IIT INDORE

27/11/2019

PROJECT REPORT

Parallel Processing for Face Detection and Recognition

Submitted by-

1) Seemandhar Jain
   (170001046)
2) Akshit Khatgarh
   (170001004)

Submitted To-

Dr. Surya Prakash
Associate Professor in C.S.E
. IIT INDORE
Shubham Kumar
(M.S. (Res.)))

I. INTRODUCTION 4

II. NEEDS/PROBLEMS 5

III. GOALS/OBJECTIVES 6

IV. TIMETABLE 6

V. DATASET RUN 7

VI. RESULT 8

VII. CODEPARALLEL 9

VIII. MODEL DISCUSSED 10

IX. REFERENCES 11

ABSTRACT

Image segmentation is the process of dividing an image into multiple parts. It is typically used to identify objects or other relevant information in digital images. There are many ways to perform image segmentation including Thresholding methods, Color-based segmentation, Transform methods among many others. Alternately edge detection can be used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images. Image thresholding is most effective in images with high levels of contrast. Otsu's method, named after Nobuyuki Otsu, is one such implementation of Image Thresholding which involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. An image can have horizontal, vertical or diagonal edges. The Sobel operator is used to detect two kinds of edges in an image by making use of a derivative mask, one for the horizontal edges and one for the vertical edges.

1. Introduction

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

2. Needs/Problems

There have been widely applied many researches related to face recognition system. The system is commonly used for video surveillance, human and computer interaction, robot navigation, and etc. Along with the utilization of the system, it leads to the need for a faster system response, such as robot navigation or application for public safety. A number of classification algorithms have been applied to face recognition system, but it still has a problem in terms of computing time. In this system, computing time of the classification or feature extraction is an important thing for further concern. To improve the algorithmic efficiency of face detection, we combine the eigenface method using Haar-like features to detect both of eyes and face, and Robert cross edge detector to locate the human face position. Robert Cross uses the integral image representation and simple rectangular features to eliminate the need of expensive calculation of multi-scale image pyramid.

3. Objectives

Some techniques used in this application are

1. Eigen-face technique

2. KLT Algorithm

3. Parallel for loop in openmp

4. OpenCV for face detection.

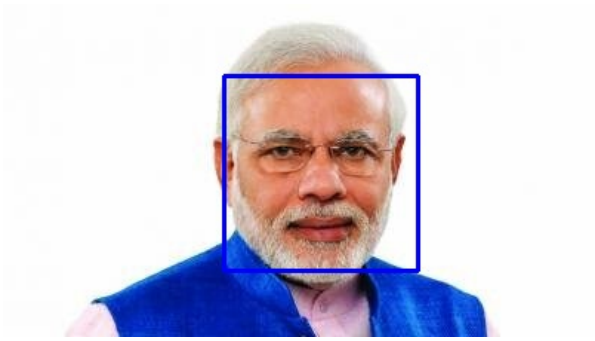
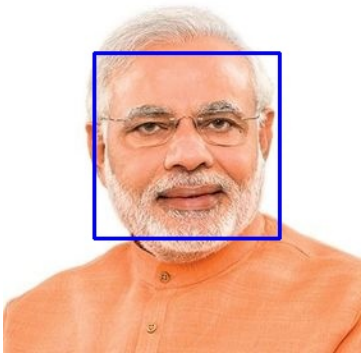
5. Further uses of the techniques

4. Timetable

	Description of Work	Start and End Dates
Phase One	Getting Ready and Collection of Data	24/10/2019
Phase Two	Implementing the diff. algorithms and comparing them. Finding the results.	24/10/2019 to 26/11/19
Phase Three	Final evaluation And Submission.	27/11/19

Dataset 1 (Narendra Modi)-





Output-

Parallel-

Photo norm: 8904.403921568626 / per pixel: 0.17295813999900211

Zero norm: 50731.0 / per pixel: 0.9853932366023735

Images Are of same Person!!!!!Hurray...!!!

Time For Parallel Execution - 0.214631790603905

Serial-

Photo norm: 8904.403921568626 / per pixel: 0.17295813999900211

Zero norm: 50731.0 / per pixel: 0.9853932366023735

Images Are of same Person!!!!!Hurray...!!!

Time For Serial Execution - 0.841835156593051

Comparison-

$$SpeedUp = TimeOfExecutionInSerial / TimeOfExecutionInParallel$$

⇨ $\boldsymbol{SpeedUp = .8418/.214 = 3.93}$

Since there has been used of only 4 Processors..=> Our parallel Algo works efficient in case of serial/parallel execution. **(WORK OPTIMAL)**

Dataset 2 (Bridge)-

Output-

Serial-

The images have same size and channels

They are not completely Equal

Time For parallel Execution - 0.344634

Parallel-

The images have same size and channels

They are not completely Equal

Time For parallel Execution - 0.114631

DataSet 3-

Output-

Use ``imageio.imread`` instead.

(480, 640)

Use ``imageio.imread`` instead.

face = misc.imread('sj149.jpg', cv.IMREAD_UNCHANGED)

(480, 640)

Photo norm: 23046.91966386555 / per pixel: 0.14090372788233756

Zero norm: 52272.0 / per pixel: 1.0

Images Are of same Person!!!!!Hurray...!!!

