class for face recognition, which helps researchers to experiment with face recognition immediately. In our project, this module is employed to implement the face detection [4].

B. Video benchmark

The first video benchmark that we use is the YouTube faces database [8]. This database includes 3425 videos of 1595 different people. All the videos are downloaded from YouTube. For each subject, there is an average of 2.15 videos available. As to the video clip, the shortest is 48 frames while the longest is 6070 frames. The average length of a video clip is 181.3 frames. We tried two different resolution categories of videos, one is 640×480, and the other one is 480×360. What's more, to examine the capability of our system, another full HD resolution video with moving target was also used.

C. GPU and CPU platform

The GPU used in this study is an Nvidia Tesla K40, the specification of which is: 2880 CUDA cores, 12GB GDDR5 memory and 288GB/s memory bandwidth. On the CPU side, we use two Intel Xeon E5-2650 v3 CPUs. Each CPU has 10 cores with 2.3 GHz base frequency. In this work, we turn off two CPUs' Hyper Threading (HT) function. So, there are 20 threads available for CPU version program to utilize.

D. Other system hardware and software settings

The system's hardware settings are: 64GB of DDR4-2133 memory, 1 TB of SSD and one GT 740 GPU card only for video output. The system's software settings are: Ubuntu 14.04 LTS 64-bit operating system, CUDA 7.0 64-bit GPU runtime environment, Nvidia Parallel Nsight Eclipse edition 7.0 IDE, OpenCV 2.4.10, Intel Thread Building Blocks (TBB) 4.3.

V. EXPERIMENT RESULTS

A. Key frame extraction results on YouTube face database and the speedup analysis

TABLE IV: SPEEDUP COMPARISON

Video Resolution	CPU (fps)	GPU (fps)	Speedup
640×480	17.17	48.25	2.81X
480×360	26.74	81.42	3.04X

We performed the key-frame extraction on Tesla K40 GPU platform and the CPU platform. The speedup of two different solutions is compared and shown in Table IV. In this experiment, the minimum detection block area of the face detector is 25 pixels by 25 pixels, which occupies about 0.21% and 0.36% of the frame area of videos with the resolutions of 640×480 and 480×360, respectively. We can see that, in both resolution categories, the CPU failed to reach the real-time (≥ 30 fps) performance, although the CPU version code is parallelized. Especially, the CPU version of OpenCV is parallelized with Intel TBB and all 20 CPU cores are fully

used. On the other side, the GPU implementation in both resolution category reached 48.25fps and 81.42fps, separately. With the GPU acceleration, the system easily reached the real-time performance level.

The key-frame analysis results are discussed next. First, for one face video shown in Figure 2, we can see the frame number, the face number or the person ID, the frame number for corresponding face's appearance and the key-frame number. The information in the extracted key-frame is attached by our implementation and saved as an output image file. Secondly, Figures 3 and 4 shows the result for the two-face case. Figure 3 represents the result of face No.1. We can see that the best frame of that face is the 87th frame. And for face No.2 in Figure 4, the best frame for that face is the 117th frame.

Fig.2. One-face video key-frame extraction result

Fig.3. Key-frame extraction result of face No.1 in two-face video

Fig.4. Key-frame extraction result of face No.2 in two-face video

B. Experiment results of more dynamically moving target

In the previous subsection, we can see that the faces in the video sequences are relatively static. Basically, people are not walking around in the scene, and they only sit or stand still and moving their heads. In this part, we will test our system on the video in which people are walking around. The video resolution is 1920×1080, which is full HD level. The minimum detection block area of the face detector here was set to 85 pixels by 85 pixels, which occupies about 0.35% area of the whole frame. Figures 5, 6 and 7 are the result of the key-frame extraction.

Fig.5. Key-frame extraction result of Face No.2

Fig.6. Key-frame extraction result of Face No.3

Fig.7. Key-frame extraction result of Face No.8

We can see that, all these three generated key-frames are with relatively high quality. For the left person in Figure 5, he is walking from his seat on the left towards the right side of the scene, who is marked as Face No.2. Figure 5 is the key-frame extracted when he was leaving his seat, and Figure 7 is

the key-frame extracted when he reached the right part of the scene and turned around, who is marked as Face No. 8. For the right person in Figure 5, he was leaving his seat and walked towards the left part of the scene. Figure 6 is the key-frame for him, who is marked as Face No. 3. We can see that he was in front of the camera, and his face image is nicely symmetric.

The reason for why the face numbers in the key-frames are not continuous is, with the movement of the people, their faces are detected or lose detection by the face detector. And for each newly detected face, the system will give this face a unique ID to identify it. Because the faces detected with ID No. 1, 4, 5, 6 and 7 did not last longer than the extraction threshold as previously explained in Section III, which is set to 10 frames in this experiment, they are discarded. The result of this experiment shows our system's performance also reached real-time level even for high-definition videos, which is 37.39fps in this experiment.

