Middle-East Journal of Scientific Research 20 (12): 2395-2397, 2014
ISSN 1990-9233
© IDOSI Publications, 2014
DOI: 10.5829/idosi.mejsr.2014.20.12.346

Face Recognition Using Opencl

M. Gopala Krishnan and P. RachelinSujae

Bharath University, Chenai, India

Abstract: Face recognition is the biometric identification of human's face and matching the image against a library of known faces. The algorithm used to simulate the above is Eigen faces algorithm. The software which is been proposed to implement is OpenCL. OpenCL(Open Computing Language) is an open standard for general purpose parallel programming across CPUs, GPUs and other processors, is portable and provides efficient access to the power of these heterogeneous processing platforms. The OpenCL API has functions to identify devices, compile programs, send and receive information and run OpenCL programs on the chosen device. The proposed system is been simulated in MSVS (Microsoft visual studio 2010) using AMD APP SDK (Advanced Micro Devices Accelerated Parallel Processing Software Development Kit).

Key words: Face recognition · OpenCL · Microsoft Visual Studio · EigenFaces

INTRODUCTION

A face recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source [1]. It is the ability to recognize people by their facial characteristics. One of the ways to do this is by comparing selected facial features from the image and a facial database. It involves the comparison of a perceived pattern with stored representations of familiar faces. One notable aspect of face recognition is the broad interdisciplinary nature of the interest in it: within computer recognition and pattern recognition; biometrics and security; multimedia processing; psychology and neuroscience [2]. Face recognition as a biometric derives a number of advantages from being the primary biometric that humans use to recognize one another. Some of the earliest identification tokens, *i.e.* portraits, use this biometric as an authentication pattern. Furthermore it is well-accepted and easily understood by people and it is easy for a human operator to arbitrate machine decisions -in fact face images are often used as a human-verifiable backup to automated fingerprint recognition systems. Face recognition has the advantage of ubiquity and of being universal over other major biometrics, in that everyone has a face and everyone readily displays the face.

The most advanced technology is based on the Eigenface algorithm, which maps the characteristics of a person's face into a multidimensional face space [3]. Computers can conduct facial database searches and/or perform live, one-to-one or one-to-many verifications with unprecedented accuracy and splitsecond processing. Users can be granted secure access to their computer, mobile devices, or for online e-commerce, simply by looking into their Webcamera.Neural networks were used for earlier face recognition systems, but with Eigenface, the computer cannot be easily fooled by photographs or by someone else with similar appearance.Inadequate constraint or handling of such variability inevitably leads to failures in recognition.These include:

Physical changes: facial expression change; aging; personal appearance(make-up, glasses, facial hair, hairstyle, disguise).

Acquisition geometry changes:change in scale, location and in-plane rotation of the face (facing the camera) as well as rotation in depth (facing the camera obliquely, or presentation of a profile, not full-frontal face).

Imaging changes: lighting variation; camera variations; channel characteristics (especially in broadcast, or compressed images).

The main challenges of face recognition today are handling rotation in depth and broad lighting changes,

Corresponding Author: P. RachelinSujae, Asst. Prof, Bharath University, ch, India.

together with personal appearance changes [4, 5]. Even improved. Many applications for face recognition have been envisaged. Commercial applications have so far only scratched the surface of the potential. Installations so far are limited in their ability to handle pose, age and lighting variations, but as technologies to handle these effects are developed, huge opportunities for deployment exist in many domains [6].

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing unit (CPUs), graphics processing unit (GPUs) and other processors. OpenCL includes a language (based on C99) for writing kernels (functions that execute on OpenCL devices), plus application programming interfaces (APIs) that are used to define and then control the platforms.

Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It is used to develop console and interface applications along with Windows Forms applications, web sites, web applications and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE,.NET Framework,.NET Compact Framework and Microsoft Silverlight. Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic.NET), C# (via Visual C#) and F# (as of Visual Studio 2010).

Methodology

Generation of Eigenfaces: Aset of eigenfacescan be generated by performing a mathematical process calledprincipal component analysis(PCA) on a large set of images depicting different human faces. Informally, eigenfaces can be considered a set of "standardized face ingredients", derived fromstatistical analysisof many pictures of faces. Any human face can be considered to be a combination of these standard faces. For example, one's face might be composed of the average face plus 10% from eigenface 1, 55% from eigenface 2 and even -3% from eigenface 3. Remarkably, it does not take many eigenfaces combined together to achieve a fair approximation of most faces. Also, because a person's face is not recorded by adigital photograph, but instead as just a list of values (one value for each eigenface in the database used), much less space is taken for each person's face.

The eigenfaces that are created will appear as light and dark areas that are arranged in a specific pattern. This pattern is how different features of a face are singled out to be evaluated and scored [7]. There will be a pattern to evaluatesymmetry, if there is any style of facial hair, where the hairline is, or evaluate the size of the nose or mouth. Other eigenfaces have patterns that are less simple to identify and the image of the eigenface may look very little like a face.

