Mathematical Techniques for Face Recognition

In this section present a high-level view of various mathematical principles and techniques are used in *face recognition* research. Though these topics are not in the scope of this course, we present at a high-level for a high level understanding.

Processing the pixel matrix of an image, extracting the features and creating a vector has been studied for a long time by mathematicians. The following work is a reference article to have a high-level understanding of various approaches.

> *Mejda Chihaumi, Akram Elkefi, Wajdi Bellil and Ben Amar, University of Sfax,* **A survey of 2D Face Recognition Techniques,** *Computers, 2016,5,21 (28 pages).*

This paper presents a detailed survey of well-known methods and principles.

They list out three categories of work on Face recognition.

1. Global Face approach
2. Local features approach
3. Hybrid approach

2.1 Global Face approach

In face recognition, there are two main types of approaches: linear and non-linear. These terms describe the way the algorithm "looks" at faces to understand and recognize them.

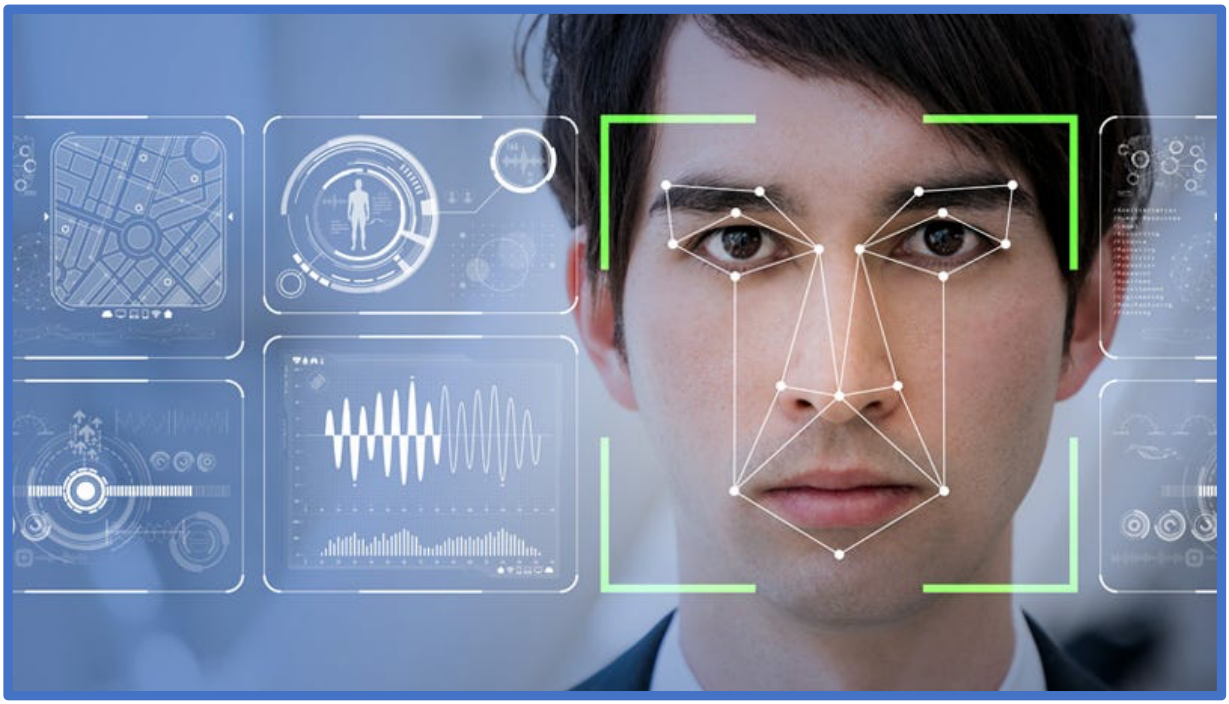

Figure 2.1 Global Face approach

Linear Approach

A linear approach in face recognition tries to separate faces by drawing straight lines (or flat planes) in a way that makes it easy to tell one face apart from another. Linear methods assume that faces can be represented and separated using straightforward mathematical patterns, like lines or flat surfaces.

Think of it like drawing a straight line between two groups of points on a graph. If all faces fit this simple pattern, a linear approach works well and is fast to calculate.

Example of Linear Approach: Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) are popular linear methods. They break down faces into basic "patterns" or "directions" that help separate one person's face from another. Linear approaches are good for situations where faces have a predictable structure, but they struggle with complex cases where faces look very different from standard patterns.

Non-linear Approach

A non-linear approach doesn't limit itself to straight lines or flat surfaces. Instead, it uses curves, complex shapes, or flexible boundaries to separate faces. This approach is more adaptable and can capture more complex patterns, like the subtle differences in how people smile or frown.

Imagine you have a wavy line or a bendy surface that adjusts to fit around groups of points on a graph. A non-linear approach can follow complex shapes to separate different faces, even when they don't fit a simple pattern.

Example of Non-linear Approach: Neural networks and support vector machines with non-linear kernels are examples of non-linear methods. They can handle more complex variations in faces, like differences in lighting, pose, or expressions, making them more accurate in real-world situations.

Linear approaches are faster, simpler, and work well when faces don't vary too much, like in controlled environments (e.g., ID photos).

Non-linear approaches are better for complex, real-life scenarios where faces come in all kinds of variations — different lighting, angles, expressions, etc. They're more accurate but need more computing power and data.

