N°d'ordre: ....

République Algerienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la Recherche scientifique

Université Djillali Liabès de Sidi Bel Abbès
Faculté de Génie Electrique
Département d'Electronique
Laboratoire RCAM

Thèse de Doctorat

en Sciences

Filière : Electronique

Spécialité : Traitement d'images

Par

Melle MEKAMI Hayat

Dimension reduction methods and artificial learning for the facial pose estimation

Soutenue le 07 Février 2022 devant le jury :

Professeur	Nassredine Taleb	UDL SBA	Président du jury
Professeur	Abdennacer Bounoua	UDL SBA	Directeur de thèse
Professeur	Miloud CHIKR EL MEZOUAR	UDL SBA	Examinateur
Professeur	Abdellatif RAHMOUN	ESI SBA	Examinateur
Directeur de Recherche	Youcef Bentoutou	CDS Oran	Examinateur
Directeur de Recherche	Moussa Sofiane Karoui	CTS Oran	Examinateur

Année Universitaire : 2021 - 2022

To My parents

To Everyone who is pleased with my success…

Acknowledgments

This thesis is the fruit of long years of efforts and work, and it wouldn't have been done without the support of many people whom I would like to thank through these lines.

First of all, I would like to express my deepest gratitude and appreciation to my supervisors, Professor. Abdennacer Bounou.

I wish to thank Professor Nassredine Taleb, who honoured me to chair the jury for this thesis. I would also like to thank Professor Miloud CHIKR EL MEZOUAR, Professor Abdellatif RAHMOUN, Mr. Youcef Bentoutou, and Mr Moussa Sofiane Karoui, for their acceptance to examine this work.

I am greatly indebted to Mr S.Benabderrahmane for all his guidance, support and inspirational ideas, from choosing the subject of the thesis to the publication of the article. I would like to thank Mrs S.Bentaieb (USTO-MB, Oran) a lot for her efforts with me in the beginning of my work. And as everyone knows that the beginnings are difficult, so I thank her for her purposeful scientific discussions which produced many ideas that influenced this work. Special thanks to those two people because they motivated me and inherit the spirit of challenge in me to continue and strive to do the best all the time.

I cannot fail to introduce many thanks to the Eskafi family, especially Fathi Eskafi (fathieskafi@gmail.com), who proofread the language of the thesis.

I indebted in gratitude to Mrs K.Sadeddine (USTHB, Algiers), who contributed to the scientific review of the thesis and the presentation of proposals and corrections that contributed to the thesis enrichment and scrutiny. I also thank her for her moral support at the moments of vulnerability and listlessness.

I sincerely would like to thank those cherished people who mean to us a lot, who stand beside me throughout the long nights of troubles, to all those who motivated me to continue in my journey, who were the first to be happy for me at moments of joy, to those whose prayers knocked at the door of heaven repeatedly, and whose hearts stirred with love at all times.

Last but not least, I am extremely grateful to M.Boutkhil, L.Mostari, K.Mechach (Oran1 University), for their dedicated endeavour and their contribution to finalizing this work. I have no words to show my gratitude for all.

Abstract

Head pose estimation is one of the most important of research areas in computer vision. Even if significant progress has been attained, and impressive number of contributions and techniques have been suggested, the research is still opened for raising performances. In this context, we present approaches to create a powerful model to estimate the head orientation.

Modern computer systems accumulate data at an almost unimaginable rate and from a huge variety of sources. Classical statistical analysis methods cannot analyse and deal with these enormous amounts of data. Data mining is a discipline that consists a family of tools that involves the statistics rules and computer science to handle and analyse this information to become meaningful and useful.

Time series is one of the most common and a special kind of data in data mining. The processing and extracting features from time series require reducing dimensionality through time series representation. Time series data can be represented and converted to discrete sequence data using an approach known as a symbolic aggregate approximation (SAX). This representation is a way to explore hidden information in time series. Thus, it can rely on this information to find the correlation between data, as well as, it can perform the clustering and classification tasks.

This thesis leverages data mining technologies to extract head pose information from face images to build a classification model that infers the head pose orientation. To reach this purpose, we propose the SAX2FACE approach, an effortless and efficient alternative solution that relies on a time series dimensionality reduction method (SAX method) to address the problem of head pose rotation. We have mapped face images into a one-dimensional vector as time series using the Peano-Hilbert and Sweep space-filling curves. These numerical series are then converted to symbolic sequences through symbolic aggregate approximation (SAX).

Our first objective is to highlight the usefulness of using powerful symbolic data mining techniques to classify face poses in any database, and thus getting effective symbolic distances for classification purposes. We have resorted to classic classifiers such as K-means, KNN, and SVM to classify frontal vs. profile face poses. Besides, we have tackled the illumination changes problem. While we have proposed to overcome these problems by processing the input image with the gradient image and the Local Binary Pattern (LBP) combined with dynamic morphological quotient image (DMQI-LBP), which are robust descriptors to changes in illumination. The results of these experiences have shown that our approach is robust and allows us to separately classify the poses even in degraded conditions.

For the second objective, we have suggested combining the expressive power of deep learning with dimensionality reduction technique with time series representation of the images for learning the suitable features to estimate the head pose rotation with a large angles range (in yaw and pitch rotation).

Lately, deep learning has witnessed huge progress and has achieved exceptional resorted only to for head pose estimation models. However, it is computationally costly due to the high dimensionality of the parameters and the features that are calculated in training (the dimension of the weights is in the order of the billion). The dimension of these parameters progresses proportionally with the dimension of the input data. Spurred with this obstacle, we propose a new approach based on the use of dimensionality reduction with time series. The model emulates the sequence-to-sequence recurrent neural network that is introduced to deal with Machine Translation (NMT) model. Sequence-to-Sequence is a deep learning model that the encoder recurrent neural network encodes and learns the relationship between words of the source language to present it into a vector, and the decoder decodes it into a sequence of words in the desired language. Here, the positions of the faces are similar to the positions of the words in a sentence. Hence, analysing the positions of the faces by taking into account that the context is useful. This is why we are motivated by the use of Seq2Seq encoder–decoder in our implementation. We built a classifier of the head pose called SAX-RED, where the SAX symbolic sequences would be the input layer of the encoder, and the decoder generates the output sequences which present the labels of head poses.

ملخص

تقدير وضعية الرأس واحد من بين اهم مجالات البحث في مجال رؤيةتقدير وضع الرأس هو أحد ميادين البحث الهامة في الرؤية بواسطة الحاسوب ، ورغم أنه سُجلت مجهوات كبيرة و أن عددا مهما من المساهمات والتقنيات تم عرضها ، فلا يزال مجال البحث مشرعا على مصرعيه بغية تحسين و تطوير النتائج وفي هدا السياق ، نقترح في هدا العرض مقاربات تروم استحداث نموذجا فعالا لتقدير توجيه الرأس في الصورة

إن أنظمة الإعلام الآلي الحديثة تراكم بيانات بوتيرة لا يمكن تصورها وهي متعددة المصادر، والأنظمة الكلاسيكية في التحليل الإحصائي لا يمكنها مواكبة والتحكم في هذا الكم الهائل من المعلومات. فالتنقيب في البيانات هو التخصص الذي يقتضي عائلة أو مجموعة الوسائل التي تتضمن القواعد والقوانين الإحصائية الإعلام الآلي لمعالجة وتحليل هده المعلومات حتى تصير ذات دلالة و فائدة.

السلاسل الزمنية هي أكثر أنواع البيانات تناولا و الأكثر شيوعا في ميدان التنقيب في بيانات. إن معالجة و استنتاج الخصائص المتعلقة بالرسائل الزمنية تتطلب تقليص الحجم عن طريق تمثيل هده السلاسسل الزمنية.

بيانات السلاسل الزمنية يمكن تمثيلها وتحويلها إلى مقاطع معلومات منفصلة بتوظيف مقاربة معروفة تحت مسمى تقريب الطبقات الرمزية SAX . و يعتبر هذا التمثيل وسيلة لاستغلال المعلومات الخفية في السلاسل الزمنية التي نعتمدها لإيجاد الترابط ـ العلاقة ـ بين هذه المعلومات وبيانات وكدلك لإنجاز مهمات التجميع والتصنيف.

هذا العرض يستغل ميزات تكنولوجيا التنقيب في البيانات لاستخراج المعلومات حول وضعية الرأس انطلاقا من صور الوجه بهدف تكوين نموذج تصنيف يسمح باعتماد توجيه وضع الرأس.

هدفنا الأول هو توضيح فائدة توظيف تقنيات البحث في البيانات وبالتالي المسافات الرمزية الفاعل لغايات الترتيب والتصنيف ولهذا الغرض استعملت

فقط المصنفات الكلاسيكية من قبيل ـ k-means, KNN, SVM لتصنيف وضعية الوجوه الأمامية والجانبية. كما تناولت مشكلة تغير الاضاءة، ولتجاوز هذا العائق قمت بمعالجة أولية لصورة المدخلات حيث استعملنا الصورة التدرج و صورة الناتجة من ادماج Local Binary Pattern (LBP) مع dynamic morphological quotient image (DMQI-LBP) اللذان يمثلان آلية توصيف ممتازة حيت انهما لا يتأثر بتغييرات الإضاءة.

والنتائج المتحصّل عليها من التجارب تبيّن أن طريقة التي اقترحتها تحقق معدلات تصنيف جيدة و تسمح بتصنيف وترتيب الصور على حدة حتى في الظروف المتدهورة.

أما بالنسبة للهدف الثاني ، فنقترح المزاوجة بين أفضلية التعليم العميق وتقنيات التصغير الحجمي المضمون بتمثيل التكرارات الزمنية للصور لنعلم الخصائص لتقدير دوران وضع الرأس مع هامش كبير للزوايا. حديثا، عرف التعلم العميق جهودا كبيرة وحقق نتائج باهرة في ميدان تقدير وضعية الرأس لكن هذه النماذج تعتبر مكلفة نظرا لارتفاع حجم المعايير والخصائص التي تحسب خلال التعلم (حجم الأوزان يضاهي المليار) بُعدُ أو حجم هذه المعايير يتطور نسبيا مع حجم المُدخَلات . هذا العائق خفّزني لاقتراح مقاربة تعتمد على توظيف تصغير الأبعاد مع السلاسل الزمنية نموذجا يتجاوز مقطع لمقطع Seq2Seq الذي يحاكي نموذج التعلم العميق للترجمة حيث يشفِرُ المشفر ويعرف العلاقة بين كلمات اللغة المصدر لتمثيلها بشعاع وفكّ الشفرة، يفككها إلى مقاطع كلمات اللغة المرغوب فيها. هنا تكون وضعية الوجه مماثلة لوضعية الكلمات في الجملة ، إذن من المهم تحليل وضعيات الوجه مع الأخذ بعين الاعتبار السياق ، لهذا نحن تحفزت لاستعمال مشفر فاك الشفرة.

في هذه المرحلة النهائية لإنجاز النموذج. - Seq2Seq أنشأنا مصنف وضع الرأس يسمى (SAX-RED) أين تمثل المقاطع الرمزية (SAX) ، مدخل التشفير وفاك الشفرة يولِد مقاطع مُخرجَات تشكل هذه الأخيرة تسمية خاصة لوضع الرأس في الصور.

Résumé

L'estimation de la pose de la tête est l'un des domaines de recherche les plus importants en vision par ordinateur. Bien que des progrès importants ont été réalisés, et un nombre impressionnant de contributions et de techniques ont été proposées, la recherche reste ouverte pour améliorer les performances. Dans ce contexte, nous présentons dans cette thèse des approches qui visent à créer un modèle puissant pour estimer l'orientation de la tête.

Les systèmes informatiques modernes accumulent des données à un rythme presque inimaginable et provenant d'une énorme variété de sources. Les méthodes classiques d'analyse statistique ne peuvent pas analyser et manipuler ces immenses quantités de données. La fouille de données est une discipline qui consiste à un ensemble d'outils englobant des règles statistiques et l'informatique pour traiter et analyser ces informations afin qu'elles deviennent significatives et utiles.

Les séries chronologiques sont un des types de données les plus courants et les plus particuliers dans le domaine de la fouille de données. Le traitement et l'extraction de caractéristiques des séries temporelles nécessitent de réduire la dimensionnalité à l'aide de méthodes de représentation des séries temporelles. Les données de séries temporelles peuvent être représentées et converties en séquences de données discrètes en utilisant une approche connue sous le nom d'approximation d'agrégats symboliques (SAX). Cette représentation est un moyen d'explorer les informations cachées dans les séries temporelles. Dont, on peut se baser sur ces informations pour trouver la corrélation entre les données, ainsi que d'effectuer des tâches de clustering et de classification.

Cette thèse exploite les avantages de la technologie de fouille de données pour extraire des informations sur la pose de la tête à partir des images du visage dans le but de construire un modèle de classification qui infère l'orientation de la pose de la tête.

Notre premier objectif est de montrer l'utilité d'utiliser les techniques de fouille de données symboliques pour classer les poses du visage dans n'importe quel ensemble de données, et donc les distances symboliques efficaces à des fins de classification. Nous avons utilisé des classificateurs classiques tels que K-means, KNN, et SVM pour classer les poses du visage de frontale vs de profil. En outre, nous avons abordé le problème des changements d'illumination, nous avons proposé pour surmonter ces problèmes d'effectuer un prétraitement à l'image d'entrée, où nous avons utlisé l'image gradient et l'image traitée avec le Local Binary Pattern (LBP) combiné avec dynamic morphological quotient image

(DMQI-LBP), qui sont des descripteurs robustes aux changements d'éclairage. Les résultats de ces expériences ont montré que notre approche est robuste et permet de classifier séparément les poses même dans des conditions dégradées.

Pour le deuxième objectif, nous avons suggéré de combiner le privilège de l'apprentissage profond avec une technique de réduction de dimensionnalité assurée par la représentation de série chronologique des images pour apprendre les caractéristiques appropriées pour estimer la rotation de la pose de la tête avec un large plage d'angles (rotation de pan et tilt).

Récemment, l'apprentissage profond a vécu un considérable progrès et a atteint des performances exceptionnelles dans le domaine d'estimation de la pose de la tête. Cependant, ces modèles sont coûteux en calcul en raison de la haute dimensionnalité des paramètres et des caractéristiques qui sont calculés lors de l'apprentissage (la dimension des poids est de l'ordre du milliard). La dimension de ces paramètres se progresse proportionnellement à la dimension des données d'entrée. Motivés par cet obstacle, nous avons proposé une approche basée sur l'utilisation de la réduction de dimensionnalité avec les séries temporelles. Notre modèle émule le modèle Sequence-to-Sequence (Seq2Seq), qui est conçu pour les modèles de traduction automatique.

Sequence-to-Sequence est un modèle d'apprentissage profond dans lequel l'encodeur encode et apprend la relation entre les mots de la langue source pour la présenter en un vecteur et le décodeur la décode en une séquence de mots dans la langue souhaitée. Ici, les positions des visages sont équivalents aux positions des mots dans une phrase. Il est donc utile d'analyser les positions des visages en tenant compte du contexte. Pour cette raison, nous sommes motivés par l'utilisation d'un encodeur-décodeur Seq-to-Seq dans notre implémentation. Nous avons construit un classificateur de la pose de la tête appelé SAX-RED, où les séquences symboliques SAX seraient l'entrée de l'encodeur et le décodeur génère les séquences de sortie qui présentent les labels des poses de la tête.

Contents

List of Figures

List of Tables

Preface

Large portions presented in this thesis have already been published or reported. This applies to

- Parts of Chapter §3 were published in:

  H.Mekami, S.Benabderrahmane, A.Bounoua, A. Ahmed Taleb. Local Patterns and Big Time Series Data for Facial Poses Classification. JCP, 2018, vol. 13, no 1, p. 18-34.

  H.Mekami, and S.Benabderrahmane. SAX2FACE: Estimating Facial Poses with Peano-Hilbert Curves and SAX Symbolic Time Series. Procedia Computer Science 109 (2017): 217-224.

