Privacy-Preserving Face Matching Using Frequency Components

by
Shikha Tripathi

A thesis submitted to the Computer Science,
College of Natural Sciences and Mathematics
in partial fulfillment of the requirements for the degree of

Master of Science

in Computer Science

University of Houston
December 2019

DEDICATION/EPIGRAPH

“Success is not final; failure is not fatal: it is the courage to continue that counts.”

-Winston S. Churchill

ACKNOWLEDGMENTS

I would like to thank Dr. Shishir Shah for his support and motivation. He has been a great mentor and has helped me to a great extent in the completion of this work. It would not have been possible without his guidance to learn many new important concepts and expand my knowledge in the field of image processing. He has been an excellent mentor in every way possible. I am grateful to my committee members for their guidance towards writing this thesis. Last, but not least, I appreciate the help extended by the research students in the Quantitative Imaging Laboratory.

ABSTRACT

Privacy-Preserving Face Matching Using Frequency Components

Faces constitute a susceptible portion of an image, which in this age has become vulnerable with the increasing use of applications involving face detectors. Increasing concerns over the privacy of people have led to research involving anonymizing faces, privacy-preserving feature selection from facial images, active learning to obfuscate the visual appearance of a detected face, and many other approaches that make the face unviewable during its use in an application, specifically in the case of facial matching or face recognition. Along the lines of preserving privacy of detected faces during a face matching process, we aim to provide a possible solution to the privacy issues by utilizing the frequency components of an image instead of using the intensity components. This process involves computing a frequency transform of the detected facial image and using the frequency coefficients in the matching process. The matching is established using metrics such as the cosine distance, correlation value, and the Manhattan distance between two facial images. We use the three metrics to compute a combined score and establish a valid match by use of empirically derived threshold values. We analyze the role of specific DCT coefficients in making the combined decision and test the overall algorithm on the Aberdeen and Utrecht face image dataset.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Privacy in computer vision

Since the 70s, the field of computer vision has seen an increase in algorithms proposed to address a broad range of object recognition challenges. “Computer vision” is a field where researchers are developing methods to mimic the human visual process to recognize various objects, especially faces. The state-of-the-art object detection algorithms today have provided us with accuracy and precision in detecting objects and faces that seem to have surpassed human capabilities. The internet started growing even earlier than computer vision and has given us access to a variety of information at our fingertips. The vast majority of the information that people access is images. In the specific case of facial images, their use has become ubiquitous with the maturity of face detection and matching algorithms. The applications on a smartphone involving unlocking the phone, taking a selfie and more, all involve the use of locating an object in an image, the face being the most utilized part of it. A face serves as a unique identifier of a person and helps the human eye to recognize a person. With such advancements in technology and accessibility to the same, there has been increasing concern over the security and privacy of users’ data. The recent case of Facebook’s inability to protecting its users’ data shook the social media world in a very different way.

Following the events when technology has failed to protect the user’s data, researchers are trying their best to provide methods to secure identities. There is a growing need to develop methods for detecting objects in images and matching them while preserving the privacy of the object. In the case of faces, this would mean the ability to use the face image without being able to visualize or distinguish any facial features.

The standard approach in preserving privacy is often to encrypt the image acquired by the camera. Nonetheless, the image is later decrypted when specific computations need to be performed on it, as would be the case in the use of facial matching algorithms. However, such cryptographic techniques do not work very well in the scenario where the user is reluctant to disclose the image information even to the specific computational process or algorithm. Avidan and Butman [2] proposed the use of secure multi-party techniques to address this problem. They rely on the use of Oblivious Transfer to input the image to the facial image processing algorithm and to report the results. This method includes intense bit by bit computation, rendering the process very slow. Alternate methods to address this problem range from approaches that transform a facial image into feature attributes that are not unique to the specific identity visible in the image [3], project the intensity values observed in the facial image into an identity-preserving subspace [9, 11], to efficient privacy-preserving Viola-Jones type object detection via random base image representation [6].

The above solutions to privacy preservation involve the use of pixel intensity information in one way or another. Hence, if an attacker tries to steal the data or even a substantial part of it, this would result in a privacy breach resulting in the possibility that the attacker would be able to reconstruct the image and thereby being able to view the face in the image. The work in this thesis tries to approach the privacy problem by considering the use of images based on their frequency content rather than their pixel intensity information.

1.2 Motivation and goal

In the effort to preserve the privacy of users and mask the ability to visualize the image during the required algorithmic processing, cryptographic, non-cryptographic, and machine learning methods have helped. The similarity among these approaches lies in the fact that they all try to modify the base image pixels in one way or another to achieve their goals. In order to preserve privacy, this thesis presents one such method that involves the use of the frequency coefficients computed from an image in a matching process rather than the pixels themselves. The Discrete Cosine Transform (DCT) is one possible transform that provides a mapping of the pixel intensity values to frequencies present in the image. We try to leverage the importance of the different frequency coefficients and to evaluate their role in producing a decision over the question, "Are the two images of the same person?".

1.3 Privacy-preserving scenario

In this section, we describe a scenario where we can observe the need for preserving a face in image data. Consider an example where Alice has some surveillance data, and she would like to use a face matcher to match facial images from the surveillance data against a database of face images. Bob has a face-matching algorithm and wants Alice to use his algorithm. Alice would like to do so provided Bob is unable to visualize the facial images from the surveillance data and hence not compromise the identity of people whose faces are present in the surveillance data. In such situations, most often, the users tend to proceed to use the algorithm without much thought, which puts their privacy at risk. Figure 1.1 illustrates such a scenario.

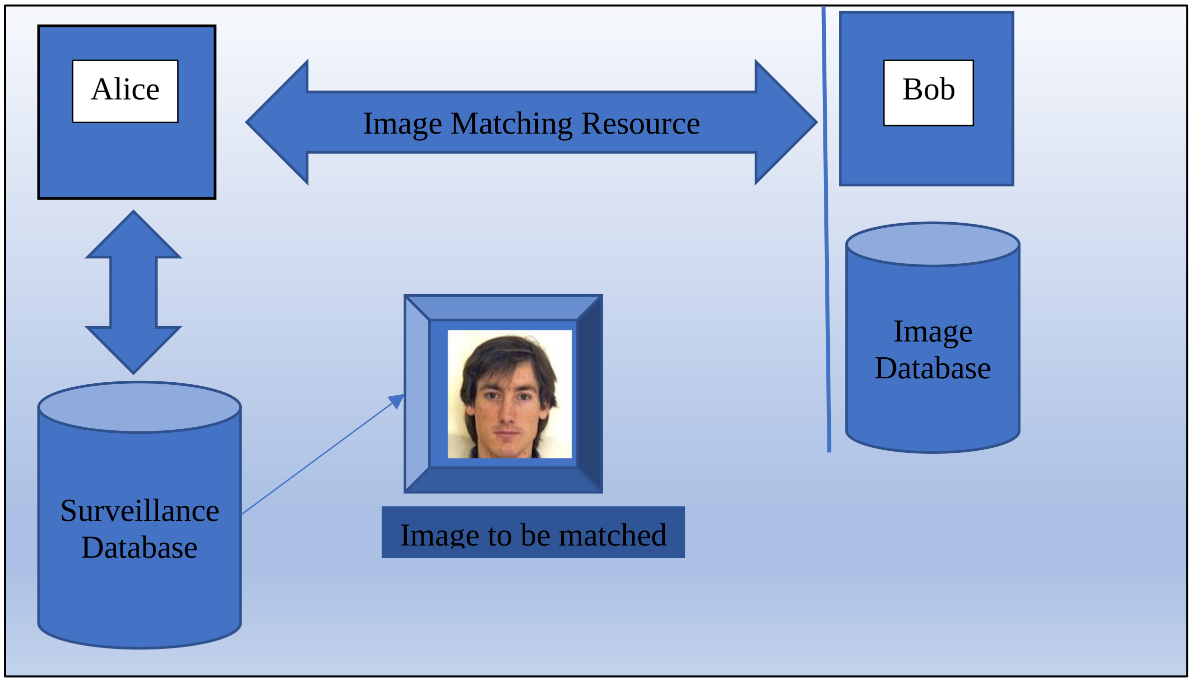


Figure 1.1: A transaction between Alice and Bob, involving image data.

1.4 Thesis outline

Chapter 2 presents related work on privacy preservation. In Chapter 3, we explain how the method we propose could potentially help strengthen the image data security. Chapter 4 explains the datasets and the significance of their use in trying to evaluate our approach. Chapter 5 presents the results obtained, and Chapter 6 outlines the conclusions and future work based on the experimental results and findings.

Chapter 2

Related work

In this section, we mention some of the work done in the domain of computer vision aimed at preserving the privacy of face images. We explain some of the methods under the category of cryptographic and non-cryptographic approaches.

2.1 Cryptographic approaches

Cryptographic approaches are the conventional methods used to secure a transaction by encrypting the data. The encryption changes the original form of the data and hence succeeds in providing security. Such a method was proposed by Avidan and Butman [2]. They constructed a secure classifier that included cryptography using oblivious transfer. The cryptographic tools used in their approach proved to be computationally intensive and thus reduced the speed of processing the image. To speed up the calculation involved, the authors suggested alternatives involving a smaller number of oblivious transfer operations, incomplete privacy preservation, and development of a one-way hash-function. Their experiments concluded that the proposed approach could be successful in preserving the identity in an image but would be impractical for typical applications.

2.2 Cryptography free approaches

The cryptographic methods, despite their effectiveness, have slow processing speeds. The speed limitation has sparked works in the field of privacy preservation by proposing cryptography-free methods.

2.2.1 Efficient privacy-preserving Viola-Jones type detector

Jin *et al.* [6] revisited the work of Avidan and Butman [2] to secure the transaction and to speed up the process by introducing the concept of a random base image. A random base image is created by separating the image in random base images, each carrying a specific weight, which is known only to the owner of the image. The random base images along with the random vector signifying the weights are utilized during the processing of the image and the results are aggregated to derive the result of the algorithmic operations. This was shown to be a relatively fast approach towards privacy preservation.

2.2.2 Learning to anonymize faces for privacy-preserving action detection

Machine learning has enabled learning using machines, thus creating more opportunities for researchers to evolve their algorithms in the best possible ways. Zhongzheng *et al.* [9] incorporated machine learning to enable learning of a video anonymizer. They used the generative adversarial network where the face anonymizer learned to generate modified pixels of an input image to decrease the accuracy in visualizing a face but at the same time trained a face detector acting as a discriminator that could increase the face identification accuracy.

2.2.3 Efficient methods for privacy-preserving face detection

Avidan and Butman developed further their work in [2] by replacing cryptographic techniques with machine learning algorithms to provide a secure pipeline during a transaction involving image data between two parties, each with sensitive information [3]. They proposed a method that is a variant of AdaBoost under which a subset of features from the input image are selected that are sufficient to classify a patch in an image. This allows partial information from an image to be utilized for algorithmic processing and hence could result in ensuring data privacy.

2.3 Conclusion

Current approaches suggest that techniques of privacy preservation are effective but can be painfully slow, or they could suffer from data leakage. The work proposed in this thesis tries to leverage the spatial independence property of the frequency components extracted by the discrete cosine transform. We intend to keep our data in a form that is difficult to comprehend and visually identify due to spatial independence. The coefficients obtained after the process of transformation and quantization are thus analyzed to choose only those who are essential in performing a secure match between image pairs.

Chapter 3

Face matching algorithm

The thesis proposes a simple approach for a face-matching algorithm to preserve the privacy of a face in an image. The process involves computing a frequency transform of the detected facial image and representing the images as its frequency coefficients. This would render the image to be visually non-recognizable. The match between any two images, represented by their frequency coefficients, is defined through a similarity function that uses metrics including the cosine distance, correlation value, and the Manhattan distance to report a matching value. The three resulting values are combined, and an empirically derived threshold value is used to report a match.

Let us assume we have an input image I_{input} and another image $I_{database}$ from a database of images that the input image is to be matched against. The discrete cosine transform, when applied to both the images, will result in the frequency transform I'_{input} and $I'_{database}$, which after quantization produces a quantized version of the transforms I'^{q}_{input} and $I'^{q}_{database}$. The quantized versions are used to extract the feature vectors for the images that are utilized in computing the matching score. Figure 3.1 below illustrates the major components of the face matching process. Upcoming sections in this chapter explain the functioning and significance of the elements in detail.

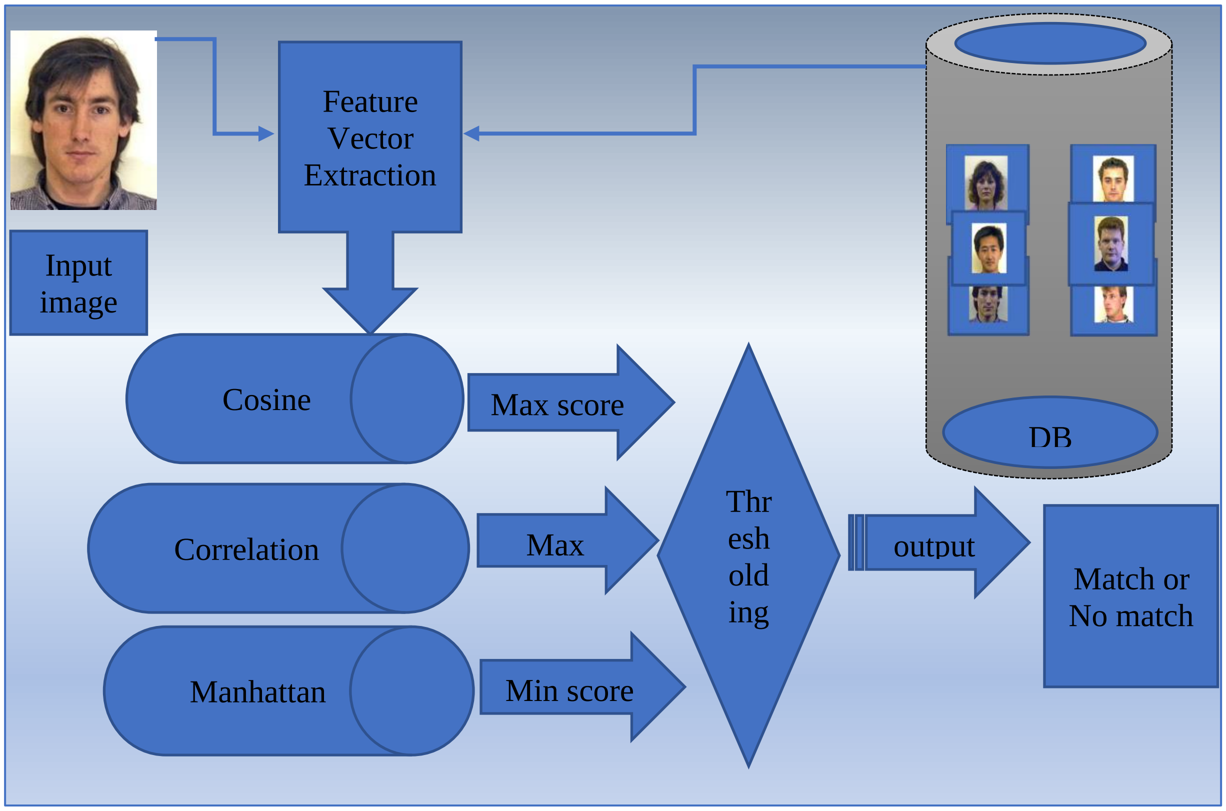


Figure 3.1: Pipeline for the privacy-preserving matching algorithm.

3.1 Frequency components of an image

A real-life image captured by a camera is stored digitally in a 2D matrix defined by its width and height, where the values in the matrix represent the intensity values of the corresponding image pixels. There are other ways to express an image, namely by its Fourier transform, discrete cosine transform, or its wavelet transform. An image can broadly be described in terms of its low-frequency and high-frequency elements in the Fourier or frequency space. Low-frequency elements tend to represent the image background or slowly varying visual components while the high-frequency components are often ephemeral forming structures like edges. Figure 3.2 below shows a 2D DCT of an image. The color scale on the right side shows the color red to represent coefficients with larger values while the color blue is used to indicate coefficients with smaller values. The DCT transformed image itself shows frequencies ranging from low to high with lower frequency starting from the top-left and increasing from left to right and top to bottom.

