Brno University of Technology

University of L'Aquila

Double-Degree Master's Programme - InterMaths
Applied and Interdisciplinary Mathematics

Master of Science **Mathematical Engineering**	**Master of Science** **Mathematical Engineering**
Brno University of Technology (BUT)	University of L'Aquila (UAQ)

Master's Thesis

IMAGE DESCRIPTORS AND THEIR USAGE FOR OBJECT DETECTION

Supervisor
Mgr. Jana Procházková, Phd

Candidate
Jonathan Bature

Student ID (UAQ): 267658
Student ID (BUT): 233659

Academic Year 2020/2021

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Master's Thesis

Institut:	Institute of Mathematics
Student:	**BSc Jonathan Bature, BSC.**
Degree programm:	Mathematical Engineering
Branch:	
Supervisor:	**Mgr. Jana Procházková, Ph.D.**
Academic year:	2021/22

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Image Descriptors and their Usage for Object Detection

Brief Description:

The student will study the area of image descriptors that are important in image analysis. Image descriptors are based on statistical evaluation, convolution filters or wavelet transformation. These descriptors describe the image and they are useful for object recognition, e.g. face or eye recognition, car or car sign detection and many others. Some descriptors are suitable for feature detection and they can be used in image composition.

Master's Thesis goals:

1. Study of theoretical background of different descriptors, learn the math formulation, compare and describe their properties.
2. Algorithm Viola–Jones (based on wavelet transformation) for face description.
3. Choose suitable descriptor and make its implementation in some application (e.g. face detection). For training phase there is a possibility to use neural networks.

Recommended bibliography:

VIOLA, P., JONES, M. J. Robust Real-Time Face Detection. International Journal of Computer Vision [online]. 2004, 57(2), 137-154 [cit. 2021-9-7]. ISSN 0920-5691. Dostupné z: doi:10.1023/B:VISI.0000013087.49260.fb

KUMAR, R. M., SREEKUMAR, K. A Survey on Image Feature Descriptors. International Journal of Computer Science and Information Technologies. 2014, 5(6), 7668-7673.

LEI, Z., LI, S. Z. Learning Discriminant Face Descriptor for Face Recognition. Computer Vision – ACCV 2012. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, 748-759. Lecture Notes in Computer Science. ISBN 978-3-642-37443-2. Dostupné z: doi:10.1007/978-3-642-37444-9_58

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2021/22

In Brno,

L. S.

prof. RNDr. Josef Šlapal, CSc. Director of the Institute	doc. Ing. Jaroslav Katolický, Ph.D. FME dean

Abstrakt
Detekce obličeje v obrazech je široce prozkoumaným tématem v počítačovém vidění. Algoritmus která umožnila rozpoznání obličeje a stanovila nové standardy v této oblasti byla Viola-Jones algoritmus. Tato práce popisuje a vysvětluje vlastní implementaci obličeje detektor založený na algoritmu Viola Jones pomocí Matlab CascadeObjectDetector.

Tato diplomová práce přispívá ke zkoumání technik, jako je hlavní součást analýza (PCA) pro rozměrovou redukci deskriptorů pro stanovení detekce objektu systém, který dosahuje nejlepšího kompromisu mezi výkonem a rychlostí. V našem přístupu, detekce obličeje se provádí analýzou hlavních součástí (PCA). Obraz obličeje je promítnutý do prostoru obličeje, který kóduje nejlepší variace známých obrazů obličeje. The prostor obličeje je definován vlastní tváří. Vlastní tvář je sada obličejových vlastních vektorů které nemusí odpovídat běžným rysům obličeje, jako jsou oči, nos a rty. Systém funguje tak, že promítá předem extrahované obličejové obrazy do řady obličejových prostorů které představují velké odchylky mezi známými obrazy obličeje. Obličeje jsou klasifikovány jako známé nebo neznámé tváře po porovnání s existujícím obrázkem tváře v databázi

Summary
Face detection in images is a widely explored topic in computer vision. The algorithm that enabled face recognition and set new standards in this area was the Viola-Jones algorithm. This thesis work describes and explain the actual implementation of a face detector based on the Viola Jones algorithm using the Matlab CascadeObjectDetector.

This thesis contributes to the exploration of techniques such as principal component analysis (PCA) for dimensional reduction of descriptors to establish an object detection system that achieves the best trade-off between performance and speed. In our approach, face detection is performed by Principal Component Analysis (PCA). The facial image is projected into the facial space that encodes the best variation of known facial images. The space of the face is defined by the eigenface. An eigenface is a set of facial eigenvectors that may not correspond to common facial features such as eyes, nose, and lips.
The system works by projecting pre-extracted facial images into a series of facial spaces that represent large deviations between known facial images. Faces are classified as known or unknown faces after matching with an existing face image on the database.

Klíčová slova
Obrazové deskriptory, SURF, Harris Corner Detector, Viola Jones, PCA, Covariance Matice, vlastní hodnoty a vlastní vektory, zobrazení obrazu, vlastní tváře, rozpoznávání tváří společnosti MATLAB.

Keywords
Image Descriptors, SURF, Harris Corner Detector, Viola Jones, PCA, Covariance Matrix, Eigenvalues and Eigenvectors, Image Representation, Eigen Faces, Face Recognition, MATLAB.

BATURE, J. *Obrazové deskriptory a jejich použití pro detekci objektů v obrazech.* Brno: Vysoké učení technické v Brně, Faculty of Mechanical Engineering, 2022. 52 s. Vedoucí Mgr. Jana Procházková, Ph.D.

I declare that I have written this diploma thesis Image descriptors and their usage for object detection on my own under the direction of my supervisor, Mgr. Jana Procházková, Ph.D., and using the sources listed in the bibliography..

Jonathan Bature

All glory to God almighty who is ever faithful and true to his promises. I am indebted to my supervisor, Mgr. Jana Procházková, Ph.D., for believing in me. Words are not enough to appreciate your support I don't think I can fully repay you for your sacrifice, but I will not stop trying to show you how grateful I am!

My earnest gratitude goes to all distinguished Professors and members of the department who has greatly inculcated in me a profound knowledge and mathematical skills in the course of the programme. To InterMaths & MathMods, I appreciate this great privilege you gave me to complete this Program, It has being of great value to me.

I have no words to express my gratitude to Pst. Oluseun Abimbola (SAN), Prophet. TB. Joshua for great contribution both financially and in the place prayer.
Finally, I would like to give thanks to my family members for supporting me throughout this work.

Jonathan Bature

Contents

List of Figures

1. INTRODUCTION

The task of object detection and recognition is certainly one of the most important topics in computer vision, and its applications are very diverse, such as gesture recognition [20], video tracking [11, 68], 3D modeling, robotic mapping [52].
Detecting objects are really common for us humans to do, but objects may look different depending on the viewing angle, lighting, etc. Objects can be recognised even if there are obstacles caused by another objects. In Fact we could group objects such as headphone, fruit, and bag into a single object class to distinguish between specific objects such as "your bag" and "another bag." But have you thought about computers detecting objects as we (humans) do.
The objective of object detection is to develop computational models and techniques that provide one of the most basic pieces of information needed by computer vision applications [71] : What objects are where? We linked detection with the problem of recognition where the aim is to identify a particular instance of a class. Facial detection systems can distinguish faces from "everything else", but facial recognition systems can tell the difference between my face and other faces [41].
As the rate of image and video information available increases, robust, configurable object detection systems for managing this data will become indispensable. There has been an explosion in the amount of information presented on the Internet as it was quickly transformed from a text-based medium to one of image and video content; object detection systems will be used to search through the growing number of image and video databases. Object detection is helping AI reach its full potential and the Viola-Jones Algorithm is one way to help with detecting objects. Great efforts have been made to build machines that can see like humans, but the fact that this ability is still a long way off [54]. To understand how a computer "sees", we first need a basic understanding of what an image is. Take a look at the image below in Figure 1.1:

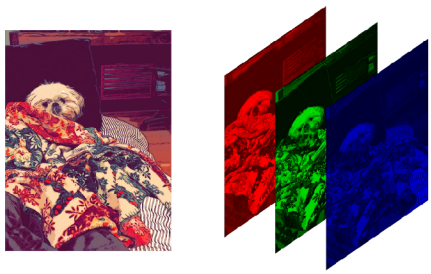


Figure 1.1: Image Illustration [45]

The left side is the original image. When the computer scans, saves, or retrieves this image, it first splits the image into three separate channels: red, green, and blue. You may recall in the science class that these are the three colors that make up white light. It is very important for us to connect these channels and lights as they will help us answer the question as whether the computer really see's images and how.
Well, they don't. At least not the conventional way. Computers only work with numbers. These numbers are often stored as scalars, vectors, or arrays and matrices. Also, these numbers can literally represent anything. But it's still a number. When the image is saved, the following process is performed:
As we have mentioned above that these image is splitted into three channels. These

channels are arrays with the same dimensions as the image's resolution (e.g. 1920×1080 pixel image will form 3 arrays with 1920 columns and 1080 rows).
After this is done, these channels are then converted into a three-dimensional array.

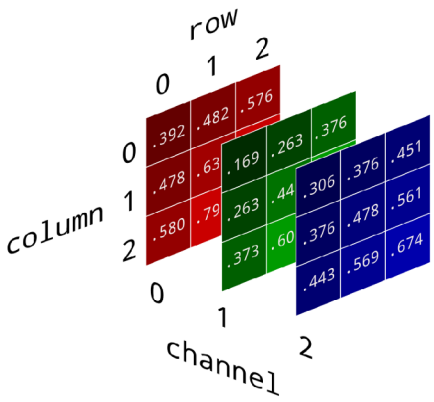


Figure 1.2: Image Channels [45]

The value stored in the cell represents the brightness of the color channel for this pixel. These values are on a scale from 0 (no light) to 255 (brightest). Therefore, if the pixel is completely black, the corresponding channel values will be {Red: 0, Green:0, Blue:0}. Conversely, if the pixel is white, its stored values will be an array {R: 255, G: 255, B: 255}.

2. Literature Review

Most of the early object detection algorithms were built on hand-crafted features. At the time, there was no effective image representation, so people had no choice but to design sophisticated feature representations and develop various speed-up skills to exhaust the limited use of computing resources.
Face recognition has been a very active field of study since the 1970s [48] due to its extensive application in many areas such as public safety, human-computer interactions, computer vision, verification [24, 58], etc.
Wei et al. [67] described some of the geometric facial features such as the eye distance to midline, horizontal eye height deviation, eye opening angle, vertical midline deviation, ear angle, mouth angle, diagonal line of eye opening, Laplace coordinates of facial landmarks, etc. The author argued that these features were comparable to those extracted from a much denser set of facial features. In addition, Gabryel and Damaševicius [19] explained the concept of key point functionality using the Bag-of-Words algorithm for image classification. The author performed image classification based on the key points obtained by the Speed-Up Robust Features (SURF) algorithm.
Face recognition is a specific and hard case of object recognition [23]. Facial recognition systems are technologies that recognize people from the acquisition sources, such as the digital image or a video frame. The first step in face recognition is face Identification or verification. This is to identify and extract areas of the human face from the background. This step uses the active contour model to detect the edges and boundaries of the face. Computer vision recognition is done in the form of face detection, face localization, face tracking, face orientation extraction, facial features, and facial expressions. Face recognition needs to take into account several technical aspects, like the poses, occlusion and illumination [25]
There are many ways in which facial recognition systems work. In general, you compare selected or global features of a particular facial image to a dataset that contains many images of the face. These extracted features are selected based on examining a person's facial color, texture, or shape pattern. The human face is not unique. Therefore, several factors cause changes in the appearance of the face. Changes in facial appearance develop two image categories, intrinsic and extrinsic factors [23].
The face recognition process [43, 69] begins with a face detection process for separating human faces in input images that include human faces. The detected face is then preprocessed to get a lower dimensional representation. This low-dimensional representation is important for effective classification.

According to Schwartz et al. (2011) [51], the face recognition method consists of the following elements. (A) Detection: detecting the face of a person in the image. (B) Identification: An image of an unknown person is matched against a gallery of known people. (C) Verification: Approval or refusal of identity claimed by the individual. Previous studies have shown that applying face recognition under well-controlled imaging conditions provides high recognition rates even with large image sets. However, when applied under uncontrolled conditions, the face recognition rate was low, resulting in a more tedious recognition process.
Jafri and Hamid [23] described the holistic approach that attempted to identify faces using global representations, i.e., descriptions based on the entire image rather than on

local features of the face. The image is represented as a 2D array of intensity values,the recognition is done through a direct correlation comparison between the input image and all other images on the database.
The main set back to the performance of the direct matching methods is the attempt to perform the classification in a very high dimensional space [22]. To counter this drawback of dimensionality, several other schemes have been proposed that use statistical dimensionality reduction methods to acquire and retain the most meaningful feature dimensions before performing recognition.
Sirovich and Kirby (1987) [58] were the first to use Principal Component Analysis (PCA) [18] to represent facial images. They showed that any face can be efficiently represented along the coordinate space of the eigenimage, and that any face can be nearly reconstructed using only a small collection of eigenimages and the corresponding projections (coefficients) along each eigenimage.
Turk and Pentland [63, 64] realized, Based on the findings of Sirovich and Kirby, that projections along eigen-images could be used as face recognition classifiers. They apply this inference to create eigenfaces that correspond to the eigenvectors associated with the highest eigenvalues of the covariance matrix of known faces (patterns) and to match the projections along the eigenfaces. We have developed a face recognition system that recognizes faces from images of known people. Eigenfaces define a feature space that significantly reduces the dimensions of the original space, and face recognition is performed in this reduced space.