  H.Mekami, A. Bounoua, S.Benabderrahmane, A. Ahmed Taleb. Facial Pose Classification using Hilbert Space Filling Curve and Multidimensional Scaling. International conference pattern analysis and intelligent systems PAIS 2015, Tebessa (Algerie) 26-27 oct. 2015

- Parts of Chapter §4 were published in:

  H.Mekami, A.Bounoua, S.Benabderrahmane. Leveraging deep learning with symbolic sequences for robust head poses estimation. Pattern Anal Applic 23, 1391–1406 (2020). https://doi.org/10.1007/s10044-019-00857-5.

Introduction 1

Contents

1.1 Context

The identification of people is one of the most important challenges to humans. Since antiquity, man has been interested in finding the ways and characteristics to know the other. For example, the ancient Egyptians (3000 BC) used their palm prints as an identification tool, so Babylonian civilization and ancient China (700 BC) used the fingerprint of the thumb in the dough used to seal documents as a signature. This technique still applied by the Chinese and Japanese until the 19th century, especially when signing commercial contracts. The search of the invariant's attributes ensuring the identification of a person by his physical characteristics was genuinely taken his scientific framework until the 19th century, with the detailed study on fingerprints and their famous "innumerable little ridges" presented by Nehemiah Grew in 1684. Two years later, Marcello Malpighi processed fingerprints under a microscope. The work of Johann Evangelista Purkinje in 1823 was the most important, where he determined that the fingerprint is unique and private for each individual. In 1880, Alphonse Bertillon performed the first scientific method of criminal identification called "Bertillonage". Since then, scientific research has progressed in a research area called "Biometrics".

Biometrics is the science concerned with studying the methods and techniques of identifying and recognizing individuals. Mainly based on the analysis and measurement of key characteristics of the human body, they have the particularity of being universal, unique, permanent, measurable, and which cannot be falsified, lost, or stolen. There is a multitude of biometrics methods grouped into two categories according to their physical and behavioural characteristics. Physiological properties can be morphological or biological. These are mainly fingerprints, the geometry of the hand and finger, the venous network, the eye (iris and retina), or even the face, for morphological analyses. In terms of biological analysis, we most often find DNA, blood, saliva, or urine used in the medical field, for criminal investigations. The most common behavioural measures are voice recognition, handwritten signatures (pen movement speed, the pressure exerted, tilt), Keystroke, how to use objects, gait, footsteps, gestures, etc…

Each of these modalities has its advantages and disadvantages. No technique used can ensure both recognition performance with optimal precision. Thus, the object of academic or commercial research is to find more efficient biometric systems with less complex technology. Generally, behavioural data are very sensitive to the physical and psychological state of the individual (e.g. emotions, stress, fatigue, ageing ...) which negatively impacts the performance and increases the rate of error. On the other hand, it is estimated that physiological characteristics have the advantage of being more stable in an individual's life. In recent years, biometric identification has become increasingly focused on physiological measurements. For example, according to the International Civil Aviation Organization (ICAO), facial recognition and fingerprints are the most frequently used.

1.2 Why facial biometric

The face represents a convenient contactless biometric descriptor. In our daily life, the human face plays an important and fundamental role, and it seems, generally, the simplest and most accessible means of identifying human beings. Technically, facial biometrics are also effective and provide essential information to identify individuals. Compared to other biological biometric recognition methods such as fingerprints and iris, the facial recognition approach remains more acceptable and practically more accessible, and has many advantages:

- **Natural**: Usually, facial biometric systems are also like humans distinguish and recognize individuals in communications by comparing the face features not the fingerprint or iris features.
- **Not intrusive**: The face analysis is performed without the subject feeling that their identity has been verified, and it does not require any user cooperation, unlike fingerprint or iris systems. This invisibility is important in some uncontrolled environments. Face recognition cameras are already installed at airports around the world.
- **Run in real-time**: Facial recognition software immediately recognizes people. When a face is captured by a camera, the correspondence performs in real-time and the software verifies and identifies the person.
- **Easy to reach**: Due to the enormous development of photography equipment, the difficulty of taking a photo has become less and less. Nowadays, we can easily use our smartphones to take high-quality photos for facial analysis.

So, facial biometric systems show good future applications in many areas. They are not only widely deployed in traditional applications such as video surveillance and access control, but they are also rapidly applied in new fields such as: mobile telephone, automatic payment, bank systems, criminal investigations, etc

Far from these applications, facial recognition could become a real tool in other fields such as health. For example, American researchers have designed the DeepGestalt system capable of detecting a genetic disease for a patient, by analysing his face [Gurovich et al., 2019].

1.3 Problematic

Some processing steps are required before the recognition of an individual in an image or video via his/her face. In general terms, the recognition process can be unrolled in four fundamental stages:

- Detection: it is the process that can identify and locate automatically a face within an image.
- Face alignment: this is a normalization step, the face detected images are geometrically normalized, so the lighting can be normalized to compensate the illumination variations.
- Facial features extraction: from face alignment images, the characteristics points are extracted. These points provide more pertinent information that produce biometric signatures called features vectors, which distinguish one individual from another. These signatures must be discriminatory and robust. A distinctive vector is discriminate when it takes completely different values for the faces of two different people. In terms of robustness, it must be invariable to all possible variations, such as expression, pose and illumination.
- The comparison step: the last step is classification, intended to classify the recorded feature vector of the face tested with the most similar images in the database.

Relatively, facial recognition is a recent biometric tool. The first analysis was realized in 1964 by Woodrow Bledsoe Team [Bledsoe et al., 1964]. Another attempt was performed by Takeo Kanade in 1973 through his doctoral thesis at the University of Kyoto.

In the current state of computer vision, face recognition is taking-off in giant steps, thanks to advances in algorithms linked to the development of artificial intelligence, particularly deep learning network that has enabled the face recognition tools to become much more efficient.

Over the past twenty years, numerous powerful and efficient face recognition results have been achieved in a controlled environment. However, these performances brusquely deteriorated when the face images were captured in a real scenario. For example, Where there is a change in the pose or different lighting conditions and a wide intra-class variation, generally, occurs by the presence of the expressions or occlusions.

In literature, different ways have been used to identify subjects in degraded illumination environments [Zou et al., 2007, Ochoa-Villegas et al., 2015, Wang and Deng, 2018], also in the presence of facial expressions [Sandbach et al., 2012, Corneanu et al., 2016, Li and Deng, 2018]. Furthermore, face recognition with frontal faces has been the subject of numerous studies and progress over these last decades. Nevertheless, face recognition in multi-poses remains a higher challenge and a largely unresolved problem. Consequently, the choice of the face image in the frontal pose allows considerably to improve the performance of the face recognition system. In this context, we present capable approaches to estimate the face orientation and select the frontal pose. These head pose estimation methods must be able to provide an accurate and robust estimation in different environments, and therefore necessarily operate independently of the identity of the person.

1.4 General objective and contributions

The objective of this thesis is to address the problem of head pose orientation. To yield this purpose, we introduce the use of time series. We propose a new approach based on the exploitation of the representation of faces as time series for building a model to classify and estimate poses. The method use dimensionality reduction through time series representation of the learning images. This technique reduced the dimension of research space from 2D to 1D space. We suggest a new face image representation, where we use the Space-Filling Curves (SFC). The SFC permitted mapping multidimensional points to a vector while preserving the locality. This representation will be the main tool of our approaches and allows us to treat the image as a time series.

Once the time series of the image is generated, a symbolic transformation is effectuated on these series to convert the numerical values into symbolic sequences of the alphabets. The Symbolic Aggregate approach approximation

(SAX) [Lin et al., 2007] is used to complete this task. Then, in the final phase of our approaches, several artificial learning algorithms such as KNN, SVM, and Recurrent Neural Network (RNN) taken these sequences as data to train models to build performance classifiers, which allow deciding if the face is in frontal or profile view even in disadvantageous external conditions. Our main contributions can be summarized as follows:

1. We convert the face image into one dimension as time series using the space-filling curve.
2. We represent these time series by Symbolic Aggregate Approximation (SAX) representation which, ensures the reduction dimensionality.
3. We investigate the resolution of SAX (frame size, alphabet size) in the classification accuracies.
4. We calculate the pair-wise similarity matrices between images of different databases using an adapted distance.
5. Several classification methods throughout the generated similarity matrices were used, and very efficient results were obtained.
6. The symbolic sequences generated by SAX are taken as the input data of Encoder Recurrent Neural Network (RNN), then the Decoder RNN provides the labels of head pose.
7. We investigate the use of Bidirectional Long Short term Memory (BLSTM) and Bidirectional Gated Recurrent Unit (BGRU).
8. The proposed approaches achieve state-of-the-art results on several public datasets.

1.5 Dissertation structure

This work is organized as follows

- **Chapter 2:** The background and the state of the art of head pose estimation methods are presented, concentrated on 2D algorithms. Subsequently, a discussion for summarizing the advantages and drawbacks of head pose estimation methods has been discussed.

- **Chapter 3:** A brief review of time series data mining is described, and it brings into light the Symbolic Aggregate approach approximation (SAX). Moreover, the method appointed for this thesis is broadly exposed. That explores the time series property and how it can be benefited for it to build a model that allows classifying the head pose using classic algorithms such as KNN, SVM. Then, evaluation and analysis of the proposed approach are performed on different databases.
- **Chapter 4:** This chapter presents the deep learning model for head pose estimation. The model is based on the symbolic representation described in Chapter 3 and the Sequence-to-Sequence (Seq2Seq) model of [Sutskever et al., 2014]. Firstly, a brief description of the deep learning and recurrent neural networks is presented. Next, a brief review of the sequence-to-sequence model and its application to the head-pose estimation problem is presented. Lastly, we report a comprehensive and experimental analysis for applying the Seq2Seq models to a head pose estimation task.

Finally, a general conclusion is presented. Furthermore, some perspectives regarding this work will be discussed.

Head pose estimation methods: State of the art

2

Contents

2.1 Introduction

In our daily life, there is a camera that acquires very often images of people, particularly their faces for different purposes, e.g.: personal leisure's, biometrics identification or security monitoring. This process is useful in many applications such as face recognition, human-computer interaction, video surveillance, human behaviour analysis, and driver assistance system. The performance of such applications would obviously increases with the best acquisition of images from the camera. Usually, the human faces can be detected in a stream of images in different directions and views, which would deteriorate drastically the robustness of the previous systems, because the rotation of the head can affect the recognition of some features of the face

In computer vision, the head pose estimation is the discipline that attempts to resolve the problem of the head's orientations according to the camera point of view. It is becoming a long-term trend and an active research topic. Several approaches have been suggested in the literature by employing different techniques and ways to achieve strong accuracy and create an efficient classification model.

2.1.1 What is the head pose estimation

The problem of head pose orientation is illustrated typically through three types of rotations relative to the frontal face view or a global coordinate system. The yaw angles, when the head rotates on the Y-axis from 90° at -90°. The pitch where the target looks up or down (X-axis), and the roll rotation is the inclination from left to the right relative to the Z-axis as shown in figure 2.1.

Most of the previous research works have given more attention to both yaw and pitch since the problem of roll rotation is solved easily by using the position of some feature points such as the centre of the eyes.

2.2 The head pose estimation challenges

The head poses estimation is a complex task due to the wide variety of physiological parameters, such as position, view angle, and peripheral acquisition conditions, etc...

A successful system that is capable of performing this task must take into account the many sources of variation that can affect the face image:

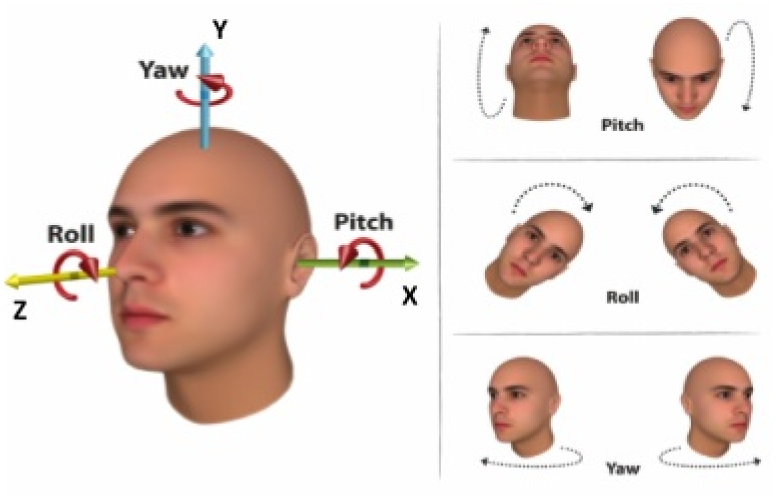


Figure 2.1 – *Example of the different pose variations.*

- **Geometric Deformations:** mainly due to facial expressions variations (intra-personal variations) and those linked to morphology variations (inter-personal variations).
- **Occlusion:** Certain parts of the face can be masked by simple gestures, e.g.: the person running his hand over his face, adjusting his glasses, or turning around. Also, he can wear a hat or scarf.
- **illumination conditions:** the type, position, and orientation of light sources directly influence the appearance of the face in the image.
- **Variations related to the acquisition device:** the type of camera and the focal length between it and the individual, the resolution and the compression of the image are also sources of variations that will have an impact on the image.
- **Radiometric variations:** depend on the properties of the skin (colour, wrinkles ...), the presence of a beard, or even the colour of the eyes. So, the radiometric variations correspond to changes in the texture.

2.3 Head pose estimation approaches: state of the art

The objective of the head pose estimation methods is to determine the head orientation from face images. Generally, the head pose estimation method is assumed to have the following properties:

- Independent of the person.

- Robust and powerful.
- Run fast to improve applications in real-time.

In this chapter, we present a panorama of the main state-of-the-art of head pose estimation methods. Many works have been proposed attempting to tackle the task of head pose estimation. The classification of existing approaches for head pose estimation into a single taxonomy is quite a difficult task, as there are many approaches and the classification can be ambiguous. Murphy Chutorian et al. [Murphy-Chutorian and Trivedi, 2009] published a survey that presents the most commonly head pose estimation methods proposed up to 2009. Recently, new techniques have been developed, and new data have become available such as methods based on 3D data, methods based on depth images, and methods based on neural networks and Deep Learning. In some other studies, these methods can split into two main categories according to the tools used to solve the problem: appearance-based methods and feature-based methods (as shown in figure 2.2).

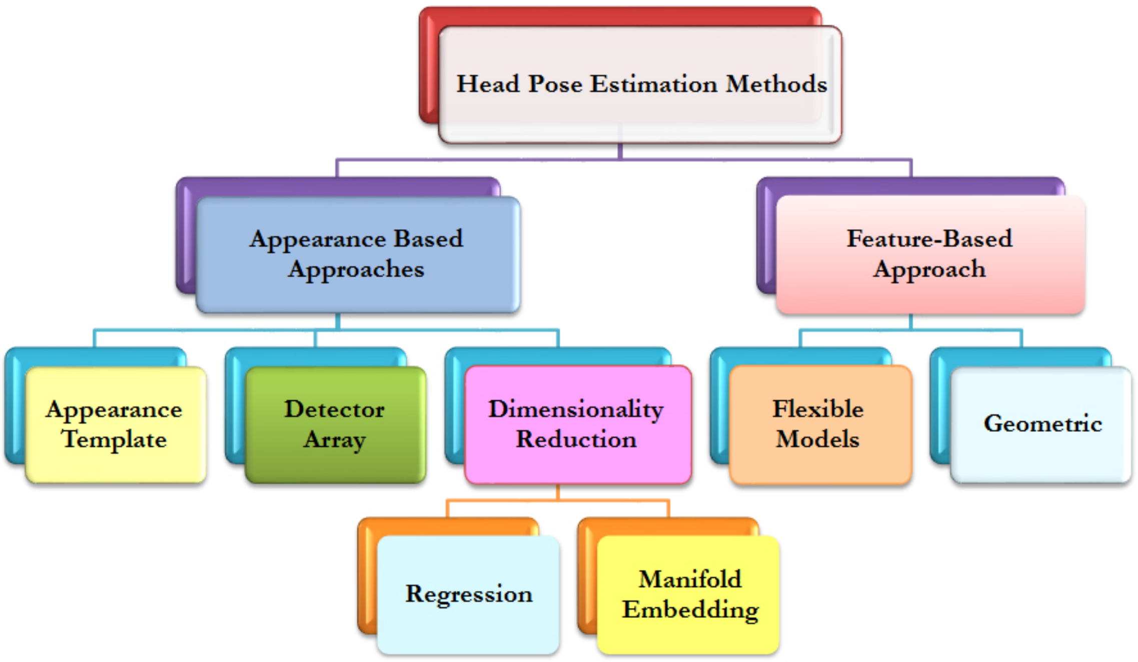


Figure 2.2 – *Head pose estimation methods.*

2.3.1 Feature based methods

These approaches are considered local methods. They mainly used the points of reference to estimate the head pose. The features are extracted from the face, for example, the eyes, mouth, or even nose tips. The correspondence between these key points and the geometrical model allows the prediction of the head pose rotation. Therefore, the head pose estimation in these approaches is needed to recognize the position of the facial key points which are tracked, and then, the correlation between the next and previous positions admits to estimate the pose. These approaches contain two methods: flexible models methods and geometric methods.