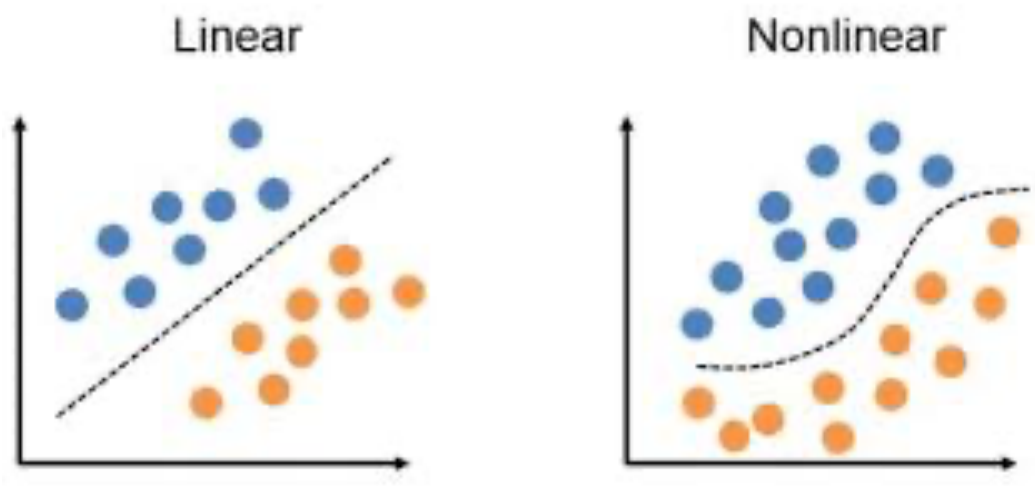

Figure 2.2 Linear and Nonlinear approaches

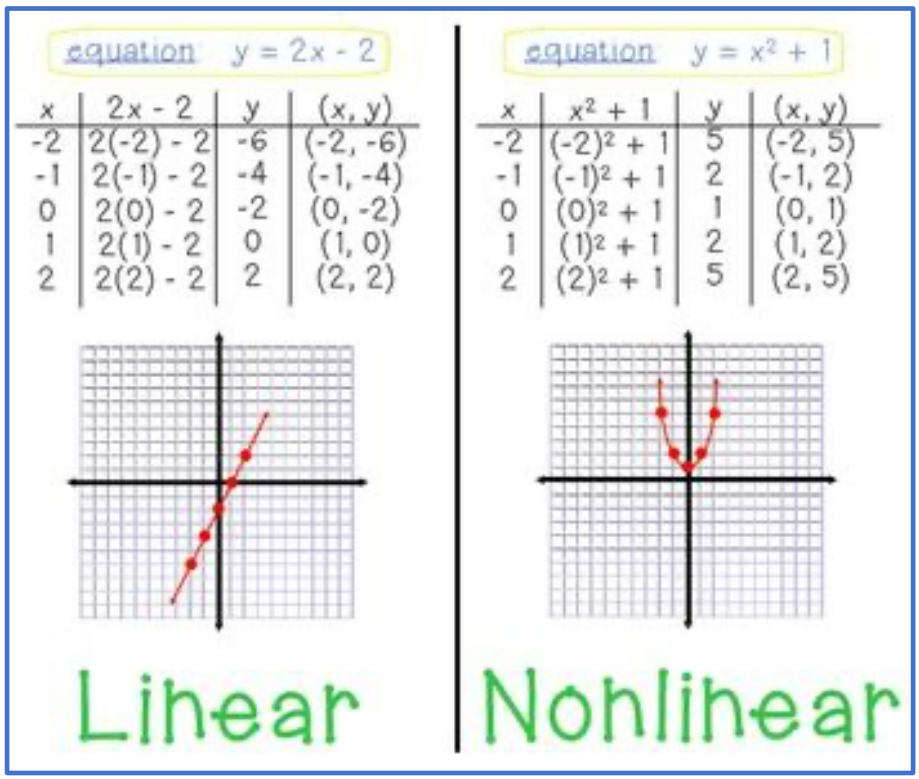

Figure 2.3 Linear and Nonlinear approaches

In Simple Terms think of a linear approach like drawing straight lines to divide faces into groups. It's quick and works fine if faces look somewhat similar. Non-linear approaches, however, draw curvy, flexible lines that adjust to fit all kinds of faces, even in tricky situations. They're more powerful and accurate, especially for recognizing faces in real-life conditions where there are more differences and variations.