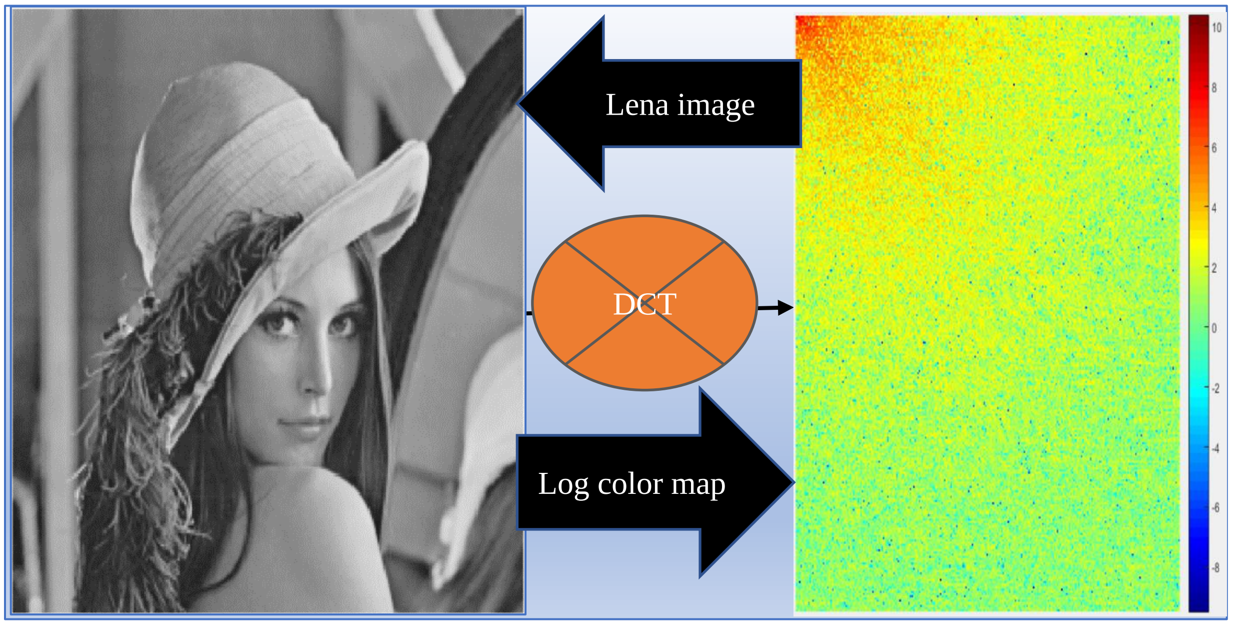


Figure 3.2: 2D image and its DCT transform.

3.1.1 DCT

The goal of applying a frequency transform on an image is to simplify an image into the sum of its sine and cosine components with different amplitudes [8]. A discrete Fourier transform is like a continuous Fourier transform known only at specific instants separated by a sample time. DFT is known for its computational efficiency but also has weak energy compaction as it calculates both the sine and cosine parts of an image. DCT [5], on the other hand, analyzes an image with few low-frequency components calculating only the cosine parts of the image resulting in the decorrelation of an input signal providing higher energy compaction. DCT [5] was first proposed by Ahmed *et al.* in 1974. DCT [5] takes an image block as an input and transforms it into a linear combination of its basis functions. Each of the basis functions is unique and represents the decorrelated data. The DCT equation for an image I of size $M \times N$ transformed into an image I' is

$$I'_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \qquad 3.1$$

where

$$0 \leq p \geq M - 1 \qquad 3.2$$

$$0 \leq q \geq N - 1 \qquad 3.3$$

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \frac{\sqrt{2}}{\sqrt{M}}, & 1 \leq p \leq M - 1 \end{cases} \qquad 3.4$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \frac{\sqrt{2}}{\sqrt{N}}, & 1 \leq q \leq N - 1 \end{cases} \qquad 3.5$$

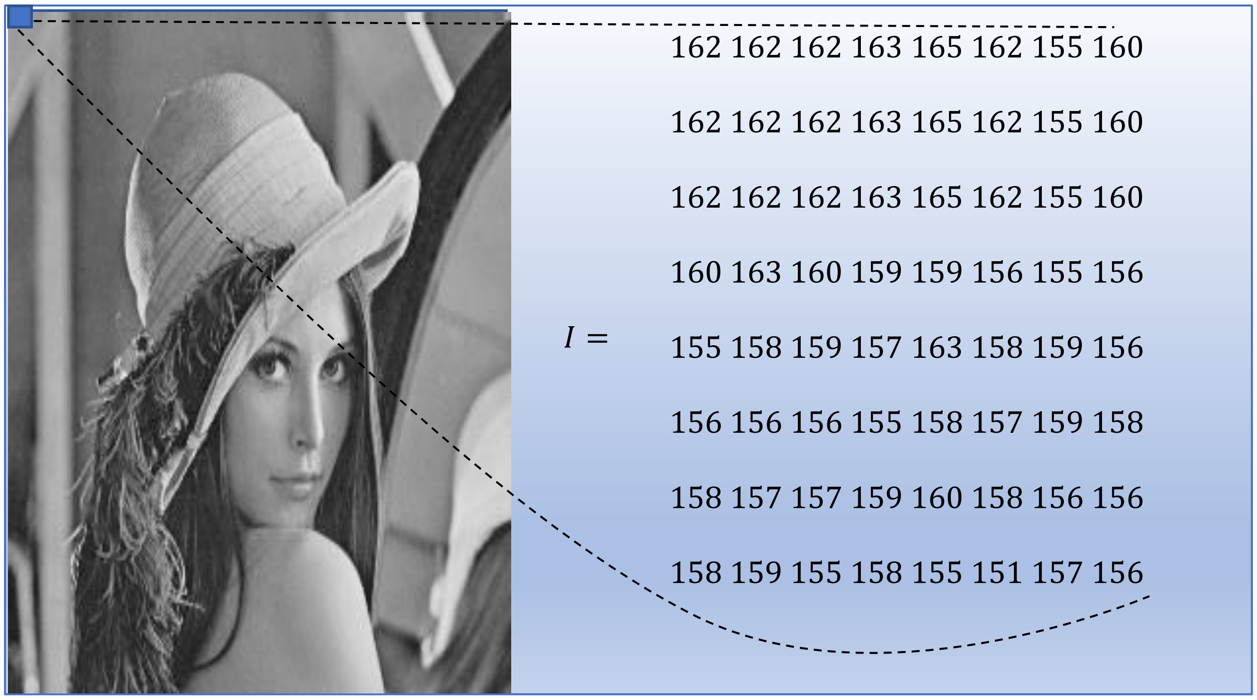
As a DCT calculates only the cosine parts of the frequency representation of the image, the result depends on the horizontal, diagonal, and vertical components of the frequencies. The value for the size of a DCT block is chosen to be 8 in this work, and thus an 8×8 block of DCT is represented by a predefined transformation matrix given in Figure 3.3.

$$T = \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix}_{8 \times 8}$$

Figure 3.3: 8×8 discrete cosine transformation matrix.

The matrix T represents the transform matrix that is multiplied to each 8×8 image block to transform the corresponding pixel values to their frequency representation. T is an orthogonal matrix, the columns of which are orthonormal to each other [10].

Let us consider the grayscale image in Figure 3.4 and its 8×8 block of pixels represented by a blue square in the upper left part of the image. The pixel intensity values are shown on the right side of the figure.



$$I = \begin{matrix} 162 & 162 & 162 & 163 & 165 & 162 & 155 & 160 \\ 162 & 162 & 162 & 163 & 165 & 162 & 155 & 160 \\ 162 & 162 & 162 & 163 & 165 & 162 & 155 & 160 \\ 160 & 163 & 160 & 159 & 159 & 156 & 155 & 156 \\ 155 & 158 & 159 & 157 & 163 & 158 & 159 & 156 \\ 156 & 156 & 156 & 155 & 158 & 157 & 159 & 158 \\ 158 & 157 & 157 & 159 & 160 & 158 & 156 & 156 \\ 158 & 159 & 155 & 158 & 155 & 151 & 157 & 156 \end{matrix}$$

Figure 3.4: Lena image 8x8 block denoted on the image by the blue square and its corresponding pixel intensity values.

A DCT works on the pixel values ranging from -128 to 128. We subtract 128 from each entry in the image block shown in Figure 3.4. The resulting matrix with the correct range for the pixels is shown in Figure 3.5.

$$I - 128 = \begin{matrix} 34 & 34 & 34 & 35 & 37 & 34 & 27 & 32 \\ 34 & 34 & 34 & 35 & 37 & 34 & 27 & 32 \\ 34 & 34 & 34 & 35 & 37 & 34 & 27 & 32 \\ 32 & 35 & 32 & 31 & 31 & 28 & 27 & 28 \\ 27 & 30 & 31 & 29 & 35 & 30 & 31 & 28 \\ 28 & 28 & 28 & 27 & 30 & 29 & 31 & 30 \\ 30 & 29 & 29 & 31 & 32 & 30 & 28 & 28 \\ 30 & 31 & 27 & 30 & 27 & 23 & 29 & 28 \end{matrix}$$

Figure 3.5: 8×8 image block leveling and the resulting pixel values.

After multiplying the image values with the values in the DCT transform matrix, we obtain the matrix in Figure 3.6.

$$I' = \begin{matrix} 247.3750 & 6.2619 & -5.8351 & 0.7143 & 4.1250 & -5.4437 & -0.6949 & 0.9123 \\ 15.3306 & 3.8938 & -4.7945 & 0.8608 & 2.3670 & -4.7028 & 4.8982 & -1.6560 \\ 1.5491 & 1.5286 & 0.6919 & -0.4326 & 3.1823 & 0.9709 & 0.4508 & -2.3150 \\ -1.3976 & -5.3607 & -1.1326 & 1.8457 & -1.5784 & -0.0535 & 1.0039 & 2.0622 \\ -3.1250 & 2.2919 & 1.2853 & -1.8988 & -1.3750 & 0.4214 & -3.1031 & 0.8073 \\ 0.6060 & 2.6198 & -0.0861 & 0.3733 & 0.4852 & -0.1866 & 1.2182 & -1.5752 \\ -0.5064 & -1.4340 & 2.2008 & -0.5673 & 0.1701 & 0.4124 & -0.1919 & -0.0379 \\ 3.2734 & -1.7293 & -4.0723 & 1.4173 & 0.6948 & -0.8212 & 1.0147 & 0.9472 \end{matrix}$$

Figure 3.6: DCT coefficients of an 8×8 image block.

I' represents the transformed matrix in Figure 3.6, where each value corresponds to a coefficient number. Figure 3.7 illustrates the ordering of the coefficients from highest to lowest.

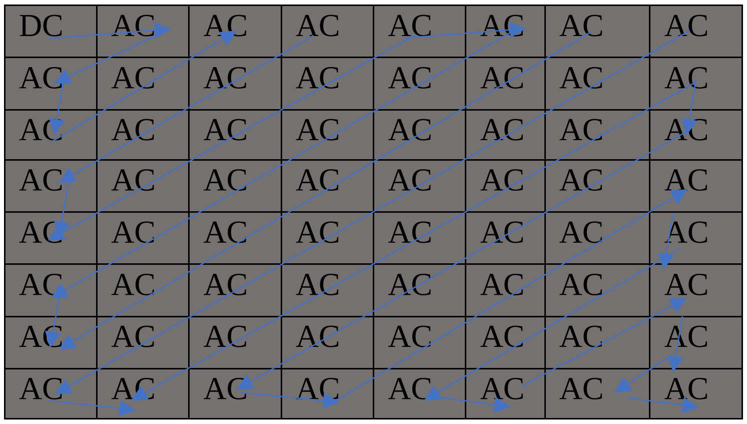


Figure 3.7: AC and DC coefficients depicted in zig-zag order.

The first component is called the DC component, the essence of the 8×8 transformed block. The rest are called the AC components, which are the combination of low and high-frequency elements that have a decreasing order of values when traversed in a zig-zag manner. High-frequency elements are the changes in the signal, which oscillate at a higher rate. The high order coefficients represent the small spanned rapid changes, and the low order coefficients represent the gradual changes in the image.

3.2 Quantization

The frequency transform process is often followed by a quantization step to reduce the number of bits required to store the transformed values. This is useful in image compression but can also be used to make a decision regarding some of the transformed information to retained or to be discarded. A quantization matrix [10] is used to perform this process. The standard quantization table [10] to perform compression representing a level of 50 as per the JPEG standard is given in Figure 3.8.

$$
Q_{50} = \begin{matrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{matrix}
$$

Figure 3.8: Quantization matrix at compression level of 50 as per the JPEG standard.

A different quantization matrix can be obtained by the following equation [10] given below.

$$
Q_n = f(x) = \begin{cases} \frac{100 - n}{50} Q_{50}, & n < 50 \\ \frac{50}{n} Q_{50}, & n > 50 \end{cases} \qquad 3.6
$$

where

$n > 50$ - represents less compression and better quality.

$n < 50$ - represents more compression and less quality.

Quantization is performed by dividing each element of the matrix I' by the corresponding element in the quantization matrix and finally rounding the values to the nearest integers. The matrix given below in Figure 3.9 represents an example of a resultant quantized matrix.

$$C_{i,j} = round\left(\frac{I'_{i,j}}{Q_{i,j}}\right) \qquad 3.7$$

$$C = \begin{matrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Figure 3.9: Final quantized matrix applied to the image block rendering the higher-order coefficients zero.

As we can observe, the values in the upper left side of the matrix represent the low frequencies to which the human eye responds well. The quantization saves these values, whereas the values corresponding to the higher frequencies are rounded to zero as they carry redundant information. The choice of the quantization matrix can produce a varying number of zeroes in the high-frequency positions. The quantization matrix used in this thesis is Q_{50}.

3.3 Feature vector

Mathematically a vector is viewed and understood as an object which has magnitude and direction in some vector space of a specific dimensionality. In computer vision and machine learning, we define another term complementing the vectors called feature vectors. A feature represents an attribute of an object. A feature vector is defined as a set of values for a specific attribute, organized in a row or column vector.

A feature vector is a convenient method of representing the many data points in space. The matrix of the quantized values is used to extract the feature vectors. In a DCT of an image, we have the coefficients ordered from high to low in a zigzag traversal, as depicted in Figure 3.7. The very first element in Figure 3.7 represents the frequency element, which contains the highest value in the entire block. This reflects the strength of the energy present in a DC component. We use the following naming convention for every coefficient in that matrix. Let us assume we have the coefficients in the zigzag order in a list called $C = [Coeff-1, Coeff-2, \dots, Coeff-64]$. Given an image of size 256×256, we can divide it into non-overlapping blocks, each of size 8×8. The division results in a total of 1024 blocks for that image, each of which is transformed into their frequency domain using DCT. The quantization matrix reduces the coefficients of low value to zero leaving only a few non-zero coefficients in every block. The quantized image blocks are each processed one by one to extract the corresponding coefficient values and place them under their respective coefficient numbers.

Every coefficient vector is a column vector which can be given by

$$Coeff_i = [Coeff - i_{b_1}, Coeff - i_{b_2}, Coeff - i_{b_3}, \dots, Coef - i_{b_{1024}}] \qquad 3.8$$

where i ranges from 1 to 64.

Thus, the feature extraction results in a total of 64 column vectors, each of size 1024. Hence the size of the coefficient feature matrix is 1024×64.

$$Coeff\_FV = \begin{bmatrix} Coeff-1_{b_1} & \cdots & Coeff-64_{b_1} \\ \vdots & \ddots & \vdots \\ Coeff-1_{b_{1024}} & \cdots & Coeff-64_{b_{1024}} \end{bmatrix}_{1024\times 64} \qquad 3.9$$

As every coefficient in a quantized block is a unique feature of the 8×8 image block, while calculating similarity scores between the feature vectors, we collect the values from the quantized blocks under similar features. The reasoning behind the approach is that if two images are the same, then their unique frequency components must remain the same. This assumption is the foundation of the application of the similarity functions, namely cosine distance, correlation value, and Manhattan distance, which are expected to produce a high score for cosine distance and correlation value calculations and a low score for Manhattan distance calculation.

Every coefficient feature vector is processed separately, and the similarity scores of each of the feature vectors are independent in deciding a face match. The reason to consider the feature vectors independently is to understand the similarity score pattern they carry amongst them. This kind of analysis is of importance in determining the number of coefficients we must include in the decision process while preserving the identity of the face image. Figure 3.10 below shows the process of feature extraction.