2.3.1.1 Flexible models

The flexible models are one of the methods in this category. Initialization and tracking phases ensure the estimation of the head. In the early step, the flexible model attempts to fit the face region to build the initial absolute head pose. Then the face region has to be tracked to get the relative change in head pose. The most frequently used models are the following ones: Active Shape Model (ASM), Active Appearance Model (AAM), and Elastic Bunch Graph Matching (EBGM).

The ASM model was proposed by Cootes and Taylor [Cootes et al., 1995]. The ASM is formed from a set of corresponding characteristic points (landmarks) collected from the training images. The location of landmarks can be connected to each other to represent the shape of the face. Enough number of features point allows covering the whole face shape, and can be constructed contours that seek the main variations in sets of training data, using principal component analysis (PCA). Once the ASM model is created, an initial contour is iteratively deformed until it adapts to the shape of the face fitting (figure 2.3), which determines the pose of the face tested.

The algorithm of ASM model can be realized in five steps:

1. Labelling the landmark points for the shape of the training set.
2. Extracting grey profile to all landmarks.
3. Aligning training set.
4. Apply the PCA to calculate the statistics on the aligned training set.
5. Repeat steps from step 1 to step 4 until convergence.

Cootes et al. [Cootes et al., 2001; 2015] proposed the combination of a statistical shape model (ASM) and texture to create the active appearance model(AAM). The face is split into small areas using triangles whose vertexes are

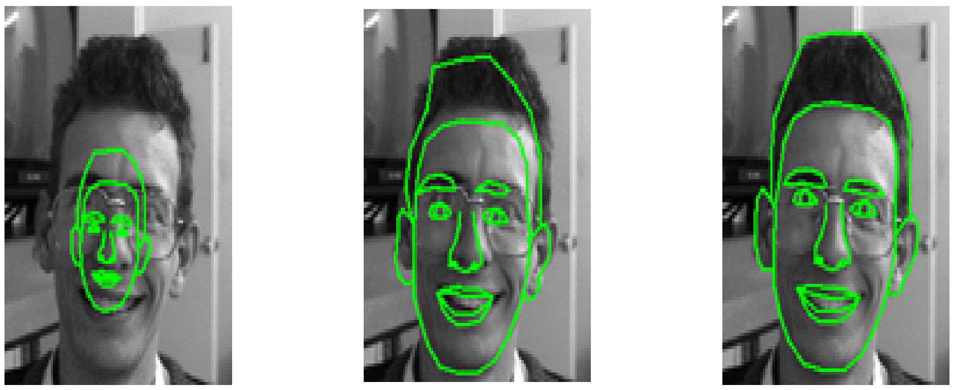

(a) (b) (c)

Figure 2.3 – *Example ASM model. (a) Initial Pose, (b) After a few iterations, (c) Convergence.*

the landmark points. The shape is determined through interesting points that had manual annotations on the face training images, the pixels surrounded by this shape describe the texture. Thus, fitting is to adjust the shape statistical model and appearance of the face to a new image (figure 2.4). Thus, the sufficiently close correlation between the model and the face being tested allows us to estimate their pose.

The algorithm of the AAM model can be performed by pursuing these steps:

1. Labelling the landmark points and extracting the shape model of the training set.
2. Warping or mapping the landmark points into the mean shape.
3. Vectorizing the warped images.
4. Performing the PCA on the shape-normalized of training images.
5. Repeat steps from step 1 to step 4 until convergence.

Constrained local models (CLMs) are an improved version of two previous models. The CLM model combines the performance of approaches based on feature detection, the flexibility of the AAM model, and the constraints of the ASM model bring a deformable model created from the local textures which are found around specific feature positions such as the eyes and mouth. Similar to the AAM model in the learning phase, the CLM builds as a shape and texture model from landmark points, and the texture is sampled into patches around these points [Cristinacce and Cootes, 2008], as shown in figure 2.5. To increase the accuracy of the traditional CLM technique and decreasing the computational cost Rajamanoharan et al. [Rajamanoharan and Cootes, 2015] suggested a multi-view Constrained Local Model (MV-CLM) combined with a global shape model

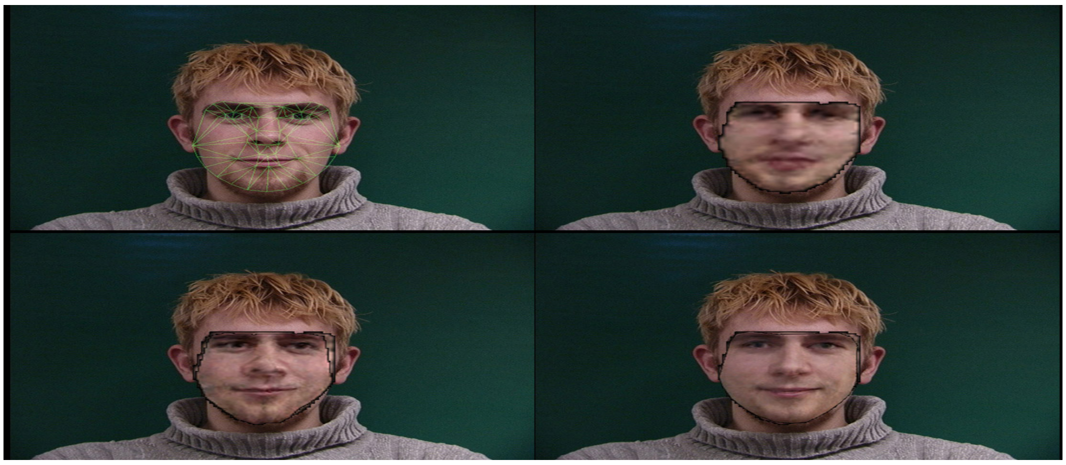


Figure 2.4 – *Example of AAM model.*

and different response maps per pose to determine the optimal pose-specific CLM, while Kim et al. proposed Holistically Constrained Local Model (HCLM) to detect the facial landmark and estimate the head orientation.

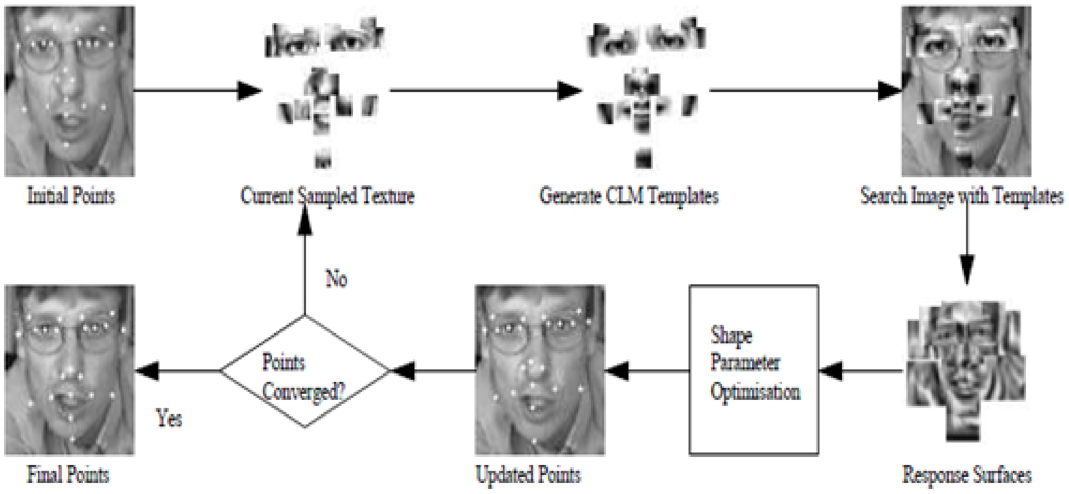


Figure 2.5 – *Example of CLM model [Cristinacce and Cootes, 2008].*

Elastic Bunch Graph Matching (EBGM), is one method among the techniques used for the construction of flexible models. EBGM uses Gabor wavelet convolution to represent the landmark points(fiducial points), such as each fiducial point is represented with Gabor jet, which is - jet- all wavelet convolution values in this point. Then, the model jet is arranged in a structure named "Bunch graph". So, the landmark points in this model are characterized by their locations and the Gabor jet elicited from these locations. Therefore, the face is described with the nodes and edges connected with these nodes (figure 2.6). Thus, the pose of the new face image is achieved by the most correlated face graph [Wiskott et al., 2014, Elagin et al., 1998].

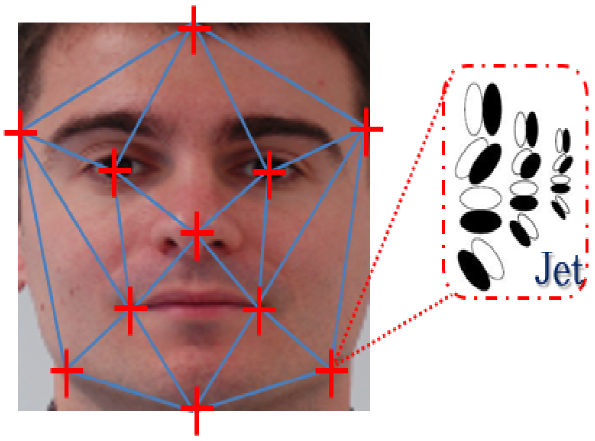


Figure 2.6 – *Example of EBGM model.*

2.3.1.2 Geometric models

Geometric methods consider the head shape and the position of the facial features as principal tools to estimate the head pose, from these landmark points, a geometric model is constructed to allow the determination of the pose of human faces. These landmark locations can use directives as it is the case in [Wang and Sung, 2007] where the authors used the six points: two outer eye corners, two inner eye corners, and two mouth corners that include the use of vanishing points to estimate the head pose as shown in figure 2.7. In our work [Mekami and

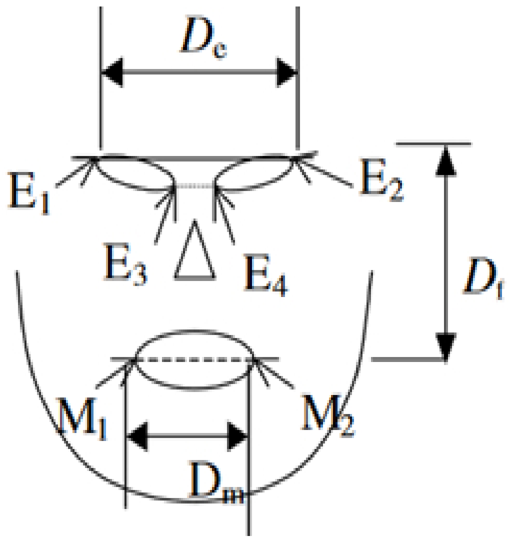


Figure 2.7 – *Example of geometric model [Wang and Sung, 2007].*

Benabderrahmane, 2010] we used the two eyes centres and the mouth position to estimate the pose from the isosceles triangles formed by these key points. Cylindrical and elliptical face models are other ways to estimate the head pose geometrically. From the three facial points: the right facial edge, the left facial edge, and the face centre. A cylindrical model is constructed and the rotation around the face centre axis permits the prediction of the head pose [Narayanan et al., 2014; 2016]. The geometric model established by Riener et al. [Riener and Sippl, 2014] is based on location and dimension information for the left and the right eyes, nose, mouth, even the mouth corners, and the face as a whole. As the yaw features are relatively related to facial symmetry, this allowed the authors to extract six features: the horizontal distances ($a_{yaw} - b_{yaw}$ and $a_{yaw} : b_{yaw}$), the diagonal distances (c–d and c:d), and the angles ($\alpha - \beta$ and $\alpha : \beta$) as shown in figure 2.8-a. As well as the difference of the horizontal distances between the left eye and the face centre and between the right eye and the face centre ($p_{yaw} - q_{yaw}$) and the horizontal distance between the nose and the face centre (r_{yaw})as shown in figure 2.8-b. Further, the head movement in the left or right is deduced from the difference in the two eyes' sizes (figure 2.8-c). Another feature was the nose's horizontal distance from the facial-symmetry line [x_{yaw}]. Meanwhile, the features

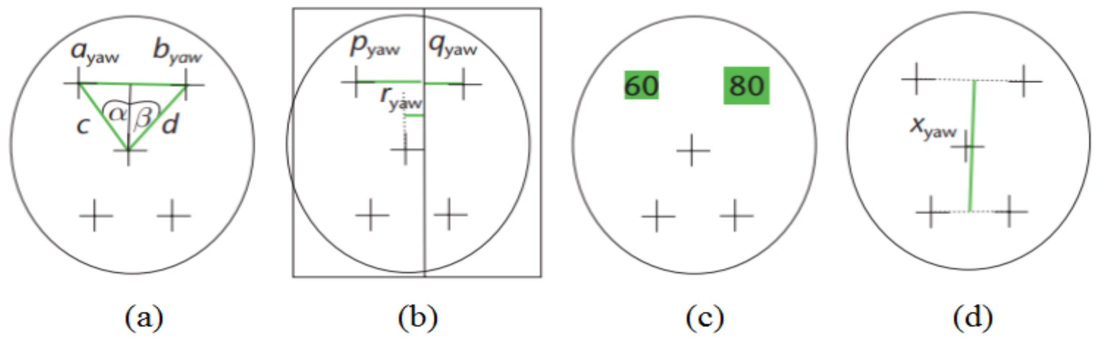


Figure 2.8 – *The geometric facial features for yaw rotation (cross lines indicate a feature location; rectangles describe dimensional features) [Riener and Sippl, 2014].*

used to estimate the pitch rotation are: the sum of and the difference between the distances from the eye line to the nose and from the nose to the mouth line [a_{pitch} + b_{pitch} and a_{pitch} + b_{pitch}](figure 2.9-a). So p_{pitch}, q_{pitch}, and r_{pitch} identify the distances from the eye line, nose, and mouth line to the vertical face centre(figure 2.9-b). Another feature was the distance between the vertical face centre and the centre between the eye and mouth lines x_{pitch} (figure 2.9-c), while figure 2.9-d indicates the dimensions of the four facial parts which are used.

Marcialis et al. [Marcialis et al., 2014] introduced the concept of "Golden Ratio" to carry out a geometrical model to estimate the head pose. Golden Ratio is the proportionality constant adopted by Leonardo da Vinci in the "Vitruvian man's".