Rouhi etal. (2012) [47] reviewed four facial recognition techniques during the feature extraction phase. The four methods are the 15Gabor filter, the 10Gabor filter, the Optimal Elastic Base Graph Method (EBGM), and the Local Binary Pattern (LBP). They concluded that weighted LBPs had the highest detection rates and that unweighted LBPs performed best in feature extraction among the methods used. However, the detection rates extracted by the 10 Gabor and 15 Gabor filtering techniques are higher than the EBGM and Optimal EBGM methods, even with long vector lengths.
Schwartz et al. [67] utilized a large and rich set of face identification feature descriptors.This was done by using a partial squares of multichannel weighting. Then they extended the method to a tree-based discriminant structure to reduce the time it takes to evaluate the sample. Their research shows that the proposed identification method outperform the state of the art results, especially in identifying faces captured under different conditions.
Kumar et al. (2014) [34] conducted a comparative study to compare global and local characteristic descriptors. This study examined experimental and theoretical aspects to verify the efficiency and effectiveness of these descriptors.
Soltanpour et al. (2017) [60] conducted a study explaining the state of art for 3D facial recognition using local features, with a major focus on the feature extraction process.
Nanni et al. [40] also presented a novel facial recognition system for identifying faces based on different descriptors using different pre-processing techniques.
Khanday et al. (2018) [31] reviewed various face recognition algorithms based on local feature extraction, hybrid, and dimensionality reduction approaches. They aimed to study the main methods / techniques used in face recognition. They provide an important overview of facial recognition technique and a description of key facial recognition algorithms.

2.1. Image Descriptors

In this section, the performance of different image descriptors used to characterize the face in an image are analyzed and one of the important uses of face detection is security surveillance. Where captured images are compared to thousands or millions of stored images. It becomes challenging when working with images that capture different types of noise. [5].
In this work it was proven that image descriptors can be efficient in face detection/recognition, even if noise is added to the image.
We know that an image is made of pixels as we have rightly explained, and if you want to quantify an image using only a list of numbers, we use the image descriptors.
The process of quantifying an image is called feature extraction, and the process of feature extraction determines the rules, algorithms, and methods we use to quantify the content of an image in the abstract, using only a list of numbers known as feature vectors.
So there are three things involved in it; Image descriptors , feature descriptors, feature vectors.
Image descriptors and feature descriptors have thus determined how an image is extracted and quantified, while feature vectors are the output of the descriptors used to quantify an image. So overall the process is called feature extraction.

Figure 2.1: Scheme of an Image descriptor

An input image is presented to the descriptor, the image descriptor is applied, and a feature vector (i.e., a list of numbers) is returned that is used to quantify the image content

So literally, any algorithm that attempt to understand and interpret the content of an image, utilizes feature extraction at some stage. So we could be extracting features from from an image for the variety of reasons:

- To compare the images for similarity.
- To rank images in search result while building an image search engine or to use when training an image classifier to recognise the content of image.

To make this comparison, we need a feature vector or simply features. The feature vector is used to represent various properties of an image such as shape, color, and texture of an object in an image. Combined with these features, the feature vector represents the shape and the color or the texture and the shape or it can represent all three.
Image Descriptor An image descriptor is an algorithm and methodology that governs how an input image is globally quantified and returns a feature vector abstractly representing the image contents.
Feature Descriptors A feature descriptor is an algorithm and methodology that governs how an input region of an image is locally quantified.
So, unlike an image descriptor, where you get one feature vector for each input image, a

feature descriptor can return multiple feature vectors.
Now, let us classify the importance of some specific descriptors to it usage in certain cases [5];

- The Edge Histogram could best be used with glow, gray and inverted images.
- The Fuzzy Color and Texture Histogram Descriptor (FCTH), Color Histogram, Color layout and Joint Histogram , could best be used with cropped images.
- Color and Edge Directivity Descriptor (CEDD) could be best used with images with random noise or rotated images.

2.1.1. Color and Edge Directivity Descriptor (CEDD)

Descriptor CEDD deals with a low level feature that is extracted from the images and can be used for indexing, retrieval and incorporates color and texture information in a histogram [14].
CEDD size is limited to 54bytes per image, rendering the descriptor suitable for use in large databases. One of the most important attribute of the CEDD is the low level computational power needed for its extraction, in comparison with the needs of the most MPEG-7 descriptors.

2.1.2. Fuzzy Color and Texture Histogram Descriptor

This type of descriptor deals with the extraction of low level feature that combines in one histogram, color and texture information. FCTH size is limited to 72 bytes per image, rendering this descriptor suitable for use in large image databases. This kind of descriptor is appropriate for accurately retrieving images even in distortion cases such as deformations, noise and smoothing [15].

2.1.3. Color Histogram Descriptor

Color Histogram Descriptor is one Color Histogram Descriptor of the most generally used color descriptors that characterizes the color distribution in an image. It represent the frequency distribution of color bins in an image.
It counts and save similar pixels. There are two types of color histograms ; Global color histogram and Local color histogram.
Color histogram that analyzes all spatial color frequencies in an image are termed global color distributor and they are used to solve problems such as translation, rotation and changing the angle of view. While the local color histogram focuses on individual parts of the image. The local color histogram takes into account the spatial distribution of pixels lost in the global color histogram.
This color histograms are very important when indexing and retrieving image databases because they are easy to calculate and are not affected by small changes in the image. Apart from this advantages, there are two major drawback, first, a total of geospatial data is not taken into account.
Second, histogram are not robust and unique, this is because two different images with similar color distributions result in similar histograms and the same view image with different exposure to light produces different histograms.

2.1.4. Color Layout Descriptor

Color is the most basic quality of visual content and so it could be used to describe and represent images [26].
This CLD is used in recording the spatial distribution of colors in an image. The feature extraction process consists of two parts; grid base typical color selection and discrete cosine transform (DCT) with quantization [12, 36].
The MPEG-7 standard tested the most efficient way to describe colors and chose one that gave the more satisfactory results. The standard proposes different ways to identify these descriptors, and one tool to identify these descriptors, and one tool for describing colors is the CLD. This allows you to describe the color relationships between sequences or groups of images.
Note that CLD is one of the most accurate and fastest color descriptors available.

2.1.5. Edge Histogram Descriptor (EHD)

The edges of an image are considered to be an important feature to represent the content of the image [5, 42]. The human eye is known to be sensitive to edge features for image perception proposed for MPEG-7, this edge histogram descriptor consist of only local edge distributions in the image. This means that the MPEG-7 standard egde histogram is designed to have only a local edge distribution, as it is important to keep the histogram size as small as possible in order to store the metadata efficiently.
Thus, for the edge identification and localization, we then:

- Divide the image space in 4×4 sub-images.
- Each sub-image divided into non-overlapping squared image block.
- Each image block is classified into 5 edges categories or non-edge block.

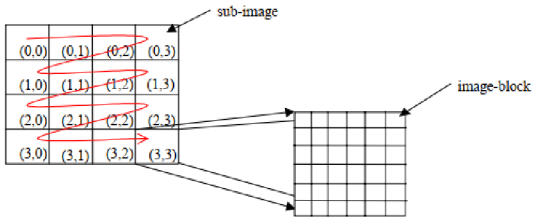


Figure 2.2: The Sub-Image and the Image-blocks

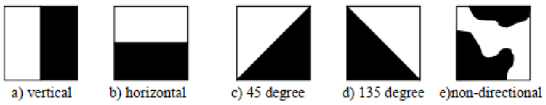


Figure 2.3: The five edge categories

2.2. Feature Detectors and Descriptors

2.2.1. Speeded Up Robust Feature (SURF)

SURF(Speeded-Up Robust Features) is a detector not influenced by the change of scale and rotation in the plane and descriptor [13]. As we'll see, these are not only scale-invariant features, but they also have the advantage of being very efficient at computing. The detector identifies the point of interest in an image, the descriptor describes the feature of the point of interest, and constructs the feature vector of the point of interest. The SURF framework described in this work is based on the Ph.D. thesis of H. Bay [ETH Zurich, 2009], and more specifically on the paper co-written by H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool [9].
Conceptually, the scale-invariant feature transformation (SIFT) proposed by David G. Lowe [37] is widely used for object recognition and detection. Nonetheless, despite the various work involved in using SIFT features in face recognition, these methods still fail to meet the speed requirements of online applications.
The SURF introduced by Bay et al.[9] on the otherhand, attempts to speed up the computation using the image integral that is, the sum of all pixels in the upper left rectangular window of the image with respect to the point of interest. It provides both a feature detector and a descriptor. Similar to SIFT, SURF first uses a detector to find the points of interest in the image, and then uses a descriptor to extract the feature vector at each point of interest. However, instead of the Gaussian difference (DoG) filter used in SIFT, SURF uses a Hessian matrix approximation that works with integral images to identify points of interest. This will significantly reduce the computation time.

2.2.2. Interest Point Detection

Our approach to identifying points of interest uses a very basic Hessian Matrix approximation. This is suitable for use with integral images that significantly reduce computational time, such as those popularized by Viola and Jones [66].

Figure 2.4: Locations of interest points in an image.
This was implemented by MATLAB using the inbuilt function `detectSURFFeatures`:Detect SURF features and return SURFPoints object

2.2.3. The Point of Interest in the Hessian Matrix

Unlike the scale normalized Laplacian of Gaussian (LoG) approximated by the Difference of Gaussian (DoG) used in SIFT as mentioned above, the determinant of Hessian matrix is a known measure for detecting scale-invariant interest points.

SURF uses the Hessian matrix for excellent performance in terms of computational time and accuracy. Instead of using different measurements for location and scale selection (Hessian Laplace detector), the SURF relies on the determinant of Hessian matrix for both. For a pixel, the Hessian matrix for that pixel is defined as this:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{2.1}$$

For any scale, we filtered the image by a Gaussian kernel, so given a point $X = (x, y)$, the Hessian matrix $\mathcal{H}(x, \sigma)$ in x at scale σ is defined as:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{2.2}$$

where $L_{xx}(x, \sigma)$ is the convolution of the Gaussian second order derivative with the image I in point x, and similarly for L_{xy} (x, σ) and L_{yy} (x, σ) [62]. Gaussians are optimal for scale-space analysis but in practice, they have to be discretized and cropped [9]. This leads to a loss in repeatability under image rotations around odd multiples of $\pi/4$. This weakness holds for Hessian-based detectors in general. Nevertheless, the detectors still perform well, and the slight decrease in performance does not outweigh the advantage of fast convolutions brought by the discretization and cropping.

In order to calculate the determinant of the Hessian matrix, first we need to apply convolution with Gaussian kernel, then second-order derivative. After Lowe's success with LoG approximations(SIFT), SURF pushes the approximation(both convolution and second-order derivative) even further with box filters. These approximate second-order Gaussian derivatives and can be evaluated at a very low computational cost using integral images and independently of size, and this is part of the reason why SURF is fast.