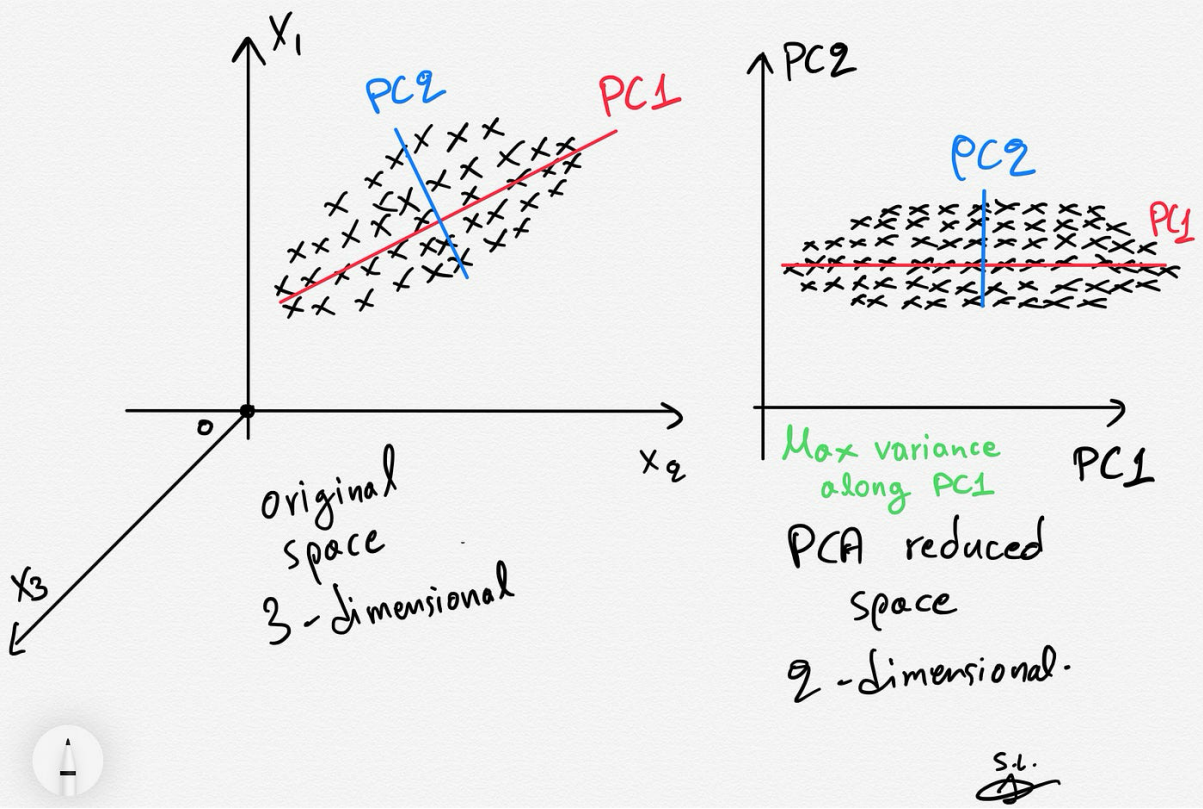

Figure 2.4 Three and two dimensional approaches

In global face approach, researchers have worked on linear and non-linear techniques. The following are Linear techniques:

- Eigen vectors and Eigen Faces
- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Multidimensional Scaling (MDS)

- Non-negative Matrix Factorization (NMF)
- Liner Discriminant Analysis (LDA)
- Gabor Wavelets

Researchers have also used the following non-linear techniques for Global Face Recognition approach.

- Kernel Functions
- Support Vector Machines (SVM)
- Nearest Manifold Approach

2.2 Local Feature Approaches

Local approaches study only same local features of the face such as mouth, eyes etc. Local methods are also called *Feature Based Methods*. There are two categories of research in this approach.

1. Interest Point based
2. Local Appearance based

In the interest point based approach we first detect the point of interest and then extract features around the point.

In local-appearance based methods, the face is divided into smaller regions or patches. From each patch, the local features are extracted.

2.3 Hybrid Approaches

To make the algorithm more efficient people have tried hybrid approaches combining some prominent features of global face and some local ones. Figure 2.5 gives a high-level view of all the key methods.

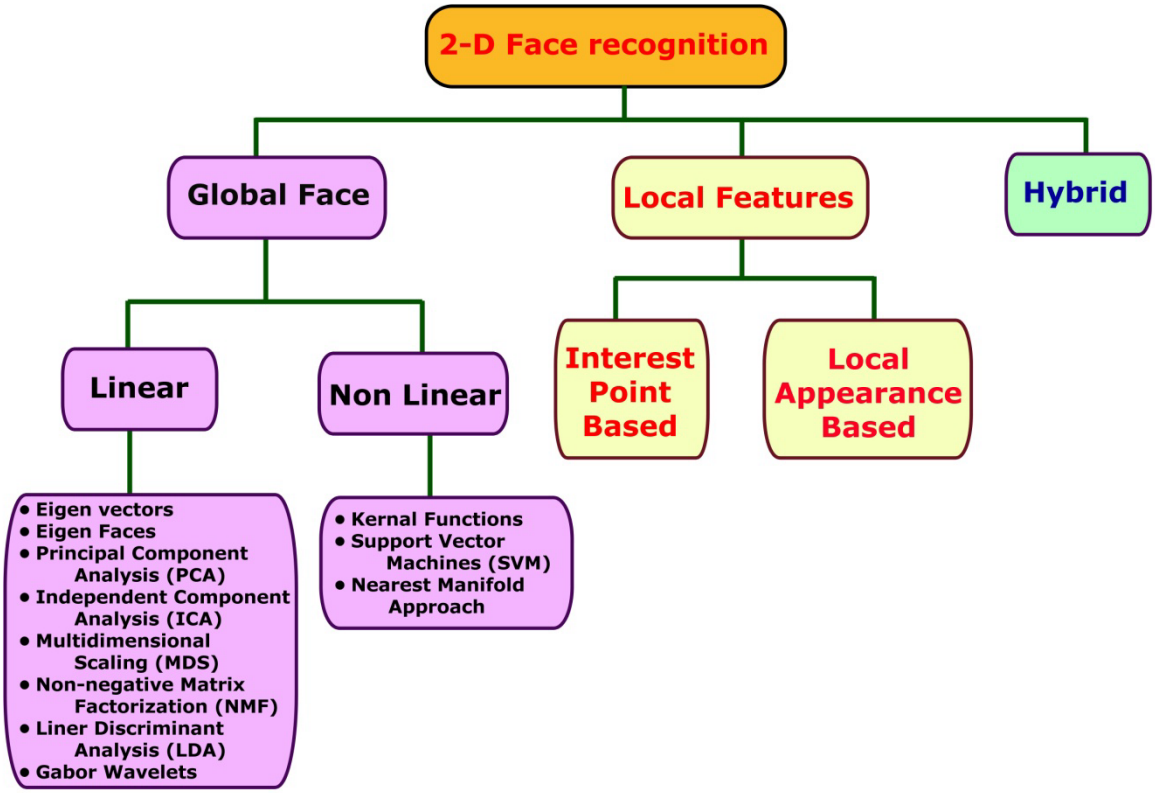

Figure 2.5 2D Face Recognition Algorithms

2.4 Eigenvectors and Eigenvalues

Eigenvectors and eigenvalues are fundamental concepts in linear algebra, with applications in numerous fields like physics, engineering, and computer vision. For a given square matrix A, an eigenvector X and eigenvalue λ satisfy the equation:

$$AX = \lambda X$$

Here:

- X is an eigenvector, which is a non-zero vector that, when the matrix AAA is applied to it, results in a scalar multiple of itself.
- λ is the eigenvalue associated with X, representing the factor by which the eigenvector is scaled during this transformation.