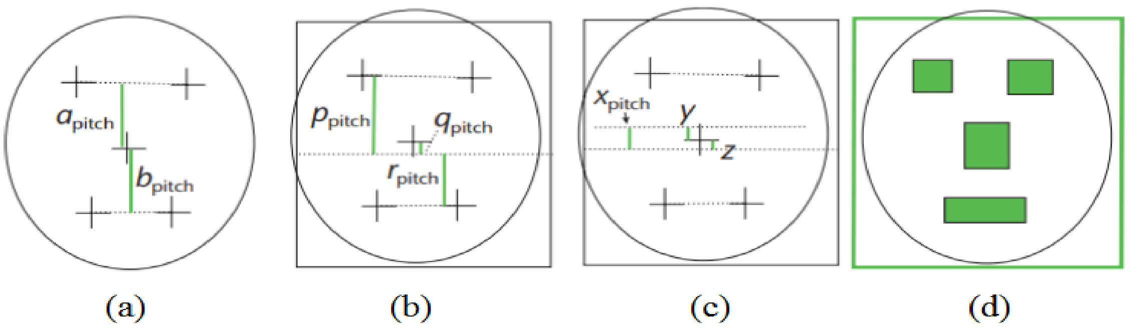


Figure 2.9 – *The geometric facial features for pitch rotation [Riener and Sippl, 2014].*

The head orientation is yield thanks to the ratio between distances of eyes and nose, pursuant to the specific assumption of the Golden Area as illustrated in figure 2.10 . Similarly, Hatem et al. [Hatem et al., 2015] built a model from the relative position of eyes and mouth corners, so that the nose coordinates were the origin of the coordinate system.

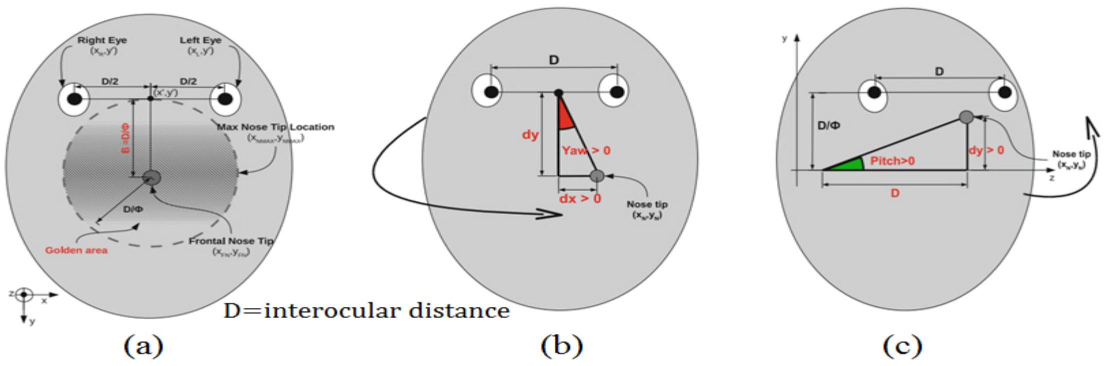


Figure 2.10 – *The geometric facial models proposed by Marcialis et al.[Marcialis et al., 2014]. **(a)** frontal head pose, **(b)** Yaw angle computation, **(c)** Pitch angle computation.*

2.3.2 The appearance-based approaches

Unlike the previously mentioned methods, the whole face is the principal mean used in the appearance-based approaches to extract features that permit the estimation of the head pose. Consequently, the head pose estimation task required a critical step to represent the face image with feature vectors. In this step, it is necessary to emphasize that feature vectors extracted from a face image can handle pose variation avoiding all other variations irrelevant to pose, e.g., identity, expression, and lighting. These approaches reduced or transformed the problem of head estimation to pattern classification. A model is trained with a learning database and during the execution, it is used to find the best matching between the query image and the training images.

Appearance-based methods can be classified into different categories according to the technology used to determine the pose: the template-based methods, classification-based methods, and the dimensionality reduction methods, which can be split into two sub-types: regression methods and manifold embedding methods.

2.3.2.1 The template-based methods

The template-based methods or the methods by comparison with prototypes are one of the oldest approaches, which are relatively very simple and intuitive. These approaches effectuate through the direct comparison of a new image with a set of examples of database images labeled with their pose and the most similar template that determines the pose of the tested image. Various metrics were used to measure this similarity criterion,Mean Squared Error (MSE) [Niyogi and Freeman, 1996], Normalized cross-correlation (NCC) [Beymer, 1994], as well mutual information [Goudelis et al., 2008] and Normalized co-occurrence mutual information [Qing et al., 2010].

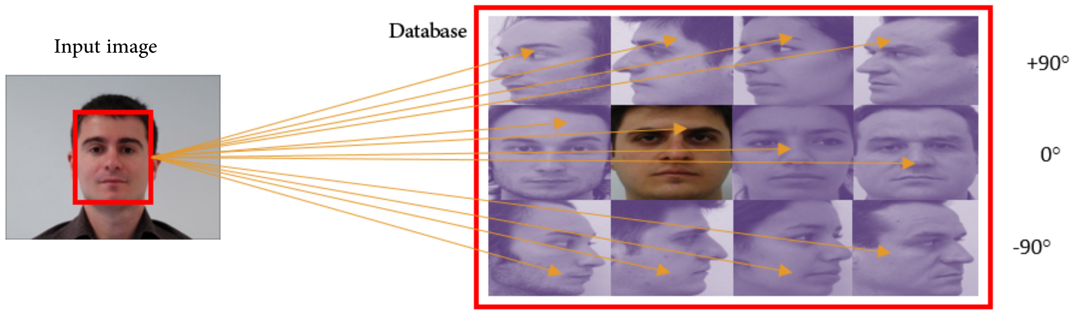


Figure 2.11 – *Estimation of the face pose by The template-based method.*

These direct approaches suffer from numerous limitations, the main is that the similarity between different poses for the same person can obviously be higher than the similarity between two face images of different people in the same pose, which leads to a poor pose estimate of the head.

2.3.2.2 The classification-based methods

In the classification-based methods, the head pose is acquired from groups or classes where the training images are grouped into a limited number of classes labelled with discrete poses, which are obtained via supervised algorithms. Most of the techniques proposed in these methods can jointly address the problems of face detection and estimation of the pose.

The classification methods are performed in two stages, features extraction, then these features are used to learn a model for estimating the head poses using well known supervised algorithms as shown in figure 2.12. Therefore, to achieve a high classification accuracy, the techniques reported in these methods are attempting either to extract the suitable features that represented the face orientation or a powerful algorithm of classification. The work of Rowley et al.

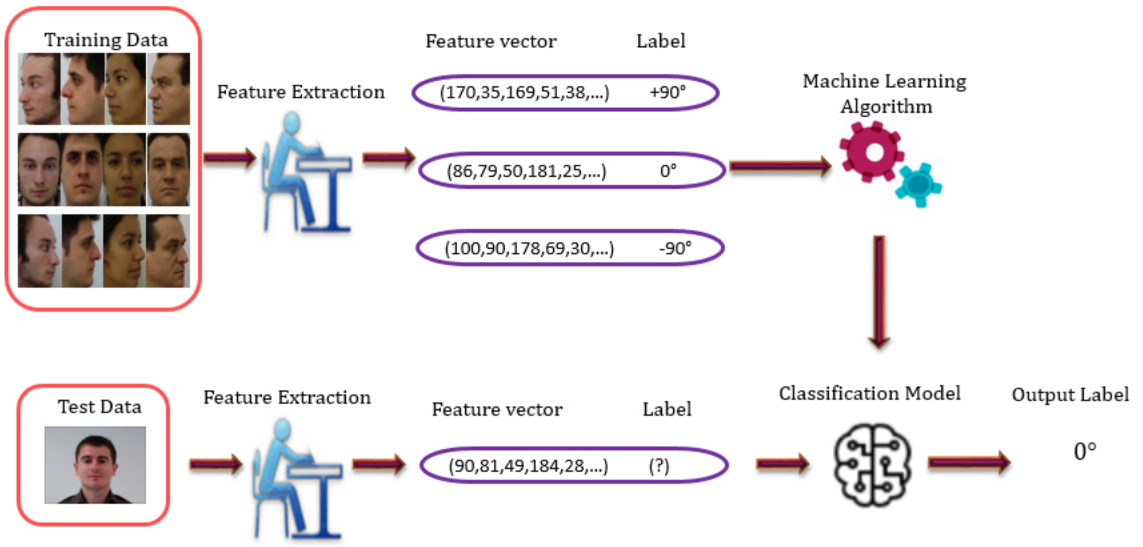


Figure 2.12 – *Illustration of the classification method.*

[Rowley et al., 1998] is among the earlier essays proposed in these methods, where they used multi-layer perceptron to classifier the face pose. The first layer is a "router" network that tackles each input image to determine its orientation and then transmits this information to a single detector to support or discard the decision. However, the reliability of these methods depends on the performance of the pose estimator. Furthermore, this technique cannot extend to out-of-plane (yaw, pitch) rotations of the face where it was only tested for the roll (in-plane) rotation. Huang et al. [Huang et al., 1998] implemented support vector machines (SVM) with three classes to discriminate the frontal view vs the left and right view. The classifier of Viola-Jones [Viola and Jones, 2004] is the leading model used in multi-view face detection and it serves as a foundation for a modern detector. The Viola-Jones model is based on three main ideas that make it possible to build a successful milt-view face detector: the integral image, classifier learning with AdaBoost, and the attentional cascade structure. The integral image is used for the fast computation of Haar-like features and utilizing the AdaBoost to train cascaded classifiers.

A naive Bayesian classifier is used to estimate the head pose in five orientations corresponding to frontal view, left half-profile, right half-profile, left profile, right profile [Zhang et al., 2006]. Chen et al. [Chen and Lien, 2009] developed a statistical system capable of detecting and estimating the face pose that is composed of five modules, where the Eigenfaces were chosen as the face features, and the Gaussian mixture model(GMM) was used to classify the pose into seven yaw poses(0, $\pm 30°$, $\pm 60°$, and $\pm 90°$). The authors in [Ma et al., 2008] exploited the facial symmetry which is vanished with the yaw rotation, and introduced a new descriptor called GFour to represent this asymmetry. The input image is convoluted with multiple 1D Gabor filters which provide the advantage of mitigating the influence of illumination and noise, then they used the Fourier transformation to extract the facial asymmetry, the nearest centroid classifier (NC) is employed to determine the head pose categories. Further, Ma et al. [Ma et al., 2014; 2015] proposed a set of novel descriptors named covariance descriptor of Gabor filters (CovGa) and fisher vector of local descriptors (VoD), which are improved by combining them with a metric learning method named: Keep It Simple and Straightforward Metric Learning (KISSME). So they have produced other descriptors K-CovGa and K-VoD. These descriptors extracted the asymmetry features of the head. The nearest centroid classifier (NC) was used to classify the yaw pose. Furthermore, the biologically inspired features (BIF) and the combination of the BIF with the local binary pattern (LBP) feature produced the local biologically inspired features (LBIF) are other new feature descriptors presented also by Ma et al. in [Ma et al., 2013]. However, all these features proposed by Ma el al. [Ma et al., 2008; 2013; 2014; 2015] are focused to represent only the yaw angle variation. Alternatively, an image abstraction is proposed by Han et al. [Han et al., 2014] to represent the variations of face orientation. They created binary images with the most important features of the face, then utilized local directional quaternary patterns (LDQP) to achieve better representation discriminative for head pose estimation. Subsequently, a multi-class SVM classifier is learned to interpret the orientation of the head. Random Forest (RF) is used by Huang et al. [Huang et al., 2010], as an ensemble learning framework to classify the head pose based on features extracted from Gabor Filtre. kang et al. [Kang et al., 2017] suggested multi-block local binary pattern (MB-LBP) feature, and training the RF algorithm to obtain head pose estimation. Liu et al. [Liu et al., 2016] introduced an improved version of the RF. They combined multiple texture descriptions, i.e., Gabor feature-based PCA, LBPH, Sobel with two geometric features to extract the face orientation representation, that is distinguished by the RF classifier reinforced through the use of Dirichlet-tree distribution. While, Zhao et al. [Zhao et al., 2019] proposed the multi-feature fusion, where the second-order HOG and UP-LBP features are extracted separately for the face image, then fuse the two features to form the final image features. Thereafter, in the construction stage of Random Forest, the CART algorithm is introduced and trained independently, and all the decision trees are constructed by weighted combination to participate in the final decision of the Random Forest algorithm. Afterward, the improved random forest is learned to determine the head pose. Dictionary-Learning and a classifier based on sparse representation are exploited in [Zhang et al., 2013, Liao et al., 2016]. However, Zhang et al. [Zhang et al., 2013] employed this technique to determine only the facial left and right orientation (i.e. yaw pose). While Liao et al. [Liao et al., 2016] have developed a model based on this technology to recognize the facial up and down orientation(i.e. pitch pose) too. Even, they built a dictionary of face occlusion that permits to solve the estimation problem when a face is occluded. Another initiative to detect and estimate the face pose was conducted by Yoon et al. [Yoon and Kim, 2019]. The author constructed a model with four successively blocks: background rejector, pose classifier, pose-specific face detectors, and face validator as shown in figure 2.13. The first module facilitates the removal of the background. The face pose is estimated in the second module earlier than the detection of a face that is performed in the next module. The final module has the role to decide, and valid is that the input image is a face or non face.

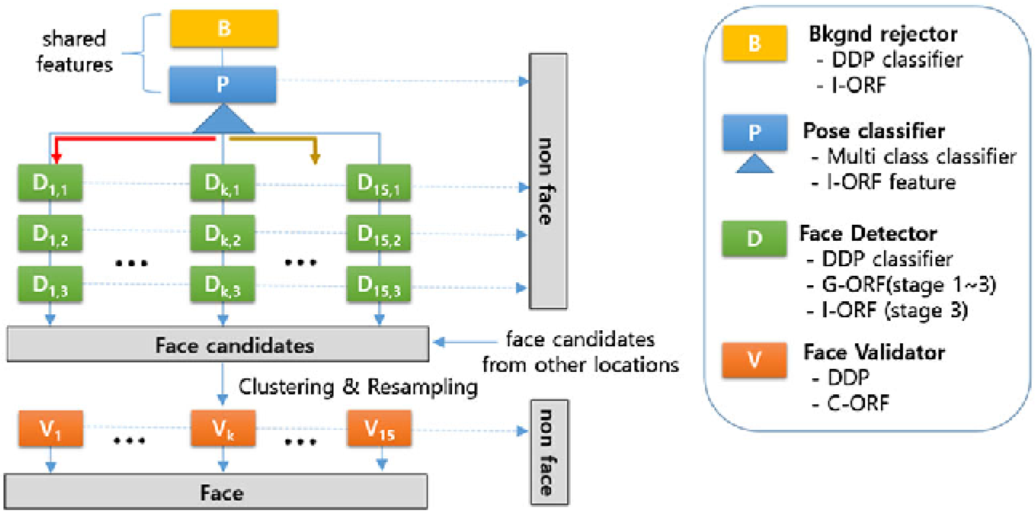

shared features
B
P
$D_{1,1}$
$D_{k,1}$
$D_{15,1}$
$D_{1,2}$
$D_{k,2}$
$D_{15,2}$
$D_{1,3}$
$D_{k,3}$
$D_{15,3}$
non face
Face candidates
face candidates from other locations
Clustering & Resampling
V_1
V_k
V_{15}
non face
Face
B Bkgnd rejector - DDP classifier - I-ORF
P Pose classifier - Multi class classifier - I-ORF feature
D Face Detector - DDP classifier - G-ORF(stage 1~3) - I-ORF (stage 3)
V Face Validator - DDP - C-ORF

Figure 2.13 – *The structure of multi-view face detector proposed in [Yoon and Kim, 2019].*

2.3.2.3 Regression Methods

Contrary to classification methods, the regression methods attempt to find the relationship between the feature vectors and the pose variation to predict the continuous head pose angle. Regression methods are categorized under the umbrella of Dimensionality Reduction methods that mapping the face representation feature from high dimensional into low dimensional, denoted the head pose rotation. The figure 2.14 illustrates an example of the regression method.