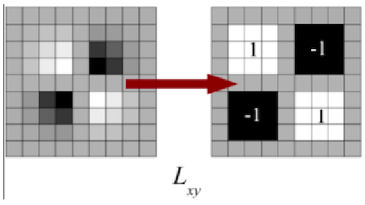


Figure 2.5: Gaussian partial derivative in xy-direction

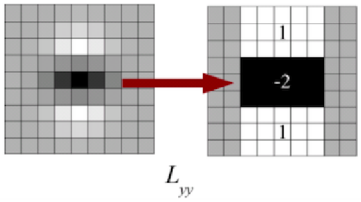


Figure 2.6: Gaussian partial derivative in y-direction

The 9×9 box filters in the above images are approximations for Gaussian second order derivatives with $\sigma = 1.2$. We denote these approximations by D_{xx}, D_{yy}, and D_{xy}. Now we can represent the determinant of the Hessian (approximated) as:

$$det\left(\mathcal{H}_{\text{approx}}\right) = D_{xx}D_{yy} - \left(wD_{xy}\right)^2 \tag{2.3}$$

$w = 0.9$ (Bay's suggestion)

2.2.4. Harris Corner Detection

Harris corner detectors [50] are the standard method for identifying points of interest on an image. Despite the emergence of many feature detectors over the last decade [38, 46, 65, 9], it continues to be a commonly used reference technique, which is typically used for camera calibration, image matching, tracking [55] or video stabilization [49].
Chris Harris and Mike Stephens came up with an algorithm that is super powerful in being able to detect corners, a specifically useful feature in images. To define a corner, let's talk in terms of image processing techniques. Corners can be considered as a junction of two edges, where an edge is a sudden change in the pixel brightness or intensity. Corners are important as they are considered feature points that are invariant to translation, rotation and illumination, but sadly not scale-invariant.
The Harris Corner Detector wasn't the first of its kind. In fact, Moravec's algorithm called the Moravec's corner detector was pretty well known before. The Moravec detector uses a small window to look for changes in average light intensity within that window. The "scan" process was performed by moving the window in different directions for a short distance. Three cases occur in his situation [39] .
1. The change in intensity is about 0. i.e., the intensity is almost constant. In this case, the window patch is said to be flat.
2. If there are edges in this window, two cases will occur. The edges are either perpendicular to or parallel to the movement of the window. A shift perpendicular to the edge leads to a significant shift.
3. Now the case we are interested in, is the the corner. All shifts, in this case, will result in a significant change. Equation (2.5) is the mathematical translation of "moving the window around."
One of the main drawbacks of this approach is that the windows move by discrete units of distance. In the original paper, only the following shifts were considered: $(x, y) = \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$ [16], [21].
This can be fixed by an analytical approach to solving this problem. In other words, instead, you only need to consider the differential shift from the original position (x, y). This was done in the Harris Corner Detector Algorithm [21].
Another advantage of the Harris Corner Detector over Moravec's detector is the nature of window/mask $w(x, y)$ used. Moravec's Detector used a binary mask as it comprised of all $1's$ within the window region and all $0's$ outside this region. This made the window much more prone to noise. Harris Corner detector used a smooth and circular window which was achieved with the help of Gaussian function:

$$w(u, v) = e^{-(u^2+v^2)/2(\sigma^2)} \tag{2.4}$$

To make things simple, this algorithm basically finds the difference in intensity of pixels for a displacement (u, v) for all directions. Then, while sliding a window across pixels

in an image, we find the sum of squared differences (SSD) of the pixel values before and after the shift to determine the intensity change. The function $E(u, v)$ is defined as the sum of all the sum of squared differences. I_x and I_y are image derivatives in the x and y directions, and can also be considered as the intensity value of the pixel.
Without loss of generality, we will assume a grayscale 2-dimensional image is used. Let this image be given by I. Consider taking an image patch $(x, y) \in W$ (window) and shifting it by $(\Delta x = u, \Delta y = v)$. The sum of squared differences (SSD) between these two patches, denoted by E, is given by:

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \underbrace{[I(x + u, y + v)}_{\text{shifted intensity}} - \underbrace{I(x, y)]}_{\text{intensity}}^2 \tag{2.5}$$

Where:
E is the difference between the original and the moved window.
u is the window's displacement in the x direction
v is the window's displacement in the y direction
$w(x, y)$ is the window at position (x, y). This acts like a mask. Ensuring that only the desired window is used.
I is the intensity of the image at a position (x, y)
$I(x + u, y + v)$ is the intensity of the moved window
In order to solve this equation, we will have to maximize this equation, specifically, the second term of equation (2.5) , by applying Taylor expansion series. For this computation, article [70] is used.

So, we maximize this term:

$$\sum_{x,y} [I(x + u, y + v) - I(x, y)]^2$$

Then, we expand this term using the Taylor series. It's just a way of rewriting this term in using its derivatives.

$$E(u, v) \approx \sum_{x,y} [I(x, y) + uI_x + vI_y - I(x, y)]^2 \tag{2.6}$$

Next, we expand the square. The $I(x, y)$ cancels out, so its just two terms we need to square. It looks like this:

$$E(u, v) \approx \sum_{x,y} u^2 I_x^2 + 2uvI_x I_y + v^2 I_y^2 \tag{2.7}$$

Now the equation can be written in a matrix form like this:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.8}$$

We obtain an equation as this and calculate the covariance matrix M :

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{2.9}$$

This part of the equation can be summarized by the following steps:
1. Apply the Sobel operator to find the image derivatives in the x and y directions.
2. Multiplying with the window means for each pixel in the grayscale (we usually turn images to grayscale for image processing) image, we slide a window across the image to find the Harris value R.
3. Set a threshold and for pixels that exceed such threshold, consider that the local extrema in that window, and the region contains a corner
4. We compute the feature descriptor here.
And the function $E(u, v)$ becomes:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.10}$$

However, to find out if a point in an image is a feature point, the value of $E(u, v)$ has to be large. Meaning, if there is a corner, $E(u, v)$ would be a big value. And to compute that we can solve for the eigenvectors of M to determine the directions for both the largest and smallest increases in SSD. Basically, they came out with such conditions such that:

$$R = \det(M) - k(\text{trace}(M))^2 \tag{2.11}$$

Where
$\det(M) = \lambda_1 \lambda_2$
$\text{trace}(M) = \lambda_1 + \lambda_2$
λ_1 and λ_2 are the eigenvalues of M and k is an empirically determined constant; $k \in [0.04, 0.06]$

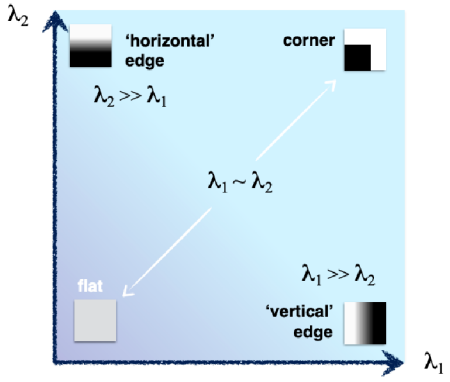


Figure 2.7: interpreting eigenvalues

By solving for R, we can know if there is a corner at the point in an image by the following conditions:

- When $|R|$ is small, which happens when λ_1 and λ_2 are small, the region is flat.
- When $R < 0$, which happens when $\lambda_1 >> \lambda_2$ or vice versa, the region is edge.
- When R is large, which happens when λ_1 and λ_2 are large and $\lambda_1 \sim \lambda_2$, the region is a corner.

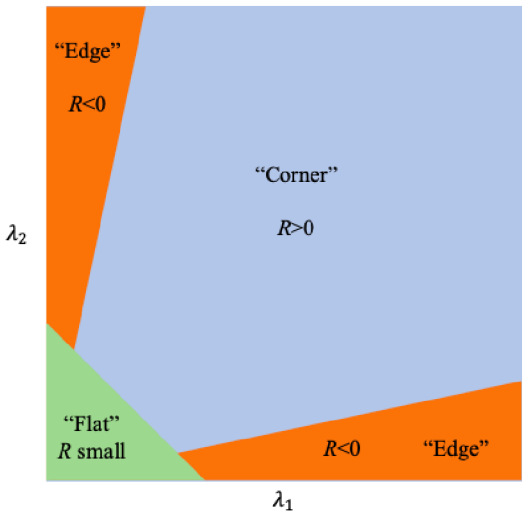


Figure 2.8: Harris measure

In summary, The Harris Corner Detector is just a mathematical way of determining which windows produce large variations when moved in any direction. With each window, a score R is associated. Based on this score, you can figure out which ones are corners and which ones are not.

Here are some result of the Matlab Implementation for Harris 2D corner detector.

Figure 2.9: Image Derivatives I_x and I_y direction

Figure 2.10: Corner Detection on a Test Image

3. The Viola Jones Object Detection

3.1. Object Detection with the Viola-Jones

Object detection is one of the computer technologies related to image processing and computer vision. This involves recognizing objects such as human faces, buildings, trees and cars. The main purpose of the face detection algorithm is to determine if a face is present in the image.

Figure 3.1: Face Detection
A simple MATLAB code is initiated to detect the entire face of an Image with the help of inbuilt MATLAB functions.

Most recently, smartphones have been able to unlock devices using the human face as a password. Like fingerprints, faces are unique and have millions of small features that differ from each other. It may not always be obvious to us humans, but machines synthesize and evaluate all the small data, which gives us more objective accuracy. Like other database-driven models, face detection/recognition isn't completely perfect. Although it has reached the stage of being commercially accepted in daily life. Facial recognition built into the device not only unlocks your smartphone, but can also be used in a variety of ways, from sending money to accessing personal information[44].
Computer vision technology is now deeply integrated into the lifestyles of many. An important aspect of it application is the ability to make a computer recognize objects in an image. Of these objects, the human face is the focus of attention because it has many useful uses in security and entertainment[27]. Therefore, this chapter will focus on a popular facial detection framework called the Viola Jones object detection framework.

The purpose of this chapter will be to explain and emulate the face detection algorithm presented by Viola Jones. The Viola-Jones algorithm was developed in 2001 by Paul Viola and Michael Jones. It's best described as an object-detection algorithm that was trained on faces. This means that, even though it was designed for and functions as a face-detection algorithm, its functionality can extend to any object if it were retrained. This algorithm should work in an unconstrained environment. That is, we would need

to detect all visible faces in every conceivable image. The ideal goal of a facial detection algorithm is to do the same work as a human inspecting the same image.
The basic problem to solve in this chapter is to explain the implementation of this algorithm that recognizes faces in images. At first glance, the task of facial detection may not seem so overwhelming, especially when it comes to how easy it is for humans to do it. However, this is in stark contrast to how difficult it really is to get a computer to successfully complete this task.
For ease of work, Viola Jones limits themselves to a full view frontal upright faces as seen in Figure 3.2. This means that the entire face must point towards the camera and not lean to one side in order to be recognized.
Figure 3.2 shows a typical input image of the face recognition algorithm. This image has relatively low contrast, has different types of textures, and ultimately has many faces. Since all faces are more or less upright from the front, it is expected that they will be recognized by the algorithms developed in this project.

Figure 3.2: Input Image[27]

To ensure optimal performance of the developed algorithms, most of the images that will be used for training, evaluation, and testing are on the Internet or from personal collections. In other words, these images are not generated under controlled conditions and are therefore considered a reliable representation of the unrestricted environment in which the face detector is expected to work. Now let us explore the Viola Jones algorithm in details.

Figure 3.3: Detected faces of the input image in Figure 3.2

3.2. The Concept of Viola-Jones Algorithm

Given an image (this algorithm works for grayscale images), the algorithm looks at many small sub-areas and tries to find the face by looking for specific features in each sub-area. Images can have many faces of different sizes, so it looks for different positions and scales. The algorithm will divide the image into sections as seen in Figure 3.4 (the green box). It will examine each section and try to see if it sees any features of a face in there. If it does not see all of the features it will move a little bit further to another section and tries to find the features again. The feature that the Viola-Jones Algorithm mainly looks for in a face are eyes, mouth, nose, cheeks, and so on.

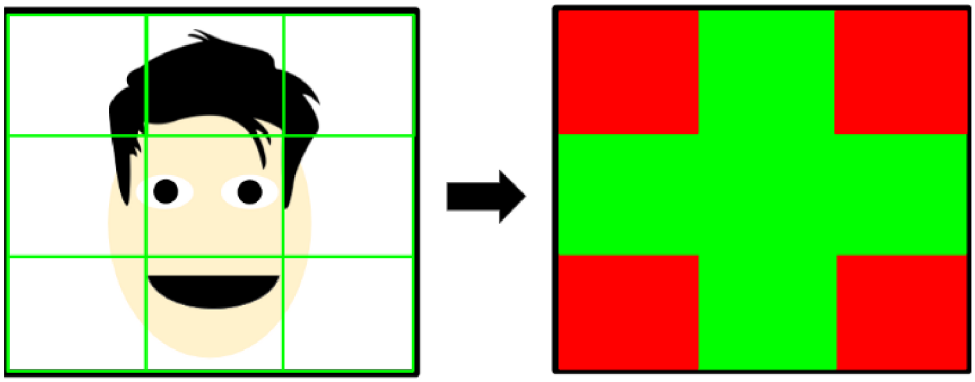


Figure 3.4: Face feautures[27]

The face can represent a map and the face features can represent the landmarks on the map. The algorithm is like a person trying to find the area that has all the landmarks in one location.
Viola and Jones uses the Haarlike features to detect faces in this algorithm.

3.3. Haar-like Features

In the $19th$ century, the Hungarian mathematician Alfred Haar introduced the concept of Haar Wavelet. Haar wavelets are a series of rescaled "square-shaped" functions that form the wavelet family or foundation. Viola and Jones adopted the idea of using Haar wavelets and developed so-called Haar-like features.

Haar-like features are features of digital images used in object recognition. All human faces share some of the universal characteristics of human faces, such as, the eye area is darker than the adjacent pixels and the nose area is brighter than the eye area.
An easy way to find out which area is bright or dark is to add up and compare the pixel values in both areas. The sum of the pixel values in the dark areas is less than the sum of the pixels in the light areas. If one side is brighter than the other, it may be the edge of the eyebrows. Alternatively, the central part may be glossier than the surrounding boxes. This can be interpreted as a nose. This can be achieved with hair-like features, and with

their help we can interpret different parts of the face [1].