0.2018456266988866

Code-

1)This Parallel loop for extracting faces

```
with concurrent.futures.ThreadPoolExecutor() as executor:
    for (x, y, w, h) in faces:
        cv.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

2)This Parallel Loop for conversion into grey code

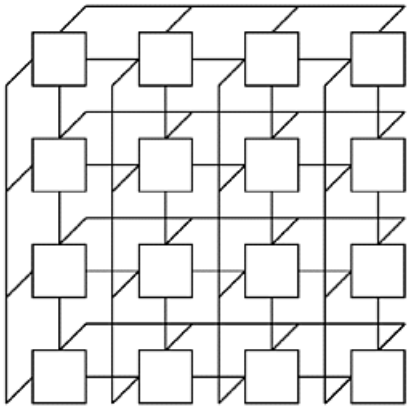
```
with concurrent.futures.ThreadPoolExecutor() as executor:
    for i in range(H):
        with concurrent.futures.ThreadPoolExecutor() as executor:
            for j in range(W):
                gray[i,j] = np.clip(0.07 * img[i,j,0]  + 0.72 * img[i,j,1] + 0.21 * img[i,j,2], 0, 255)
```

3)This function for comparing two extracted Face

```
def compare_images(img1, img2):
    img1 = normalize(img1)
    img2 = normalize(img2)
    diff = img1 - img2
    m_norm = sum(abs(diff))
    z_norm = norm(diff.ravel(), 0)
    return (m_norm, z_norm)
```

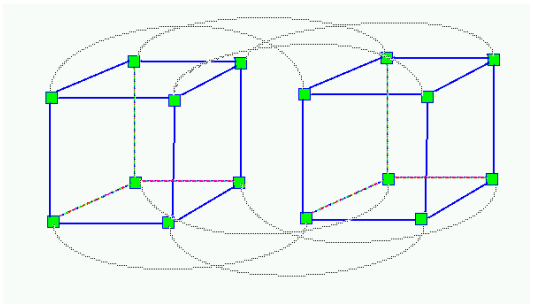
Models Discussed-

1) Mesh Connected Model



Dividing Each Photo into N*N Network and assign each pixel to Each processor and then each processor Will compare the photo and save the result in shared Memory.

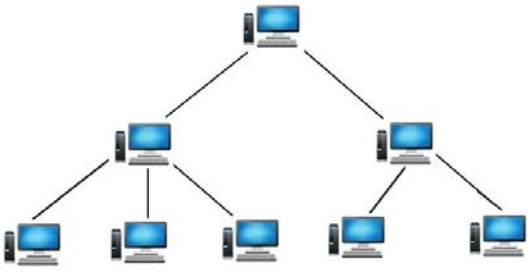
2) Hypercube Model



Each vertex is given with each row of image and same comparison row wise happens and will compare the photo and save the result in shared Memory.

3) Tree Connected Model

It works best in our case as each pixel was assigned to the leaf processor and then each processor Will compare the photo and save the result and passed the result into upper processor and final comparison happens at root node whether they are same or not



References-

1. C. Akinlar and C. Topal, "ColorED: Color edge and segment detection by Edge Drawing (ED)," Journal of Visual Communication and Image Representation, vol. 44, pp. 82–94, 2017. View at Publisher · View at Google Scholar · View at Scopus
2. Hildreth E C., Edge detection, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, 1985.
3. B. Julesz, "A Method of Coding Television Signals Based on Edge Detection," Bell System Technical Journal, vol. 38, no. 4, pp. 1001–1020, 1959. View at Publisher · View at Google Scholar · View at Scopus
4. L. D. Roberts, Machine perception of three-dimension solids in optical and electro-optimal information processing, Massachusetts Institute of Technology Press, Cambridge, UK, 1966.
5. B. Tang and W. Long, "Fast Canny algorithm based on GPU + CPU," Chinese Journal of Liquid Crystals and Displays, vol. 31, no. 7, pp. 714–720, 2016. View at Publisher · View at Google Scholar · View at Scopus
6. H. Cui Y, J. Cao F, and H. Shi, "Parallel PSO-BP neural network algorithm based on MapReduce," in Bulletin of Science and Technology, vol. 33, pp. 110–115, 110-115, 33(4, 2017. View at Google Scholar
7. W. Zhu S and P. Wang, "Large-scale image retrieval solution based on Hadoop cloud computing platform," Journal of Computer Applications, vol. 34, no. 3, pp. 695–699, 2014. View at Google Scholar
8. A. L. Wang and X. S. Liu, "Vehicle license plate location based on improved Roberts operator and mathematical morphology," in Second International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC 2012), Harbin, Peoples R China, 2012. View at Publisher · View at Google Scholar
9. T. Yang, HW. Tian, XM. Liu, Ke. XT, and PJ. Xing, "Otsu thresholding segmentation method based on two boundaries and its fast algorithm," Application Research of Computers, vol. 33, no. 12, pp. 3872–3875, 2016. View at Google Scholar
10. H. Zhang, Q. Zhu, C. Fan, and D. Deng, "Image quality assessment based on Prewitt magnitude," AEÜ - International Journal of Electronics and Communications, vol. 67, no. 9, pp. 799–803, 2013. View at Publisher · View at Google Scholar · View at Scopus
11. N. Dwivedi, K. Srivastava, and N. Arya, "Sanskrit word recognition using Prewitt's operator and support vector classification," in Proceedings of the IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICE-CCN'13)., , MAR 25-26, Infant Jesus Coll Engn Technol, Dept Elect Commun Engn, Tirunelveli, India, 2013.
12. P.-M. Nguyen, J.-H. Cho, and S. B. Cho, "An architecture for real-time hardware co-simulation of edge detection in image processing using Prewitt edge operator," in Proceedings of the 13th International Conference on Electronics, Information, and Communication, ICEIC 2014, Malaysia, January 2014. View at Publisher.