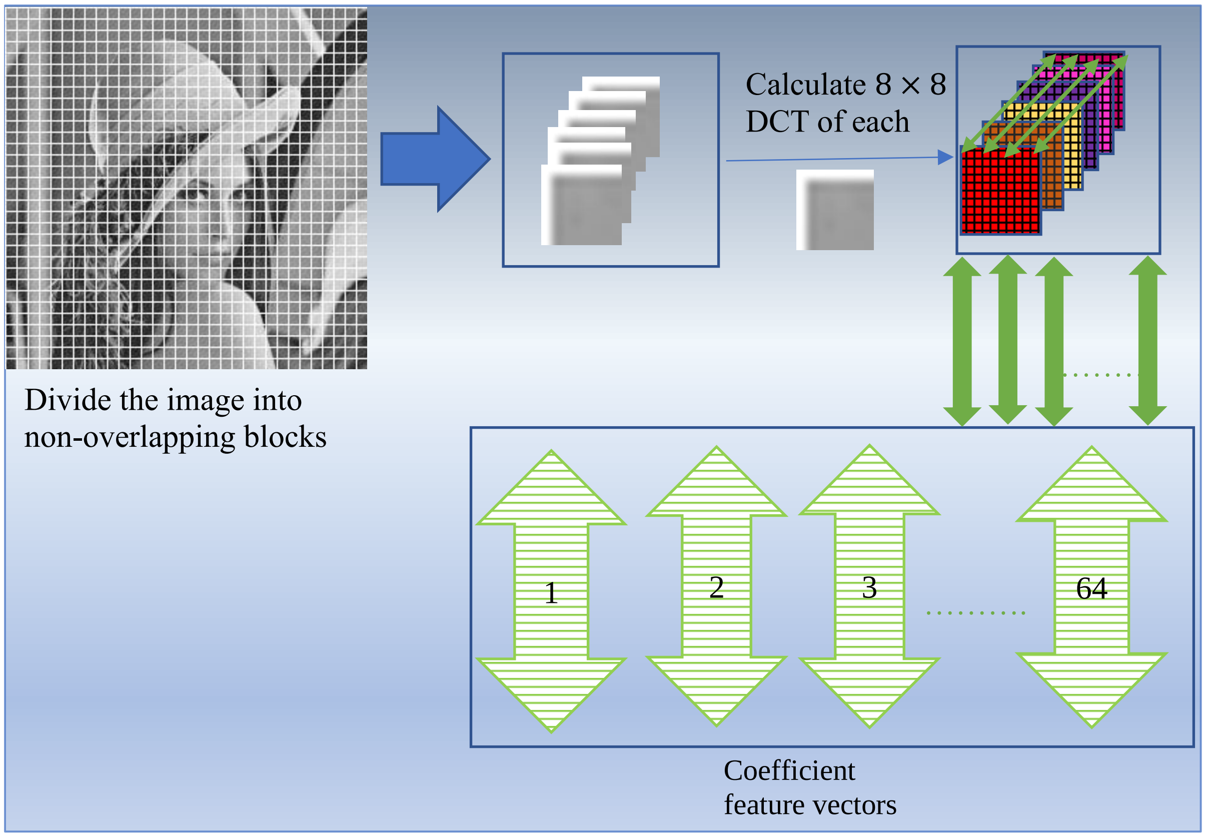


Figure 3.10: Coefficient feature extraction process.

3.4 Similarity functions

This section explains the working and significance of the similarity functions in detail.

3.4.1 Cosine distance

The cosine similarity calculates the angle of cosine between two vectors in the same vector space. The orientation of a vector is important in a vector space as two vectors despite having the same magnitude may end up being completely different from each other. In situations when we have large sets of data, we often need a measure that can compare two objects and best describe their similarity in terms of their orientations in the vector space. A cosine similarity measure is the dot product of the vector quantities given below.

$$similarity = \cos\theta = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\ \sqrt{\sum_{i=1}^{n} B_i^2}} \qquad 3.10$$

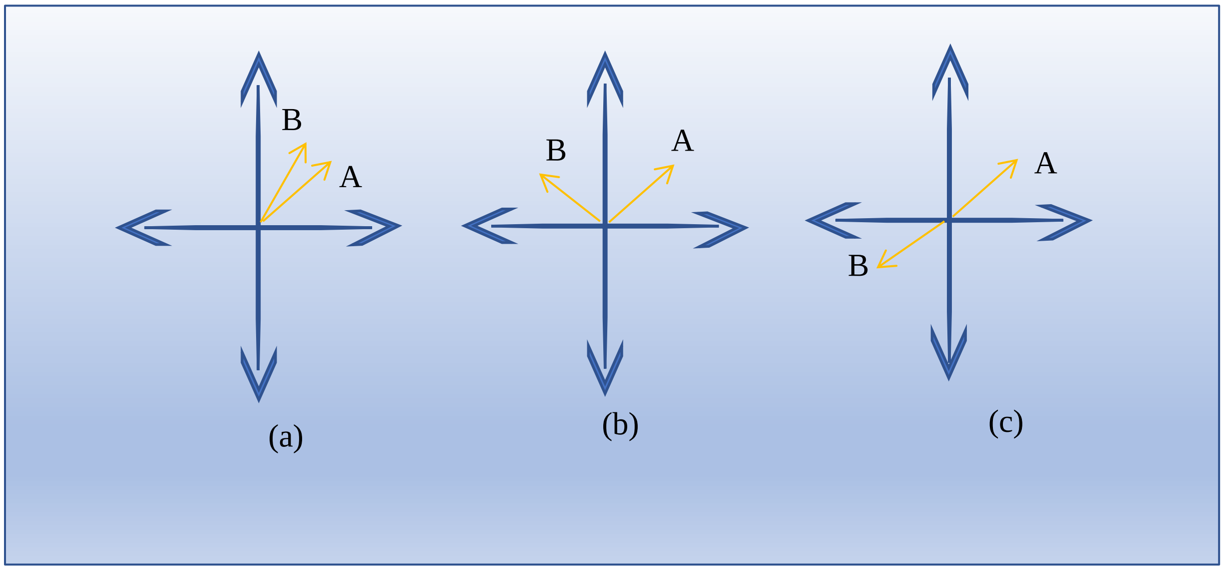


Figure 3.11: Vectors A and B with (a) Similar, (b) Orthogonal, and (c) Opposite cosine relation.

Figure 3.11 above illustrates the results of a cosine similarity score in a 2D Euclidean space. Calculating cosine similarity requires non-zero vectors for the calculation. A zero vector represents a non-existent magnitude and hence, a non-existent orientation. This requirement carries a significance in analyzing the role every coefficient feature vector plays in the decision-making process. The fact that after quantization, many of the low order high-frequency elements are zero removes specific coefficient feature vectors from the decision-making process. The range of the cosine similarity lies from -1 to 1, with -1 representing the opposite and 1 representing identical entities. Figure 3.12 shows the feature vectors extracted from each of the images as the inputs to the cosine similarity function.

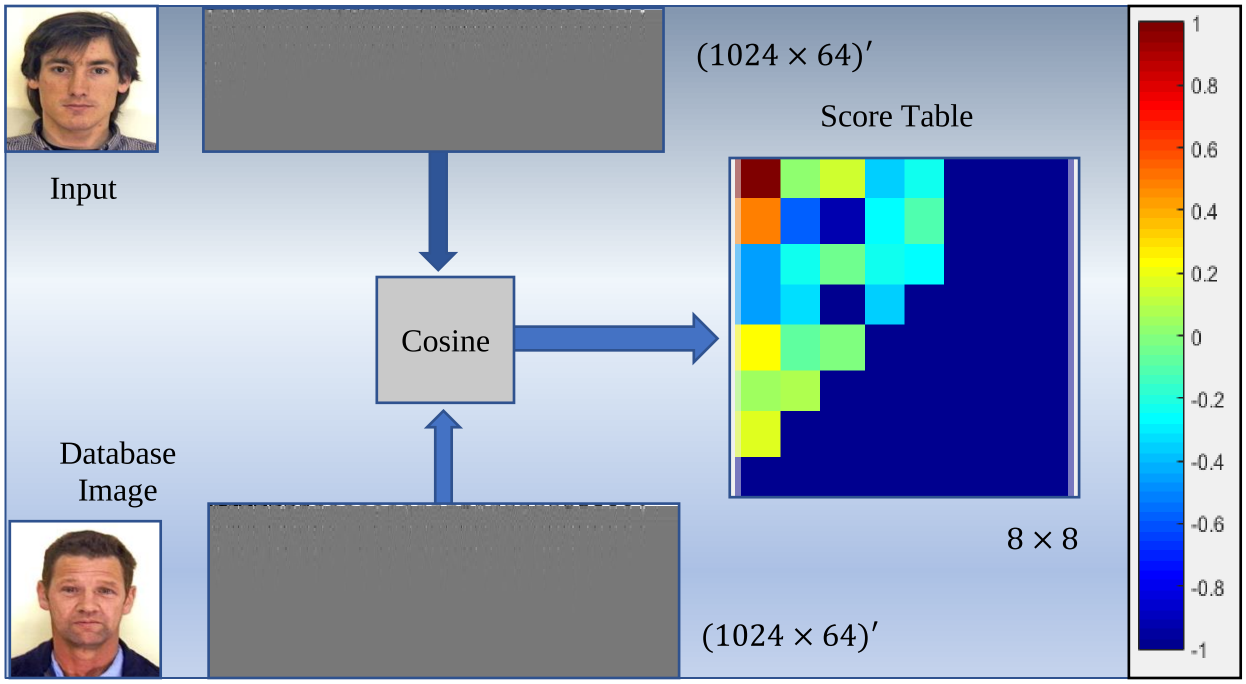


Figure 3.12: Feature vectors each of size 1024×64 are inputs to the cosine similarity function, which outputs the score for each of the 64 coefficients represented by the log color map ranging from -1 to 1.

3.4.2 Correlation value

The correlation measures the linear dependence between two variables. Pearson's correlation coefficient is used to calculate the correlation between the pair of the coefficient feature vectors. The equation for Pearson's correlation is below.

$$r_{XY} = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{nS_X S_Y} \tag{3.11}$$

where

X and Y represent the subject variables or in our case, the feature vector pairs.

X and Y have a positive linear relationship if the value of $r_{XY} = 1$.

X and Y have a negative linear relationship if the value of $r_{XY} = -1$.

X and Y have no linear relationship if the value of $r_{XY} = 0$.

The significance of calculating the correlation score helps us to reduce the number of coefficient feature vector we would use in the decision-making process. In the absence of the variance of each of the feature vectors, a condition arises where the correlation score results in not a number value denoted by NAN. The NAN score can be thus analyzed to understand the pattern of low-frequency and high-frequency components in the set of feature vectors. A correlation matrix best represents the correlation scores between the variables. Figure 3.13 shows the feature vectors extracted from each of the images as the inputs to the correlation similarity function and the resulting correlation values.

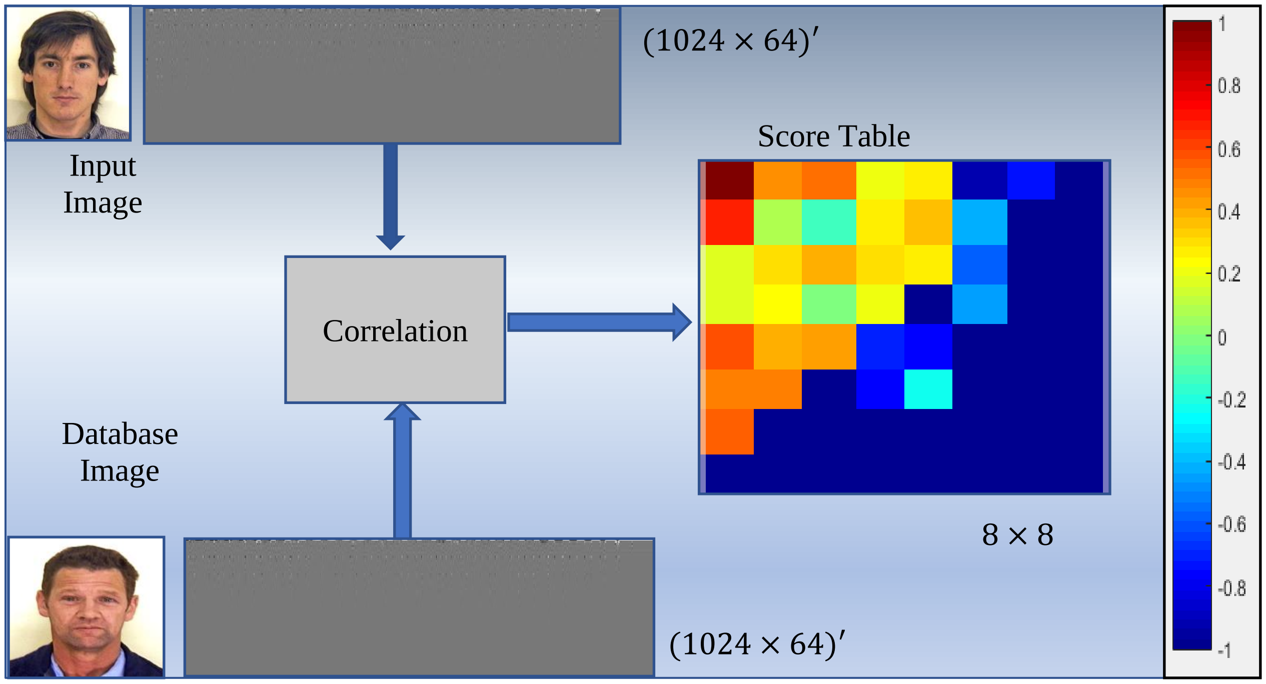


Figure 3.13: Feature vectors each of size 1024×64 are inputs to the correlation similarity function, which outputs the score for each of the 64 coefficients represented by the log color map ranging from -1 to 1.

3.4.3 Manhattan distance

The Manhattan distance introduced by Hermann Minkowski is also known as the L1 norm or city block distance. It is the measure of the distance between two points in an N-dimensional vector space. Given a 2D vector space, let us suppose we have points A $(x1, y1)$ and B $(x2, y2)$. The equation below provides the Manhattan distance between points A and B.

$$|x_1 - x_2| + |y_1 - y_2| \tag{3.12}$$

Regression analysis and frequency distribution applications very frequently use the Manhattan distance. The frequency distribution application utilizes the benefits of this metric to find out similar frequency distributions in a pair of images. The Manhattan distance has been used previously in the process of facial recognition in [7]. The range of the score calculated can range from $0\ to\ +\infty$. Figure 3.14 shows the feature vectors extracted from each of the images as the inputs to the Manhattan similarity function and the resulting distance values.

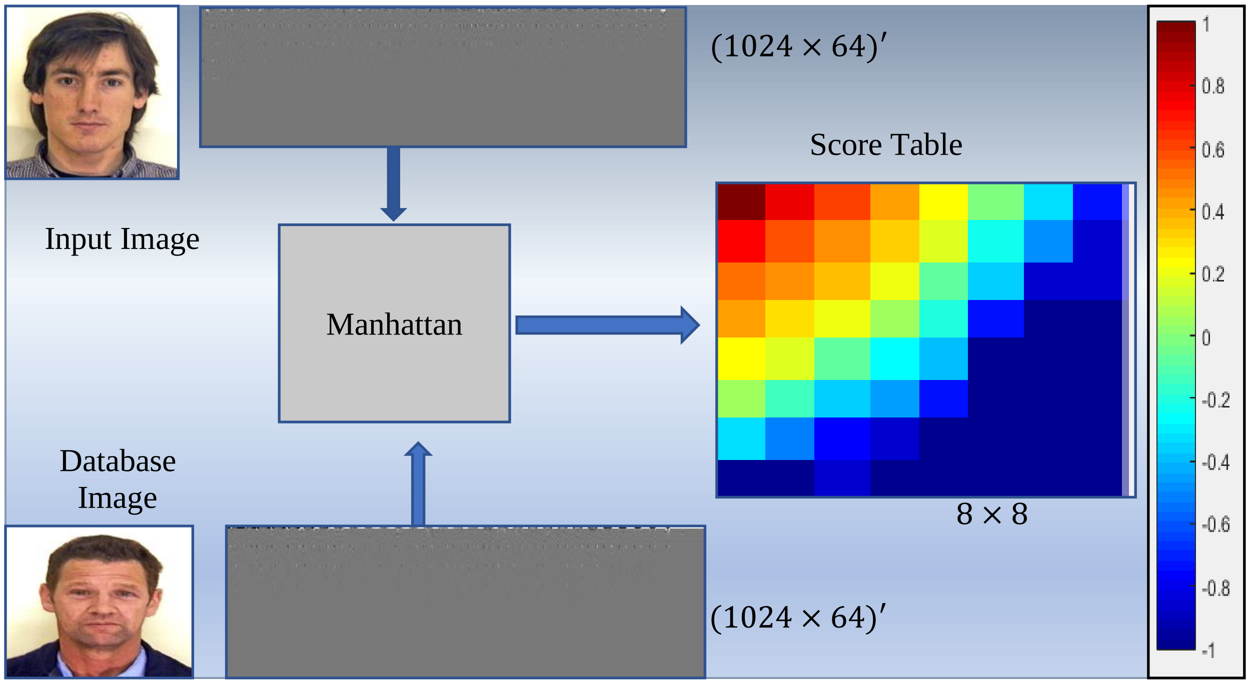


Figure 3.14: Feature vectors each of size 1024×64 are inputs to the Manhattan similarity function, which outputs the score for each of the 64 coefficients represented by the log color map ranging from -1 to 1.

3.5 Preprocessing

The preprocessing step in the algorithm is used to maintain a standard format for all the images to be processed. In this work, we have converted all the images to the grayscale format with size 256×256. In Figure 3.15, we show the preprocessing of an input image before we proceed to the coefficient wise feature extraction.