The basic idea behind the Haar-like feature is using machine learning in which a cascade function trained by numerous positive and negative images and after upgrading of their classifier, it utilizes it to locate objects within different image (The positive and negative images are those images which include human face and without human face, respectively) [53].

Haar-like features are calculated as the difference between the sum of the pixels from the white region and the sum of the pixels from the black region. In this way, it is possible to detect contrast differences.
There are three basic types of Haar-like features as seen in Figure 3.5: Edge features, Line features, and Four-rectangle features. The white bars represent pixels that contain parts of an image that are closer to the light source, and would therefore be "whiter" on a grayscale image. The black bars are the opposite. These are pixels whose image features are farther away from the light source (like a background), or are obstructed by another object (such as the eyebrows casting slight shadows over the eyes below). These features, similar to before, would appear "blacker" in a grayscale image. This comparison between white and black pixels is the most important reason that we transform the image to grayscale.

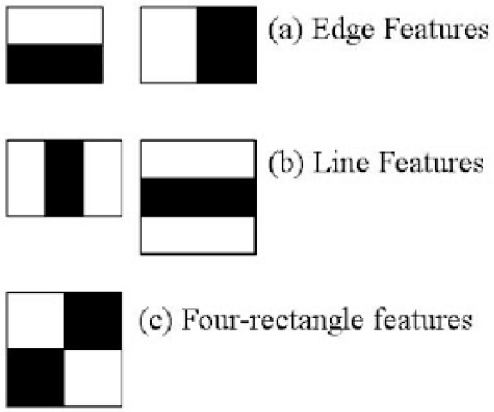


Figure 3.5: The types of Haar-like feautures[27]

Let's take a look at the Figure 3.6 below to understand how all of this is tied together.

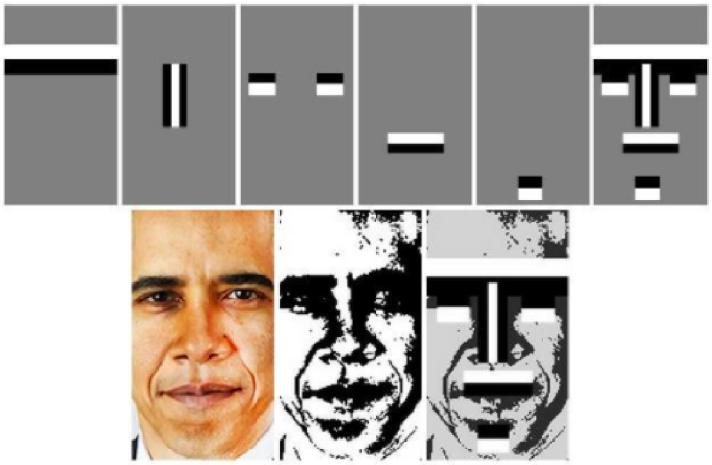


Figure 3.6: Applying the Haar-like feauture types on a face [29]

From Figure 3.6, the algorithm classifies the transition from the forehead to the brow, the eyes to the cheeks, the upper-lip to the mouth, and the jaw to the chin as Edge-features. The nose follows a black-white-black pattern as light reflects off of the top. The highlight here creates a line and, thus, it is classified as a Line-feature.
Notice that the Figure 3.6 above is not only converted to grayscale, but also has improved contrast. This is not a conversion done by the process. Instead, it would look for features on an image similar to this Figure 3.7

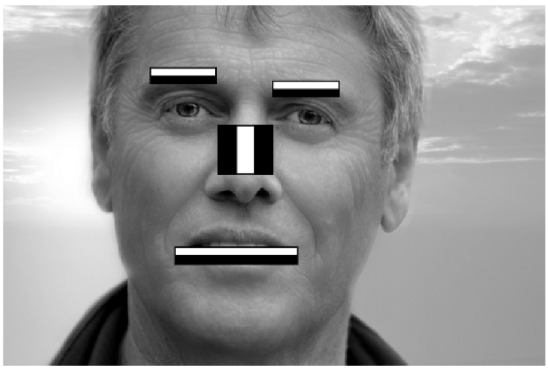


Figure 3.7: Applying the Haar-like feauture types on a face with no contrast tuned up [29]

If we decide to take into consideration the mask M from Figure 3.7, the Haar feature associated with the image I behind the mask is defined by:

$$\sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)_{\text{white}} - I(i,j)_{\text{black}} \tag{3.1}$$

Consider the image in Figure 3.7 with 24×24 pixel and the line feature Haar mask. Features are extracted for a shifted 24×24 pixel window on the image where you want to detect faces. For such windows, the Haar mask is scaled and moved to produce 162,336 features. Integral images were used to reduce the calculation time for Haar features but this vary depending on the size and type of the feature.

3.3.1. The Integral Image

The first step of the Viola-Jones face detection algorithm is to turn the input image into an integral image. This is done by making each pixel equal to the entire sum of all pixels above and to the left of the concerned pixel. This is demonstrated in Figure 3.8 . This image integral is utilize to speed up the keypoint detection process. This detection process is done speedily by summing up the area table called integral image, which is an algorithm for obtaining the sum values in a rectangular sum of grid.
Thus, the mathematical formular for this fast computation of box type convolution filters is given below as;

$$I(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \tag{3.2}$$

The entry of an integral image $I(\mathbf{x})$ at a location $\mathbf{x} = (x, y)$ represents the sum of all pixels in the input image I within a rectangular region formed by the origin and $\mathbf{x}$.
The goal here is to reduce the number of computations needed to obtain the summations of pixel intensities within a window.
To illustrate, in the below shows an image and its corresponding integral image. The integral image is padded to the left and the top to allow for the calculation [66]. The pixel value at $(2, 1)$ in the input image becomes the pixel value $(3, 2)$ in the integral image after adding the pixel value above it $(2 + 1)$ and to the left $(3 + 0)$.

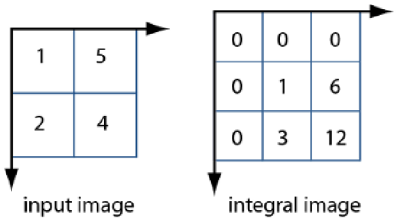


Figure 3.8: Integral Image

Each pixel in the integral image represents the corresponding input pixel value and the sum of all input pixels above and below the input pixel. Because `integralImage` (a Matlab function) zero-pads the resulting integral image, a pixel with (row,column) coordinate (m, n) in the original image maps to the pixel with coordinate $(m + 1, n + 1)$ in the integral image.
In this Figure 3.9, the current pixel of the input image is the dark green pixel at coordinates $(4, 5)$. All pixels in the input image at the top left of the input pixels are displayed in light green. The sum of the green pixel value is returned as the integral image pixels with the coordinates $(5, 6)$ colored in gray.

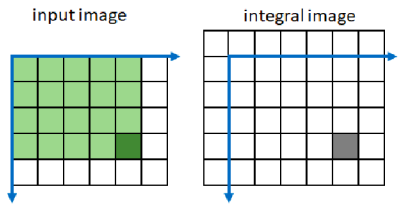


Figure 3.9: Integral Image

3.4. Face Detection Implementation With the Matlab

Face detection can be achieved with MATLAB code. You can detect faces, eyes, nose, and mouth using the classes and functions built into MATLAB. The Computer Vision System Toolbox's `ObjectVision.CascadeObjectDetector` system detects objects based on Viola Jones face detection algorithms.

3.4.1. Description of the Syntax

- `detector = ObjectVision.CascadeObjectDetector`: This syntax is used for the creation of a detector that detects objects using the Viola-Jones algorithm.
- `detector = ObjectVision.CascadeObjectDetector(mode1)`: This syntax is used to create a detector that is configured to detect an object defined in the input vector mode1.
- `detector = ObjectVision.CascadeObjectDetector(Name, Value)`: This syntax is used to set properties using one or more name-value pairs, and each property name is enclosed in quotation marks.
  Example: `Detector = vision.CascadeObjectDetector (`ClassificationModel`,` UpperBody`)`
- `bbox =detector(I)` Returns a bbox, an $M \times 4$ matrix that defines an "M" bounding box containing the detected objects.

Here are some result for detecting some part of the faces like the Nose ,Mouth, Eyes et al.

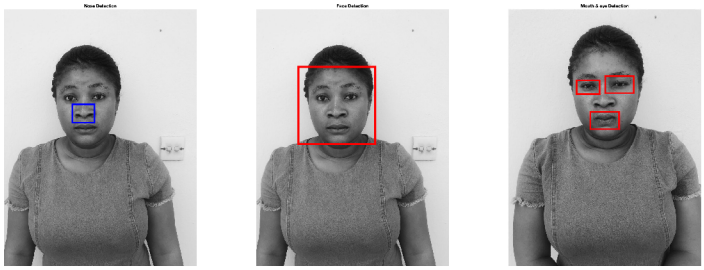


Figure 3.10: Detecting some interest point on the face

4. Eigen Vectors for Image Recognition

4.1. Principal Components Analysis (PCA)

This section is designed to give us an understanding of Principal Components Analysis (PCA). PCA is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. The PCA method aims to project data in the direction that has the greatest variation (indicated by the eigenvector) corresponding to the largest eigenvalues of the covariance matrix [6]. Before getting to a description of PCA, this section first introduces mathematical concepts that will be used in PCA. It covers covariance, eigenvectors and eigenvalues. This background knowledge is meant to make the PCA section very straightforward. We introduces some examples all the way through this section that are meant to illustrate the concepts being discussed. If further understanding is required, the mathematics textbook [7] is a good source of information regarding the mathematical background I have included in this section the statistical background which looks at distribution measurements, or, how the data is spread out. Another subsection 4.1.4 is on Matrix Algebra which looks at eigenvectors and eigenvalues, important properties of matrices that are fundamental to PCA.

4.1.1. Intuition behind PCA

Our data almost always comes with information, redundancy, and noise. The purpose could be to extract the most feasible information from the data even as decreasing the noise and ignoring the redundant information. By decreasing this dataset, we also are decreasing the accuracy. However, PCA works at the precept of trading little accuracy for simplicity. This is due to the fact that smaller datasets are less difficult to discover and visualize, as a consequence making data analysis less difficult and quicker for machine learning algorithms. This forms the purpose of PCA [35].

Imagine you want to capture a photo of a large group of friends in one frame. We find the best possible angle or rearrange the group so that everyone can be captured in a single frame instead of clicking on multiple images. PCA, similarly, it transforms the correlated features in the data into linearly independent features (Orthogonal) component so that all important information from the data is captured while reducing the dimension

This thesis work adopt the Principal Component Analysis (PCA) technique as a mechanism for extracting facial features for facial recognition in an efficient manner. There are several problems to consider every time we are deciding on a face recognition technique. The primary element is: accuracy, time limitations, processing speed and availability. With such in mind, PCA way's of facial recognition is chosen due to the fact that it's far without a doubt a handiest and simplest technique to implement, very fast computation time. PCA is an interest that extracts absolutely the maximum applicable and relevant information in a face and after which it attempts to construct a computational model

that best describes it [30].

4.1.2. The Background Mathematics of PCA

In this section we will attempt to give some mathematical elementary background skills that will be required to understand the process of Principal Components Analysis.
Not all of these techniques are used in PCA, but the ones that are not explicitly required do provide the grounding on which the most important techniques are based. Next, we introduce some basic concepts that are essential for the explanation of statistical techniques. We will take a look at the difference between covariance and variance. Variance measures the variation of a single random variable (like the height of a person in a population), whereas covariance is a measure of how much two random variables vary together (like the height of a person and the weight of a person in a population). The formula for variance is given by

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4.1}$$

where n is the number of samples (e.g. the number of people) and $\bar{x}$ is the mean of the random variable x (represented as a vector). The covariance $\sigma(x, y)$ of two random variables x and y is given by

$$\sigma(x, y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \tag{4.2}$$

with n samples. The variance σ_x^2 of a random variable x can be also expressed as the covariance with itself by $\sigma(x, x)$.
The formula for covariance is very similar to the formula for variance as in equation (4.1) and equation (4.2) The only difference between variance and covariance is using the values and means of two variables instead of one.
Note that when we work on sample data, we don't know the population mean, we know only the sample mean. That's why we should use the formula with $n - 1$. But when we have the all population of the subject, we can use with n.