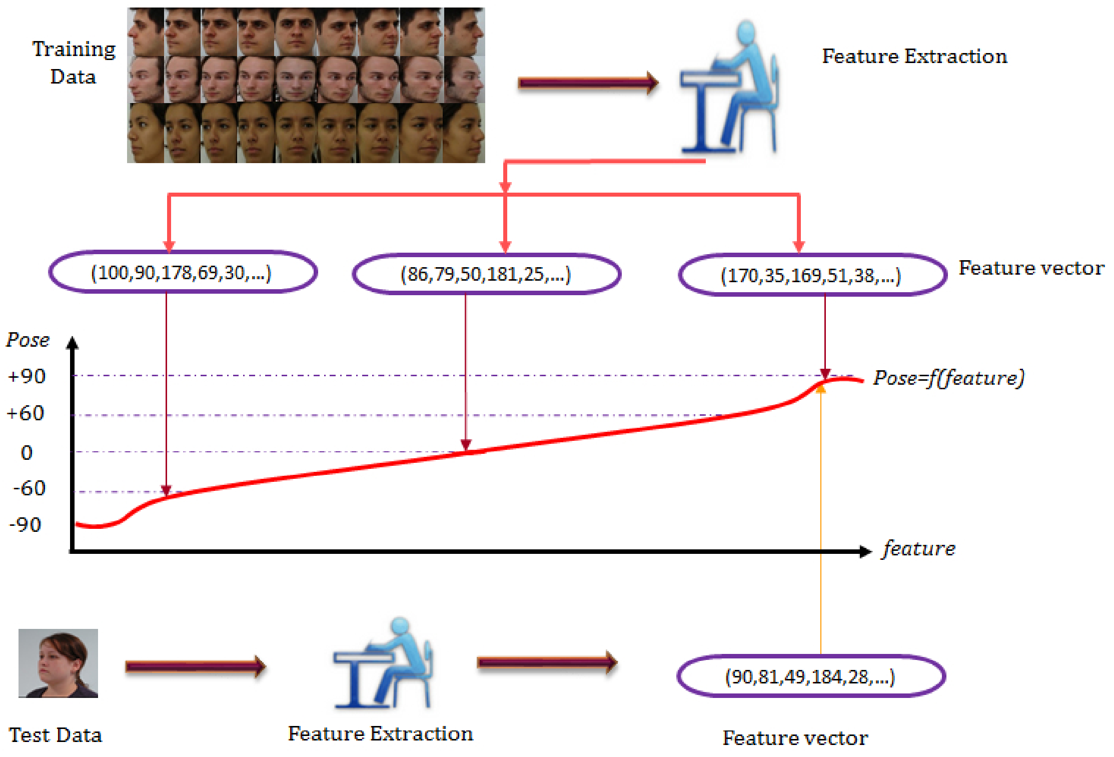


Figure 2.14 – *Illustration of the regression method.*

Numerous regression models have been employed in the literature. A regression via the neural network named GRNN (Generalized Regression Neural Network) to achieve the linkage between the input features and the output head rotation angle that is presented in the work of Bailly ([Bailly and Milgram, 2009]). They pioneered the Fuzzy Functional Criterion (FFC) to extract features. Thereby, the boosted tree algorithm uses this criterion to select the efficient feature that is taken as the input of a neural network. Random regression forests are widely used to express the variation of the head rotation angle. Among the first published researches, that exploited the random forest as a regression for head pose estimation task, the paper of Fanelli et al. [Fanelli et al., 2011]. The authors constructed the tree predictors in random forests using the depth information as feature vectors. Since it is used as a regression tool to determine the continuous head pose, many methods have been proposed to improve and optimize the construction of tree forests. As suggestion of Zhu et al. [Zhu et al., 2013] who proposed some improvements to Random Regression Forests for estimating the head pose in 2D face images using Linear discriminant analysis (LDA) to reduce the dimension of the features that were extracted with Gabor filters and weighted accordingly to the eigenvalues associated with their corresponding LDA axes to generate tree predictors in the forests. Whereas, Ying et al. [Ying and Wang, 2013] proposed the dynamic random regression forests (DRRF), which introduced some actions that allow dynamically building the tree forest, aimed to avoid the redundant tree in RF that enhances the learning quality, the boosting strategy for data induction is used. With this strategy, the samples for building the next tree are the data with a large prediction error of the previous tree. Further, the author embedded a stem operator into the conventional tree-shaped to ensure the division's continuity of the nodes and thus growing the possibility of optimal data separation. Lately, Hara et al. [Hara and Chellappa, 2017] performed the Adaptive K-clusters Regression Forest (AKRF) to upgrade the RF. They optimized the RF using the k-means clustering to each node and Bayesian information criterion(BIC) to determine adaptively the number of child nodes. On the other hand, Haj et al. [Al Haj et al., 2012] used Partial Least Squares (PLS) regression model to learn a mapping from HOG features to the head pose estimation. While, Fenzi et al. [Fenzi et al., 2013] learned a set of local feature generative models using radial basis function (RBF) networks and formulates the head pose as a Maximum A Posterior (MAP) inference task. Li et al. [Li et al., 2014] extracted face features using block energy maps (BEM) and the head pose is estimated using support vector regression and Gaussian processes regression respectively.

In contrast to the traditional regression that leads from a high dimensional to low dimensional to yield the regression to interpret the head pose, Drouard et al. [Drouard et al., 2015; 2017b;a] suggested the use of the inverse regression through a Gaussian mixture model. In other words, unlike learning and formulating the regression from the face representation (i.e. high dimensional) to head pose (i.e. low-dimensional), the authors used GLLiM (Gaussian Locally Linear Mapping) to learn and predict the pose parameters from the head pose. Thereby, the problem turns out from a low dimensional to a high problem. The HOG features and a Gaussian locally-linear mapping model are used in [Drouard et al., 2015], as well as, an extension of this work that associated the partially latent variables in their mixture of the linear regression approach reported in [Drouard et al., 2017b]. Furthermore, they combined a Gaussian mixture of linear inverse-regression model into a dynamic Bayesian model to predict pose parameters [Drouard et al., 2017a]. Beyond two-stage Cumulative Attribute (CA) regression is carried out in a regression model rather than a single layer regression architecture. The features extracted are mapped to Cumulative Attribute space (Attribute Learning) that provides the first regressor. Then the output of this stage is mapped to a two-dimensional output space (i.e. head yaw and pitch angles) in the second regressor using the Partial Least Squares (PLS) regression [Chen et al., 2019].

2.3.2.4 Manifold embedding methods

Manifold embedding methods provide the ability to map the facial features that represent the continuous variation in the head pose in a high-dimensional into a low-dimension space. The underlying idea behind these methods is to generate a dimensionality reduced features space from the high dimensional feature vector extracted from all training images set, where each feature point is connected with the closest neighbours to create a manifold embedding, depending on two parameters necessary: the number of nearest neighbours as well as the number of dimensions to which the high-dimensional space should be reduced. The head pose estimation of a new example that is not in the training set is a huge problem in these methods because there is no direct way to map an image onto the manifold. For this reason, a supervised or unsupervised learning models is used to embed the features of a new image to predict the orientation of the head. The major challenge of these methods is to know how to choose the dimensionality reduction space that efficiently recovers face pose while ignoring other changes in the image. An example of the manifold embedding method is illustrated in figure 2.15

Principal Components Analysis (PCA) [Srinivasan and Boyer, 2002] and the nonlinear kernel KPCA [Chen et al., 2003], locally linear embedding (LLE) [Roweis and Saul, 2000], Laplacian eigenmaps (LE) [Belkin and Niyogi, 2003], Locally Embedded Analysis (LEA) [Fu and Huang, 2006] and isometric feature mapping (Isomap) [Raytchev et al., 2004, Tenenbaum et al., 2000] have been used as classical techniques of dimension reduction. The interested reader can refer to [Wang et al., 2017] which presented a discussion and comparison of classical manifold learning algorithms as well as several

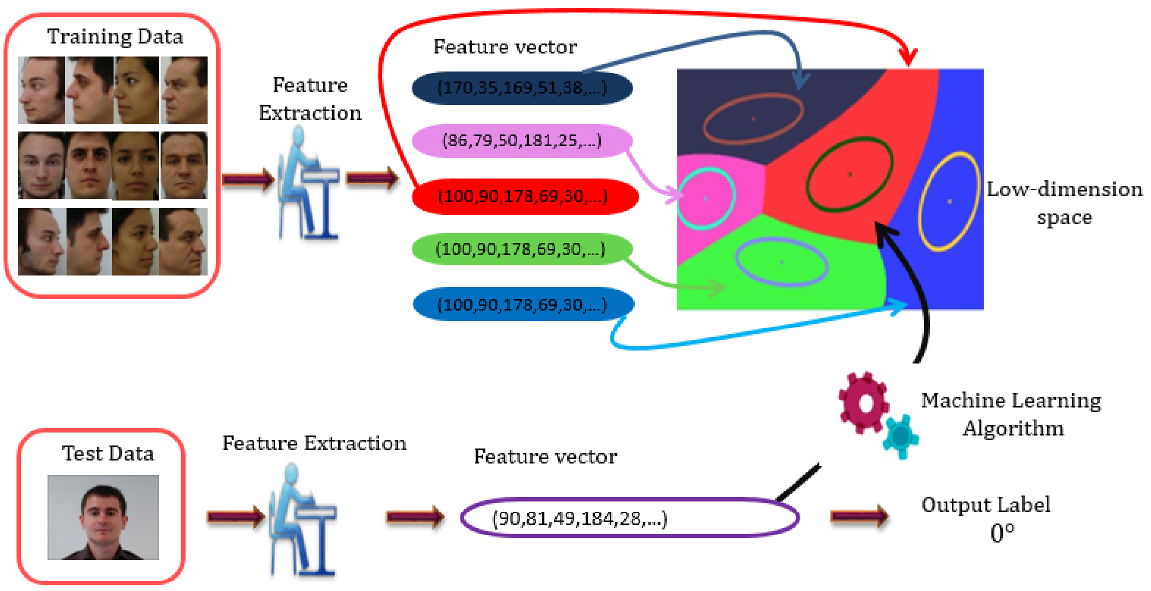


Figure 2.15 – *Illustration of the manifold embedding models.*

extensions of these manifold learning algorithms developed for head pose estimation or the taxonomy presented [BenAbdelkader, 2010] for a taxonomy of such techniques. Generally, these methods either treat the problem as a complex nonlinear embedding process, or by using linear approximations of nonlinear techniques using unsupervised or supervised algorithms to construct a simplified system. Balasubramanian et al. [Balasubramanian et al., 2007] enhanced these manifold embedding techniques by using a Biased Manifold Embedding (BME) framework. BME uses a Generalized Regression Neural Network (GRNN) to learn the complex nonlinear mapping from the high-dimensional feature space to the low-dimensional embedded space. Another extension of the classical algorithm, Wang et al. [Wang et al., 2008] used Isometric feature mapping (Isomap) to increase the level of oversight of traditional methods by the implementation of a Fisher Local Discriminant Analysis. BenAbdelkader [BenAbdelkader, 2010] combined the Neighbourhood Preserving Embedding (NPE) and Locality Preserving Projection (LPE) which are deemed linearized versions of LLE and Laplacian Eigenmaps (LE) respectively. Huang et al. in [Huang et al., 2011b] proposed Supervised Local Subspace Learning (SL^2) that builds local linear models from a sparse and non-uniformly sampled training set. SL^2 learns a mixture of local tangent spaces that are robust to under-sampled regions for continuous head pose estimation so that it can be robust to overfitting and to noise. Foytik and Asari [Foytik and Asari, 2013] employed two layers for head pose estimation. They applied linear discriminant analysis (LDA) in the first layer to localize the given observation to a determined region of the pose manifold and establish coarsely pose estimation. Canonical correlation analysis (CCA) allows producing a linear transform for fine pose estimation in the second layer. Peng et al. in [Peng et al., 2014; 2015] proposed multiple manifolds as well, where Homeomorphic Manifold Analysis (HMA) is used to construct a separate manifold and learns the mapping from the low-dimensional uniform geometry representation to each instance manifold (the high-dimensional feature space) as shown in figure 2.16. After that, the mapping coefficients matrices are arranged as a tensor and perform decomposition along the instance direction to separate the pose-related/unrelated factors. The pose of the testing image is achieved by parameterizing it in the instance parametric space and search the domain of uniform geometry representation corresponding.

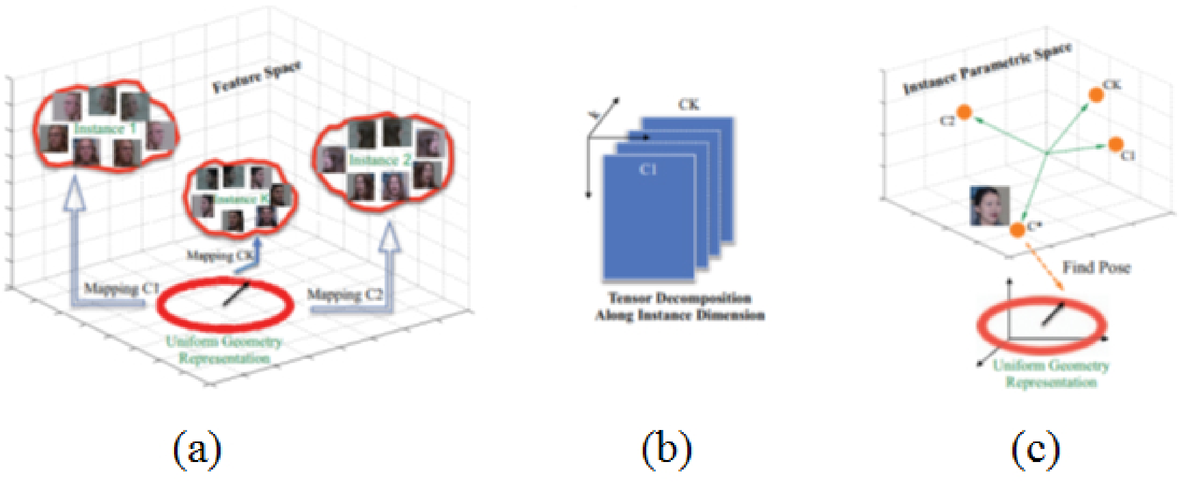

Feature Space
Instance 1
Instance K
Instance 2
Mapping CK
Mapping C1
Mapping C2
Uniform Geometry Representation
CK
C1
Tensor Decomposition Along Instance Dimension
Instance Parametric Space
C2
CK
C1
C*
Find Pose
Uniform Geometry Representation
(a)
(b)
(c)

Figure 2.16 – *Illustration of the training and learning procedure of approach proposed in [Peng et al., 2015].*

Synchronized submanifold embedding (SSE) is combined with random regression forests to estimate the head pose in [Zhu et al., 2014]. SSE and random regression forest methods are used to map the nonlinear data into linearly separable low-dimensional data. Interpolation techniques are used as well to identify the missing range of pose values. Liu et al. [Liu et al., 2014] introduced supervised locality discriminant manifold learning (SLDML) to decrease the effects of the unrelated pose. The SLDML approach is executed in two phases: the graph embedding stage, and the regression stage. The manifold learning and the regression are incorporated with a learned discriminant manifold-based projection function yield by discriminatively Laplacian regularized least squares to learn a low dimensional space represented the continuous head pose angles. The Supervised Neighbourhood-based Fisher Discriminant Analysis (SNFDA) algorithm is proposed in [Wang and Song, 2012; 2014]. The authors presented a framework that permits to keep the input data with similar pose angles close together, although the input images with dissimilar pose angles have been very far apart. This framework incorporates the pose angle information into three stages of manifold learning. Before, the supervised neighbourhood-based Fisher Discriminant Analysis (FDA) was used to constrain the projection learning the inter-point distance for neighbourhood construction is redefined as well as graph weight. A similar approach was adopted in [Wang et al., 2015] where supervised Laplacian regularization is combined with sparse regression into manifold learning to determine the head orientation. Recently, an unsupervised pose estimation method for face images based on clustered locally linear manifolds using discriminant analysis was reported by Hari et al. in [Hari and Sankaran, 2017]. The authors carried out a multilayer nonlinear manifolds framework to abstract discriminant functions. In the first layer, the clustering process performed on the entire data set to obtain the pose-dependent classes for training, with the aim to supply the discriminant features. The benefit of this step is to divide the manifold into small regions of focus, where the manifold can assumed to be linear now. In the second layer, unsupervised smaller classes are yield from another clustering approach that is proceeding on these local regions of interest. Then, these classes are used to extract a second set of discriminant features that can be used to obtain the ultimate pose estimation model. More recent, Diaz et al. [Diaz-Chito et al., 2018] proposed a continuous head pose estimation system based on manifold learning-based methods. The histogram of Oriented Gradients (HOG) is extracted from the image as a preliminary feature extraction step. Subsequently, the HOG features mapping onto a feature manifold using Generalized Discriminative Common Vectors (GDCV). Then a continuous regression composed of spline fitting and multivariate local regression is learned to build the final head pose estimation model, while Derkach et al. [Derkach et al., 2018; 2019] estimated the head pose by using multi-linear or tensor decomposition to split the pose variation factors into separate sub-spaces and demonstrated that each angle orientation (i.e. pitch, yaw, and roll) has its own structure. Thus, cosine functions enable to model variation factors from a unique shared parameter per angle.