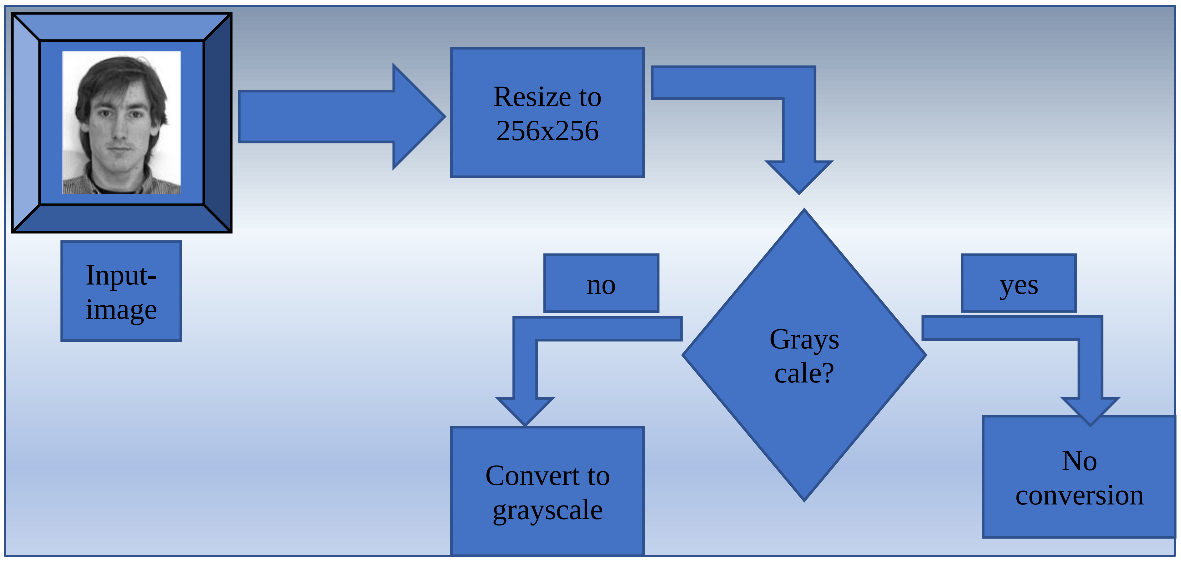


Figure 3.15: Image preprocessing steps.

3.6 Image variants

In the process of image matching algorithm, an essential component is a threshold. In the absence of a threshold, we would not be able to decide on a face match. To establish an empirical value for the threshold, we generated five affine variants of an image, as explained in this section. Image variants are created to analyze the differences in the thresholds. Given the affine transforms, we are analyzing the strength of the coefficients in the order from high to low.

The variants were created by following the concept of geometric affine transforms of an image [1]. The geometric transform of an image is the process of modifying the locations of the pixel values from one point in an image to another image. A list of geometric transforms includes translation, rotation, scaling, and many more.

We chose these four transforms, namely translation, rotation, scaling, and shear, to create image variants. We also created a transform called Puzzle. Geometric transforms can generalize by the equation given below.

$$Forward\ transform \rightarrow g(x,y) = f\big(u(x,y), v(x,y)\big) \tag{3.13}$$

$$Inverse\ transform \rightarrow f(u,v) = g(x(u,v), y(u,v)) \tag{3.14}$$

The transform can be represented in a 2D Cartesian coordinate, as shown in Figure 3.16 below.

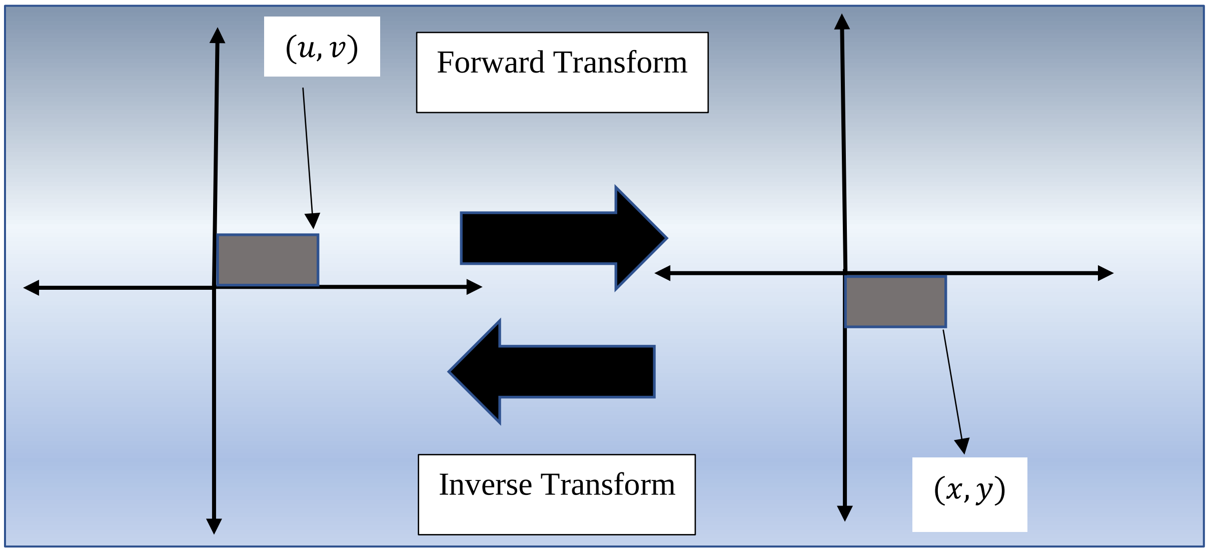


Figure 3.16: A general forward and inverse geometric transform.

The coordinate pair (u,v) represents a pixel in the original image which is transformed into the coordinate pair (x,y) in the new image.

3.6.1 Translation

Translation of an image is the mapping of the image pixels using a transformation matrix T in Equation 3.16 depicted in Figure 3.17 and described by Equation 3.13.

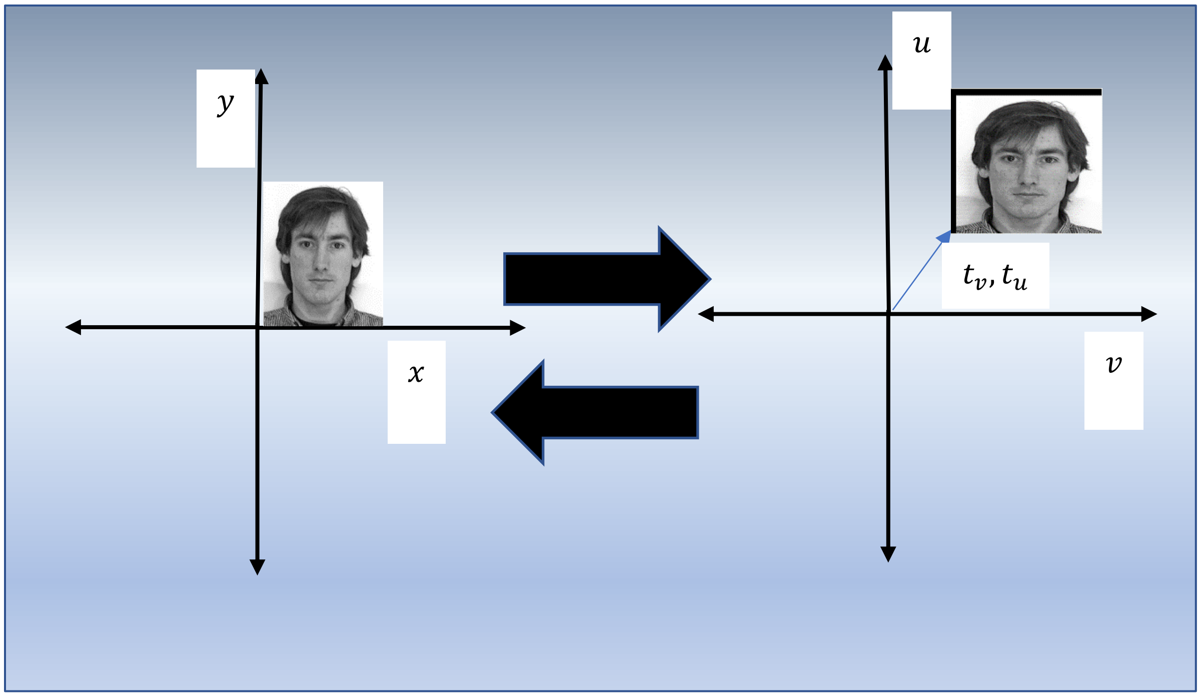


Figure 3.17: Translation of an input image from a vector space (x, y) to (v, u).

$$\begin{aligned} v &= x + t \\ u &= y + t \end{aligned} \qquad 3.15$$

The translation matrix for a 2D affine geometric transform used in this thesis is the following

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 100 & 0 & 1 \end{pmatrix} \qquad 3.16$$

3.6.2 Rotation

The rotation transform maps a pixel U located at (u, v) to a new pixel X located at (x, y) by rotating it by an angle θ. The transformation matrix for the rotation transform is given by the following equation.

$$X = RU, \quad U = R^T X \qquad 3.17$$

where

$$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & co\theta \end{bmatrix} \tag{3.18}$$

Figure 3.18 below illustrates the output of a rotation transform on an input image.

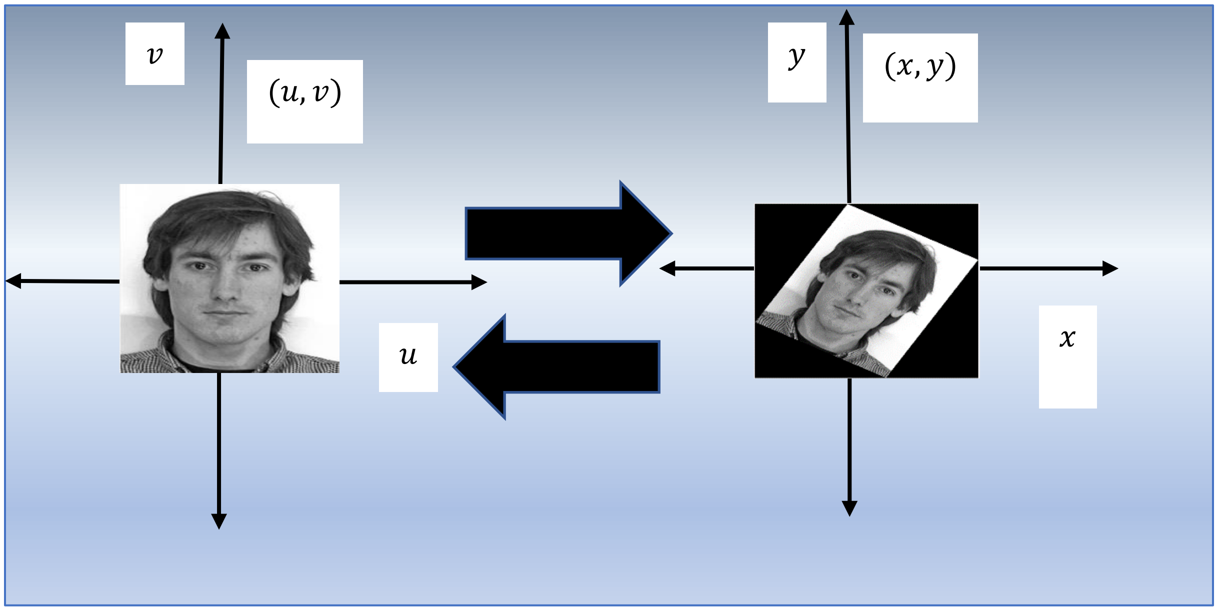


Figure 3.18: Rotation of an input image by an angle of 30 around the origin.

The rotation angle θ used in this thesis is 30 and results in the following rotation matrix.

$$R = \begin{pmatrix} \cos(30) & \sin(30) & 0 \\ -\sin(30) & \cos(30) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

3.6.3 Shear

The shear transform changes the position of the pixels, so that the output is the slanted version of the original object. The value used in this work to change the y and x coordinates proportionally is 0.5. Figure 3.19 represents the shear transform applied to an input image and is described by Equation 3.20.

$$\begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + az \\ y + bz \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} \qquad 3.20$$

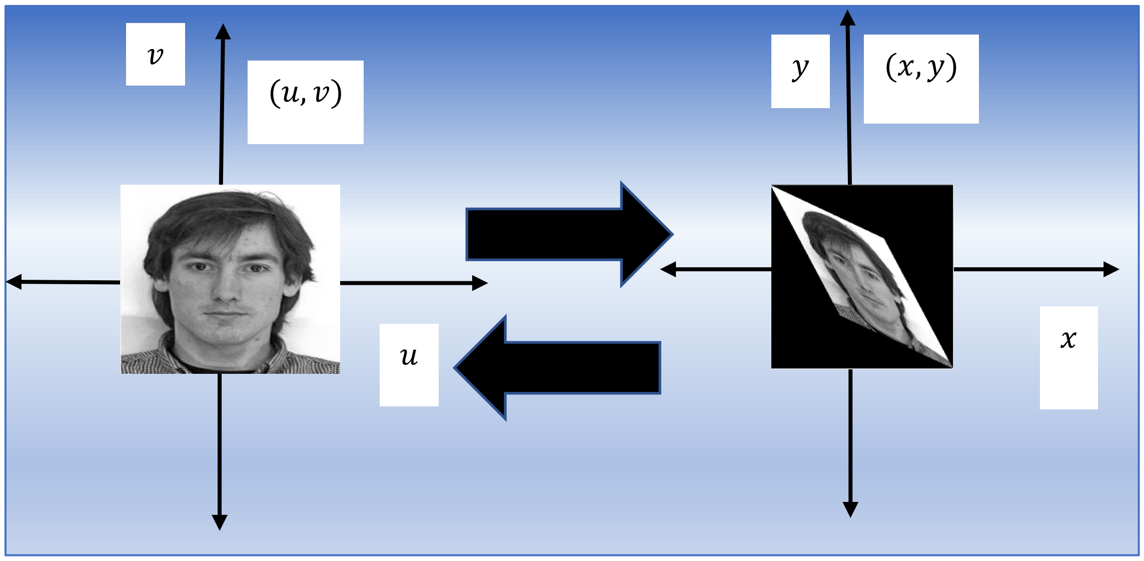


Figure 3.19: The Shear transformation of an input image.

3.6.4 Scale

Scaling of an image causes the magnification or diminution of that image, transforming a pixel U located at (u, v) to a new pixel X located at (x, y). The scale factor used in the transform matrix shown in Equation 3.21 is 5.

$$Scale = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.21}$$

We can scale an image by following Equation 3.22

$$X = SU, \quad U = S^{-1}X. \tag{3.22}$$

The Figure 3.20 given below shows the scaling of an input image.

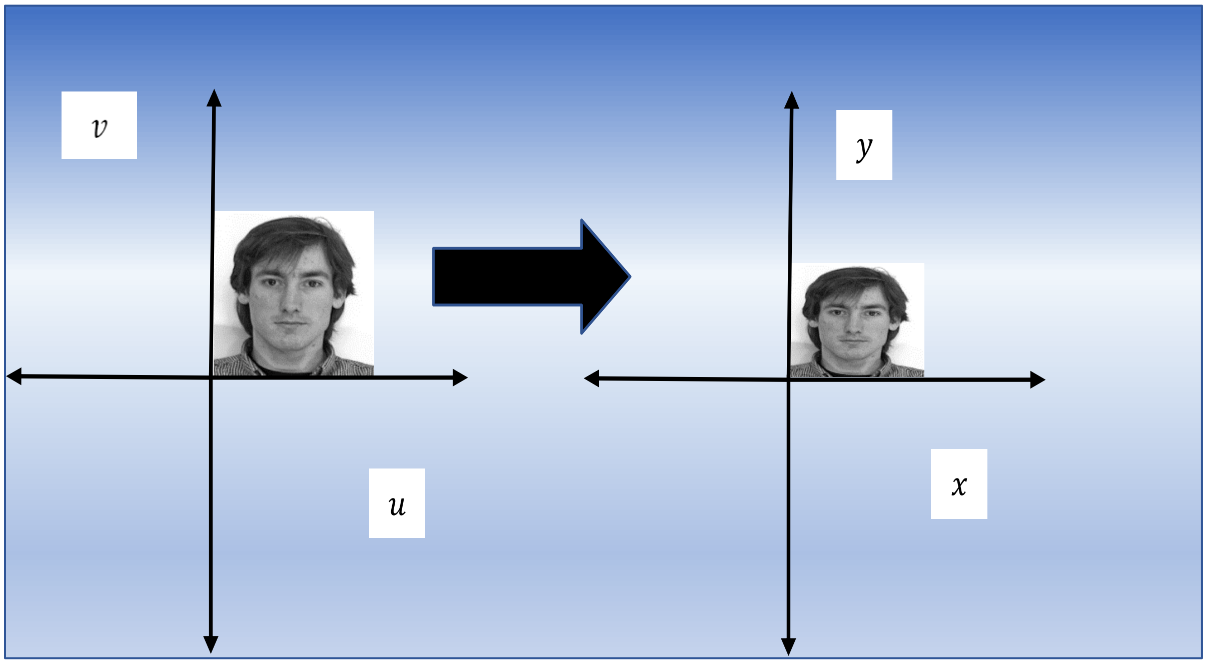


Figure 3.20: Scale transformation of an input image.

3.6.5 Puzzle

The transform called “Puzzle” is a straightforward concept of generating random pixel locations within the limits of the image width and height to create a puzzle of an image. We divided an image into non-overlapping image blocks of size 8×8 and then shuffled their locations based on a random number generator. The random number generator generates the numbers in the range of the image width and height. Figure 3.21 depicts the process of generating a puzzle transform.