4.1.3. The Covariance Matrix

Next thing that we should know is the covariance matrix. Because covariance can only be calculated between two variables, covariance matrices represent covariance values of each pair of variables in multivariate data. Also, the covariance between the same variables equals variance, so, the diagonal shows the variance of each variable. Suppose there are two variables as x and y in our data set. The covariance matrix should look like

$$\begin{bmatrix} \mathrm{var}(x) & \mathrm{cov}(x, y) \\ \mathrm{cov}(x, y) & \mathrm{var}(y) \end{bmatrix} \tag{4.3}$$

With the covariance we can calculate entries of the covariance matrix, which is a square matrix given by

$$C_{i,j} = \sigma(x_i, x_j) \tag{4.4}$$

where $C \in \mathbb{R}^{d \times d}$ and d describes the dimension or number of random variables of the data (e.g. the number of features like height, width, weight). Also the covariance matrix is symmetric since $\sigma(x_i, x_j) = \sigma(x_j, x_i)$. The diagonal entries of the covariance matrix are the variances and the other entries are the covariances. For this reason, the covariance matrix is sometimes called the variance-covariance matrix. A useful way to get all the possible covariance values between all the different dimensions is to calculate them all and put them in a matrix.

$$C^{n \times n} = (c_{i,j}, c_{i,j} = \text{cov}(\text{Dim}_i, \text{Dim}_j))$$

where $C^{n \times n}$ is a matrix with n rows and n columns, and Dim_x is the xth dimension. What the formula says is that if you have an n-dimensional data set, then the matrix has n rows and columns (so is square) and each entry in the matrix is the result of calculating the covariance between two separate dimensions. Eg. the entry on row 2 , column 3 , is the covariance value calculated between the 2nd dimension and the 3rd dimension. But we recall that covariance is always measured between 2 dimensions. If we have a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated. For example, from a 3 dimensional data set (dimensions x, y, z) you could calculate $\text{cov}(x, y), \text{cov}(x, z)$, and $\text{cov}(y, z)$. In fact, for an n-dimensional data set, you can calculate $\frac{n!}{(n-2)!2}$ different covariance values [59].

Here is an example of covariance matrix for an imaginary 3 dimensional data set, using the usual dimensions x, y and z. Then, the covariance matrix has 3 rows and 3 columns, and the values are this:

$$C = \begin{pmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{pmatrix}$$

We will lay emphasis on the main diagonal, we observe that the covariance value is between one of the dimensions and itself. These are the variances for that dimension. The other point is that since $\text{cov}(x, y) = \text{cov}(y, x)$, the matrix is symmetrical about the main diagonal.

4.1.4. Eigenvalues and Eigenvectors

We will consider here the computation of the eigenvalues and eigenvectors. We suppose that matrix **A** is symmetrical. In our work we would deal with covariance matrix that is symmetrical.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \dots & a_{pp} \end{pmatrix} \tag{4.5}$$

Note that we would refer to the matrix symmetric if the elements a_{ij} are equal to a_{ji} for each i and j.
For the present we will be primarily concerned with eigenvalues and eigenvectors of the variance-covariance matrix [28].

If we have an $n \times n$ matrix $\mathbf{A}$ we are going to have n eigenvalues, $\lambda_1, \lambda_2 \ldots \lambda_n$. They are obtained by solving the equation given in the expression below:

$$|\mathbf{A} - \lambda \mathbf{I}| = 0 \tag{4.6}$$

When we calculate the determinant of the resulting matrix, we end up with a polynomial of order n. Setting this polynomial equal to zero, and solving for λ we obtain the desired eigenvalues. In general, we will have n solutions and so there are n eigenvalues, not necessarily all unique.

The corresponding eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ are obtained by solving the expression below:

$$(\mathbf{A} - \lambda_j \mathbf{I})\, \mathbf{e}_j = \mathbf{0} \tag{4.7}$$

This will obtain the eigenvector e_j associated with eigenvalue λ_j but does not generally have a unique solution [28]. To illustrate these calculations, we consider the correlation matrix M.

Example 4.1.1. *Computation of eigen values and vectors for symmetrical matrix M 2×2*

$$\mathbf{M} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

By using equation (4.6), we have the equation,

$$|\mathbf{M} - \lambda \mathbf{I}| = 0 \tag{4.8}$$

Where I is the $m \times m$ Identity matrix.
This is a homogeneous system of equations, and from fundamental linear algebra, we know that a nontrivial solution exists if and only if

$$det|\mathbf{M} - \lambda \mathbf{I}| = 0 \tag{4.9}$$

This is known as the characteristic equation of $\mathbf{M}$, when evaluated, becomes a polynomial of degree 2 and λ here is the eigenvalue that we wish to solve for. Carrying out the calculation we end up with the matrix with $1 - \lambda$ on the diagonal and ρ on the off-diagonal. Then calculating this determinant we obtain

$$\begin{vmatrix} 1-\lambda & \rho \\ \rho & 1-\lambda \end{vmatrix} = (1-\lambda)^2 - \rho^2 = \lambda^2 - 2\lambda + 1 - \rho^2$$

Setting this expression equal to zero we end up with the following.

$$\lambda^2 - 2\lambda + 1 - \rho^2 = 0 \tag{4.10}$$

The characteristic polynomial is of degree 2. If $\mathbf{M}$ is $n \times n$, then there are n solutions or n roots of the characteristic polynomial [33]. Thus there are n eigenvalues of $\mathbf{M}$ satisfying the equation (4.7),

To solve for λ we use the general result that any solution to the second order polynomial below:

$$a\lambda^2 + b\lambda + c = 0$$

is given by the following expression:

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Here, $a = 1, b = -2$ (the term that precedes λ) and c is equal to $1 - \rho^2$ Substituting these terms in the equation above, we obtain that λ

$$\begin{aligned}\lambda &= \frac{2 \pm \sqrt{2^2 - 4\left(1 - \rho^2\right)}}{2} \\ &= 1 \pm \sqrt{1 - (1 - \rho^2)} \\ &= 1 \pm \rho\end{aligned}$$

Here we will take the following solutions:

$$\begin{aligned}\lambda_1 &= 1 + \rho \\ \lambda_2 &= 1 - \rho\end{aligned}$$

Next, to obtain the corresponding eigenvectors, we must solve a system of equations below:

$$(\mathbf{M} - \lambda\mathbf{I})\mathbf{e} = \mathbf{0}$$

This is translated for this specific problem in the expression below:

$$\left\{ \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right\} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

This simplifies as follows:

$$\begin{pmatrix} 1 - \lambda & \rho \\ \rho & 1 - \lambda \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Yielding a system of two equations with two unknowns:

$$\begin{aligned}(1 - \lambda)e_1 + \rho e_2 &= 0 \\ \rho e_1 + (1 - \lambda)e_2 &= 0\end{aligned}$$

We understand that this does not have a unique solution. Therefore, we will require the additional condition that the sum of the squared values of (e_1 and e_2) are equal to 1 (ie., $e_1^2 + e_2^2 = 1$). Consider the first equation:

$$(1 - \lambda)e_1 + \rho e_2 = 0$$

Solving this equation for e_2 and we obtain the following:

$$e_2 = -\frac{(1 - \lambda)}{\rho} e_1$$

Substituting this into $e_1^2 + e_2^2 = 1$ we get the following:

$$e_1^2 + \frac{(1 - \lambda)^2}{\rho^2} e_1^2 = 1$$

Recall that $\lambda = 1 \pm \rho$. In either case we end up finding that $(1-\lambda)^2 = \rho^2$, so that the expression above simplifies to:

$$2e_1^2 = 1$$

Or, in other words:

$$e_1 = \frac{1}{\sqrt{2}}$$

Using the expression for e_2 which we obtained above,

$$e_2 = -\frac{1-\lambda}{\rho}e_1$$

we get

$e_2 = \frac{1}{\sqrt{2}}$ for $\lambda = 1 + \rho$ and $e_2 = \frac{1}{\sqrt{2}}$ for $\lambda = 1 - \rho$ Therefore, the two eigenvectors are given by the two vectors as shown below:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \text{ for } \lambda_1 = 1 + \rho \text{ and } \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \text{ for } \lambda_2 = 1 - \rho$$

4.2. Eigenfaces for Face Recognition

Eigenfaces are fundamentally nothing more than basis vectors for real faces. This algorithm follows the concept that all the parts of face are not equally important or useful for face recognition. When we look at a face we look at the places of maximum variation so that we can recognise that person. For example from nose to eyes there is a huge variation in everyone's face. Eigenfaces algorithm works at the same principle [8].

In the language of information theory, we want to extract relevant information from a facial image, encode it as efficiently as possible, and compare the facial encoding to a database of similarly encoded models. Somehow an easy way to extract the information contained in the face image. It captures variations of a collection of facial images and uses this information to code and compare individual facial images, regardless of feature rating.

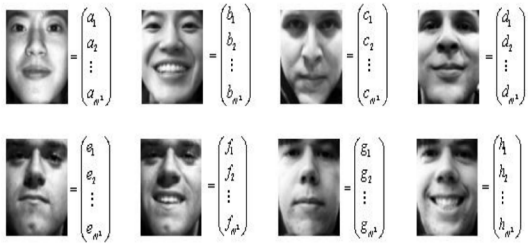
In the mathematical sense, we wish to find the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images. These eigenvectors can be thought of as a set of features which together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that we can display the eigenvector as a sort of ghostly face which we call an eigenface [64, 32].

Computational models of face recognition, in particular, are interesting because they can contribute not only to theoretical insights but also to practical applications. Computers that recognize faces could be applied to a wide variety of problems, including criminal identification, security systems, image and film processing, and human-computer interaction. For example, the ability to model a particular face and distinguish it from a large number of stored face models would make it possible to vastly improve criminal identification. Even the ability to merely detect faces, as opposed to recognizing them, can be important [32].

4.2.1. Face Image Representation

Let a face image $I(x, y)$ be a two-dimensional N by N array of intensity values, or a vector of dimension N^2. A typical image of size 256 by 256 describes a vector of dimension 65,536 or, equivalently, a point in 65,536-dimensional space. An ensemble of images, then, maps to a collection of points in this huge space. Each face is represented by $\Gamma_1, \Gamma_2, \Gamma_3, \ldots\ldots, \Gamma_M$ Feature vector of a face is stored in a N × N matrix. Now, this two dimensional vector is changed to one dimensional vector.

Let us assume size of rectangular face image with height = N and width = N and consists of h samples image, $\{\vec{a}, \vec{b}, \ldots, \vec{h}\}$ and p class $\{x_1, x_2, \ldots, x_p\}$.



Calculating the average of all face images

$$\vec{\Psi} = \frac{1}{M} \begin{pmatrix} a_1 + b_1 + \cdots + h_1 \\ a_2 + b_2 + \cdots + h_2 \\ \vdots + \vdots + \cdots + \vdots \\ a_{N^2} + b_{N^2} + \cdots + h_{N^n} \end{pmatrix} \tag{4.11}$$

we can also write as

$$\vec{\Psi} = \frac{\vec{a} + \vec{b} + \ldots + \vec{h}}{M} \tag{4.12}$$

Each face differs from the average by:

$$\vec{a}_m = \begin{pmatrix} a_1 - \Psi_1 \\ a_2 - \Psi_2 \\ \vdots \\ \vdots \\ a_{N^2} - \Psi_{N^2} \end{pmatrix}, \qquad \vec{b}_m = \begin{pmatrix} b_1 - \Psi_1 \\ b_2 - \Psi_2 \\ \vdots \\ \vdots \\ b_{N^2} - \Psi_{N^2} \end{pmatrix},$$

$$\vec{c}_m = \begin{pmatrix} c_1 - \Psi_1 \\ c_2 - \Psi_2 \\ \vdots \\ \vdots \\ c_{N^2} - \Psi_{N^2} \end{pmatrix}, \qquad \vec{d}_m = \begin{pmatrix} d_1 - \Psi_1 \\ d_2 - \Psi_2 \\ \vdots \\ \vdots \\ d_{N^2} - \Psi_{N^2} \end{pmatrix},$$

$$\vec{e}_m = \begin{pmatrix} e_1 - \Psi_1 \\ e_2 - \Psi_2 \\ \vdots \\ e_{N^2} - \Psi_{N^2} \end{pmatrix}, \qquad \vec{f}_m = \begin{pmatrix} f_1 - \Psi_1 \\ f_2 - \Psi_2 \\ \vdots \\ f_{N^2} - \Psi_{N^2} \end{pmatrix},$$

$$\vec{g}_m = \begin{pmatrix} g_1 - \Psi_1 \\ g_2 - \Psi_2 \\ \vdots \\ g_{N^2} - \Psi_{N^2} \end{pmatrix}, \qquad \vec{h}_m = \begin{pmatrix} h_1 - \Psi_1 \\ h_2 - \Psi_2 \\ \vdots \\ h_{N^2} - \Psi_{N^2} \end{pmatrix}$$

or presented as

$$\vec{A} = [\vec{a} - \vec{\Psi}, \vec{b} - \vec{\Psi}, \ldots, \vec{h} - \vec{\Psi}] \quad = \left[\vec{a}_m, \vec{b}_m, \ldots, \vec{h}_m\right]$$

For easy calculation and mathematical illustration for the steps involved in this method of recognition, we shall refer to the example below:

Example 4.2.1. *We rewrite the matrix as vector:*

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

STEP I : Image representation
Each face image is represented by the vector Γ_{i}

$$\Gamma_1 = \begin{bmatrix} 1 \\ -2 \\ 2 \\ -3 \end{bmatrix} \Gamma_2 = \begin{bmatrix} 2 \\ 3 \\ -2 \\ -3 \end{bmatrix} \Gamma_3 = \begin{bmatrix} 3 \\ 1 \\ -2 \\ 3 \end{bmatrix} \Gamma_4 = \begin{bmatrix} 2 \\ 3 \\ 1 \\ -1 \end{bmatrix} \ldots \Gamma_M$$

we choose $M = 4$ in our case

STEP II : Mean and Mean Cantered Images
Average Face Image is calculated by

$$\Psi = \frac{1}{M}\sum_{i=1}^{4}\Gamma_i \tag{4.13}$$

$$\begin{bmatrix} 1 \\ -2 \\ 2 \\ -3 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \\ -2 \\ -3 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \\ -2 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \\ 1 \\ -1 \end{bmatrix}$$

$$\Psi = \frac{(\Gamma_1 + \Gamma_2 + \Gamma_3 + \cdots + \Gamma_M)}{M} \tag{4.14}$$

Therefore

$$\Psi = \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \end{bmatrix} \tag{4.15}$$

Each face differs from the mean thus The deviation vector for each image Φ_i is given by:

$$\Phi_i = \Gamma_i - \Psi \qquad i = 1, 2, ..., M \tag{4.16}$$

This Face Image is called mean centered image.

$$\Phi_1 = \begin{bmatrix} -1 \\ -1 \\ 2 \\ 4 \end{bmatrix}, \Phi_2 = \begin{bmatrix} 0 \\ 2 \\ -2 \\ -2 \end{bmatrix}, \Phi_3 = \begin{bmatrix} 1 \\ 0 \\ -2 \\ 4 \end{bmatrix}, \Phi_4 = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

STEP III : Covariance Matrix
Consider a difference matrix A = $[\Phi_1, \Phi_2, \ldots\ldots \Phi_4]$ that just keeps the distinguishing facial image features and remove common features. Then the eigenface is calculated from finding the covariance matrix C of the training image vector by the following method:

$$C = \tfrac{1}{M}\textstyle\sum_{n=1}^{M}\Phi_n\Phi_n^T \quad = AA^T \tag{4.17}$$

where $A = [\Phi_1, \Phi_2, \ldots, \Phi_M$ of size $N^2 \times N^2]$

$$A = \begin{bmatrix} -1 & 0 & 1 & 0 \\ -1 & 2 & 0 & 2 \\ 2 & -2 & -2 & 1 \\ -4 & -2 & 4 & 0 \end{bmatrix} \quad A^T = \begin{bmatrix} -1 & -1 & 2 & -4 \\ 0 & 2 & -2 & -2 \\ 1 & 0 & -2 & 4 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

Size of covariance matrix will be $N \times N (4 \times 4$ in our case). Eigen vectors corresponding to this covariance matrix is needed to be calculated, but that will be a tedious task due to large dimension of matrix C, we consider matrix L of size $(M \times M)$ which gives the same effect with reduces dimension.
Therefore, for simplicity we calculate $L = A^{\mathrm{T}}A$ which would be a 2×2 matrix in this case.

$\mathrm{A^TA} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ size of this matrix is $M \times M$.

The eigenvectors of C (Matrix U) can be obtained by using the eigenvectors of L (Matrix V) as given by:

$$\mathrm{U_i = AV_i} \tag{4.18}$$

$$\text{eigenface } = [\mathrm{U_1, U_2, U_3, \ldots\ldots..U_M}]$$

Consider the eigenvectors V_i of A^TA such that,

$$A^TAV_i = \lambda_i V_i \tag{4.19}$$

The eigenvectors V_i of $L = A^TA$ are $\mathrm{V_1}$ and $\mathrm{V_2}$ which are 2×1. Now multiplying the above equation with A both sides we get,

$$AA^TAV_i = A\lambda_i V_i \tag{4.20}$$

$$AA^T\left(AV_i\right) = \lambda_i \left(AV_i\right) \tag{4.21}$$

Eigen vectors corresponding to AA^T can now be easily calculated now with reduced dimensionality where AV_i is the Eigen vector and λ_i is the Eigen value.
The M eigenvectors of L are finally used to form the M eigenvectors U_k of C that form our eigenface basis:

$$U_k = \sum_{k=1}^{M} \Phi_k V_{ik} \qquad k = 1, 2, \ldots, m \tag{4.22}$$

It turns out that only $M - k$ eigenfaces are actually needed to produce a complete basis for the face space, where k is the number of unique individuals in the set of known faces [8].

4.2.2. Recognition Algorithm

Facial recognition is essentially a computer system answering the query of "who's this person ?" mainly based on their facial features. Now, the question is, what does face recognition, eigenvalues and eigenvectors have in common?
To let your computer know that a said photo (Image) is just the face of a friend or a family member, you have to train it on lots of face images of that friend or family member using various deep learning algorithms. Now training a computer to understand the information(i.e. who is this person) carried in these matrices (Recall that we said images are just matrices) would be computationally expensive. So in order to make things easier and somewhat faster, we need to reduce the dimension of these images while still retaining as much information as possible.
This is where eigenvectors and eigenvalues come to the rescue, from this point on, things become easier. When the matrices (face images) dimensions have been reduced, the resulting eigenvector is known as an Eigen face because when displayed it produces a ghostly face.
Now, this N dimensional vectors(eigenvectors) span an N dimensional subspace called a "face space" which represents all the possible Eigen faces.
Mathematically, what happens is, we find the Euclidean distance between the new vector

to the face space. If the minimum distance passes a specified threshold to the face space we can say this is a face, the farther it is, the more confidence we have to say it is not a face.

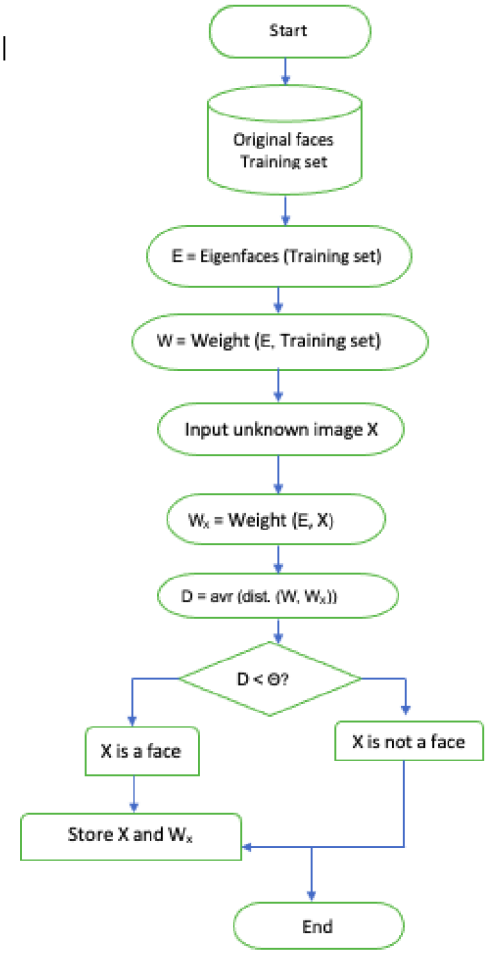


Figure 4.1: Proposed Face recognition frame work [57]

4.2.3. Using Eigenfaces to classify/detect a face image

Instead of using M eigenfaces, the highest $m' \leq M$ is chosen as the eigenspace [3]. Then the weight of each eigenvector W_k to represent the image in the eigenface space, as given by:

$$W_k = U_k^T(\Gamma - \Psi) \qquad k = 1, 2, \ldots, m' \tag{4.23}$$

Weight matrix

$$\Omega = [W_1, W_2 \ldots . W_{m'}]^T \tag{4.24}$$

Recognition is simply a problem of finding the closest database image, or mathematically finding the minimum Euclidean distance between a test point and a database point. The distance of Ω to each face is called Euclidean distance and defined by

$$\eta_k^2 = \|(\Omega - \Omega_k)\|^2 \qquad k = 1, 2, \ldots, m' \tag{4.25}$$

Where Ω_{k} is a vector describing the kth face class. A face is classified as belonging to class k when the minimum η_{k} is below some chosen threshold $\Theta_{threshold}$. Otherwise the face is classified as unknown [64].
When a new image comes into the system, there are three special cases for recognition.

- Image is a known face in the database
- Image is a face, but of an unknown person
- Image is not a face at all. A building, a tree, or a dog.

So we define some threshold values to characterized our images
$\Theta_{threshold}$ is half the largest distance between any two face images:

$$\Theta_{\mathrm{threshold}} = \frac{1}{2} \max_{j,k} \|\Omega_j - \Omega_k\| \tag{4.26}$$

We have to find the distance η between the original test image Γ and its reconstructed image from the Eigen face Γ_{f}.

$$\eta^2 = \|(\Gamma - \Gamma_f)\|^2 \tag{4.27}$$

where

$$\Gamma_f = U * \Omega + \Psi \tag{4.28}$$

If $\eta \geq \Theta_{threshold}$ then input image is not even a face image and not recognized.

If $\eta < \Theta_{threshold}$ and $\eta_k \geq \Theta$ for all k then input image is a face image but it is recognized as unknown face.

If $\eta < \Theta_{threshold}$ and $\eta_{\mathrm{k}} < \Theta$ for all k then input images are the individual face image associated with the class vector Ω_{k}.

5. RESULTS AND DISCUSSION

In this chapter, I will explain the simplest way of implementing a face recognition system with the MATLAB. To keep the face recognition system as simple as possible, I used eigenvector based recognition system.

5.1. Face Recognition With The MATLAB

According to the MathWorks, facial recognition is the process of identifying a person in an image or video by analyzing and comparing patterns [4].
Face recognition algorithms usually extract human facial features and thoroughly compare them with the facial features in the database to find the one with the best match.
It has become an important part of biometrics, security,surveillance systems, Image and video indexing system.

Face recognition systems include face detection and recognition methods in image analysis. The detection process is utilized to locate the face in the image. While recognition methods are used to classify a particular image using the structured features used in most computer vision applications.

Figure 5.1: Block Diagram of a Face Recognition System

Figure 5.1 shows the application flow of the face recognition system.
The first step in a face recognition system is to acquire the image via an external device such as a camera. The next step is face detection from the captured image. Then the system moves to the face recognition part, where the captured image is subjected to faces in the face database and proceed to the final step. The identity of the individual has been found.

5.1.1. Methodology

The methodology involved for Facial recognition using MATLAB has the part for face detection and facial recognition from the database [57, 61, 10]. The flowchart of this process is in Figure 5.2.
For a face recognition system, there should be a database of faces of people to be recognized. Then, the feature extraction step is done [56]. In this step, discriminative information about each face is stored in a compact feature vector. Then there are learning or modeling steps that use machine learning algorithms to fit the model of the facial appearance in the gallery. This is where the faces are identified in the gallery, and the output of this step is a classifier.
A classifier is a model that recognizes input images and compares them to input query images. The face detection algorithm is used to find the position of the face in the input

query image.
If a face is found, it is cropped, resized, normalized to the size and pose of the image used in the face database, and the feature extraction process is performed. If there is a match between the input query image and the database, the classifier indicates which person in the database the face belongs to.

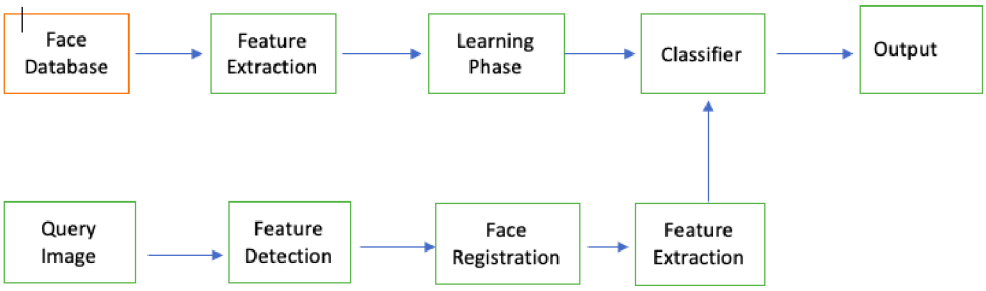


Figure 5.2: Facial Recognition Scheme using MATLAB

5.2. Database Description

I have prepared a dataset of 2 persons. Each of these persons has 10 images with different poses. That means in total there are 2 x 10 = 20 images. For every individual, there is separate folders. Explaining it in this manner for a larger dataset will be confusing. Hence, we prepared figure 5.3 for better understanding on how to create a dataset, most especially in a situation where we want to prepared a large dataset.