Practical Implementation

To Create a Set of Eigenfaces, One Must:

- Prepare a training set of face images. The pictures constituting the training set should have been taken under the same lighting conditions and must be normalized to have the eyes and mouths aligned across all images. They must also be all resampled to a common pixel resolution (r × c). Each image is treated as one vector, simply by concatenating the rows of pixels in the original image, resulting in a single row with r × c elements. For this implementation, it is assumed that all images of the training set are stored in a single matrixT, where each row of the matrix is an image.
- Subtract the mean. The average image a has to be calculated and then subtracted from each original image in T.
- Calculate the eigenvectors and eigenvalues of the covariance matrixS. Each eigenvector has the same dimensionality (number of components) as the original images and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called eigenfaces. They are the directions in which the images differ from the mean image. Usually this will be a computationally expensive step (if at all possible), but the practical applicability of eigenfaces stems from the possibility to compute the eigenvectors of S efficiently, without ever computing S explicitly, as detailed below.
- Choose the principal components. The D x D covariance matrix will result in D eigenvectors, each representing a direction in the r × c-dimensional image space. The eigenvectors (eigenfaces) with largest associated eigenvalue are kept.

These eigenfaces can now be used to represent both existing and new faces: we can project a new (mean-subtracted) image on the eigenfaces and thereby record how that new face differs from the mean face. The eigenvalues associated with each eigenface represent how much the images in the training set vary from the mean image in that direction. We lose information by projecting the image on a subset of the eigenvectors, but we minimize this loss by keeping those eigenfaces with the largest eigenvalues. For instance, if we are working with a 100 x 100 image, then we will obtain 10,000 eigenvectors. In practical applications, most faces can typically be identified using a projection on between 100 and 150 eigenfaces, so that most of the 10,000 eigenvectors can be discarded.

Computing the Eigenvectors: Performing PCA directly on the covariance matrix of the images is often computationally infeasible. If small, say 100 x 100, greyscale images are used, each image is a point in a 10,000-dimensional space and the covariance matrix S is a matrix of 10,000 x 10,000 = 10^8 elements [8]. However the rank of the covariance matrix is limited by the number of training examples: if there are *N* training examples, there will be at most *N-1* eigenvectors with non-zero eigenvalues. If the number of training examples is smaller than the dimensionality of the images, the principal components can be computed more easily as follows.

Let T be the matrix of preprocessed training examples, where each column contains one mean-subtracted image. The covariance matrix can then be computed as $S = TT^T$ and the eigenvector decomposition of S is given by

$$Sv_i = TT^T v_i = \lambda_i v_i$$

However TT^T is a large matrix and if instead we take the eigenvalue decomposition of

$$T^T T u_i = \lambda_i u_i$$

then we notice that by pre-multiplying both sides of the equation with T, we obtain

$$TT^T T u_i = \lambda_i T u_i$$

Meaning that, if u_i is an eigenvector of $T^T T$, then $v_i = Tu_i$ is an eigenvector of S. If we have a training set of 300 images of 100 x 100 pixels, the matrix $T^T T$ is a 300 x 300 matrix, which is much more manageable than the 10000 x 10000 covariance matrix. Notice however that the resulting vectors v_i are not normalised; if normalisation is required it should be applied as an extra step.

CONCLUSION

Thus, a face recognition algorithm (PCA based recognition algorithm) using OpenCL is simulated. The input image is the one existing in the database (here, five sample images are taken under the same illumination conditions). The algorithm gets compiled if the input image matches with the sample images in the database. The program does not get compiled for an unmatch.

Future Work: The implementation of several other algorithms can be done using the proposed run time environment and the SDK. VB (Visual Basic) interface can be showed on to project a pop up window in case of a match or unmatch. Real time face recognition can be done using a video camera or a web camera.

REFERENCES

1. Dimitri PISSARENKO Eigenface-based facial recognition.
2. Ilkeratalay, 2002. Face Recognition using Eigen faces-Computer Engineer B.ScL.I.Smith. A tutorial on principal components analysis URL http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. URL accessed on November, pp: 27.
3. Matthew, A. Turk and Alex P. Pentland, Face Recognition using Eigen Faces.
4. Digital Image Processing. Prntice-Hall, Inc., third edition, ISBN 0-201-18075-8.
5. Saravanan, T. and R. Udayakumar, 2013. Comparision of Different Digital Image watermarking techniques, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1684-1690.
6. Saravanan, T. and R. Udayakumar, 2013. Optimization of Machining Hybrid Metal matrix Composites using desirability analysis, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1691-1697.
7. Saravanan, T. and R. Udayakumar, 2013. Simulation Based line balancing of a single piece flow line, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1698-1701.
8. Saravanan, T., R. Udayakumar and G. Saritha, 2013. Simulation Based line balancing of a single piece flow line, Middle-East Journal of Scientific Research, ISSN:1990-9233, 16(12): 1790-1793.