Eigenvalues and eigenvectors are important in many applications because they provide insights into the structure of a matrix. For example, they allow us to analyze linear transformations in systems, make matrix decompositions, and solve differential equations.

Eigenfaces

Eigenfaces are an application of eigenvectors in the field of computer vision, specifically in face recognition. This technique represents faces as a combination of *eigenfaces* — the

principal components that capture the most significant variations in facial features among a set of training images.

In the context of face recognition, a dataset of face images is represented as a matrix. By applying *Principal Component Analysis (PCA)* to this matrix, we can identify the eigenvectors (or principal components) that capture the main features of faces. These eigenvectors are termed *eigenfaces.*

Each face in the dataset can then be approximated by a combination of these eigenfaces, allowing for dimensionality reduction and efficient face recognition. When a new face is introduced, it can be projected onto the eigenface space and matched with known faces based on the closeness of its projection.

2.5 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while preserving as much variance (or information) as possible. In high-dimensional data, PCA helps by finding patterns, compressing data, and removing noise, making it particularly useful for data visualization, preprocessing in machine learning, and image compression.

PCA transforms the data into a new coordinate system where the greatest variance lies along the first principal component (PC1), the second greatest variance lies along the second principal component (PC2), and so on. Each principal component is an eigenvector of the covariance matrix of the dataset, representing an "axis" in the new space. These principal components are ordered by the amount of variance they capture from the data. By selecting only the top few components, we effectively reduce the dimensions of the dataset.

The process of PCA involves several steps. First, the mean of each feature is subtracted from the dataset to center the data around the origin. This makes it easier to analyze the directions of variance without the data being biased by its mean. Next, the covariance matrix of the centered data is computed to understand the relationships between features. Then, the eigenvalues and eigenvectors of the covariance matrix are calculated. The eigenvectors are the directions (principal components) of the data's greatest variance, and the eigenvalues represent the magnitude of variance in each direction.

Once the principal components are determined, the original data can be projected onto this new basis. For example, if a dataset originally had 100 dimensions, it can be transformed to a new set of 100 principal components. By keeping only the top components that capture, say, 95% of the variance, we can reduce the dataset to fewer dimensions, such as 10 or 20, while still retaining most of its informative content.

PCA has numerous applications in fields like image processing and computer vision. For instance, in face recognition, PCA is used to identify and capture key features in a dataset of face images by transforming them into a "face space" defined by eigenfaces. PCA is also widely used as a preprocessing step in machine learning, where reducing dimensionality can help models train faster and improve generalization by minimizing noise and redundant features.

However, PCA has limitations. It assumes that the directions with the most variance are the most informative, which may not always be true. PCA also performs a linear transformation, which means it may not capture complex, non-linear relationships within the data. Despite these limitations, PCA remains a foundational technique in data analysis due to its simplicity, efficiency, and ability to reveal structure in high-dimensional data.

2.6 Independent Component Analysis (ICA) Explained

Independent Component Analysis (ICA) is a statistical and computational technique used to separate a multivariate signal into additive, independent components. ICA is especially useful in fields like signal processing, neuroimaging, and financial analysis, where it is often used to extract independent signals from mixed sources — a classic example being the "cocktail party problem," where ICA helps to separate individual voices from a recording of a conversation.

Unlike PCA, which seeks to maximize the variance in the data, ICA assumes that the data is composed of independent sources mixed together and tries to separate them by finding statistically independent components. Independence here means that the components do not contain information about each other, and their statistical properties are unrelated.

The process of ICA involves representing the observed data as a linear combination of unknown independent source signals. In contrast to PCA, ICA does not rely on maximizing variance; rather, it maximizes statistical independence, often measured through higher-order statistics, such as kurtosis or mutual information. This difference is crucial, as ICA can extract non-Gaussian sources from mixtures, while PCA is limited to uncorrelated, often Gaussian sources.

Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a way to visualize the similarities or distances between different items in a simple, easy-to-understand map, usually in 2D or 3D. Imagine you have a list of cities and know the distances between each pair of cities, but you don't have a map. MDS helps create a map where the cities are positioned so that the distances between them on the map closely match the real distances.

In simpler terms, MDS takes complex information about how close or similar things are to each other and arranges them in a way we can easily see. This is useful in many fields, like psychology, biology, and marketing, to help us understand relationships and patterns that might not be obvious in raw data. For example, it could help group similar types of customers based on their preferences or show how different products relate to each other based on customer ratings.

Non-negative Matrix Factorization (NMF) is a technique in data analysis and machine learning used to break down data into parts, specifically focusing on cases where the data and the factors are non-negative (i.e., no negative numbers). This is particularly useful for datasets like images, where pixel values represent intensities and are naturally non-negative.

Imagine that NMF works like a set of "face puzzle pieces." Each face image in a collection is made up of different parts: eyes, nose, mouth, etc. NMF looks at all the faces in the collection and tries to find common parts — like puzzle pieces that make up different faces.