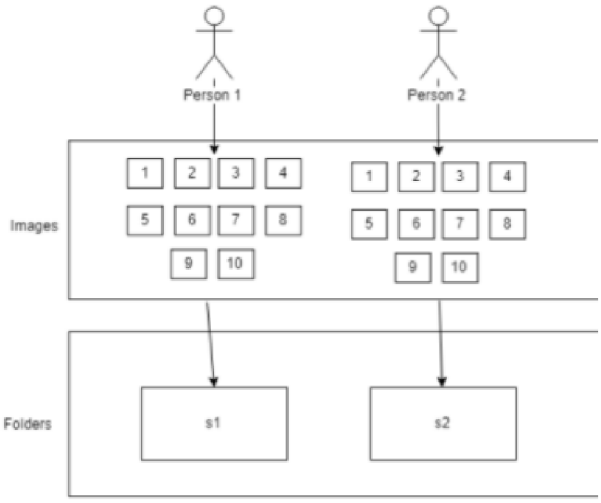


Figure 5.3: Preparing Dataset

The size of the images in the database was originally 960 ×1280 pixels. This creates a huge one-dimensional vector with a size of 1 ×1228800 pixels, which causes Matlab to have memory allocation problems when computing the covariance matrix. Therefore, the size of each image is reduced to 68x90 pixels. The top row in Figure 5.3 shows 2 persons labeled 1, 2. Each has 10 images. These images are grayscale. All of these images should be the same size and resolution. Finally, I saved everyone's photos in a different folder.

In Figure 5.3, s1, s2 represent folders. You can find the database used in this diploma thesis here.
The summary for prepaing a dataset is thus as follows;

- 10 images for every person
- separate folder for each person (s1, s2, for this case, but could be more in a general case)
- The Images saved in the folder above must be in grayscale
- The images must have the same resolution and size. I captured an image of 68 ×90 pixels.
- The name of the image must be numeric. Example: 1, 2, 3…10
- Also, the extension of the image must be the same as bmp, pgm, etc. Do not confuse different image extensions.

There are many other ways to prepare a dataset. This is just one of them.

5.3. Loading the Dataset for Face Recognition

After preparing the dataset, the next task is to load the dataset. Implement a function in Matlab that loads the dataset. You can also use this function to load other datasets. Let's call it `load_database.m`

```
function returned_value = load_database();
persistent loaded;
persistent numeric_Image;
if(isempty(loaded))
    all_Images = zeros(6120,2);
    for i=1:2
        cd(strcat('s',num2str(i)));
        for j=1:10
            image_Container = imread(strcat(num2str(j),'.pgm'));
            all_Images(:,(i-1)*10+j)=reshape(image_Container,
            size(image_Container,1)*size(image_Container,2),1);
        end
        display('loading Database');
        cd ..
    end
    numeric_Image = uint8(all_Images);
end
loaded = 1;
returned_value = numeric_Image;
```

The material used in this section 5.3 is thus referenced here [2, 35, 17].
The first line declared the name of the function `'load_database.m ()'` no input is required. Therefore, we used empty parenthesis. However, it returns numeric form of an

image. The returned image is stored in a variable called `'returned_value'`.
Then we got two variables, `'loaded'` and `'numeric_Images'`. These variables are persistent.
The persistent variables are local to the function in which they are declared; yet their values are kept in memory between calls to the function [17].
Then, at the `'if'` condition, we confirm if the `'loaded'` variable is empty. If it is empty only then we will load the dataset. The persistent variables permanently save the data and we need to load this dataset only once. This is why it is necessary we confirm if the variable is empty or not at the beginning.
We took 6120 zero's (which translate to the pixel value) for 2 times (number of faces s1,s2). Thereafter, the pixel values of the images will replace the zero's. Then we introduce a `'for loop'` using the `'strcat'` function, here we concatenate the names of our folders (which in our own case is s1,s2), the name we called the images in our folder (1,2,...10) and the extension of this images (we used `.pgm`).

5.3.1. MATLAB Implementation

This algorithm uses the eigenface system (based on Pricipal Component Analysis - PCA) to recognize faces.

```
 %% Loading the database into matrix A
loaded_Img=load_database();
 % We randomly pick an image from our database and use the rest of the
 % images for training. Training is done on 19 pictures. We later
 % use the randomly selected picture to test the algorithm.
rand_Index=round(20*rand(1,1)); % Randomly pick an index.
rand_Img=loaded_Img(:,rand_Index); %image we will use to test the alg.
w=loaded_Img(:,[1:rand_Index-1 rand_Index+1:end]); %rest of the 19 images.
img_Signature=2; % Number of signatures used for each image
```

I first loaded the dataset to recognize the face. Then I used a random function to generate a random index. I used a random index sequence to load an image that will be recognized later. The rest of the images are also loaded into a seperate variable.

```
%% Subtracting the mean from w
white_Image=uint8(ones(1,size(w,2)));
mean_val=uint8(mean(w,2)); % mean_val is the mean of all images
mean_Removed=w-uint8(single(mean_val)*single(white_Image));% The diff.
%% Calculating eignevectors of the correlation matrix
L=single(mean_Removed)'*single(mean_Removed);
[V,D]=eig(L);
V=single(mean_Removed)*V;
V=V(:,end:-1:end-(img_Signature-1));% Pick eigenvectors corresponding to
     %to the 10 largest eigenvalues.
%% Calculating the signature for each image
all_img_Signature=zeros(size(w,2),img_Signature);
for i=1:size(w,2);
    all_img_Signature(i,:)=single(mean_Removed(:,i))'*V; % Each row here
```

```
    % is the signature for one image
end
```

Then I calculated the mean of all the images (you can refer to equation (4.13) for the calculation) and subtracted the mean from each image (also refer to equation (4.16)). The eigenvectors were calculated on these images. After getting the eigenvalues, I created a matrix where each row contains the signature of the individual image. In other words, we have the eigenvalues of the images and the signature to identify them.

```
  %% Recognition
%Now, we run the algorithm and see if we can correctly recognize the face.
subplot(121);
imshow(reshape(rand_Img,90,68));
title(' Query Face','FontWeight','bold','Fontsize',16,'color','red');
subplot(122);
p=rand_Img-mean_val; % Subtract the mean
s=single(p)'*V;
z=[];
for i=1:size(w,2)
    z=[z,norm(all_img_Signature(i,:)-s,2)];
    if(rem(i,2)==0),imshow(reshape(w(:,i),90,68)),end;
    drawnow; %updates figures and processes any pending callbacks
end
[a,i]=min(z);
subplot(122);
imshow(reshape(w(:,i),90,68));
title('Recognized face','FontWeight','bold','Fontsize',16,'color','red');
```

In the remaining section, I subtracted the mean value from the face which we need to recognize. Finally, based on the difference between current image signatures with the signature we have mentioned above, we are able to predict the recognized face.
Here is the result our implementation

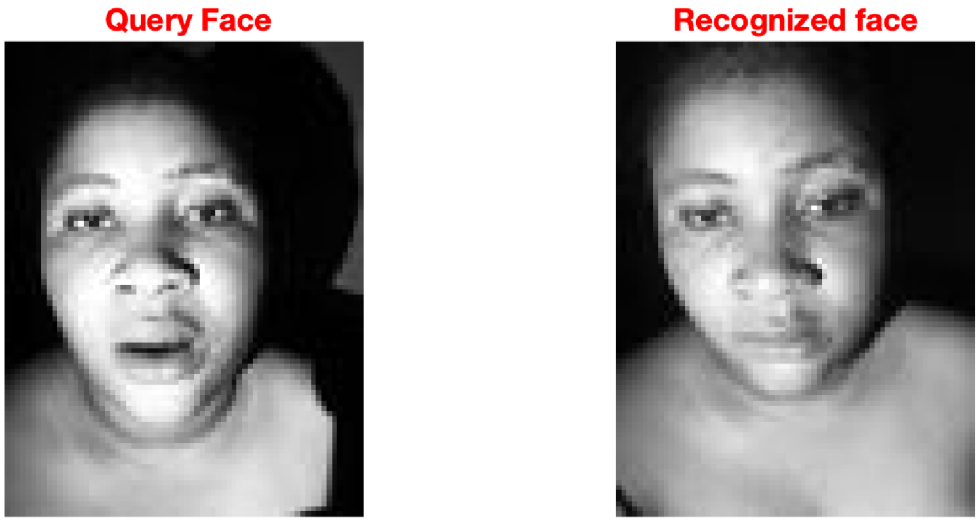


Figure 5.4: A queried face is recognized

6. CONCLUSION

The goal of this diploma thesis was to study the area of image descriptors that are important in image analysis. The main objectives were satisfied as thus described in the diploma thesis assignment.

First, in section 2.1 we wrote an overview of image descriptors with different types of image noises and the determination of the best descriptors that can be used with each noise type were proposed. We described that this image descriptors can be efficient in face detection even if some noise type are present in the image.
In section 2.2 we describe the mathematical formulation of different detectors (2.2.1, 2.2.4), in the calculation, we solve and determine the interest point in image which are invariant to translation, rotation and illumination. We further discussed the comparison with respect to computational speed and accuracy. We made the implementation with Matlab describing interest point in an image.

In chapter 3, we introduce the Viola Jones algorithm. The algorithm uses the the concept of Haar like features and as such, it computes the Haar like feature by the concept of integral image.
We examined and described Matlab's implementation by detecting faces in images. When applying, our main focus was to a) see if a particular image contained a face, and b) if so, to see where the face was. The algorithm successfully detected the face in a particular image. Use the same concept to detect other upper body parts such as eyes, nose, and lips.

Furthermore, in chapter 4, we choose the eigenvectors for image recognition and we described some theoretical and statistical background that is based on Principal component analysis (PCA).
In our study, we used eigenfaces to represent the feature vectors of human faces. Features are extracted from the original image and represent a unique identity used as an input to measure the similarity between classification and recognition. We used some acquired faces as our data set for the Matlab implementation. The faces were detected and recognized. More so,the Matlab code and syntax were explicitly explained for practical understanding.

The possible future work of this diploma thesis are:

- To examine the accuracy level and reliability of the face recognition on a bigger dataset.
- Try a dataset that contains images other than the face.

Bibliography

[1] Hussain Mujtaba (2020). Face detection using viola jones algorithm.

[2] Ali Behboodian (2022). Face recognition.

[3] Manal Abdullah, Majda Wazzan, and Sahar Bo-Saeed. Optimizing face recognition using pca. *arXiv preprint arXiv:1206.1515*, 2012.

[4] Aaron Don M Africa, Ara Jyllian A Abello, Zendrel G Gacuya, Isaiah Kyle A Naco, and Victor Antonio R Valdes. Face recognition using matlab. *International Journal of Advanced Trends in Computer Science and Engineering*, 8(4):1110, 2019.

[5] Eissa Alreshidi, Rabie A Ramadan, Md Sharif, Omer Faruk Ince, Ibrahim Furkan Ince, et al. A comparative study of image descriptors in recognizing human faces supported by distributed platforms. *Electronics*, 10(8):915, 2021.

[6] Mustamin Anggo and La Arapu. Face recognition using fisherface method. In *Journal of Physics: Conference Series*, volume 1028, page 012119. IOP Publishing, 2018.

[7] Howard Anton and Chris Rorres. *Elementary linear algebra: applications version.* John Wiley & Sons, 2013.

[8] Richard Baraniuk et al. Elec 301 projects fall 2004. 2010.

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[10] M Parisa Beham and S Mohamed Mansoor Roomi. A review of face recognition methods. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(04):1356005, 2013.

[11] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[12] Aman R Chadha, Pallavi P Vaidya, and M Mani Roja. Face recognition using discrete cosine transform for global and local features. In *2011 International Conference On Recent Advancements In Electrical, Electronics And Control Engineering*, pages 502–505. IEEE, 2011.

[13] Ahmed Chater and Abdelali Lasfar. Comparison of robust methods for extracting descriptors and facial matching. In *2019 international conference on wireless technologies, embedded and intelligent systems (WITS)*, pages 1–4. IEEE, 2019.

[14] Savvas A Chatzichristofis and Yiannis S Boutalis. Cedd: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *International conference on computer vision systems*, pages 312–322. Springer, 2008.

[15] Savvas A Chatzichristofis and Yiannis S Boutalis. Fcth: Fuzzy color and texture histogram-a low level feature for accurate image retrieval. In *2008 ninth international workshop on image analysis for multimedia interactive services*, pages 191–196. IEEE, 2008.

[16] Nilanjan Dey, Pradipti Nandi, Nilanjana Barman, Debolina Das, and Subhabrata Chakraborty. A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique. *arXiv preprint arXiv:1209.1558*, 2012.

[17] Nuruzzaman Faruqui. Loading the dataset for face recognition.

[18] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[19] Marcin Gabryel and Robertas Damaševičius. The image classification with different types of image features. In *International conference on artificial intelligence and soft computing*, pages 497–506. Springer, 2017.

[20] Dariu M Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

[21] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[22] Ren-Jay Jeffrey Huang. *Detection strategies for face recognition using learning and evolution*. George Mason University, 1998.

