Parallel Implementation of Eye Detection Algorithm on Color Facial Images

Jalal A. Nasiri, Amir H. Bahmani, Hissein Deldari, H. Sadoghi Yazdi

1Communication and Computer Research Center, Department of Computer Engineering, Ferdowsi University of Mashhad, Iran
2Department of Computer Engineering, Islamic Azad University of Mashhad, Mashhad, Iran
3Computer Engineering Department, Ferdowsi University of Mashhad, Iran

Abstract: As processing power becomes cheaper and more available by using cluster of computers, the need for parallel algorithms which can harness this computing potentials is increasing. Eye detection is an application of parallel algorithms. Detecting eyes in images plays an important role in many applications such as face detection/recognition. In addition, its widespread usage as a part of series applications made it a nontrivial task which should be worked on. Moreover, existing algorithms are usually much time-consuming. In this paper we have proposed a parallel algorithm in EREW PRAM model for eye detection in color images. Using color characteristics is a useful way to detect eyes. We use special color space, YC'C, which its components give us worthwhile information about eyes. We make two maps in parallel according to their components and merge them to obtain a final map. The proposed algorithm has been examined with MPI and its implementation results on CVL and Iranian databases showed that parallel approach reduces the time of detection efficiently. Exploiting p processors has reduced the time of detection to \( n/p + c \) which c is the communication overhead between the processors and n is the number of pixels of a particular image.

Key words: Parallel algorithm, Eye detection, and Color images.

INTRODUCTION

Image processing applications can be computationally intensive due to large amount of data which is processed and complexity of image processing algorithms. The best known approach to overcome this issue is using parallel computing in image processing applications. One of the widely used applications in this area of image processing is Eye Detection. Eye detection is a crucial step in many applications such as face detection/recognition, face expression analysis, gaze estimation, criminal investigation, human interactions and surveillance systems (Chia-Feng, J. and S. Shen-Jie, 2008; Gokberk, B., H. Dutagaci 2008; Yang, J., X. Ling, 2007).

Existing works in eye detection can be classified into two major categories: traditional image-based passive approaches and the active IR based approaches. The former uses intensity and shape of eyes for detection and the latter works on the assumption that eyes have a reflection under near IR illumination and produce bright/dark pupil effect.

The traditional methods can be broadly classified into three categories: template based methods (Feng, G. and P. Yuen, 2001; Huang, W.M. and R. Mariani, 2000), appearance based methods (Huang, J. and H. Wechsler, 1999; Pentland, A., B. Moghaddam, 1994) and feature based methods (Sirohey, S.A. and A. Rosenfeld, 2001; Kawato, S. and N. Tetsutani, 2002). Our approach is considered to be in first category.

Color is one of the useful features used for eye detection. Thilak et al.(2002) proposed an algorithm which by three levels detects eyes. First they localized eye candidates by simple thresholding on HSV color space and normalized RGB color space sequentially. It is then followed by connected component analysis to determine spatially connected regions and reduce the search space to determine the eye pair windows. Ultimately, the mean and variance projection function is applied to validate the presence of eye in each window. Lin et al.(2004) proposed an algorithm which uses HSI color space to extract skin color pixels and uses region growing algorithm to group these pixels. Then by the means of Face Circle Fitting (FCF) method, they detect face region and thereafter apply Dark-pixel Filter (DFP) to identify candidate’s eye. At last, they use geometric relation to find eye positions. Gargesha et al.(2002) combine the techniques of chrominance and...
luminance and curvature analysis to compute eye maps. The exact position of eyes could be determined using either PCA or Random transform.

We construct Eye Maps and by combining them, determine candidate’s eye from the final EyeMap. Eye Map is obtained from a facial image that is transformed into YC, color space (Douglas, C. and N. King, 1999). The two highest peaks (brightest regions) in Eye Map are supposed to be eyes (Rein-Lien, H., A.M. Mohamed, 2002).

Our simulation results showed that two highest peaks do not always correspond to eyes, i.e. input image is noisy or under poor lighting conditions. An extra phase is designed to overcome these situations. In this phase, the bright regions that satisfy some special features are considered as eye pair. Experimental results showed this phase improved detection rate saliently.

The rest of this paper is organized as follows: In section 2, the proposed algorithm is discussed. Section 3 describes the implementation result on a cluster. We conclude in 4.