When you use NMF for face recognition, here's what happens:

1. **Learning Phase**: The system looks at a bunch of face images and learns a set of basic "face parts" that can be combined in different ways to create any face. For example, it might learn one piece that looks like an eye, another that looks like a nose, and another that looks like a mouth.
2. **Breaking Down Faces**: For each face, NMF breaks it down into a mix of these basic parts. Each face is represented as a unique combination of eyes, nose, mouth, and other parts. This way, each face has its own "recipe" based on these parts.
3. **Recognizing a New Face**: When a new face shows up, the system tries to match it by breaking it down into the same parts it learned before. Then, it compares this new face's "recipe" of parts to the stored recipes for all known faces. If it finds a close match, it recognizes the person.

So, in simple terms, NMF helps a face recognition system by breaking faces down into their basic parts, and then using these parts to recognize people based on how their faces are built. It's like having a puzzle where each person's face is a unique combination of common pieces.

Linear Discriminant Analysis (LDA) is a method used in statistics and machine learning to help find patterns in data. It's especially useful for recognizing different groups or categories, like separating apples from oranges in a basket of fruits. LDA tries to find the best way to separate groups by projecting data onto a new line (or space) that maximizes the differences between groups. This helps make distinctions clearer and improves classification.

Assume that we have points on a graph representing two groups, say Group A and Group B. These groups might overlap, making it hard to separate them. LDA finds a new "line" or "direction" where the groups are as separated as possible. This new line is chosen so that:

- Distance between groups is maximized (groups are far apart).
- Distance within each group is minimized (points in the same group are closer together).

LDA essentially creates a new "view" of the data where it's easier to tell the groups apart.

In face recognition, LDA helps distinguish between different people's faces by finding patterns that separate one person's face from another. Here's how it works in simple terms:

1. **Training with Known Faces**: First, the system is given a set of face images, where each face belongs to a specific person (for example, faces of Person A, Person B, and Person C). LDA analyzes these faces and finds the features (like distance between eyes, shape of jawline) that best separate each person.
2. **Creating a New Space for Faces**: Using LDA, the system creates a new "space" or "direction" where faces of the same person are grouped closer together, and faces of different people are far apart. This makes it easier to tell faces apart because each person's face now has a unique "signature" in this LDA space.

3. **Recognizing New Faces**: When a new face image is shown to the system, it projects this face onto the LDA space and checks its position. If the face lands near the "signature" of Person A, it recognizes it as Person A. If it lands near Person B's signature, it recognizes it as Person B, and so on.

LDA is Useful for Face Recognition due to the following reasons.

- *Better at Separating Classes*: Unlike other techniques like PCA, which only captures the main patterns in faces, LDA focuses on the patterns that specifically distinguish one person's face from another.
- *Enhanced Accuracy*: By concentrating on the differences between people's faces, LDA reduces the chance of confusing similar-looking faces, making face recognition more accurate.
- *Efficient Data Representation: L*DA represents each face with fewer features, reducing the amount of data the system needs to process. This makes it faster and more efficient.

In simple terms, LDA helps a face recognition system by finding the "directions" that make it easy to tell people apart, making it a powerful tool for accurate, quick face identification

Gabor Wavelets are mathematical functions used to analyze textures and features in images, especially for patterns that have a particular orientation and frequency. They are often used in image processing because they are good at capturing both spatial (position-based) and frequency (detail-based) information. Imagine a Gabor wavelet as a filter that looks for specific patterns in an image, like lines, edges, or textures.

2.7 Gabor Wavelets

Gabor wavelets resemble small waves, which vary in orientation and frequency. When applied to an image, they highlight parts that match their shape and orientation. For example, if a Gabor wavelet is set to detect horizontal lines, it will respond strongly (i.e., give a high value) wherever it finds horizontal lines in the image, but not so much to vertical lines.

Gabor wavelets are particularly effective because:

They capture both edges and textures at different orientations (like vertical, horizontal, or diagonal lines).

They focus on specific scales, meaning they can detect both large and small features.

How Gabor Wavelets Are Used in Face Recognition

In face recognition, Gabor wavelets help extract unique patterns in a face image. Here's how they're typically used:

Applying the Wavelet Filters: When we want to recognize a face, we apply a set of Gabor wavelet filters with various orientations and frequencies across the image of the face. This produces several filtered images, each highlighting specific features, such as edges around the eyes, nose, or mouth.

Creating a Gabor Feature Map: The filtered images are combined to form a "Gabor feature map." This map acts as a unique "fingerprint" of the face, capturing essential details in different directions (like horizontal and vertical features) and scales (large and small features).

Feature Extraction and Matching: The Gabor feature map is then used to represent the face in the recognition system. When a new face needs to be identified, its Gabor features are extracted and compared to stored Gabor feature maps of known faces. If the new face's map closely matches one of the stored maps, the system can recognize the person.

Gabor Wavelets Are Effective for Face Recognition because of the following reasons:

Capturing Local Details: Faces have many small, distinctive features, like the curve of eyebrows or the contour of lips. Gabor wavelets excel at capturing these local details.

Robust to Changes in Lighting and Expression: Unlike simpler methods, Gabor wavelets are less sensitive to variations in lighting or slight changes in facial expression, which helps improve recognition accuracy.

Orientation-Specific Information: Since Gabor wavelets are applied at different angles, they capture detailed information on the structure of the face, making it easier to distinguish between individuals.

In simple terms, Gabor wavelets help face recognition systems by breaking down a face into specific patterns that capture unique features, making it easier for the system to tell different faces apart.

2.8 Kernel Functions

Kernel functions are mathematical functions used in various machine learning and statistical applications, particularly in algorithms like Support Vector Machines (SVM), Gaussian Processes, and kernelized principal component analysis. In the context of face recognition, kernel functions play a crucial role in transforming data into higher-dimensional spaces, making it easier to separate different classes or patterns.