[23] Rabia Jafri and Hamid R Arabnia. A survey of face recognition techniques. *journal of information processing systems*, 5(2):41–68, 2009.

[24] Anil K Jain and Stan Z Li. *Handbook of face recognition*, volume 1. Springer, 2011.

[25] Sushma Jaiswal. Comparison between face recognition algorithm-eigenfaces, fisherfaces and elastic bunch graph matching. *Journal of global research in computer science*, 2(7):187–193, 2011.

[26] Hamid A Jalab. Image retrieval system based on color layout descriptor and gabor filters. In *2011 IEEE Conference on Open Systems*, pages 32–36. IEEE, 2011.

[27] Ole Helvig Jensen. Implementing the viola-jones face detection algorithm. 2008.

[28] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 6. Pearson London, UK:, 2014.

[29] Kushsairy Abdul Kadir, Mohd Khairi Kamaruddin, Haidawati Md Nasir, Sairul I. Safie, and Zulkifli Abdul Kadir Bakti. A comparative study between lbp and haar-like features for face detection using opencv. *2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)*, pages 335–339, 2014.

[30] Ramandeep Kaur and Er Himanshi. Face recognition using principal component analysis. In *2015 IEEE international advance computing conference (IACC)*, pages 585–589. IEEE, 2015.

[31] AMUD Khanday, Aamir Amin, Irfan Manzoor, and Rumaan Bashir. Face recognition techniques: a critical review. *STM Journals [Internet]*, 5(2):24–30, 2018.

[32] Hyun Hoi James Kim. Survey paper: Face detection and face recognition. In *Proceedings, International Conference on Computing and Video Processing*, pages 224–5. Citeseer, 2007.

[33] VP Kshirsagar, MR Baviskar, and ME Gaikwad. Face recognition using eigenfaces. In *2011 3rd International Conference on Computer Research and Development*, volume 2, pages 302–306. IEEE, 2011.

[34] Rekhil M Kumar and K Sreekumar. A survey on image feature descriptors. *Int J Comput Sci Inf Technol*, 5:7668–7673, 2014.

[35] Peter Kayere Linet Achieng. Face recognition using principal component analysis (pca).

[36] Christoph Loeffler, Adriaan Ligtenberg, and George S Moschytz. Practical fast 1-d dct algorithms with 11 multiplications. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 988–991. IEEE, 1989.

[37] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[38] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[39] Hans Peter Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. PhD thesis, Stanford University, 1980.

[40] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. Survey on lbp based texture descriptors for image classification. *Expert Systems with Applications*, 39(3):3634–3641, 2012.

[41] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International journal of computer vision*, 38(1):15–33, 2000.

[42] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. Efficient use of local edge histogram descriptor. In *Proceedings of the 2000 ACM workshops on Multimedia*, pages 51–54, 2000.

[43] P Jonathon Phillips, Patrick J Flynn, Todd Scruggs, Kevin W Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 947–954. IEEE, 2005.

[44] Gupta Rohan. Breaking down facial recognition: The viola-jones algorithm.

[45] Brandon Rohrer. Course 137 signal processing techniques.

[46] E Rosten. & drummond, t.(2006). In *Machine learning for high-speed corner detection In European conference on computer vision*, 2006.

[47] Rahimeh Rouhi, Mehran Amiri, and Behzad Irannejad. A review on feature extraction techniques in face recognition. *Signal & Image Processing*, 3(6):1, 2012.

[48] Toshiyuki Sakai, Takeo Kanade, Makoto Nagao, and Yu ichi Ohta. Picture processing system using a computer complex. *Computer Graphics and Image Processing*, 2(3):207–215, 1973.

[49] Javier Sánchez. Comparison of motion smoothing strategies for video stabilization using parametric models. *Image Processing On Line*, 7:309–346, 2017.

[50] Javier Sánchez, Nelson Monzón, and Agustín Salgado De La Nuez. An analysis and implementation of the harris corner detector. *Image Processing On Line*, 2018.

[51] William Robson Schwartz, Huimin Guo, Jonghyun Choi, and Larry S Davis. Face identification using large feature sets. *IEEE transactions on image processing*, 21(4):2245–2255, 2011.

[52] Stephen Se, David Lowe, and Jim Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 2051–2058. IEEE, 2001.

[53] Ali Sharifara, Mohd Shafry Mohd Rahim, and Yasaman Anisi. A general review of human face detection including a study of neural networks and haar feature-based cascade classifier in face detection. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pages 73–78. IEEE, 2014.

[54] Riti Sharma. *Object detection using dimensionality reduction on image descriptors*. Rochester Institute of Technology, 2014.

[55] Jianbo Shi et al. Tomasi. good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[56] Qiquan Shi, Yiu-Ming Cheung, Qibin Zhao, and Haiping Lu. Feature extraction for incomplete data via low-rank tensor decomposition with feature regularization. *IEEE transactions on neural networks and learning systems*, 30(6):1803–1817, 2018.

[57] Sanjay Kr Singh, Ashutosh Tripathi, Ankur Mahajan, and S Prabhakaran. Analysis of face recognition in matlab. *International Journal of Scientific & Engineering Research*, 3(2):48–54, 2012.

[58] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[59] Lindsay I Smith. A tutorial on principal components analysis. 2002.

[60] Sima Soltanpour, Boubakeur Boufama, and QM Jonathan Wu. A survey of local feature methods for 3d face recognition. *Pattern Recognition*, 72:391–406, 2017.

[61] Nisha Soni, Garima Mathur, and Mahendra Kumar. A matlab based high speed face recognition system using som neural networks. *International Journal of Engineering Research and Applications (IJERA)*, 3(4):785–790, 2013.

[62] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*, pages 1–10. IEEE, 2018.

[63] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[64] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society, 1991.

[65] Tinne Tuytelaars and Krystian Mikolajczyk. *Local invariant feature detectors: a survey*. Now Publishers Inc, 2008.

[66] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[67] Wei Wei, Edmond SL Ho, Kevin D McCay, Robertas Damaševičius, Rytis Maskeliūnas, and Anna Esposito. Assessing facial symmetry and attractiveness using augmented reality. *Pattern Analysis and Applications*, pages 1–17, 2021.

[68] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):780–785, 1997.

[69] Mingfu Xue, Can He, Jian Wang, and Weiqiang Liu. Backdoors hidden in facial features: a novel invisible backdoor attack against face recognition systems. *Peer-to-Peer Networking and Applications*, 14(3):1458–1474, 2021.

[70] Na Yao, Tiecheng Bai, Xia Jiang, and Haifang Lv. Improved harris corner detection for chinese characters. In *2013 Fourth World Congress on Software Engineering*, pages 321–325. IEEE, 2013.

[71] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

A. Harris 2D Corner Detector

```
% Harris 2D
im = imread('face1.jpeg');
im = rgb2gray(im);
im = double(im);
im = padarray(im,[1 1],0,'both')


k = 0.04;
%(between 0.04 and 0.06)
%Definition of filters used to compute the image derivatives by utilizing
%the Sobel operator
dx = [-1 0 1; -2 0 2; -1 0 1];
dy = dx';
%Computation of image derivatives Ix and Iy using convolution
Ix = conv2(im, dx, 'same');
Iy = conv2(im, dy, 'same');


% show results
figure(1)
subplot(2,1,1)
imshow(uint8(Ix))
subplot(2,1,2)
imshow(uint8(Iy))

[m,n] = size(im)
outImage = zeros(m,n);

for i = 3:m-2
    for j = 3:n-2
        meanDerX(i,j) = sum(sum(Ix(i-2:i+2, j-2:j+2)))/25.0;
        meanDerY(i,j) = sum(sum(Iy(i-2:i+2, j-2:j+2)))/25.0;
        IxCenter2(i,j) = sum(sum((Ix(i-2:i+2, j-2:j+2) - meanDerX(i,j)).^2)
            );
        IyCenter2(i,j) = sum(sum((Iy(i-2:i+2, j-2:j+2) - meanDerY(i,j)).^2)
            );
        IxyCenter(i,j) = sum(sum(    (Ix(i-2:i+2, j-2:j+2)-meanDerX(i,j) .*
            (Iy(i-2:i+2, j-2:j+2)-meanDerY(i,j))   )         ));
        C = [IxCenter2(i,j) IxyCenter(i,j); IxyCenter(i,j)  IyCenter2(i,j)
            ];
        d = IxCenter2(i,j) * IyCenter2(i,j) - IxyCenter(i,j).^2;
        traceC = trace(C);
        response = d - k*(traceC).^2;
        if response > 1000000000
            outImage(i,j) = 255;
        end
    end
end
figure(2)
imshow(uint8(outImage));
```

B. Face Detection With the Viola Jones

```
1  %%FACE DETECTION:
2  clear all
3  clc
4  %Detect objects using Viola-Jones Algorithm
5
6  %To detect Face
7  FDetect = vision.CascadeObjectDetector;
8
9  %Read the input image
10 I = imread('11.jpg');
11 I = rgb2gray(I);
12 %Returns Bounding Box values based on number of objects
13 BB = step(FDetect,I);
14
15 figure,
16 imshow(I); hold on
17 for i = 1:size(BB,1)
18     rectangle('Position',BB(i,:),'LineWidth',5,'LineStyle','-','EdgeColor',
       'r');
19 end
20 title('Face Detection');
21 hold off;
22
23 %To detect Nose
24 NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);
25
26
27
28 BB=step(NoseDetect,I);
29
30
31 figure,
32 imshow(I); hold on
33 for i = 1:size(BB,1)
34     rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor',
       'b');
35 end
36 title('Nose Detection');
37 hold off;
38
39 %To detect Mouth
40 MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',16);
41
42 BB=step(MouthDetect,I);
43
44
45 figure,
46 imshow(I); hold on
47 for i = 1:size(BB,1)
48  rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','r'
       );
```

```
49 end
50 title('Mouth Detection');
51 hold off;
52
53 %To detect Mouth
54 MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',16);
55
56 BB=step(MouthDetect,I);
57
58
59 figure,
60 imshow(I); hold on
61 for i = 1:size(BB,1)
62  rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','r'
       );
63 end
64 title('Mouth Detection');
65 hold off;
```

C. Face Recognition: load database

```
1  function returned_output = load_database();
2  persistent loaded;
3  persistent numeric_Image;
4  if(isempty(loaded))
5      all_Images = zeros(6120,2);
6      for i=1:2
7          cd(strcat('s',num2str(i)));
8          for j=1:10
9              image_Container = imread(strcat(num2str(j),'.pgm'));
10             all_Images(:,(i-1)*10+j)=reshape(image_Container,size(
                   image_Container,1)*size(image_Container,2),1);
11         end
12         display('loading Database');
13         cd ..
14     end
15     numeric_Image = uint8(all_Images);
16 end
17 loaded = 1;
18 returned_output = numeric_Image;
```

D. Face Recognition: Implementation

```
%% Loading the database into matrix w
loaded_Img=load_database();

 % We randomly pick an image from our database and use the rest of the
 % images for training. Training is done on 19 pictures. We later
 % use the randomly selected picture to test the algorithm.
rand_Index=round(20*rand(1,1));      % Randomly pick an index.
rand_Img=loaded_Img(:,rand_Index); %rand_Image contains the image we later
    on will use to test the algorithm
w=loaded_Img(:,[1:rand_Index-1 rand_Index+1:end]); % rest_of_the_images
    contains the rest of the 19 images.
img_Signature=2;   % Number of signatures used for each image

%% Subtracting the mean from w
white_Image=uint8(ones(1,size(w,2)));
mean_val=uint8(mean(w,2));  % mean_val is the mean of all images.
mean_Removed=w-uint8(single(mean_val)*single(white_Image)); % The
    difference of each Image from the mean.
%% Calculating eignevectors of the correlation matrix
L=single(mean_Removed)'*single(mean_Removed);
[V,D]=eig(L);
V=single(mean_Removed)*V;
V=V(:,end:-1:end-(img_Signature-1));  % Pick the eignevectors corresponding
     to the 10 largest eigenvalues.
%% Calculating the signature for each image
all_img_Signature=zeros(size(w,2),img_Signature);
for i=1:size(w,2);
    all_img_Signature(i,:)=single(mean_Removed(:,i))'*V; % Each row in cv
        is the signature for one image
end
%% Recognition
%  Now, we run the algorithm and see if we can correctly recognize the face
    .
subplot(121);
imshow(reshape(rand_Img,90,68));
title(' Query Face','FontWeight','bold','Fontsize',16,'color','red');
subplot(122);
p=rand_Img-mean_val; % Subtract the mean
s=single(p)'*V;
z=[];
for i=1:size(w,2)
    z=[z,norm(all_img_Signature(i,:)-s,2)];
    if(rem(i,2)==0),imshow(reshape(w(:,i),90,68)),end;
    drawnow;
end
[a,i]=min(z);
subplot(122);
imshow(reshape(w(:,i),90,68));
title('Recognized face','FontWeight','bold','Fontsize',16,'color','red');
```