2.4 Discussion

We mentioned in the previous section several aspects of the various techniques and methods published to overcome the head orientation problems. In this section, we will provide the most advantages and disadvantages of each method, as illustrated in Table 3.1. As well as some suggested approaches in order to proliferate the performance.

As it is known, no approach or system performs with perfect efficiency, and there are drawbacks that decrease the performance, so all methods proposed in this field of research attempt to compensate and minimize these limitations. For example, the first researches introduced to determine the head orientation were the methods using the templates [Beymer, 1994] which seek to classify a test face image by matching it with other labelled annotated images. That is based on the assumption that the faces of a discrete pose are almost similar, which is not necessarily exact in all cases. Whereas the similarity between different poses for the same person can be more than two face images of different people in the same pose. To carry out this matching, it is necessary to compare the query pose with all images in the database, which makes the process more time-consuming. However, due to the serious limitations of these methods, they are almost abandoned.

Afterward, classification methods [Huang et al., 1998] have been developed to reduce the computational time. On the contrary to the appearance template method, the training images in classification methods are grouped into classes with corresponding labels of the head angle. Thus, the test face image is correlated with the features of each class to deduct the head pose. These methods are quite simple to implement, and they are not affected by the identity, and they considered more accurate than the appearance template method. In fact, this technique is usually limited. Moreover, it is required to train large samples to achieve powerful classification accuracy, which is hard and computationally expensive. However, a common solution to compensate for this problem has been developed based on deep learning networks, which will be detailed later.

Table 2.1 – *Comparison of various head pose estimation methods.*

Method	***Advantage***	***Disadvantage***
Flexible Models	• Invariant to head localization error • Invariant to the appearance of a human face • less inter-subject variation	• Computationally expensive • Sensitive to occlusions • One model is unable to present all head precisely • Unsuitable performance with low resolution images
Geometric Method	• Simple and fast • A good accuracy with few information	• Error in the location of the features decreases the accuracy significantly • Sensitive to occlusions • Weak precision with low resolution
Template Methods	• Database can be expanded easily • Insensitive to resolution • No negative training samples and facial landmarks are required	• Computationally expensive for big databases • Pairwise similarity does not significant necessarily the pose similarity
Classification Methods	• The same classifier can be simultaneously detected and estimated the head pose • Independent to resolution • Appearance variations do not affect the training algorithms • Robust to large face pose changes.	• Computationally expensive. • The number of classes limited by the number of detectors • Required negative images to learning (images with no heads)
Regression Methods	• High accuracy and fast • The cropped labelled faces are only required to training • Insensitive to zoom • Dimensionality Reduction	• Difficult to develop an exact function for robust head pose estimation • The complexity of the non-linear and linear mappings that connect the facial images and pose labels
Manifold Embedding Methods	• A good precision • Embedding can be achieved through a simple matrix multiplication • Dimensionality Reduction	• The heterogeneity of the training data • Limited of representational capacity for nonlinear techniques • Hard to find the optimal dimensions that represent the continuous variation in head pose
Deep Neural Networks	• Very powerful accuracy • Features extraction automatically • Reduction of dimensionality	• Hard training • Computationally expensive • Difficult to choose the hyper-parameters (filters size, batch size, the number of epochs...) that allow to avoid the over-fitting status

The geometric method [Gee and Cipolla, 1994] is one of the first attempts that has been proposed to accelerate the head pose estimation process. The head orientation is inferred by assuming fixed geometric relationships between the key-points and comparing the position of these points with an average model acquired through anthropometric measurements. Although these methods are simple and run fast, the accuracy is correlated with the quality and quantity of the geometric cues inferred from the image. Therefore, the major drawback of the features-based approaches is the necessitated of high precision in facial key-point detection. It is hard to estimate head poses from images with occluded face regions.

However, the recent advent of depth cameras and 3D acquisition systems have led to enriching the geometric information and supply the head shape more distinctively. As well as, it allows the development of large annotated databases [Fanelli et al., 2011, Koestinger et al., 2011, Ariz et al., 2016, Lüsi et al., 2016].

Typically, depth (2.5D) and 3D data provide less sensitivity to light changes, noise, and partial occlusions, which help to improve the weakness in using RGB image only. The use of these alternatives modalities to carry out geometric models has significantly progressed. For example, Gurbuz et al. [Gurbuz et al., 2012] extracted a face plane and the eye positions in 3D and uses this information to develop the head pose matrix uniquely, and defining the orientation and the position of a face and used it to estimate head pose. Rely on the visibility of the nose region, which is least occluded by accessories and confirmed to be greatly discriminant in all poses from profile to frontal. Cai et al. [Cai et al., 2012] suggested an automatic method using the 3D nose tip location, and it is normal to estimate head pose. Further, Li et al. [Li et al., 2018b] used the nose tip and the nose bridge to estimate the face pose. To determine these notables features, the authors relied on the salient crest lines that were closely connected with the 3D facial skeletons of convex regions. Li and Pedrycz [Li and Pedrycz, 2014] benefited from the bilateral symmetry of the human face to construct a central profile-based 3D face pose estimation model. The central-profile is the resulting curve from the intersection between the symmetry plane and the 3D face surface. That starts from the forehead centre, goes down through the nose ridge, nose tip, mouth centre, and ends at a chin.

Baltrušaitis et al. [Baltrušaitis et al., 2012] combined depth information alongside the intensity to enhance the Constrained Local Model. The employment of depth data helps to detect the facial features when an intensity signal is not available, or the lighting conditions are insufficient. Moreover, Ackland et al. [Ackland et al., 2019] have developed the 2.5D Constrained Local Model (2.5D CLM) for head pose estimation, which is combined a deformable 3D shape point model with 2D texture information to provide a direct assessment of the pose parameters.

The use of the 3D Morphable Model (3DMM) is another way to exploit the 3D data pioneered by Vetter et al. [Blanz and Vetter, 1999]. A 3DMM is defined as a dense statistical model of 3D face geometry and texture information that is transformed into a vector space representation. It is a direct extension of the 2D Active Appearance Model (AAM) that allows for more efficient modelling in the presence of pose and illumination variations. The first version of 3D faces representation is relied upon Principal Component Analysis (PCA) to describe face shape and texture variations. The 3DMM can be fit to 2D image data, or depth, or 3D face data to adapt the model to the subject similar to AAMs. Egger et al. [Egger et al., 2020] provided a detailed survey of 3D Morphable Face Models.

To date, the Basel Face Model (BFM) [Paysan et al., 2009] is the publicly accessible variant of the morphable model, and that is the most widely used representation for 3D face shape. The BFM has various applications such as face recognition, face analysis, and 3D face reconstruction [Kittler et al., 2016], thus the 3D head pose estimation. Among the published papers based on the BFM model to estimate the head orientation, there was an approach proposed by [Cai et al., 2015b] which employs the shape model of the Basel face model and five landmarks on the 2D face image (left and right eye centres, nose tip, and left and right mouth corners). The fitting of the shape is preformed according to Laplace distribution to be able to pick up the shape variation across different persons. They implement a particle swarm (PSO) optimization algorithm to minimize the fitting error between the five feature points on the 2D input image and 3D shape model. While Meyer et al. [Meyer et al., 2015] fitted a morphable face model to the measured depth data to affect the head pose estimation. To register the reference model to the input data, they combine the iterative closest point (ICP) and PSO to avoid local minima due to the poor initialization from ICP, and to speed-up the convergence rate of the PSO algorithm.

Usually, 3 MMD models represent only the frontal face region, which decreases the performance with extreme head poses. Spurred by this lack, Yu et al.[Yu et al., 2017; 2018] extended this model with an online 3D reconstruction of the full head that can be suitable to deal with large head pose variations. Instead of modelling the complete 3D head, Ghiass et al.[Shoja Ghiass et al., 2018] based on the subject's face to infer the head pose through a fitting process with a 3D morphable model and depth data only rather than the 2D image.

Besides, other alternative approaches associated the usefulness of the high-resolution RGB image with the 3D information. These techniques use a predefined 3D facial model to fit a single RGB or depth face image. Usually, these methods localize the facial landmarks using the RGB image as the first step. Then these key points are combined with 3D features. Finally, the head pose is estimated by minimizing the re-projection error from the 3D-2D correspondences [Li et al., 2018a, Barros et al., 2018, Madrigal and Lerasle, 2020, Yuan et al., 2020].

A recent approach suggested by Barra et al. [Barra et al., 2020] proceeded from a web-shaped model to develop a feature vector to infer the head pose. Each detected facial landmark is associated with a specific face sector within the model. The orientation of the input image is performed through the reference vector that is more correlated to the pose feature vector. The reference vector is extracted from the reference prototypes, which are built from the 3D model of a standard synthetic head that is automatically rotated along all three axes.

Although the introduction of 3D data to tackle the head pose orientation issue allows yielding a model with high accuracy, the application of these data remains suffering from some limitations. It is still tougher to label 3D facial landmarks on 2D images or 3D face scans. The lack of a 3D facial database with a multitude of 3D annotations compared to the highly available 2D database [Wu and Ji, 2019]. Further, the 3DMM fitting algorithms require accurate initialization of certain parameters and are sensitive to outliers and partial face occlusion. As well as, the fitting procedure is computationally expensive. Moreover, the depth image has some handicaps, such a lot of noise at the edge. Also, it is hard to attain invariant features because the depth information depends only on distance. Furthermore, 3D scans and RGB-D sensor technology are not widely used in real-life applications such as a webcam or video surveillance.

The purpose it is still always focusing on the improving of the performance of such a head pose estimation model, suggested dimensionality reduction methods have received considerable attention in this domain. These approaches attempt to overcome certain drawbacks of the other techniques, mainly the time-consuming ones. Relying on the assumption that a high-dimensional feature space can be presented with few dimensions that describe the pose changes. The objective of these methods is to reduce the search space. Either by a regression model as a function, which determines the relationship between the appearance of a face and its poses. Or by using an alternative strategy to model the continuous variation of the head pose, based on manifold embedding techniques. Indeed, these methods are fast and achieve high accuracy. However, this approach is not enough accurate to predict head pose on a large database. They have somehow been limited by some problems. Due to the complexity of the linear/non-linear mappings that relates to the facial images and pose labels, it is difficult to learn an exact function for robust head pose estimation. Further, not all variations that may appear in the face are due to the pose variation only.

Nevertheless, a multitude of ways has been developed to surmount these issues. Usually, these methods leverage the robustness of two approaches or more to construct a model that can run with them jointly. For example, The integration of the strengths of manifold embedding methods and regression methods have been suggested. The manifold-based approaches lead to building discriminative sub-spaces. That maximized the distance between classes of poses with very different angles while minimizing the distance between the samples with the very close poses, intending to reach an effective regression later. Wang et al. [Wang et al., 2015] proposed a Supervised Sparse Manifold Regression (SSMR) method that combines the supervised graph Laplacian regularization and the sparse regression into manifold learning. The supervised Laplacian regularization allows manifold learning to maintain the preservation of local geometry structure to yield more discriminative projection embedding, while the sparse regression can create a projection mapping matrix to perform a sparse representation of the high-dimensional features to improve the feature extraction capacity. As well as Diaz-Chito et al. [Diaz-Chito et al., 2016] used various geometric features combined with two manifold embedding methods and linear regression to predict head pose orientation. To remove the samples that do not represent the pose variations, the authors relied on a small set of geometric features computed from just three representative facial key-points. While in [Diaz-Chito et al., 2018], they proposed to combine manifold learning alongside regression methods. The Generalized Discriminative Common Vectors (GDCV) technique grants the projection of the features onto a feature manifold space. The features that discriminate the pose orientations are extracted using the Histogram of Oriented Gradient (HOG) descriptor. Afterward, the head pose is inferred from a continuous regression composed of split fitting and multivariate local regression.

As the classification methods are robust to large head pose changes, and the regression methods are sensitive to a small change of head pose, the incorporation of classification and regression methods has been used to enhance the head pose estimation because performing the regression technique solely to handle large head angle rotation raises the model complexity. Liu et al. [Liu et al., 2017] combined random forest classification and regression into a multi-level structured hybrid forest (MSHF) system. Wang et al. [Wang et al., 2019b] proposed a cascaded structure for face pose estimation, which can perform classification and regression concurrently. First, they trained the model to classify the input image into four categories defined by the authors to narrow down the estimation range. Next, they employed a regression method on the output of this classifier to determine the final head angle rotation. In the same way, Huang et al. [Huang et al., 2020] carried out a model that combined classification and regression, but in this case, they tackled the methods separately into two subtasks. The first stage is a binned classification subtask with the optimal pose partition. The second stage achieves average top-k regression based on the former prediction.

Another work from Tan et al. [Tan et al., 2018] used the geometric method and the regression together on RGB-D data to upgrade the head pose estimation. Luo et al. [Luo et al., 2019] utilized a random forest classification and regression independently and a 3D model on the depth image. Whilst, Abate et al. [Abate et al., 2020] applied a regression algorithm that is applied to the Web-Shaped Model (WSM) algorithm [Barra et al., 2020], to determine the head pose estimation.

Despite all these works, the problem of research of the optimal dimension, which reliably considers only variations from pose and ignoring the others, is still always open.

In light of the above, the most common lack of both feature based methods and appearance-based methods is feature extraction. Deep neural network methods can overcome the challenges of feature extraction, which can automatically extract more discriminative features for face pose estimation.

In a recent advance, deep neural networks have been widely investigated for head pose estimation, and more eminently, a deep convolutional networks (CNNs). CNNs can be a perfect tool for processing multidimensional data. They are particularly suited to image processing. The Tasks-Constrained Deep Convolutional Network(TCDCN) [Zhang et al., 2016; 2014], is one of the first works on the problem of head pose estimation using CNN. The authors learned multi-task architecture where the facial landmark detection is the principal task, whereas the head pose is considered as an auxiliary task. Venturelli et al. [Venturelli et al., 2017] used Siamese CNN to improve the loss function of neural networks which has been fed by depth images. Patacchiola et al. [Patacchiola and Cangelosi, 2017] trained different CNN models in a wild environment. Ruiz et al. [Ruiz et al., 2018] proposed to learn multi-loss deep network on synthetic data to predict intrinsic Euler angles from image intensities by using joint binned pose classification and regression. Lathuiliere et al. [Lathuiliere et al., 2017], suggested a hybrid model based on CNN and Gaussian mixture. Borghi et al. [Borghi et al., 2017] are considered as the pioneers who used the recurrent network to estimate the head pose. They have created a system that consists of 5 convolutional layers and 3 max-pooling layers followed by 2 LSTM (Long Short Term Memory) in which, the depth images are the input data and the continuous 3D Euler head angles are the output. Xia et al.[Xia et al., 2019] feed the facial landmark and landmark heatmap into a CNN to enhance the performance of the head pose estimator. Gupta et al.[Gupta et al., 2019] take the five facial key points: left ear, right ear, left eye, right eye, and nose, and input them into a convolutional neural network to regress the head pose. While Xu et al. [Xu et al., 2019] built soft labels using a Gaussian distribution function from entire images, then feed them into a convolutional neural network which combines the Kullback–Leibler divergence loss and Jeffrey's divergence loss as optimizers. Hsu et al. [Hsu et al., 2018] suggested to estimate the head pose a CNN model combined with multi-regression loss function to avoid the ambiguity problem in Euler angles. Vo et al. [Vo et al., 2019] proposed a method called SAE-XGB, SAE for Stacked Autoencoder, and XGB for Extreme Gradient Boosting. To reduce the dimensionality of feature vectors, the authors have stacked multi-fully connected hidden layers with different hidden units to build a Stacked Autoencoder (SAE). The autoencoder is an unsupervised learning neural network used to encode the global features extracted from the Histogram of Oriented. Then, a supervised learning model named XGB classifies the output of this autoencoder in two classes, pitch and yaw angle. To improve the performance and deep metric learning, a convolutional neural network is trained to learn head features under the joint supervision of classification and regression loss [Wang et al., 2019a, Dai et al., 2020, Sun and Lu, 2020]. Based on convolutional neural networks and geometric projection, Gao et al. [Gao et al., 2020] proposed a method to infer the face pose as well as the more frontal face of the input images. This method performs in two stages. At the first stage, the face images are classified into five categories through the CNN model. Secondly, geometric projection is applied to reach the offset angles and scores of the face in the three directions of roll, yaw, and pitch. Finally, the final score of face poses yields by combining the decisions of the two previous stages. The higher score defines the more frontal face pose.