**Proposed Parallel Algorithm:**

In this section we discuss the details of our proposed parallel algorithm. We first build two separate eye maps from facial image, EyeMapC from the chrominance components and EyeMapL from the luminance component. These two maps are then combined into a single eye map, EyeMap. The facial image should be frontal and not occluded by objects like glasses, mask and so on. Also both eyes should be visible in input image so head rotation at most 30° around vertical axis and 10° around horizontal axis is acceptable. The flowchart of parallel algorithm is shown in Fig. 1.

![Fig. 1: Parallel EyeMap Construction](image)

**Parallel EyeMapC Construction:**

**Idea of Algorithm:**

The main idea of EyeMapC is based on characteristics of eyes in YC, color space which demonstrates that eye regions have high C, and low C, values (Nasiri, J.A., S. Khanchi, 2007). It is constructed by:

\[
\text{EyeMap} \quad C = \frac{1}{3}(\overline{C_y^2} + \overline{C_r^2} + \frac{\overline{C_b^2}}{C_y})
\]

Where \((\overline{C_y^2}), (\overline{C_r^2})\) and \(\frac{\overline{C_b^2}}{C_y}\) all are normalized to the range [0 1] and \((C)_2^2\) is the negative of \((\overline{C_r^2})^2\)
(i.e., 1-Cr). This formula is designed to brighten pixels with high \( C_b \) and low \( C_r \) values. \((C_{\hat{b}})^2\) emphasizes pixels with higher \( C_b \) value and causes pixels with lower \( C_b \) value become weaker, also \( \left( \frac{C_b}{C_r} \right) \) results in pixels with low \( Cr \) become brighter. Finally \( \left( \frac{C_b}{C_r} \right) \) component completes our idea that eye regions have high \( C_b \) and low \( Cr \) values. The 1/3 scaling factor is applied to ensure that the resulted \( EyeMapC \) stays within the range of [0 1]. The process of \( EyeMapC \) construction is depicted in Fig. 2.

\[
(C_{\hat{b}})^2
\]

\[
(C_b)^2
\]

\[
(C_r)^2
\]

\[
Y(x,y) \quad \text{and} \quad g(x,y)
\]

\[
EyeMapL = \frac{Y(x,y) \circ g(x,y)}{Y(x,y) \bullet g(x,y)}
\]

Where the grayscale dilation and erosion operations on a function: \( f : F \subseteq \mathbb{R}^d \rightarrow \mathbb{R} \) using a structuring function \( g : G \subseteq \mathbb{R}^d \rightarrow \mathbb{R} \) are defined in (Jackway, P. and M. Deriche, 1996). Since this step uses the \( Y \) component of \( YC_C \) color space constructing of \( EyeMapL \) can be in parallel to the previous section that is \( EyeMapC \). Pseudocode of this step is depicted in Fig. 4.
Parallel EyeMapC Algorithm

1: if rank = 0  
2: for i = 1 to p do  
3: \[ \text{send}((\text{Image} \times (i - 1)) \mod p, \text{processor}_i) \]  
4: else  
5: \[ \text{receive}(\text{MyPartC}_i, \frac{n}{p}, \text{processor}_i) \]  
6: for all processors  
7: for i = 1 to \( \frac{n}{p} \) do  
8: \[ \text{compute MyPartC} = \frac{1}{4}((C_0)^2 + (C_r)^2 + (C_p)^2) \]  
9: if rank \( <> \) 0  
10: \[ \text{send(MyPartC}_i, \frac{n}{p}, \text{processor}_i) \]  
11: else  
12: for i = 1 to p do  
13: \[ \text{EyeMapC} \leftarrow \text{receive(MyPartC}_i, \frac{n}{p}, \text{processor}_i) \]

Fig. 3: Pseudocode of Parallel EyeMapC Construction

Parallel EyeMapL Algorithm

1: for all processors  
2: for i = 1 to \( \frac{n}{p} \) do  
3: \[ \text{compute MyPartL} = \frac{Y(s,0)}{Y(s,0) \times Y(0,v)} \]  
4: if rank \( <> \) 0  
5: \[ \text{send(MyPartL}_i, \frac{n}{p}, \text{processor}_i) \]  
6: else  
7: for i = 1 to p do  
8: \[ \text{EyeMapL} \leftarrow \text{receive(MyPartL}_i, \frac{n}{p}, \text{processor}_i) \]

Fig. 4: Pseudocode of Parallel EyeMapL Construction

In which MyPartL is the part that is received from the Master node’s image.