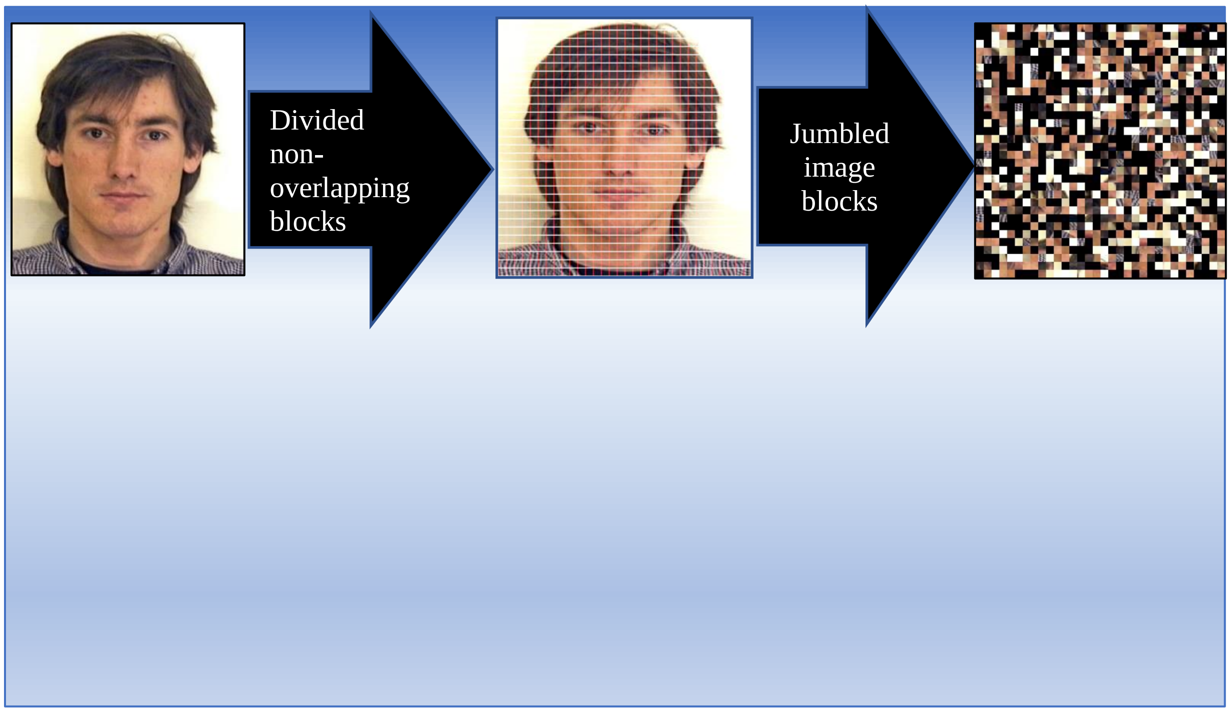


Figure 3.21: Transforming an input image to jumbled pieces.

3.7 Not a number

This section provides details about how a coefficient’s significance depends upon individual factors for each of the similarity functions. We encounter the “not a number” or “NAN” situation because of division by zero. We calculated the number of times we encountered NAN for every coefficient feature vector. The Aberdeen dataset [1] has 687 images. Figure 3.22 shows the graph that represents the NAN scores calculated for all images in the Aberdeen dataset. The Y-axis of the graph represents the number of NAN values, and the X-axis represents the coefficient feature vectors from 1,2, … ,64.

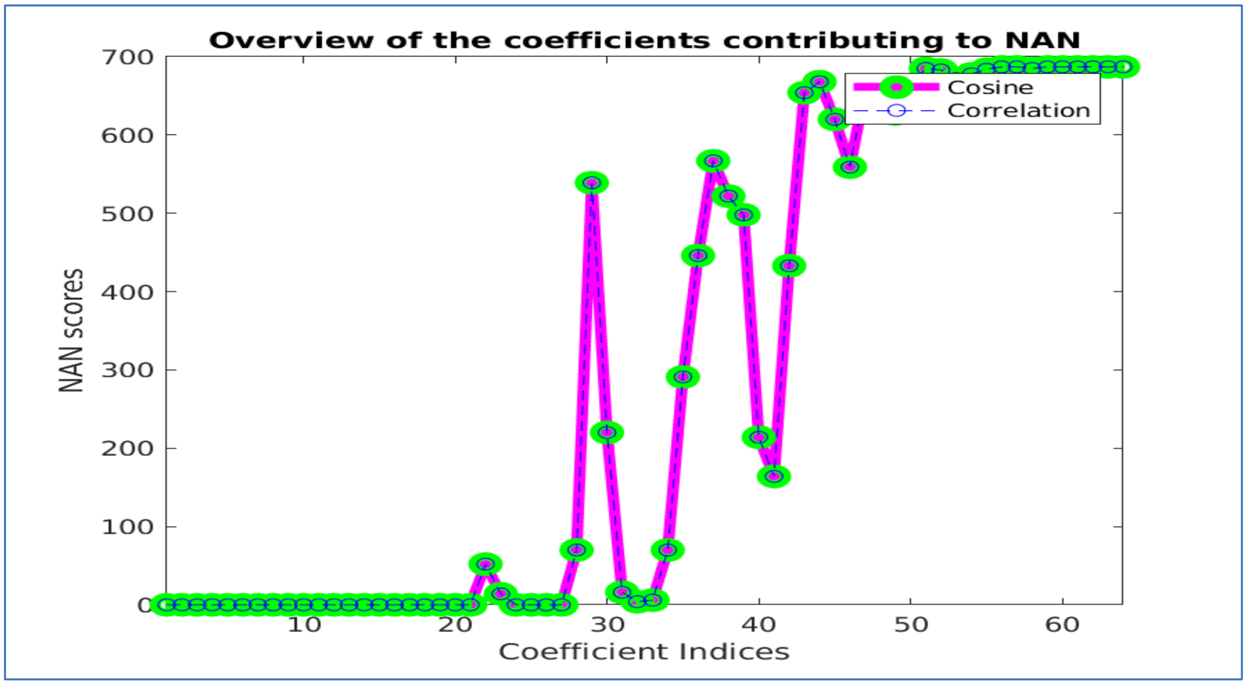


Figure 3.22: Graph representing the total number of images for which the cosine and correlation of the coefficients are NAN. The Y-axis represents the number of images that were calculated to have NAN, and X-axis represents the coefficient numbers.

- As mentioned earlier, a cosine metric is impossible to calculate in the presence of a zero vector. Many of the high-frequency coefficients are rendered zero after the quantization step. The zeros in the high order coefficient positions result in zero magnitude feature vectors at the higher orders and hence make it impossible to calculate a cosine score.

- Under correlation, the coefficients due to the absence of variance in the feature vector set to result in a NAN value as a score.
- The Manhattan distance does not suffer from these effects created by a zero vector or a non-varying feature vector. Instead, we observed elementary effects after quantization for a Manhattan score, where if the two coefficient vectors were rendered 0, their distance metric resulted in a value 0. A 0 value helped to eliminate those coefficient feature vectors from the list of the credible feature vectors.

3.8 Conclusion

In this chapter, we explained the components involved in the privacy-preserving face matching algorithm. We observed how the quantization step results in a varying number of zeros in the high order coefficient positions. This, in turn, has specific implications in the use of the resulting coefficient feature vectors for the purpose of computing a similarity score between two feature vectors.

Chapter 4

Datasets

In this section, we describe the datasets utilized in the process of testing the utility of the proposed matching process. Two datasets are used, namely:

- 2D Face sets - Aberdeen
- 2D Face sets - Utrecht ECVP

2D face sets are a collection of images used primarily for psychological experiments. The dataset is comprised of facial images of subjects with varying pose and expression. In the upcoming sections, we describe the arrangement of the dataset and the extraction process of the data in order to create the ground truth and the test set.

4.1 2D Face sets

2D Face Sets [1] is the collection of the image data conducted by the Psychological Image Collection at Stirling. There are nine image sets, namely Aberdeen, Iranian women, Nottingham scans, Nott-faces-originals, Stirling faces, Pain expressions, Pain expression subset, Utrecht ECVP, Mooney_LR, and Mooney_MF. Out of these nine sets, we have used two of the image sets to create a ground truth set and test set.

4.1.1 Aberdeen

There are 687 Color faces from Ian Craw at Aberdeen. The images belong to 90 individuals with variations in lighting and viewpoint, as shown in the sample images in Figure 4.1. The resolution of the images varies from 336×480 to 624×544.

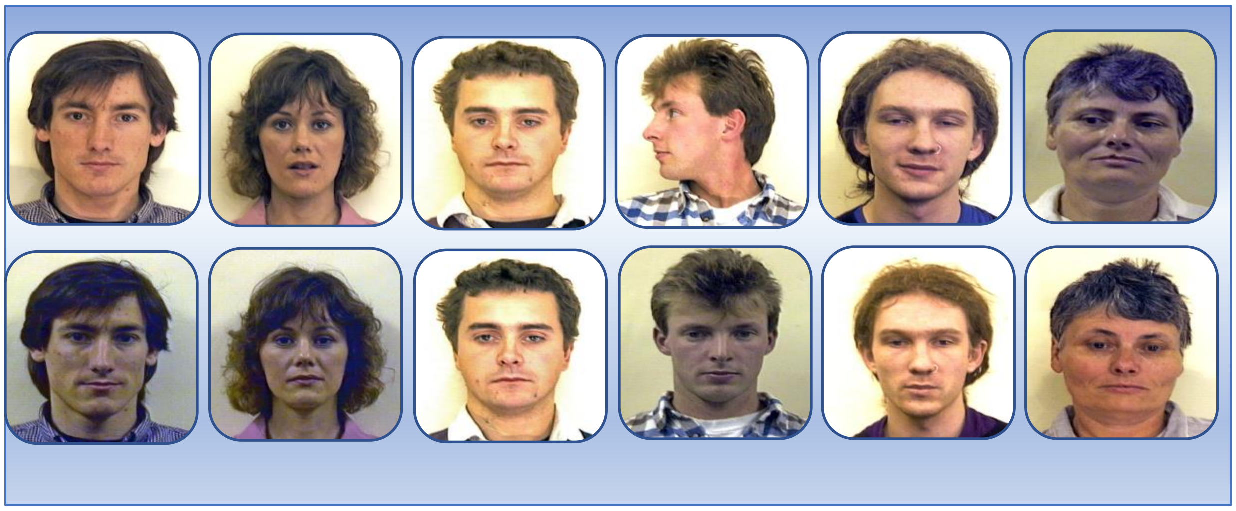


Figure 4.1: Aberdeen dataset image samples representing the viewpoint and lighting variations in the dataset.

The ground truth dataset created from this set of images involves all the 687 images for each of which we have created the five transformation variants, as shown in Figure 4.2.

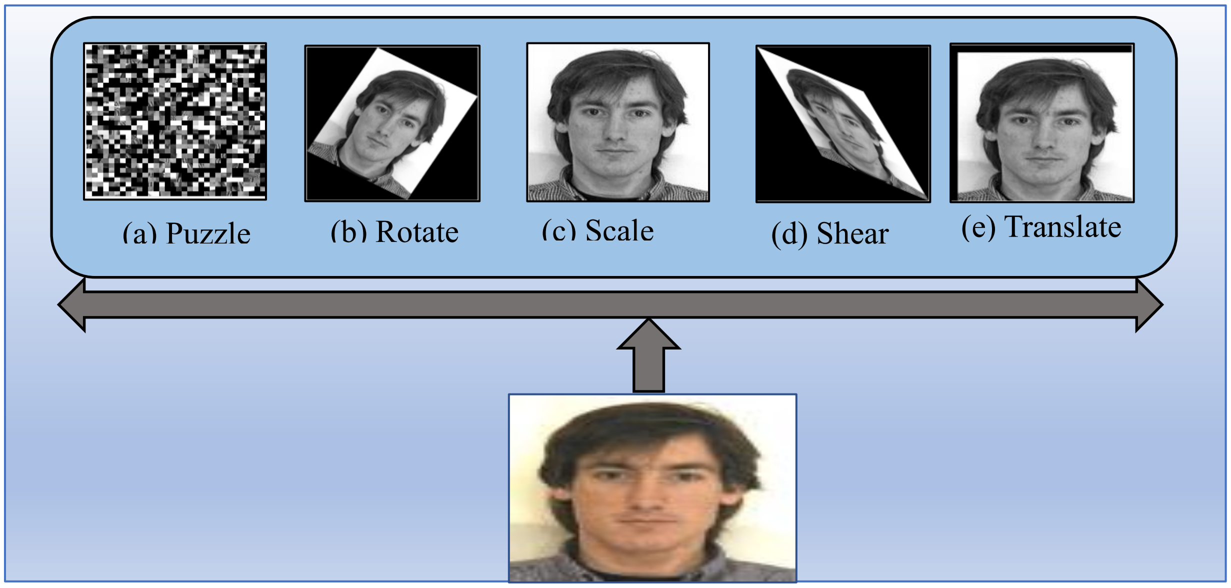


Figure 4.2: Aberdeen dataset image and its variants (a) puzzle, (b) rotate, (c) scale, (d) shear, and (e) translate.

4.1.2 Utrecht ECVP

There are 131 images of 49 men and 20 women with neutral faces and faces with a smile collected at the European conference on visual perception in Utrecht in 2008. The resolution of the images varies from 900 × 1200. Figure 4.3 shows sample images from this dataset.

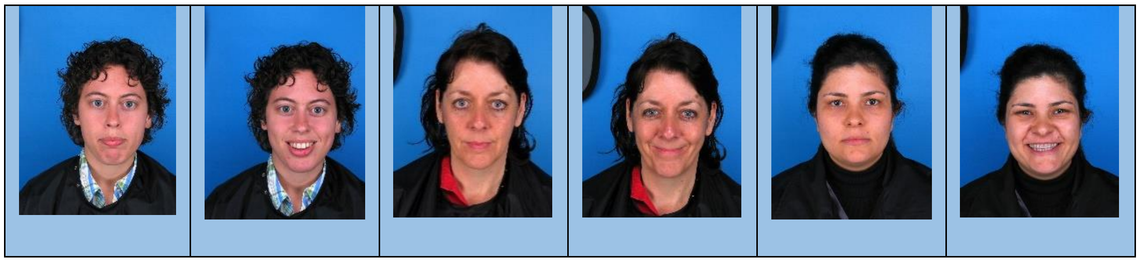


Figure 4.3: Utrecht dataset image samples.

We have created the transformation variants for all 131 images to create the ground truth. Figure 4.4 shows the variants of an example image from the dataset.

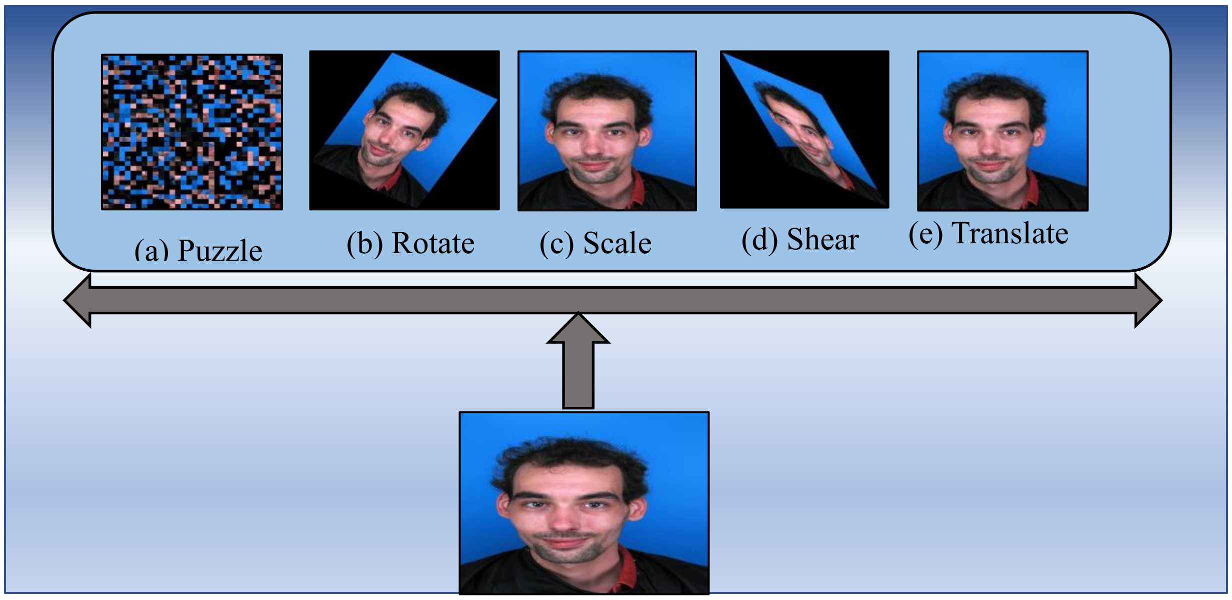


Figure 4.4: Utrecht dataset image and its variants (a) puzzle, (b) rotate, (c) scale, (d) shear, and (e) translate.

4.2 Significance of different sets in creating ground truth

We have created the ground truth using the Aberdeen dataset, which contains multiple images of the same person, and we have created another ground truth dataset which does not contain multiple images of the same person. We chose to create two ground truth datasets so that we can observe a pattern in the average and mode threshold scores, if any. Also, the number of images in the two databases is different, which can illuminate the differences in the thresholds. It was of importance to perform testing under such a set up as we wanted to analyze the behavior of the coefficients individually. Therefore, it is essential to note that the experiments involving the ground truth set, test set creation, and threshold calculations had been carried out for all the 64 coefficients separately.