According to panoramic of state of the art, we can mention that the feature-based approaches are speedy and present excellent performance, particularly in frontal images or small angles. The difference in the appearance of human faces is not affected. These methods are still sensitive to partial occlusions that obscured some landmarks, and they are not efficient for certain degenerative angles. Furthermore, these facial points are non-uniform for all human faces. Therefore, only one model has been unable to represent all the heads precisely. Contrariwise, the appearance-based approaches are quite efficient and relatively simply implemented. Since these approaches have used the full image of the head, these methods are less sensitive to partial occlusions and extreme angular views. They have attracted more and more attention, especially the dimensionality reduction methods that achieved high accuracy. However, the problem of facial features extraction without other changes in the image is still a hot topic and an open research problem.

Besides, thanks to the successful results that have been achieved relying on deep neural networks. In the recent years, the majority of contributions to estimate the head pose have been based on machine learning approaches through deep neural architectures. Nevertheless, the computational cost of deep learning is very expensive due to the high dimensionality of the parameters and the features that are calculated in training (the dimension of the weights is in the order of billion). The dimension of these parameters increases linearly with the dimension of the input data.

2.5 Conclusion

In this chapter, we first presented the set of head pose estimation methods that are mainly used or that have led to significant progress in the field. The first category depends basically on local information of the facial region, which is known as feature-based methods. The second category is based essentially on the global characteristics of the face, which is named appearance-based methods. Subsequently, we have discussed the strengths and weaknesses of each approach, as well as the proposed state-of-the-art attempts to overcome the limitations. However, further research is involved to reach the state-of-the-art performances.

In the next chapter, we will present our attempt to address these limitations, and we will expose our approach based on the use of dimensionality reduction through time series. The method is robust and fast, we narrow down the solution space from the 2D to 1D space thanks to the space-filling curve and time series representation.

3 Representation of the time series for head pose estimation

Contents

3.1 Introduction

The face is the most commonly used biometric parameter for human recognition. Although great progress has been achieved in face detection, it is still hard to estimate its poses. Head pose estimation has emerged as an important field in computer vision.

Usually, the head pose estimation problem can be handled in two main steps: feature extraction and classification model creation. Various techniques and descriptors have been used to extract a representation that can discriminate the poses of the face in an image, such as scale-invariant feature transform (SIFT) [Alioua et al., 2016], histograms of oriented gradients (HOG) [Wang et al., 2013a], speeded up robust features (SURF) [Bay et al., 2008], and the Haar-like features [Jones and Viola, 2003]. Then supervised algorithms used these features to learn a model for estimating the head poses such as support vector machine (SVM), and neural networks (NN).

Despite a lot of existing methods that have provided acceptable classification results, they are often complex to implement and computationally costly. Frequently, the complexity arises from the way that is used to extract features. Starting from these bottlenecks, we propose SAX2FACE in this thesis, a simple and efficient alternative solution that suggests a method to extract the features. SAX2FACE relies on the usefulness of extracting knowledge through time series and used a time series dimensionality reduction method (SAX) to address the problem of facial pose estimation. Thus, the head pose estimation becomes a time series classification task.

This chapter revolves around five sections. First, we briefly introduce the underlying concepts of the time series data mining and their representations, then providing sufficient knowledge background about symbolic aggregate approximation (SAX). Next, we show how to represent the face image as times series, while the use of space-filling curves allows us to yield this aim. Subsequently, we convert the face image time series to symbolic sequences using the SAX technique to build classifiers models that classify these time series, thus the face pose is inferred. Finally, we evaluate the efficiency of the methods under diverse protocols on several databases.

3.2 Time Series Data Mining

The technological advance achieved in the last decades leads to collect and tackle a huge bulk of data in many fields of business and science. The fast progressing computing power and the lessening costs of high volume data storage released the collection of enormous amounts of data.

Classical statistical analysis methods can't analyse these enormous amounts of data. Data mining is the discipline that encloses both the statistics rules and computer science to handle and analyse this information to become meaningful and useful. In other words, it is the result of the hybridization of statistics, computer science, artificial intelligence, and machine learning, to explore hidden and relevant information previously unknown through this data and provide important information as it is new, valid, and useful to the expert user.

One of the most common and a special kind of data in data mining, which presents a great interest to researchers in this field, and a lot of endeavours are put into understanding and analysing it, known as Time Series. Time series are sequences of numerical values collected nearly everywhere and every day and typically organized in non-random orders. Time series is the data model that can commonly be omnipresent in various fields, such as medicine e.g., electrocardiogram (ECG), finance (daily fluctuations of the stock market), multimedia, meteorology (variation in temperature daily, monthly or yearly), Bioinformatics, pattern recognition, text mining, computer vision, and others.

The high dimensionality is a major challenge that hampers the analysis of time series. Dealing with the time series in the original format is very costly in terms of both processing and storage. To overcome this obstacle, one can put the focus on finding the best representation, that can transform the original data to another reduced space. Therefore, the high-level representation requires the reduction of dimensionality with the preservation of the fundamental information of the original database.

3.2.1 Time Series Representation Methods

The leading motivation spurred to represent a time series than using the original values is to provide a concise display and clear notion that emphasizes the main features. Further, benefits can speed up the processing time, so that if a time series X of original length n to $m \ll n$ is represented in reducing dimensionality, the computational complexity can be reduced from $\mathcal{O}(n)$ to $\mathcal{O}(m)$. Undoubtedly, the selection of a suitable time series representation is related to the way that has the utmost preservation of local and global information of the original time series. In another meaning, the representation should allow minimizing modelling error(Lower Bounding), i.e. when two-time series are transformed into a new space, the distance between them should be less than or equal to the distance in the original space, i.e. $D_{LB}(Q', S') \leq D(Q, S)$ as illustrated in figure 3.1.

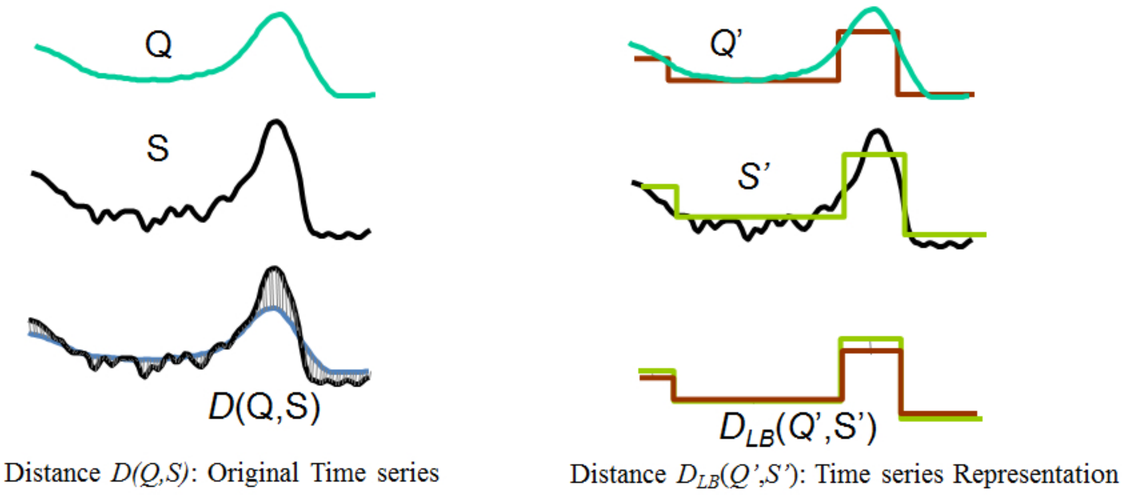

Q
S
D(Q,S)
Distance D(Q,S): Original Time series
Q'
S'
D_LB(Q',S')
Distance D_LB(Q',S'): Time series Representation

Figure 3.1 – *Illustration example of the lower bounding property.*

To achieve these goals and properties, the time-series data mining community developed many representations and techniques that can address these requirements properties. These representations methods transform and reduce time series either into a smaller number of numeric coefficients or into a discretized or symbolic representation, and can be grouped into three main categories: *data-adaptive, non-data-adaptive, or model-based*.

3.2.1.1 Non-data-adaptive

Dimension reduction methods are considered as non-data-adaptive representation when the time series transform into a different space, and the selection of a subset of the coefficients is independent of the data set. The transformation parameters are fixed a priori and remain the same for all time series regardless of their nature. One of the first works on this subject was carried out by Agrawal et al. [Agrawal et al., 1993], who pioneered the use of a discrete Fourier transform (DFT) to compress the time series. Agrawal et al.[Agrawal et al., 1993] decomposed the time series into sinusoidal waves through the Fourier coefficients. They showed that the first few waves are enough to represent the time series and can drop the rest without any impact on the reconstruction error. As well as the Discrete Wavelet Transform (DWT) [Chan and Fu, 1999], transforms the time series into a frequency domain, and take into account the local and global shape to provide the final representation of time series, where the DWT processes the series at different scales and resolutions of a mother wavelet function. The Piecewise Aggregate Approximation (PAA) [Keogh et al., 2001], is a completely different approach that proposed to generate time series representation by dividing the time series into w equi-length windows and calculating mean value of the subsequences in the corresponding windows (Details of this method will be presented in the sub-section 3.3.2).

3.2.1.2 Data-adaptive

Unlike the previously mentioned methods, the transform parameters in the data-adaptive techniques, are not fixed a priori but are chosen depending on the available data. However, all non-data-adaptive approaches can be transformed into data-adaptive ways by adding data-sensitive proceeding schemes. As an example, Chakrabarti et al. [Chakrabarti et al., 2002], proposed Adaptive Piecewise Constant Approximation (APCA) as an adaptive version of PAA. Another simple representation is the Singular Value Decomposition (SVD) [Korn et al., 1997], which carries out the final time series representation from the linear combination of the basic shapes that best represent the original data through a global transformation of the entire database, and rotates the axes to maximize variance over the first few dimensions. There are different manners to reduce the dimensionality of time series that transforms it into symbolic sequences, the most popular one of them is the method introduced by Lin et al. [Lin et al., 2007], called Symbolic Aggregate Approximation (SAX).The underlying logic of SAX will describe in section 3.3.

3.2.1.3 Model-based

Another group of representations of time series are methods based on a model. These methods depend on the assumption that the time series have been produced by a certain model. Dimensionality reduction is achieved by fitting the model to the data to infer the parameters that generate the time series.There are several approaches using parametric temporal models such as statistical modelling via feature extraction [Nanopoulos et al., 2001], or the ARMA and ARIMA models[Kalpakis et al., 2001]. More sophisticated approaches include Markov Chains or Hidden Markov Models (HMM) [Panuccio et al., 2002].

More time series representations are described in the published works such as [Fu, 2011, Kleist, 2015, Wang et al., 2013b].

3.2.2 Distance and Similarity Measures

One of the most important reasons for dimensionality reduction methods of the time series is to simplify and expedite the similarity measures task. Similarity measures are the backbone and the bottleneck of time series data mining, as well as, it is fundamental to manipulate and compare the time series. Indeed, the similarity measure means measuring the distance between time series, which permits identifying the similarity (or dissimilarity) of two-time series, thus being used for any search, clustering, or classification process.

The most common methods for numeric time series similarity computation are Euclidean distance and Dynamic Time Warping (DTW)[Ratanamahatana and Keogh, 2004]. Euclidean distance is the most widely used measure to estimate the distance between two time series x and y.

$$D_{euclide}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.1}$$

Despite Euclidean distance presents several properties, in addition to its simplicity, and its efficiency in many cases, but Euclidean distance has the drawbacks that make it unsuitable to use for some applications. Therefore, it is used only to compare two-time series of the same length. Further, it is also highly sensitive to time distortions, such as shifting, uniform amplitude scaling, uniform time scaling, uniform bi-scaling, time warping, and non-uniform amplitude scaling.

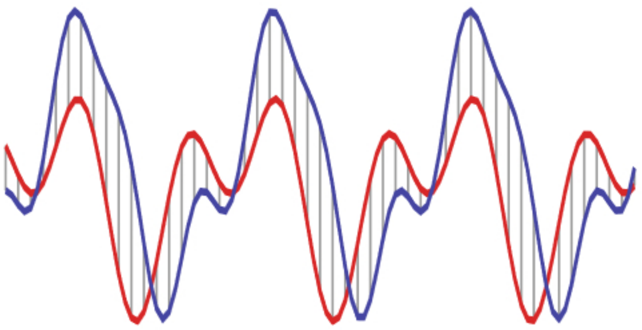


Figure 3.2 – *Representation of Euclidean Distance between two Time Series.*

Meanwhile, the DTW allows overcoming the limitations of Euclidean distance by seeking the best alignment between two time series x and y of length N and

K respectively. The optimal alignment is achieved by calculating the shortest warping path in the matrix W of dimension $N \times K$ containing the distance between each point, such that: $D(i,j) = (x_i - y_j)^2$. The path that minimizes the warping cost $DTW(x,y)$ between x and y, is defined as follows

$$DTW(i,j) = D(x_i, y_j) + min \begin{cases} DTW(i,j-1) & \text{repeat } x_i \\ DTW(i-1,j) & \text{repeat } y_j \\ DTW(i-1,j-1) & \text{repeat neithe} \end{cases} \tag{3.2}$$

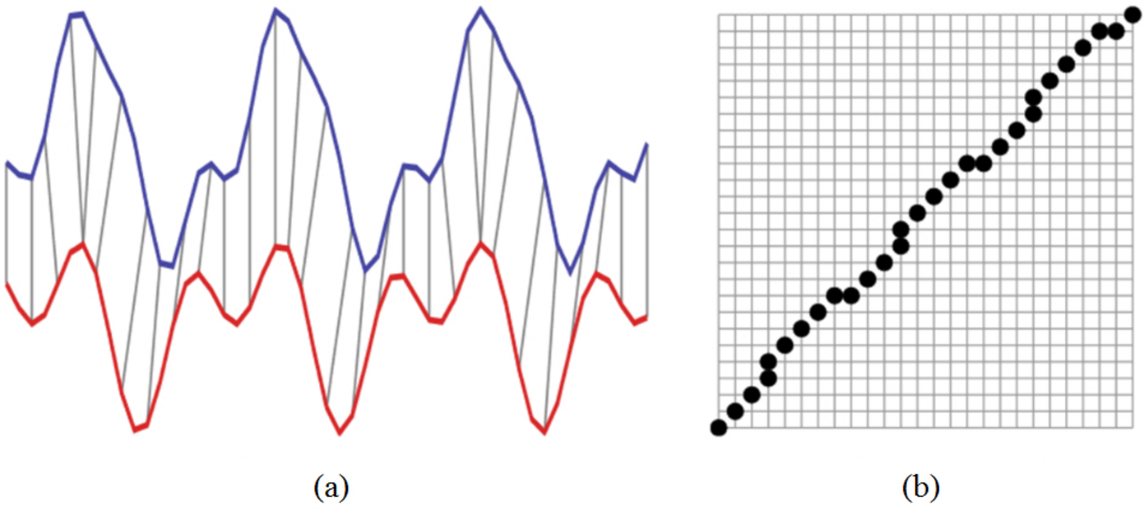


Figure 3.3 – *Representation of DTW distance between two time series and its optimal warping path. (a) DTW distance, (b) Warping path.*

Concerning the discrete or symbolic time series, other ways have been used. Among the well-known edit-based distances in time series field, we can cite: the edit-distance, also known as the Levenshtein distance [Ristad and Yianilos, 1998], and the Longest Common Subsequence (LCSS) [Das et al., 1997].