A kernel function computes a similarity measure between two data points in a potentially infinite-dimensional feature space without explicitly mapping them into that space. This is known as the "kernel trick." Some common types of kernel functions include:

1. **Linear Kernel**
2. **Polynomial Kernel**
3. **Radial Basis Function (RBF) or Gaussian Kernel**:
4. **Sigmoid Kernel**:

How Kernel Functions Are Used in Face Recognition

1. **Feature Mapping**: Face recognition typically involves extracting features from images, such as texture, shape, and color. Kernel functions help in transforming these features into a higher-dimensional space where it may be easier to find a separating hyperplane for different faces.
2. **Dimensionality Reduction**: Techniques like Kernel Principal Component Analysis (KPCA) utilize kernel functions to reduce dimensionality while preserving essential characteristics. This is helpful in face recognition as it can help focus on the most relevant features.
3. **Classification**: In classification algorithms like SVM, kernel functions enable the algorithm to learn complex boundaries between different classes (e.g., different faces). The SVM can find a hyperplane that maximizes the margin between classes in this transformed space.
4. **Similarity Measures**: Kernel functions can be interpreted as similarity measures between face representations (feature vectors). A higher kernel value indicates a greater similarity, which can help in identifying matching faces.

Practical Example

1. **Data Preparation**: Collect a dataset of face images and extract relevant features (e.g., using techniques like Local Binary Patterns or Histogram of Oriented Gradients).
2. **Kernel Application**: Choose an appropriate kernel function (like RBF) and apply it in a machine learning algorithm such as SVM to classify the images based on their extracted features.
3. **Training and Testing**: Train the model using labeled data (images with corresponding identities) and test it on unseen images to evaluate its performance.
4. **Recognition**: For new face images, extract features and apply the kernel function to compute similarities with the trained model to identify or verify the individual's identity

.

By leveraging kernel functions, face recognition systems can achieve higher accuracy and robustness, particularly in challenging conditions like variations in lighting, expression, and pose

2.8 Support Vector Machines (SVM) in Face Recognition

Support Vector Machines (SVM) are a set of supervised learning methods widely used for classification and regression tasks. Developed in the 1990s, SVM is particularly effective in high-dimensional spaces, making it suitable for various applications, including image recognition. The core idea behind SVM is to find a hyperplane that best separates the data points of different classes in a multi-dimensional feature space. The following is the approach used usually.

1. **Data Representation**: SVM takes a set of training examples, each represented as a vector in an n-dimensional space. Each example belongs to one of two classes.
2. **Hyperplane Selection**: The objective of SVM is to identify a hyperplane that maximizes the margin between the two classes. The margin is defined as the distance between the hyperplane and the nearest data point from either class. These nearest points are called support vectors, as they directly influence the position and orientation of the hyperplane.
3. **Kernel Trick**: To address non-linear relationships between classes, SVM employs the kernel trick. This technique transforms the original feature space into a higher-dimensional space where a linear hyperplane can effectively separate the classes. Common kernel functions include linear, polynomial, and Radial Basis Function (RBF) kernels.
4. **Classification**: Once the optimal hyperplane is determined, SVM can classify new data points based on which side of the hyperplane they fall on. The decision boundary can thus categorize previously unseen data into the appropriate class.

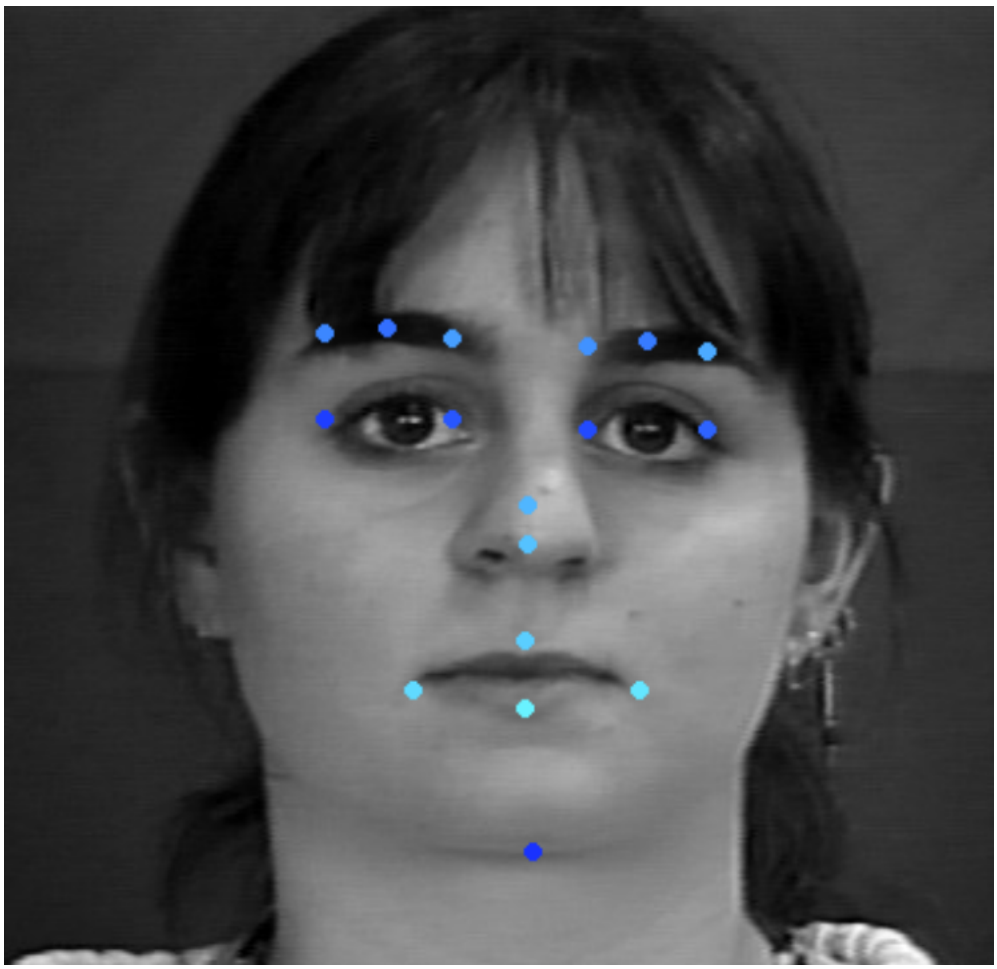

Figure 2.6 SVM for Face Recognition

Face recognition is a challenging task due to the high variability in facial appearances caused by factors such as lighting, pose, and expression. SVM has emerged as a robust method for tackling these challenges due to its effective handling of high-dimensional data.