Chapter 5

Experiments and results

In this chapter, we report the experiments performed and the results obtained. We also discuss in detail how we have prepared our ground truth, test set, and the process of thresholding. We report the average accuracy across the test sets for the two databases separately.

5.1 Ground truth set

The ground truth set is the set of images for which we have calculated the five variants using the affine transformations defined in Chapter 3. The significance of these variants is to observe a pattern in the threshold values calculated over the entire dataset. We calculated the average and mode threshold for every coefficient number from $1\ to\ 64$ across all the image-variant pairs. Given a dataset has a total number of x images, the total number of image-variant pairs are $5 \times x$.

$$Total\ data\ points = 5 \times x \qquad 5.1$$

$$Average = \begin{cases} \text{Correlation}_i \ \dfrac{\sum_{1}^{Total\ data\ points} score}{Total\ data\ points} \\ \text{Cosine}_i \ \dfrac{\sum_{1}^{Total\ data\ points} score}{Total\ data\ points} & i = 1\ to\ 64 \\ \text{Manhattan}_i \ \dfrac{\sum_{1}^{Total\ data\ points} score}{Total\ data\ points} \end{cases} \qquad 5.2$$

$$Mode = \begin{cases} Correlation_i(List[Score\ of\ total\ data\ points]) \\ Cosine_i(List[Score\ of\ total\ data\ points]) \\ Manhattan_i(List[Score\ of\ total\ data\ points]) \end{cases} \quad i = 1\ to\ 64 \qquad 5.3$$

Figure 5.1 shows the process of ground truth construction for a dataset.

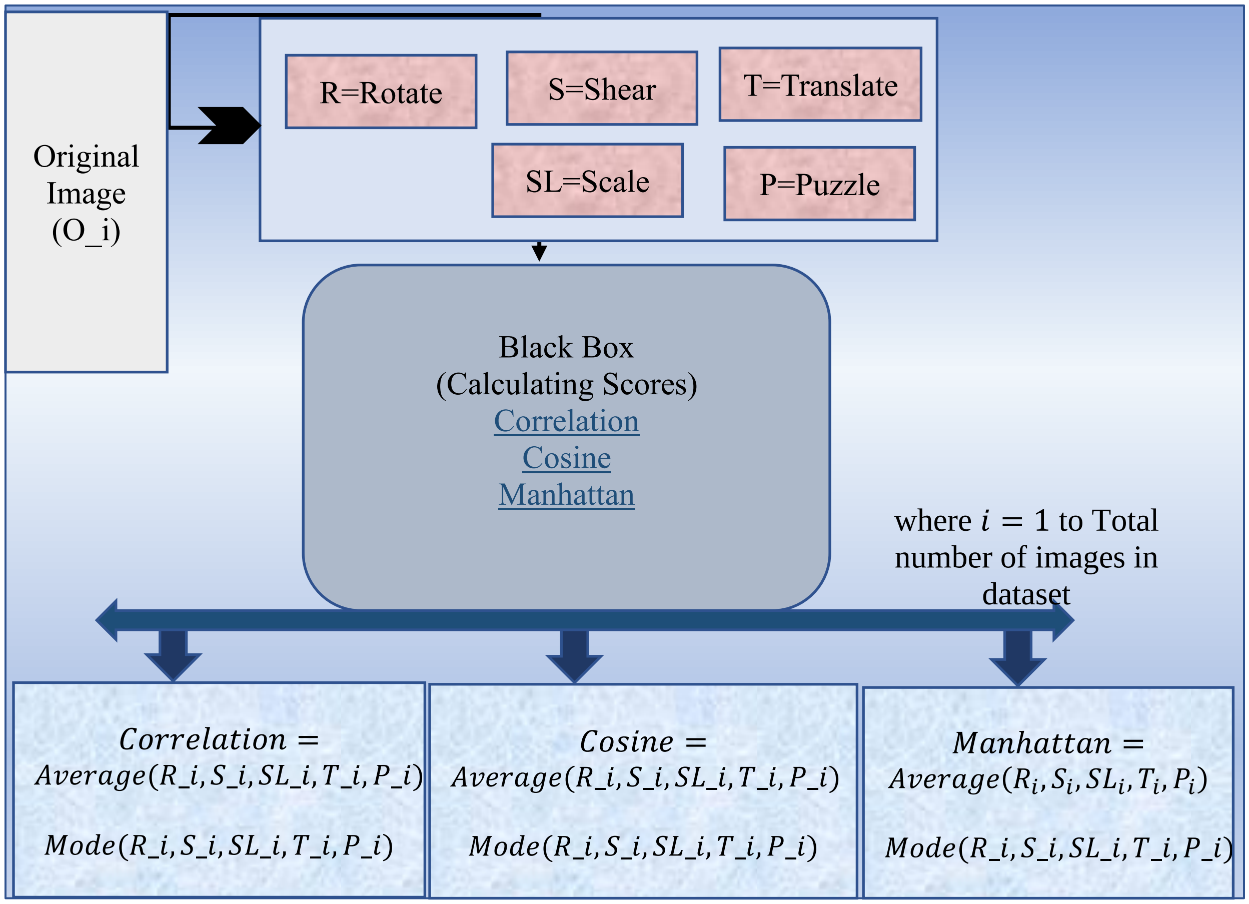

Original Image (O_i)
R=Rotate
S=Shear
T=Translate
SL=Scale
P=Puzzle
Black Box (Calculating Scores)
Correlation
Cosine
Manhattan
where i = 1 to Total number of images in dataset
$Correlation = Average(R\_i, S\_i, SL\_i, T\_i, P\_i)$ $Mode(R\_i, S\_i, SL\_i, T\_i, P\_i)$
$Cosine = Average(R\_i, S\_i, SL\_i, T\_i, P\_i)$ $Mode(R\_i, S\_i, SL\_i, T\_i, P\_i)$
$Manhattan = Average(R_i, S_i, SL_i, T_i, P_i)$ $Mode(R\_i, S\_i, SL\_i, T\_i, P\_i)$

Figure 5.1: Ground truth construction by creating variants of an image. For an image-variant pair, we calculated cosine, correlation, and Manhattan scores. The threshold for the entire dataset was calculated by averaging the scores across all such image-variant pair.

5.2 Test set

The test set construction involves the following steps:

- For every dataset, we extracted 50 images at random as the 50 faces that we want to match.
- The rest of the images in the dataset becomes the database images.
- We matched every face against the database for which we calculate the accuracy.
- We have repeated the above steps ten times, and the accuracy calculated for every test set is then averaged over the 10-test sets.
- To match an image pair, we have used two types of thresholds – average and mode. The matching is performed for every coefficient number separately.
- Figure 5.2 illustrates the process of test set generation.

$$Average\_Accuracy = \begin{cases} \frac{\sum_{i=1}^{10} Test\_Set_i}{10} \\ \frac{\sum_{i=1}^{10} Test\_Set_i}{10} \\ \frac{\sum_{i=1}^{10} Test\_Set_i}{10} \end{cases} \qquad 5.4$$

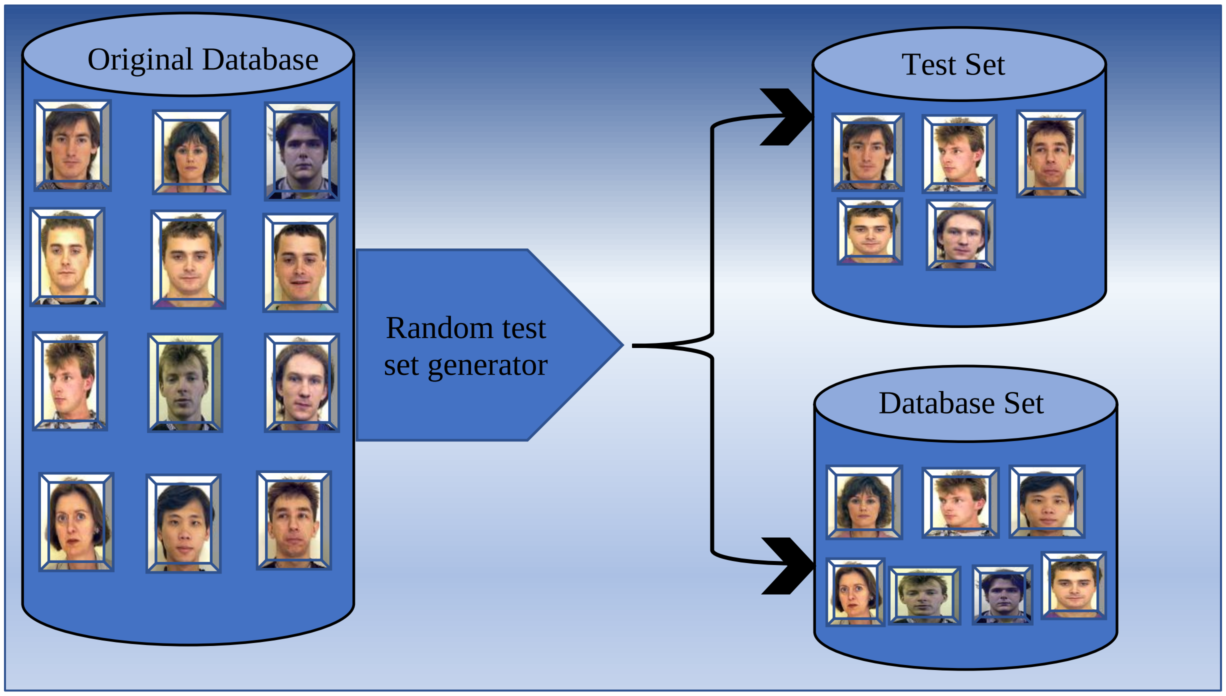


Figure 5.2: A pictorial representation of how the original database is split into a test set and a database set, such as the database is shown here, has a total of 12 images that results in a test set of 5, and a database set of 7 images.

5.3 Thresholding

Thresholding is the process where we obtain a value for an algorithm, which in turn aids the algorithm to make a certain kind of decision. In order to obtain such a significant value, we thought it would be interesting to experiment with two things, namely the average and the mode. As we have calculated the scores reflecting a match between the image-variant pairs, the prior knowledge about the pair belonging to the same person intrigued us to find out the central tendency among those scores for every coefficient.

- An average would reflect such a central tendency among the list of values.
- The mode of a list of numbers represents the value that is repeated most in the set and the minimum number in the list in the absence of a frequently used value. With the prior knowledge about the image-variant pair and the property of a number selected as the mode

of the list, we thought we could observe a pattern in the similarity scores calculated between the image-variant pairs.

We have calculated the average and mode thresholds for each of the similarity functions, namely cosine, correlation, and Manhattan, of the two datasets. The figures below reflect the importance of every coefficient number for every similarity function under two datasets separately.

We have represented the NAN values for the coefficients by the value -3 and zeroes obtained after calculating the scores by -4. This representation was adopted as we wanted to distinguish between the zeroes and NAN by looking at the charts themselves; otherwise, there was no distinction. Also, the zero values were hard to distinguish from minimal values. For the ground truth dataset for Aberdeen, we calculated the thresholds - average and mode for cosine, correlation, and Manhattan. We report the behavior of the coefficients under different similarity functions.

5.3.1 Aberdeen

In this section, we report the values for mode and average for every coefficient calculated over the entire ground truth set, which contains 687 images. The values thus are calculated over $687 \times 5 = 3435$ similarity function scores.

5.3.1.1 Manhattan

- The mode for the coefficients numbered from 1 to 21 has values very close to 0. This kind of behavior represents that most of the image-variant pairs have often had small Manhattan distances between them.

- As quantization reduces the coefficients, the average values of the Manhattan distance for the same coefficients decrease with increasing coefficient indices.

- The coefficients arranged in the decreasing order of their horizontal and vertical frequencies traversed in a zig-zag manner represent the higher indices with lower mode and average values.

- The Manhattan scores are normalized in the range [0,1] before calculating the average and mode that are plotted in the graphs shown in Figures 5.3 and 5.4.

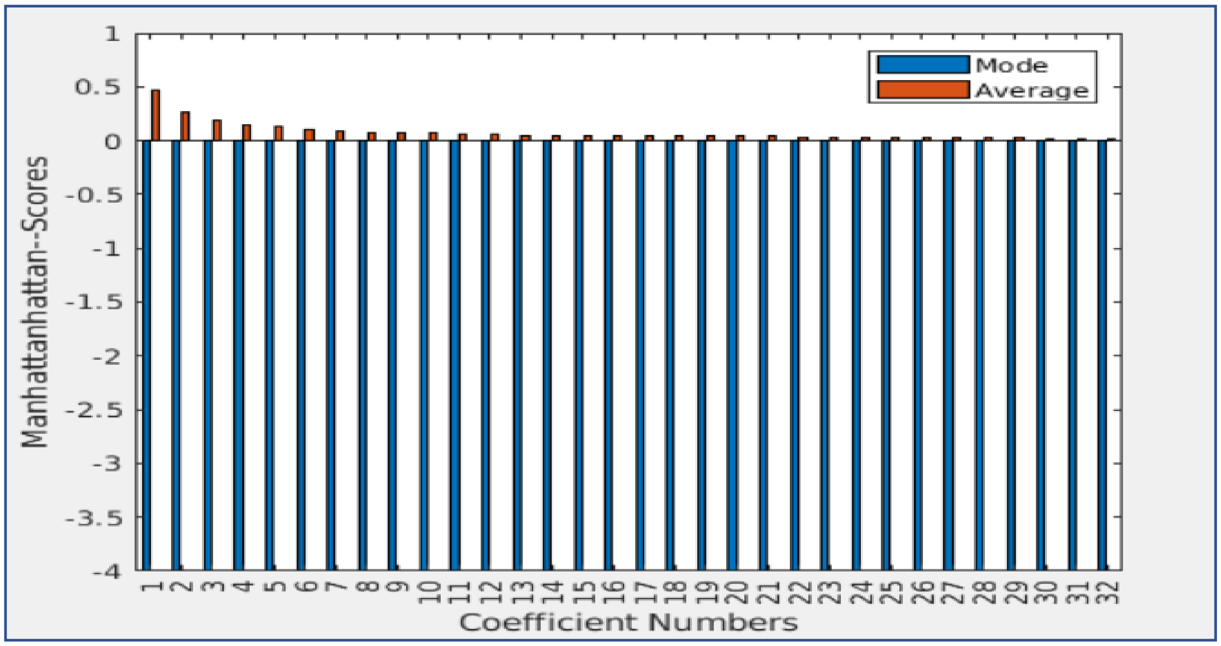


Figure 5.3: Manhattan mode and average threshold values for coefficient feature vectors from 1 to 32 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

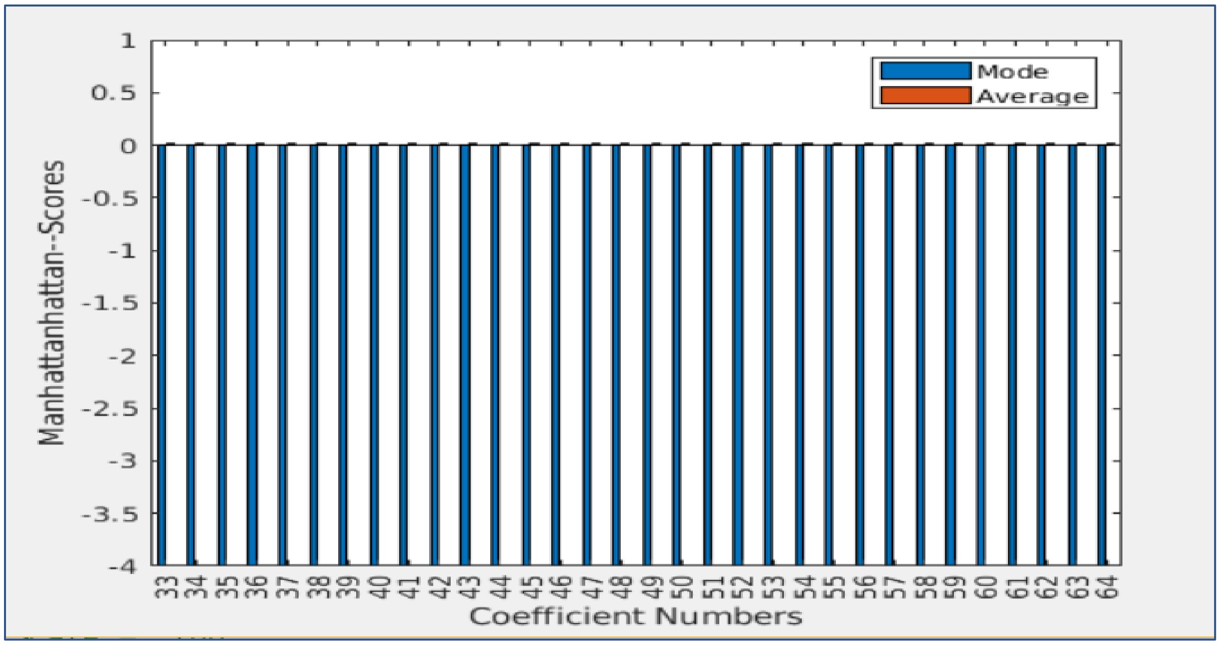


Figure 5.4: Manhattan mode and average threshold values for coefficient feature vectors from 33 to 64 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

- The mode for the coefficients numbered from 22 to 64 is 0. A 0 value shows how quantization produces minimal values in the higher positions in the coefficient list. Even in the absence of a most frequently occurring value, a 0 value reflects the presence of 0 or smaller values after the quantization process.