Parallel EyeMap Construction:

After constructing EyeMapC and EyeMapL in previous steps, we multiply them to obtain the final EyeMap in parallel, i.e., \( \text{EyeMap} = (\text{EyeMapC}) \AND (\text{EyeMapL}) \). This step is illustrated in Fig. 5. Each processor \( p \) is responsible for \( n/p \) rows of each EyeMap and multiply the corresponding elements. The resulting eye map is then dilated, masked, and normalized to brighten both eyes and suppress other facial areas. The location of the candidate’s eyes are estimated and then refined using thresholding and binary morphological erosion on this eye map.

Experimental Result:

In this section, we show data related to effectiveness of proposed parallel algorithm on CVL (Solina, F., P. Peer, 2003) and Iranian Databases. We have used a Beowulf cluster with 8 nodes and Ethernet 10/100 network infrastructure. The algorithm have been implemented with LAM/MPI ver. 7.1.14. The results of CVL and Iranian Databases are depicted in Table. 1 and Table. 2. In the following sections, we will examine the efficiency of our MPI program and the parallel speed of proposed algorithm.
**Table 1:** Results on CVL database

<table>
<thead>
<tr>
<th>Expression</th>
<th>No. of Data</th>
<th>Data rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serious</td>
<td>110</td>
<td>90</td>
</tr>
<tr>
<td>Smile</td>
<td>110</td>
<td>86.36</td>
</tr>
<tr>
<td>Grin</td>
<td>110</td>
<td>81.81</td>
</tr>
<tr>
<td>Total</td>
<td>110</td>
<td>86.06</td>
</tr>
</tbody>
</table>

**Table 2:** Results on Iranian database

<table>
<thead>
<tr>
<th>Gender</th>
<th>No. of Image</th>
<th>Data rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>28</td>
<td>85.71</td>
</tr>
<tr>
<td>Male</td>
<td>22</td>
<td>86.36</td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td>86</td>
</tr>
</tbody>
</table>

**CVL:**

CVL database consists of head and shoulder images taken from 114 people in 7 kinds of expressions. Among 7 images taken from a person, 3 of them are suitable for our purpose. These three photos are frontal view and with different expressions: serious, smile and grin. Samples of eye detection on CVL database are illustrated in Fig. 6.

**Iranian Database:**

Iranian database consists of head and shoulder images taken from 50 people. Images in Iranian database are taken under various lighting conditions. Some samples of detection are depicted in Fig. 7.
**MPI Efficiency:**

In order to examine the efficiency of a MPI program, a small arithmetic operations program should be written. The overall speed of all processors is determined by taking the amount tasks which should be done and dividing the total execution time. This is illustrated in Fig. 8. This figure shows that the ratio of growth in number of processors to growth in speed is constant. This indicates that the MPI process is efficient.

![MPI Efficiency](image)

**Fig. 8:** MPI Efficiency of a simple arithmetic program.

**Speed of Parallel Algorithm:**

One of the best performance evaluation in parallel programming is the speed comparison of parallel and serial implementations. In Fig. 9, the speed of parallel and serial implementations is depicted on a 1600x 1200 pixel color image. We have changed the number of processors from 1 to 8 gradually. The total time for the serial algorithm is constant for any value of number of processors. Surprisingly, when we increase the number of processors to 2 processors, the time gets worth. This is because of an extra cost which is the cost of communication. As the number of processors increases, the total time of parallel implementation decreases. Ideal diagram is also illustrated in this figure and it is the expected total time of parallel implementation without the cost of communication.

![Speed of Parallel Algorithm](image)

**Fig. 9:** Speed of Parallel Algorithm against Serial Algorithm.

**Conclusion:**

In this paper we have proposed a parallel algorithm for detecting eyes in color images. Our method detects eyes in face image which is extracted over the entire image. This algorithm uses a special color space called $\text{YC}_C$. It constructs two maps in parallel and merge these two maps together to achieve a final map. Implementation results in MPI on CVL and Iranian Databases indicate a considerable reduction in time for detecting eyes in a color image.
ACKNOWLEDGMENT

This work was supported by Communications and Computer Research Center, Ministry of Information Technology, Mashhad, Iran.

REFERENCES