VI. CONCLUSIONS

Video analytics as an emerging area has found itself widely used in entertainment, health-care, automotive, transportation, safety and security. Recently, the face in video recognition has gained great attention. However, running face detection and recognition on every frame poses great computational demand, especially when there is a real-time processing requirement. In this paper, we proposed a GPU accelerated key-frame extraction scheme. The key-frame of a face is extracted according to the face image quality. The four metrics, namely, symmetry, resolution, sharpness and brightness, are used to evaluate the quality of the face image. From the experimental results, our GPU accelerated approach achieved a speedup from 2.8 to 3.0 times against the parallel CPU implementation on a 20-core high performance platform. By using GPU acceleration, the system exceeds the real-time demand, the performance of which is higher than 30 frames per second (fps) by a large margin. We also tried high-definition moving target scenario, and our scheme is able to reach the real-time processing speed as well.

As for future work, first we want to implement more optimization methods on the system. One possible method is using GPU streaming, which can let GPU execute multiple streams or kernels at the same time. This method will improve the GPU occupancy rate and lead to higher frame rate. Especially, the GPU face detection module can be improved to support stream in OpenCV 3.0. With this, the speedup can be further improved. We will also integrate new features to handle more complex test dataset, especially more dynamically moving target videos.

ACKNOWLEDGEMENT

The authors would like to thanks Joseph Aguayo for the discussions that greatly improve the quality of this work. This material is based upon work supported by the Center for Identification Technology Research and the National Science Foundation under Grant No. 1068055. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Center for Identification Technology Research or the National Science Foundation.

REFERENCES

- [1] Nasrollahi, Kamal, and Thomas B. Moeslund. "Face quality assessment system in video sequences." *Biometrics and Identity Management*. Springer Berlin Heidelberg, 2008. 10-18.
- [2] Breitenstein, M. D., Kuettel, D., Weise, T., Van Gool, L., and Pfister, H. "Real-time face pose estimation from single range images." *Computer Vision and Pattern Recognition*, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.
- [3] Chen, Xiao-hua, and Chun-zhi Li. "Image quality assessment model based on features and applications in face recognition." *Signal Processing, Communications and Computing (ICSPCC)*, 2011 IEEE International Conference on. IEEE, 2011.
- [4] OpenCVFaceRecognizer
http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html
- [5] Jain, Anil, Ross, Arun A., Nandakumar, Karthik, "Introduction to Biometrics" 2011, ISBN 978-0-387-77326-1
- [6] Nasrollahi, Kamal, and Thomas B. Moeslund. "Face quality assessment system in video sequences." *Biometrics and Identity Management*. Springer Berlin Heidelberg, 2008. 10-18.
- [7] Yakhu, Sofina, and Nikom Suvonvorn. "Key Frame Extraction from Video Sequence Using a Face Quality Index Applied to Surveillance System." *Proceedings of ECTI Conference on Application Research and Development (ECTI-CARD)*, 2009.
- [8] Youtube face database: <http://www.cs.tau.ac.il/~wolf/ytfaces/>
- [9] Begum, S. A., & Askarunisa, A. Performance Analysis of Various Key Frame Extraction Methods for Surveillance Applications 2010
- [10] NIST Face In Video Evaluation: <http://www.nist.gov/itl/iad/ig/five.cfm>
- [11] https://en.wikipedia.org/wiki/Optical_flow
- [12] Jaspreet Kaur Mann, Navjot Kaur. Key Frame Extraction from a Video using Edge Change Ratio. *International Journal of Advanced Research in Computer Science and Software Engineering*. Volume 5, Issue 5, pp.1288-1233, May 2015.
- [13] Kim, Hyunhee, and Yongho Kim. "A two-step model for video key-frame determination." *Proceedings of the American Society for Information Science and Technology* 46.1 (2009): 1-16.
- [14] Şaykol, E., Baştan, M., Güdükbay, U., & Ulusoy, Ö. "Keyframe labeling technique for surveillance event classification." *Optical Engineering* 49.11 (2010): 117203-117203.
- [15] Schoeffmann, K., Del Fabro, M., Szkaliczki, T., Böszörményi, L., & Keckstein, J. "Keyframe extraction in endoscopic video." *Multimedia Tools and Applications* (2014): 1-20.
- [16] Liu, Lijie, and Guoliang Fan. "Combined key-frame extraction and object-based video segmentation." *Circuits and Systems for Video Technology*, IEEE Transactions on 15.7 (2005): 869-884.
- [17] Shih, Huang-Chia. "Key-frame extraction and key-frame rate determination using human attention modeling." *Multimedia and Expo (ICME)*, 2011 IEEE International Conference on. IEEE, 2011.
- [18] Kalirajan, K., and M. Sudha. "Moving Object Detection for Video Surveillance." *The Scientific World Journal* 2015 (2015).
- [19] Du, Y., Wang, X., Wang, S. T., Xue, S., and Qin, Z. "Hybrid moving object detection system based on key frame extraction." *Industrial Electronics and Applications (ICIEA)*, 2011 6th IEEE Conference on. IEEE, 2011.
- [20] Kim, Changick, and Jenq-Neng Hwang. "Object-based video abstraction for video surveillance systems." *Circuits and Systems for Video Technology*, IEEE Transactions on 12.12 (2002): 1128-1138.
- [21] OpenCV face detection module and API,
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_api.html.