5.3.1.2 Correlation

- A score of NAN after calculating the correlation score is due to the absence of the variance in the feature vectors, which is the result of the zeros in the higher coefficient indices.

- The mode stops existing after coefficient number 10 because of the quantization transform values. The coefficients at the indices from 11 to 64 are 0 because of quantization, which produces small values at those indices, as shown in Figures 5.5 and 5.6.

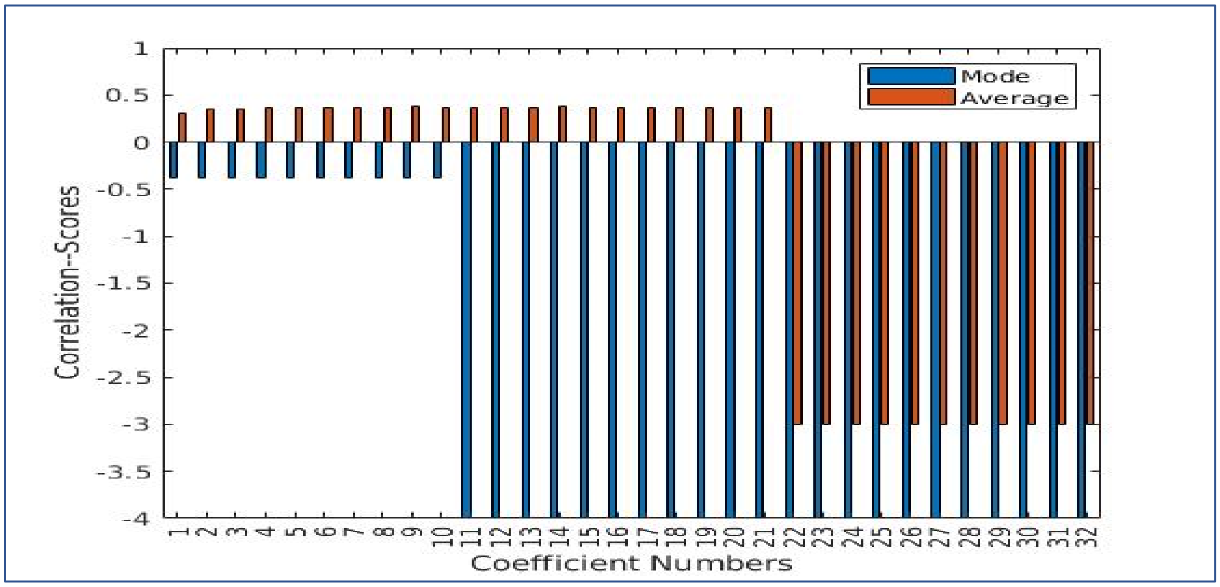


Figure 5.5: Correlation mode and average threshold values for coefficient feature vectors from 1 to 32 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

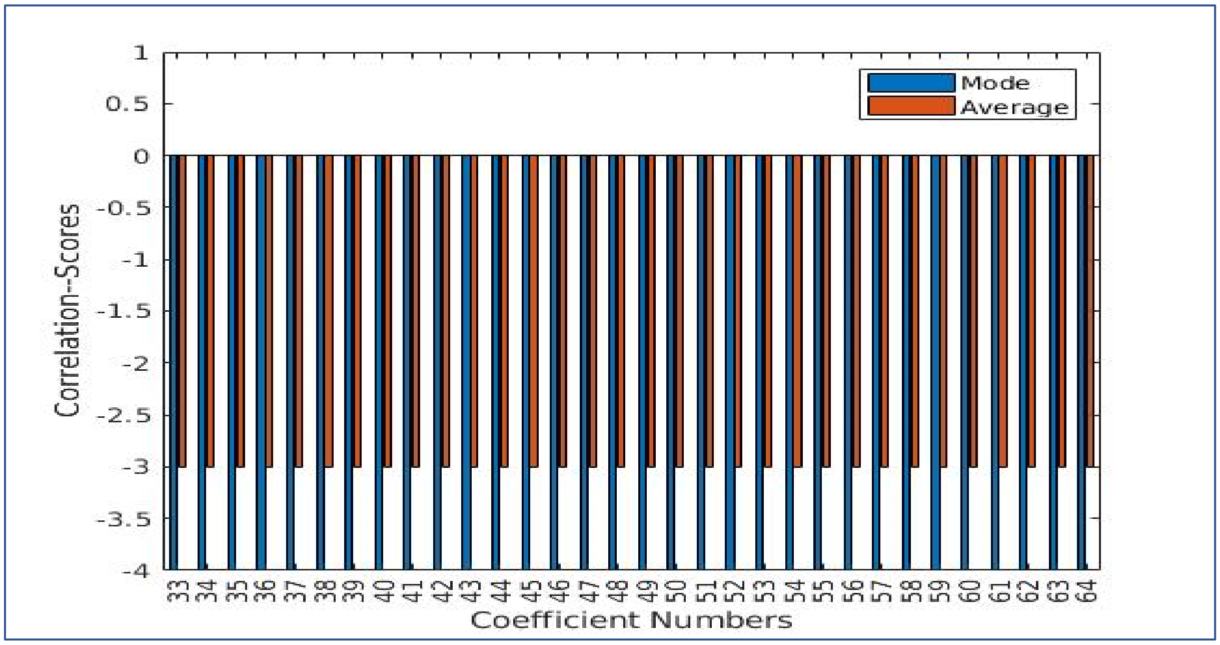


Figure 5.6: Correlation mode and average threshold values for coefficient feature vectors from 33 to 64 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

- The average value for most of the high order coefficients from 22 to 64 does not exist due to the presence of the NAN.
- It is interesting to observe how, among 64 coefficients, there are only a few which retain a valid average and mode value.

5.3.1.3 Cosine

- Calculating the cosine similarity involves the norm of a vector, and thus, when the norm is 0, the similarity score ceases to exist, resulting in NAN for average.
- As we observe in the charts in Figures 5.7 and 5.8, the mode only exists for the first three coefficients, whereas the mode for coefficients indexed from 22 to 64 again follows to be 0.
- As we are averaging the values across all the similarity scores in the ground truth dataset, the NAN values due to the zero vectors make the thresholds for the coefficients at the higher indices NAN, and hence those coefficients are excluded from the decision-making process.

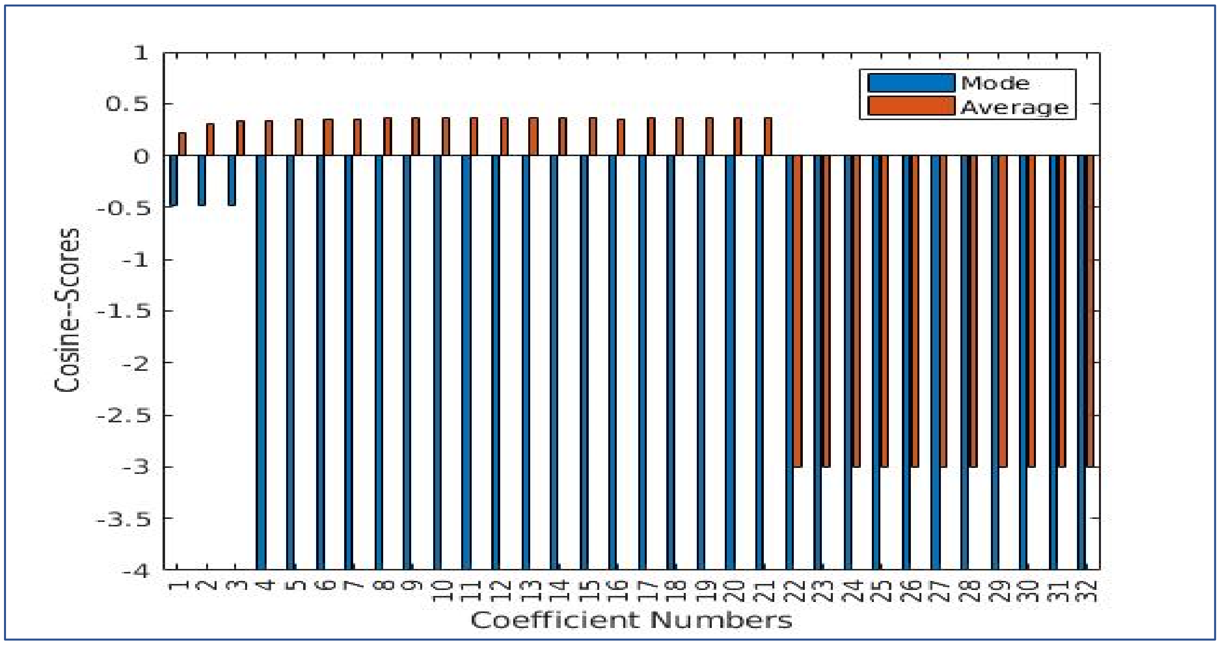


Figure 5.7: Cosine mode and average threshold values for coefficient feature vectors from 1 to 32 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

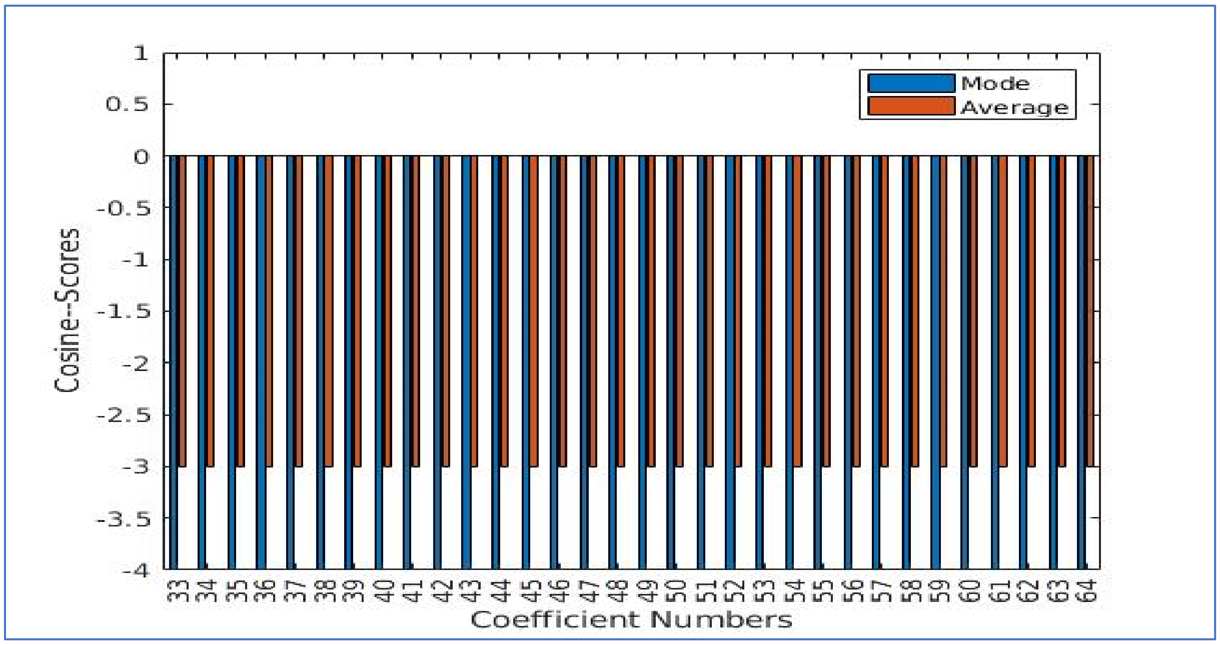


Figure 5.8: Cosine mode and average threshold values for coefficient feature vectors from 33 to 64 on Aberdeen dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

5.3.2 Utrecht ECVP

In this section, we report the values for mode and average for every coefficient calculated over the entire ground truth set, which contains 131 images. We have calculated the ground truth with 73 images, excluding multiple images. The values thus are calculated over $73 \times 5 = 365$ similarity function scores.

5.3.2.1 Manhattan

- We can observe from Figures 5.9 and 5.10 below, how the average values exist through all the coefficients but with decreasing values.

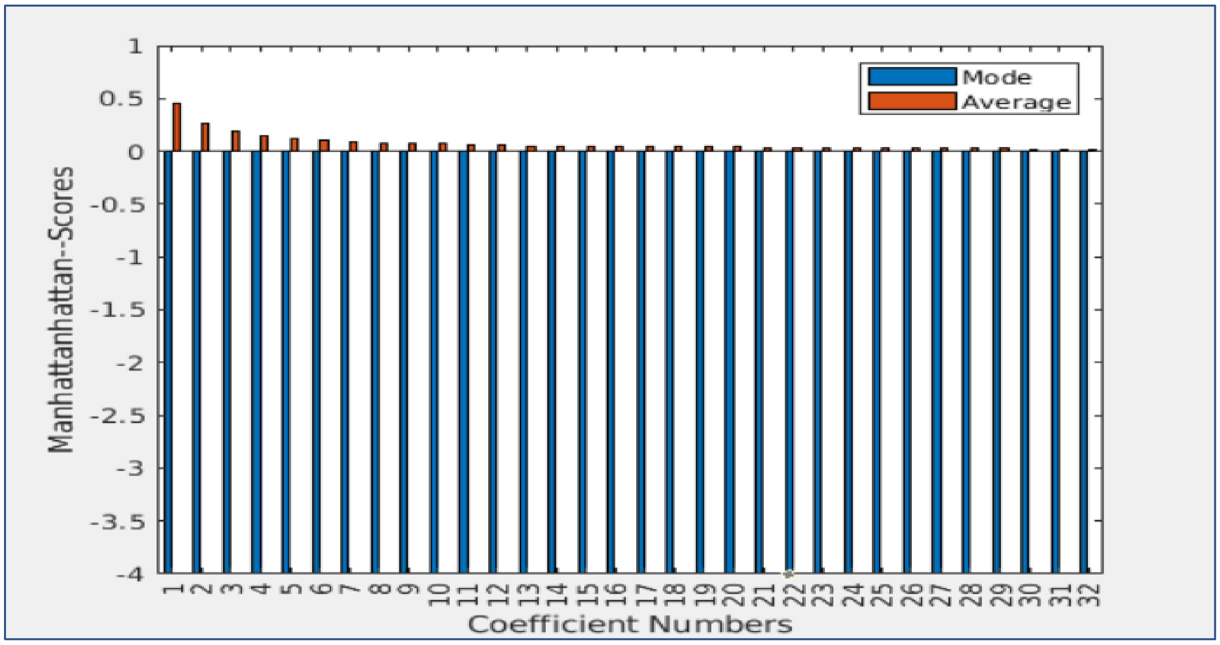


Figure 5.9: Manhattan mode and average threshold values for coefficient feature vectors from 1 to 32 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

- It is interesting to observe how the mode threshold does not exist for the first 35 coefficients, whereas, from 36 to 64, we see the NAN values.

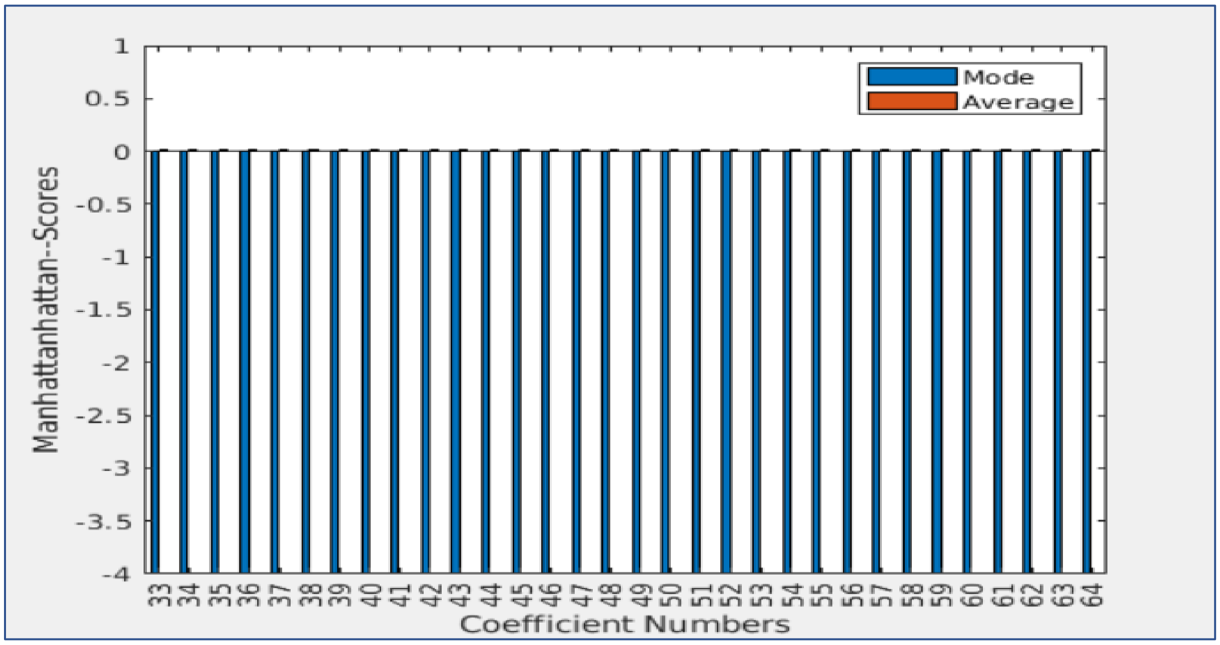


Figure 5.10: Manhattan mode and average threshold values for coefficient feature vectors from 33 to 64 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

5.3.2.2 Correlation

- The mode exists for coefficient indices 1 to 10, whereas the average exists for the coefficients from 1 to 15 as shown in Figures 5.11 and 5.12 below.