1. **Feature Extraction**: The first step in face recognition involves extracting meaningful features from face images. Techniques such as Principal Component Analysis (PCA) or Local Binary Patterns (LBP) can be employed to reduce dimensionality while retaining essential facial characteristics. The resulting features form a feature vector for each face image.
2. **Training the SVM Model**: With a labeled dataset of feature vectors representing various individuals' faces, SVM is trained to classify these vectors. During training, the SVM identifies the support vectors and determines the optimal hyperplane that separates different individuals' face representations.
3. **Recognition Phase**: For recognizing a new face, the same feature extraction methods are applied to obtain its feature vector. The SVM model then classifies this vector by determining which side of the hyperplane it lies on, effectively identifying the individual or determining whether the face matches any of the known faces in the training set.
4. **Performance Evaluation**: The effectiveness of the SVM model in face recognition can be assessed through various metrics, such as accuracy, precision, recall, and F1 score. Cross-validation techniques can also be utilized to ensure the model's robustness against overfitting.

Support Vector Machines provide a powerful framework for face recognition tasks, combining high-dimensional data handling with effective classification capabilities. By leveraging techniques like kernel functions and feature extraction, SVM can achieve high accuracy and reliability, making it a popular choice in modern face recognition systems.

2.9 Nearest Manifold Approach

The **Nearest Manifold Approach** is a method used in machine learning and computer vision, particularly for tasks such as face recognition. This approach is based on the idea that high-dimensional data (like images) often lie on or near lower-dimensional manifolds. Understanding this concept is crucial for effectively recognizing faces in images where variations (such as changes in lighting, pose, or expression) may occur. The following are the key Concepts of the Approach

1. *Manifold Learning**:*** In manifold learning, it is assumed that high-dimensional data can be represented as low-dimensional manifolds. For instance, in face recognition, the variations in facial expressions, lighting, and orientation can be considered as lying on a manifold in the high-dimensional space of image data.
2. *Local Neighborhoods*: The nearest manifold approach focuses on the local neighborhoods of data points. Instead of treating every data point independently, it looks for the nearest neighbors in the manifold structure, which helps to capture the underlying geometry of the data more accurately.
3. *Distance Measures*: The approach often employs distance measures that are sensitive to the manifold structure. This can involve techniques like geodesic distances, which

account for the true geometric distances on the manifold rather than simple Euclidean distances.

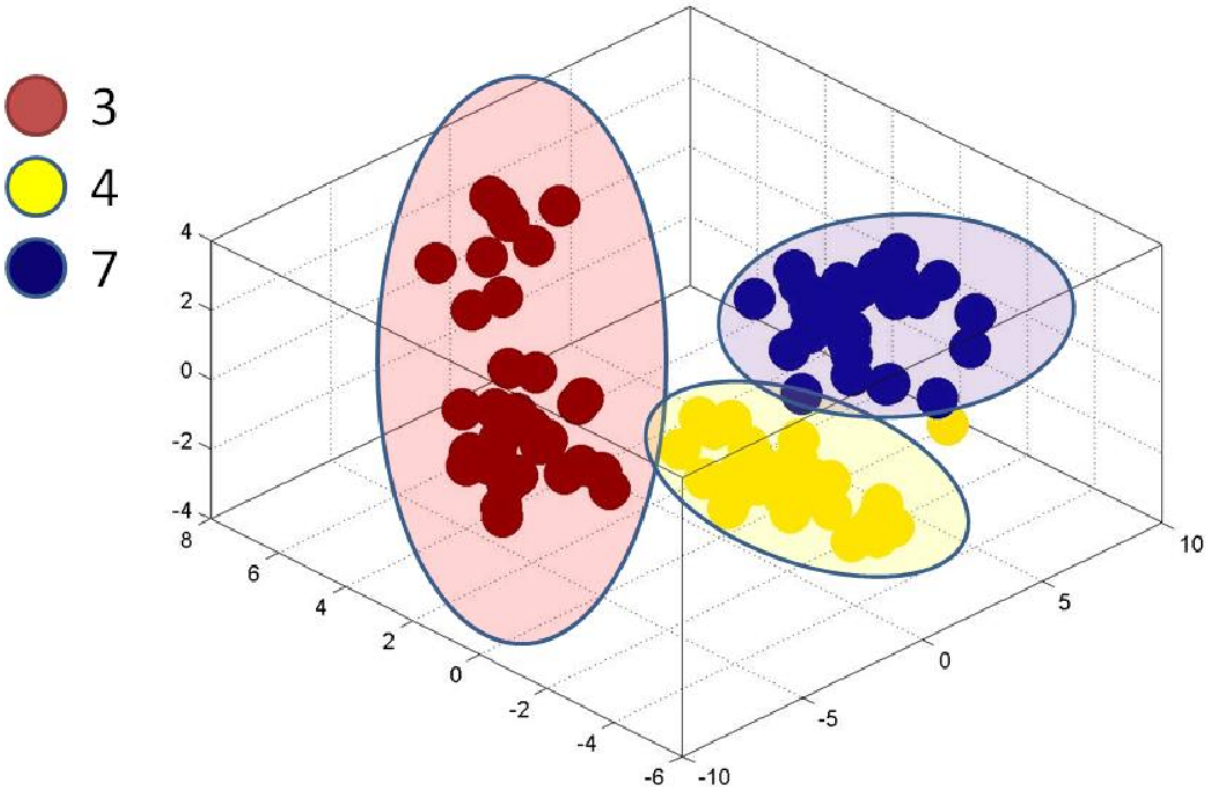

Figure 2.7 Manifold Approach

In face recognition, we use the nearest manifold approach as follows:

1. **Feature Extraction**: Similar to other methods, the nearest manifold approach begins with extracting relevant features from face images. Techniques such as PCA, Independent Component Analysis (ICA), or Local Binary Patterns (LBP) are commonly used to represent faces in a reduced-dimensional space.
2. **Manifold Learning**: Once the feature vectors are obtained, manifold learning techniques (such as Locally Linear Embedding (LLE) or Isomap) can be applied to model the manifold structure of the face data. This step aims to capture the intrinsic geometric relationships between different facial representations.
3. **Finding Nearest Neighbors**: During the recognition phase, for a new face image, its feature vector is computed, and the nearest neighbors in the learned manifold are identified. The recognition is based on the class labels of these neighbors. If most of the nearest neighbors belong to a particular individual, that identity is assigned to the new face.
4. **Classification**: Depending on the number of neighbors considered, different classification strategies can be employed. For instance, a majority voting scheme may be used, where the class with the most votes among the nearest neighbors is chosen as the final classification.
5. **Handling Variability**: One of the significant advantages of the nearest manifold approach is its robustness to variations in face images. Since it relies on local

structures, it can better handle changes in lighting, expressions, and angles, making it suitable for real-world applications.

The Nearest Manifold Approach is a powerful technique for face recognition, leveraging the intrinsic geometric properties of data in high-dimensional spaces. By effectively modeling the underlying manifolds and utilizing local neighborhood structures, it enhances the ability to accurately identify faces under varying conditions. This approach has significant implications for improving the performance of face recognition systems in practical applications.

2.10 Face Recognition based on Nose, Eyes and lips

Face recognition based on facial features such as the nose, eyes, and lips is a technique widely used in biometric identification. These features are often selected because they remain relatively stable and unique across individuals, making them reliable for distinguishing one person from another. Here's an overview of how each of these features plays a role in facial recognition:

1. **Eyes**: The eyes are crucial for facial recognition as they are symmetrically positioned and maintain a consistent spatial relationship with other facial features. The distance between the eyes, their size, shape, and the pattern of the iris are unique for each individual. Eye tracking also helps in determining the gaze direction, which adds more context to face recognition applications.
2. **Nose**: The shape and structure of the nose are also unique, with specific measurements like the length, width, and nostril shape varying widely among individuals. The nose is central on the face and is less affected by changes in expression, which makes it a stable reference point for recognition algorithms. Some advanced techniques measure the three-dimensional structure of the nose to enhance accuracy.
3. **Lips**: While the lips can change shape depending on expressions, they still carry distinctive characteristics such as the width, fullness, and contour of the lip line. For face recognition, the system may use the overall mouth shape, especially in a neutral expression, to help identify individuals. Lips can also provide cues in lip-reading applications, aiding in voice recognition or surveillance in noisy environments.

In practice, a face recognition system typically maps out these features and uses their spatial relationships to create a unique "faceprint" or mathematical representation of a person's face. This faceprint can then be matched against a database to identify or verify an individual.

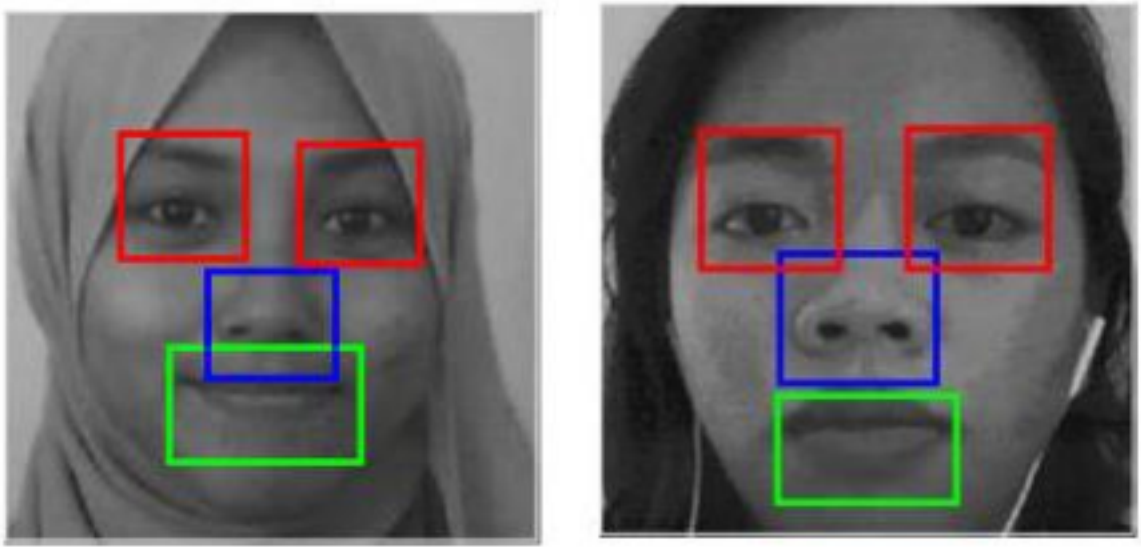

Figure 2.8 Face Recognition based on Nose, Eyes and lips

2.9 Part based Face Recognition - Eyes