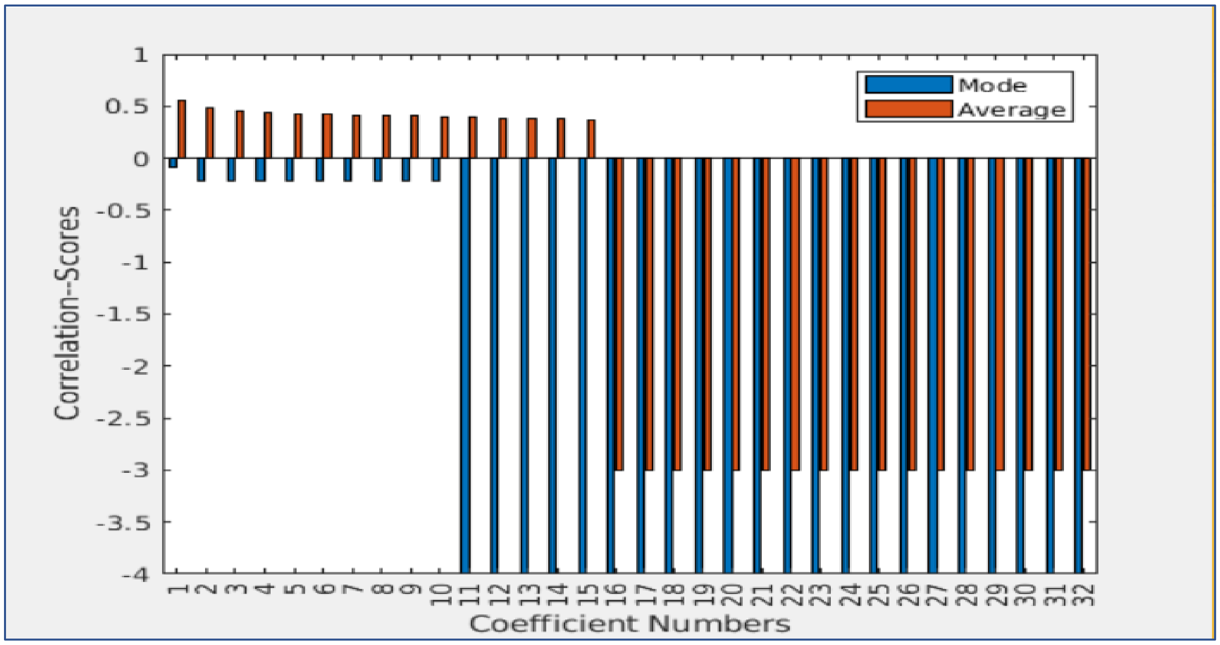


Figure 5.11: Correlation mode and average threshold values for coefficient feature vectors from 1 to 32 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

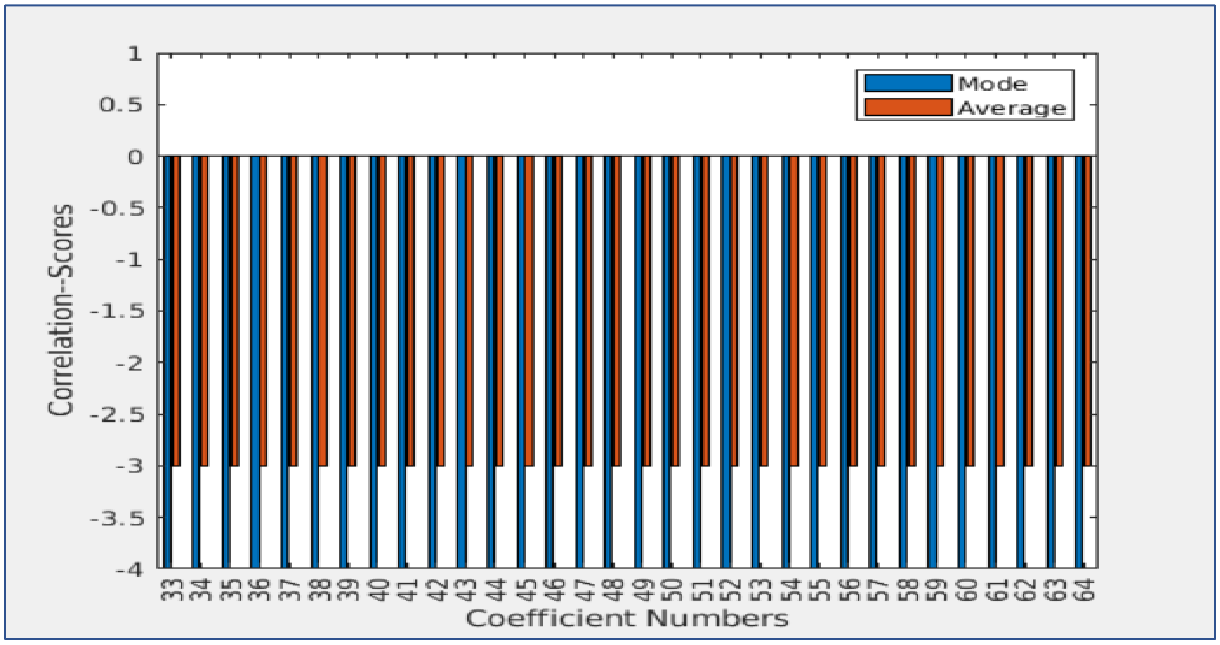


Figure 5.12: Correlation mode and average threshold values for coefficient feature vectors from 33 to 64 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

- The mode and the average both stop existing for higher coefficients.

5.3.2.3 Cosine

- As shown in Figures 5.13 and 5.14, a similar pattern is observed for the cosine mode and average threshold values for each of the coefficients when compared to the values for the correlation similarity function.

- Although the mode threshold can be seen to attain a positive value for the first coefficient.

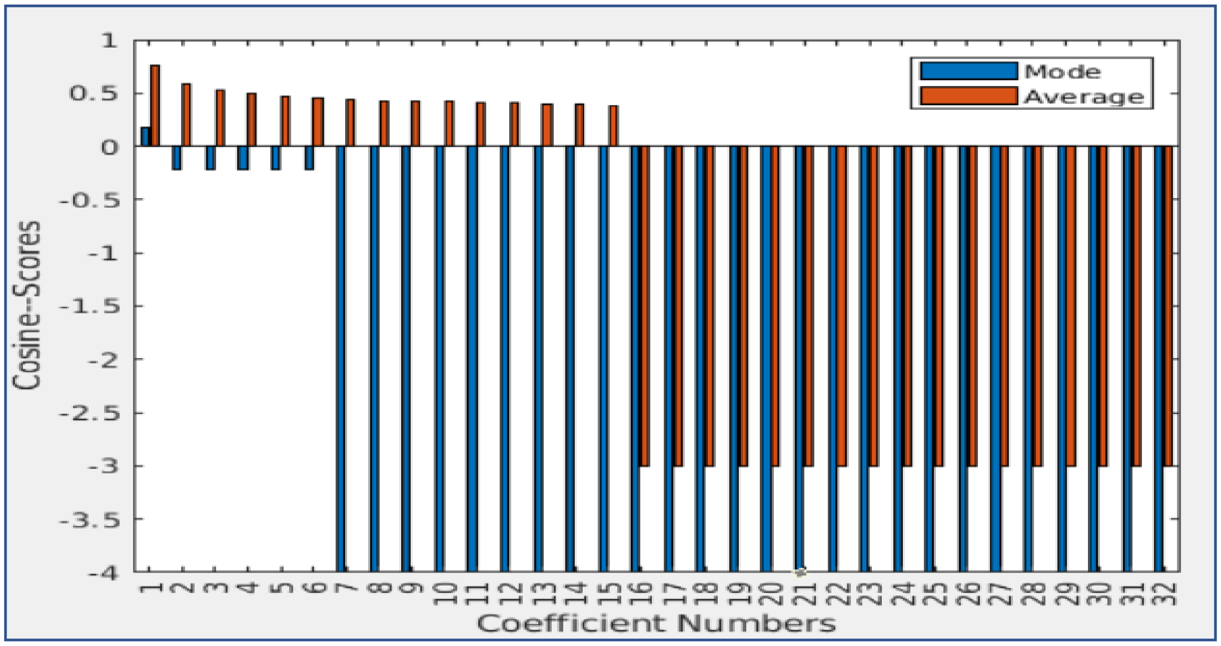


Figure 5.13: Cosine mode and average threshold values for coefficient feature vectors from 1 to 32 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

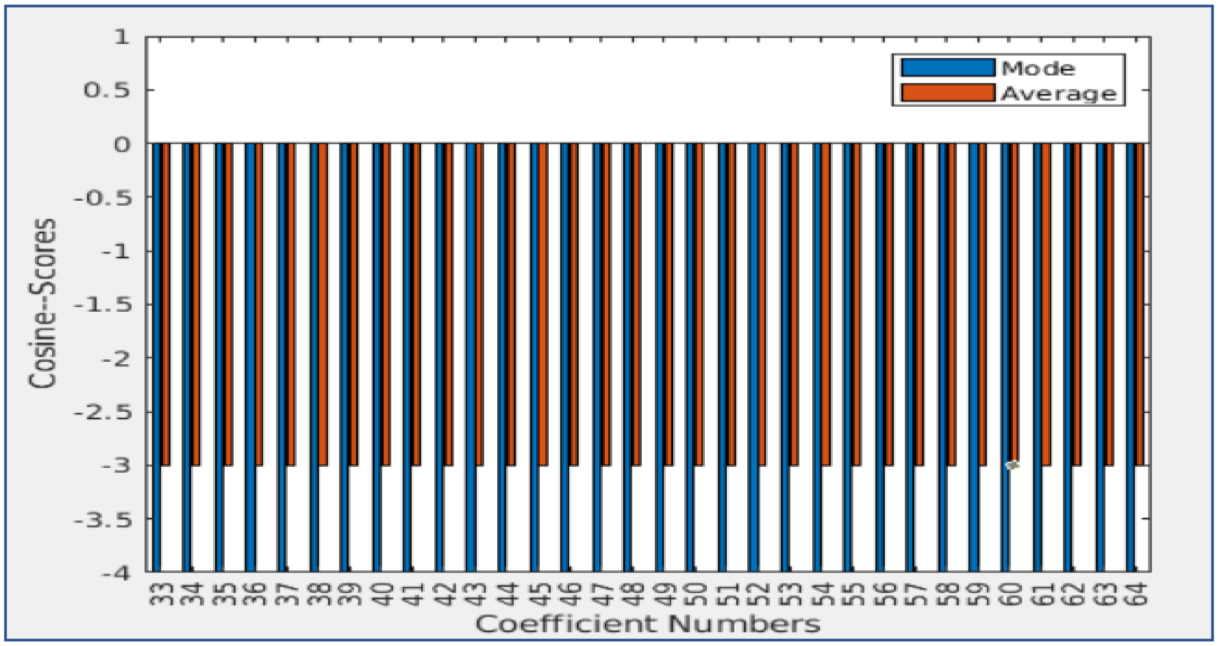


Figure 5.14: Cosine mode and average threshold values for coefficient feature vectors from 33 to 64 on Utrecht dataset. The Y-axis represents the normalized values, and the X-axis represents the coefficient numbers organized in a zig-zag traversal.

5.4 Average accuracy

We have calculated the average accuracy across 10 test sets each for Aberdeen and Utrecht ECVP. Each test set contains 50 random images; we have matched each against the images into the respective databases. We calculated the accuracy for each test over 50 images, and then the accuracy from each test set is averaged over 10 test sets. Table 5.1 below shows the average accuracy of the Utrecht dataset.

Table 5.1: The percentage average accuracy values for Utrecht

Coefficient Numbers	**Average (%)**	**Mode (%)**
1	0.194	0
2	0.094	0
3	0.088	0
4	0.086	0
5	0.084	0
6	0.082	0
7	0.08	0
8	0.076	0
9	0.048	0
10	0.014	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0

Table 5.2 below shows the average accuracy of the Aberdeen dataset.

Table 5.2: The percentage average accuracy values for Aberdeen

Coefficient Number	**Average (%)**	**Mode (%)**
1	14.8	0
2	4.4	0
3	2.4	0
4	1.6	0
5	1.2	0
6	1	0
7	1	0
8	0.6	0
9	0.2	0
10	0.2	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0

Chapter 6

Conclusions and future work

6.1 Conclusion

After analyzing the coefficients in the process of face matching, we observe how there are only a first few coefficients that contribute towards calculating an average and a mode threshold. The pattern of recurring NAN was interesting as it existed for the coefficients at the higher indices. Therefore, different quantization matrices may produce different results. It gives us an insight into the significance of the coefficients at the lower indices and the importance to choose a proper quantization matrix. This helps us to drop the coefficients at the higher indices without any worry as they do not play an essential role in decision-making. While fewer coefficients help us reduce the number of coefficient feature vectors involved in the process, this also allows us to store the data in a compact form which can be understood or recreated without complete coefficient information; this provides a reliable method to obfuscate the visual appearance of a face in an image. We also observed that Aberdeen produced lower average threshold values as compared to the Utrecht ECVP dataset. As well, several significant coefficients were dropped in the Utrecht ECVP as compared to Aberdeen; the reason may lie in the fact that the images in Utrecht ECVP were fewer than in Aberdeen.

The process of thresholding, which included the use of average and mode values of the coefficient feature vector similarity calculations, clearly indicates that the use of average values for thresholding is a better option compared to the use of mode value. The similarity functions cosine and correlation show similar results for the average threshold values. We observe that for cosine and correlation, the coefficients which actively participate in the decision-making span from 1 to 22, whereas for Manhattan, the coefficient number increases to 26 but not with a significant increase or difference in the value of the threshold score.

The average accuracy produced for average and mode thresholds reflects how mode as a threshold is not useful in this process. However, the average thresholds are not sufficient; thus, we require an improved threshold metric.

6.2 Future work

As exciting it was to observe the behavior of the DCT coefficients in the process of face matching while preserving the identity, it would be even more interesting to see its behavior at different quantization levels. We need to employ a better threshold method to obtain better average accuracy. We also need to experiment on a larger dataset to obtain more clarity about the patterns the coefficients exhibit. We want to extend the experiment to test the combination of the coefficients in matching the faces and preserving the privacy. Machine learning has become the backbone of computer society; therefore, in future work on these algorithms, we expect to include machine learning algorithms in the proposed method. As mentioned earlier, the quantization matrices affect the number of significant coefficients in the face matching process, and thus it would be interesting to calculate a quantization matrix for the process, which results in higher average accuracy and better privacy preservation.

Bibliography

[1] 10 August 2019. <pics.stir.ac.uk>.

[2] Avidan, Shai and Butman, Moshe. "Blind vision." *European Conference on Computer Vision*. Springer, 2006. 1-13.

[3] Avidan, Shai and Butman, Moshe. "Efficient methods for privacy preserving face detection." *Advances in Neural Information Processing Systems*. 2007. 57-64.

[4] Coste, Arthur. 2012. 26 June 2019. <http://www.sci.utah.edu/~acoste/uou/Image/project3/ArthurCOSTE_Project3.pdf>.

[5] Jain, Anil K. *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall,, 1989.

[6] Jin, Xin and Yuan, Peng and Li, Xiaodong and Song, Chenggen and Ge, Shiming and Zhao, Geng and Chen, Yingya. "Efficient privacy preserving Viola-Jones type object detection via random base image representation." *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017. 673-678.

[7] Ponnmoli, KM and Selvamuthukumaran, Dr S. "Analysis of face recognition using Manhattan distance algorithm with image segmentation." *International Journal of Computer Science and Mobile Computing* 3 (2014): 18-27.

[8] Raid, AM and Khedr, WM and El-Dosuky, Mohamed A and Ahmed, Wesam. "JPEG image compression using discrete cosine transform-A survey." *arXiv preprint arXiv:1405.6147* (2014).

[9] Ren, Zhongzheng and Jae Lee, Yong and Ryoo, Michael S. "Learning to anonymize faces for privacy preserving action detection." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. 620—636.

[10] Watson, Andrew B. "Image compression using the discrete cosine transform." *Mathematica Journal* 4.1 (1994): 81.

[11] Zhu, Zhenyao and Luo, Ping and Wang, Xiaogang and Tang, Xiaoou. "Deep learning identity-preserving face space." *Proceedings of the IEEE International Conference on Computer Vision*. 2013. 113-120.