The Edit distance measured the similarity between two strings S and T, by determining the minimum number of operations' insertion, deletion, and substitution required to convert one string into another. So the distance is obtained by the following formula:

$$Edit(i,j) = min \begin{cases} Edit(i-1,j-1) & \text{if } s_i = t_j & \text{copy} \\ Edit(i-1,j)+1 & \text{if } s_i \neq t_j & \text{substitute} \\ Edit(i-1,j)+1 & & \textit{insert} \\ Edit(i,j-1)+1 & & \text{delete} \end{cases} \tag{3.3}$$

The LCSS seeks to find a common subsequence of all the sequences that is of maximal length. A subsequence is a sequence that appears in the same relative order but not required to occupy consecutive positions within the original sequences. The optimal matching between two strings S and T can be expressed by enumerating these cases:

$$LCSS(i,j) = max \begin{cases} LCSS(i-1, j-1) + 1 & \text{only if } s_i = t_j \\ LCSS(i-1, j) & \text{otherwise (no match on } s_i) \\ LCSS(i, j-1) & otherwise\ (no\ match\ on\ t_i) \end{cases} \quad (3.4)$$

A more suitable distance measure for discrete time series is the $MINDIST$ function, which was especially pioneered for SAX, and we will give more detail in sub-section 3.3.4

3.3 Symbolic Aggregate Approximation (SAX)

As we have mentioned earlier, the analysis of time series requires the reduction of dimensionality by transforming the input data into a lower-dimensional representation. One of the prominent used techniques is the Symbolic Aggregate approXimation (SAX).

SAX is a symbolic representation of time series which transforms a numerical series into a symbolic sequence, that performs dimensionality reduction with a minimum modelling error (Lower Bounding), as well as, it can significantly reduce computational complexity. This method runs in three steps:

1. Normalization
2. Window-based averaging
3. Value-based discretization

3.3.1 Normalization

Time series typically need to be normalized. Normalization is a rescaling of the data from the original to zero mean and standard deviation of 1, to keep away from comparing time series with different offsets and amplitudes.

Let T be a time series of size n such that $T = (t_1, t_2, ..., t_n)$, each element of T is with:

$$s_i = \frac{t_i - \mu}{\sigma} \quad (3.5)$$

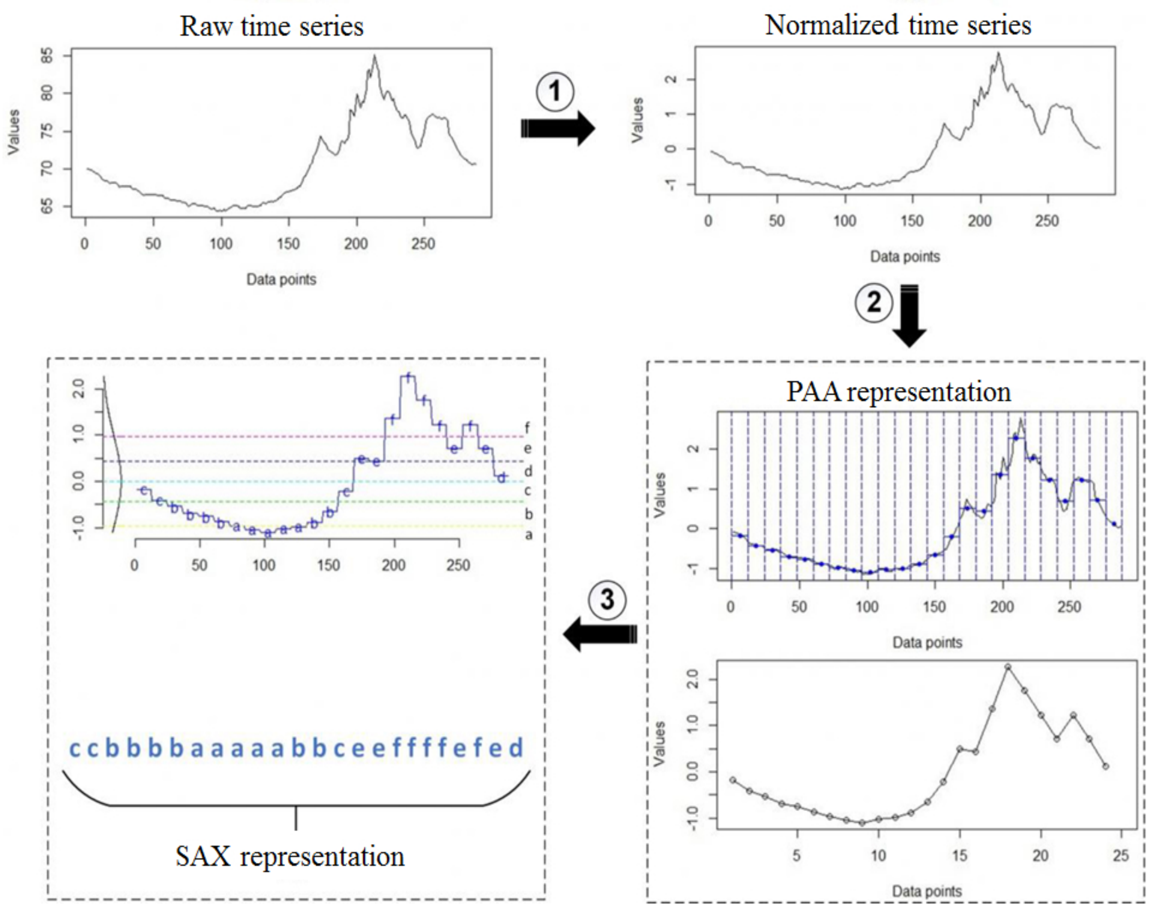


Figure 3.4 – *Illustration of SAX procedure.*

where μ is the mean value of all elements in T, and σ is the standard deviation.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} t_i \text{ , and } \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|t_i - \mu|^2} \tag{3.6}$$

And $S = (s_1, s_2, ..., s_n)$ denoted the normalized series S.

3.3.2 Window-based averaging

The main idea of SAX is to use a PAA (Piecewise Aggregate Approximation) [Keogh et al., 2001] representation as an intermediate step between the raw data and the generated symbolic sequence. This step used to divide the time series of length n into w windows of equal size (frame is also known as codeword (w)) and the average time-series value over successive and equally-spaced windows is computed, and a vector of these values becomes the data-reduced representation. The sum of these averages is based on the transformation series PAA.

As a reminder, the PAA coefficients are derived by slicing the normalized time series into w equidistant frame. The PAA converts the sequence S of size n into a new sequence, of size w with $w \ll n$, and it is denoted by $\bar{S}$, such as $\bar{S} = (\bar{s}_1, ..., \bar{s}_w)$, such that the i^{th} element of $\bar{S}$ is calculated using the following equation:

$$\bar{s}_i = \frac{w}{n} \sum_{j=\frac{w}{n}(i-1)+1}^{\frac{w}{n}i} s_j \tag{3.7}$$

The result of this approximation can be illustrated in figure 3.5.

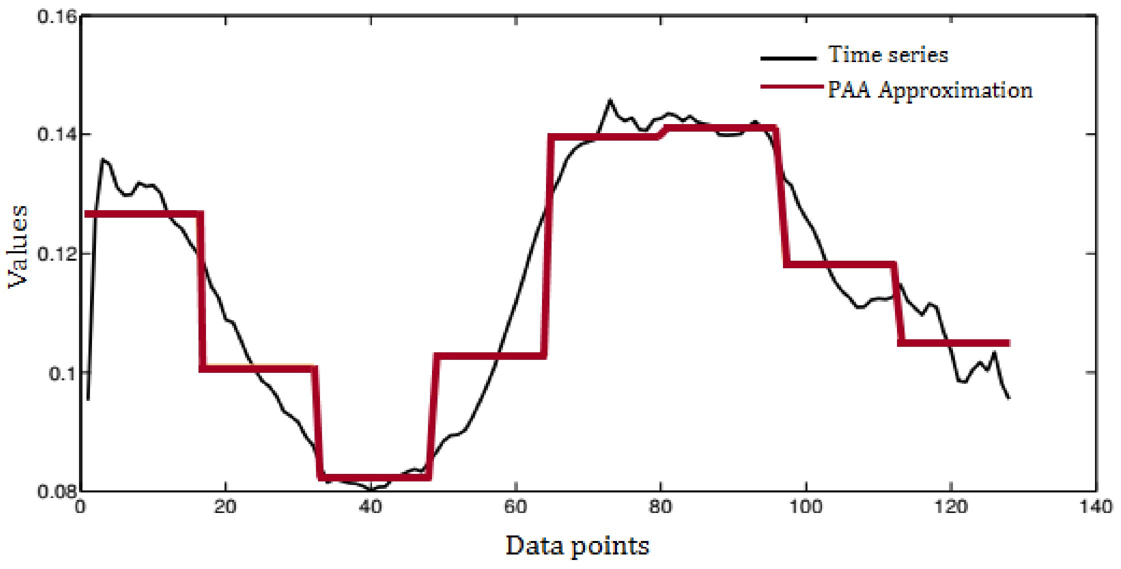


Figure 3.5 – *Example of the PAA representation to approximate a time series. The original time series T (in black) has a size n = 128 and the PAA approximation has a size w = 8 (in rad) [Flocon-Cholet, 2016].*

3.3.3 Value-based discretization

The last step is to convert the sequence PAA ($\bar{S}$) into a series of symbols, belonging to an alphabet A of size "a" (a is an alphabet size also known as a codebook), such that $A = \{\alpha_1, ..., \alpha_a\}$. To do this, it is necessary to define the $(a-1)$ breakpoints that divide the space of values, where each specific region will be associated with this a symbol α_i.

The SAX developers have shown that time series that are normalized follow a Normal distribution (Gaussian distribution). The normalized distribution can easily choose areas of equal size on the Gaussian curve, which defines the breakpoints [Lin et al., 2007]. Lin et al.[Lin et al., 2007] used a lookup table to determine breakpoints that divide a Gaussian distribution in an arbitrary number of equiprobable regions (the number of breakpoints β_i is related to the size of the alphabet "*a*", where *number (breakpoints) = alphabet size - 1*. Breakpoints are a sorted list of numbers $B = \{\beta_1, ..., \beta_{a-1}\}$, such that the area under an N(0,1) Gaussian curve from $\beta_i - \beta_{i-1} = 1/a$.

The subdivision of the normal distribution is illustrated in figure 3.6. In this figure, we can see the breakpoints defined for an alphabet of size 3, 4, 5, and 6 symbols.

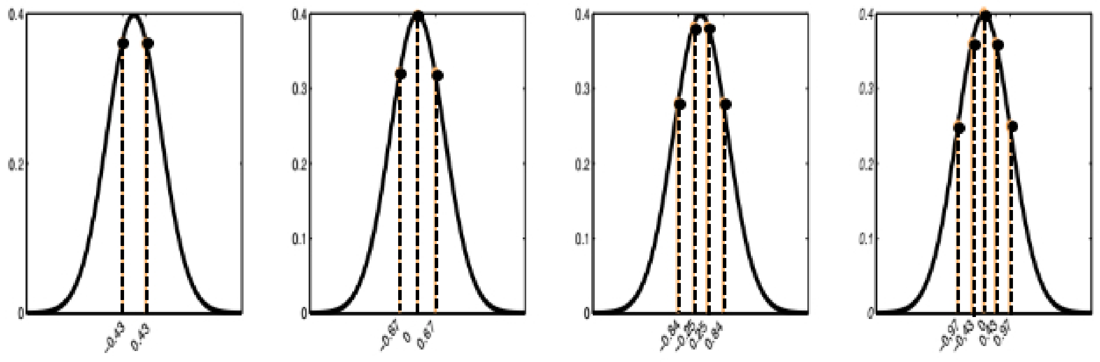


Figure 3.6 – *The breakpoints are defined so that each symbol has the same probability of occurrence. We illustrate here the cases where there are 3, 4, 5, and 6 symbols (from left to right) [Flocon-Cholet, 2016].*

A lookup table that contains the breakpoints is shown in Table 3.1.

Table 3.1 – *Lookup table that contains statistical breakpoints [Lin et al., 2007].*

a	3	4	5	6
β_1	-0,43	-0,67	-0,84	-0,97
β_2	0,43	0	-0,25	-0,43
β_3	-	0,67	0,25	0
β_4	-	-	0,84	0,43
β_5	-	-	-	0,97

Hereafter, the PAA representation is symbolized into a sequence of discrete strings. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of the PAA within this interval is discretized by this symbol. So all PAA coefficients that are below the lowest breakpoint are encoded by the symbol "*a*", then all PAA coefficients that are above or equal the lowest breakpoint and lower than the second smallest breakpoint are encoded by the symbol "*b*", the following symbol is "*c*" and so on, i.e. $\alpha_1 =$ "a", $\alpha_2 =$ "b", $\alpha_3 =$ "c", etc...

Figure 3.7 presents an example of SAX symbolization. The time series T of length $n = 128$ is converted into a SAX sequence $\hat{C}$, of size $w = 8$ with alphabet of size $a = 5$ such that $\hat{C} =$ "d", "b", "a", "b", "e", "e", "c", "b".

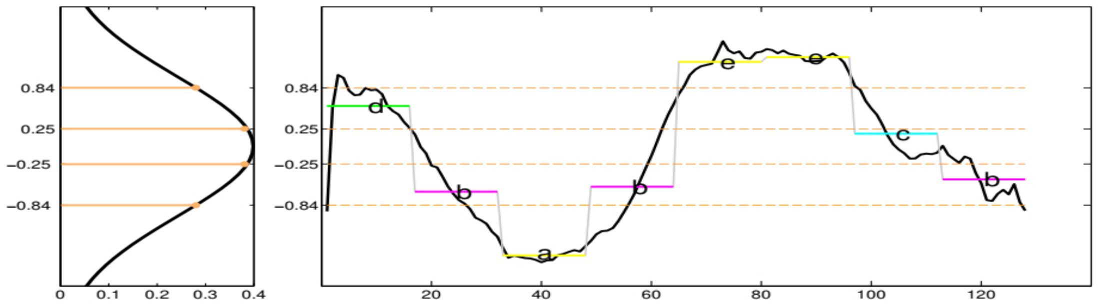


Figure 3.7 – *Example of SAX representation of a time series with the number of segments w equal to 8, and the size of alphabetic symbols a is 5. The orange dashed lines represent breakpoints [Flocon-Cholet, 2016].*

3.3.4 SAX distance function

Measuring the similarity is an important task of the time series data mining. One of the most positive aspects of SAX is that it represents lower bounding for Euclidean distance. To measure the similarity we use the following formula:

$$MINDIST(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w} dist(\hat{q}_i - \hat{c}_i)^2} \tag{3.8}$$

Where $\hat{Q}$ and $\hat{C}$ are the symbolic representation of numerical time series Q and C respectively. The *dist* function is implemented using the lookup table for the particular set of breakpoints as illustrated in Table 3.2 [Lin et al., 2007]. The distance *dist(r,c)*, between two SAX symbol values *r* and *c* is calculated by the following expression:

$$dist_{(r,c)} = \begin{cases} 0 & if \quad |r-c| \leq 1 \\ \beta_{max(r,c)-1} - \beta_{min(r,c)} & otherwise \end{cases} \tag{3.9}$$

Thus, the distance between any successive symbols of the alphabet is zero.

Table 3.2 – *A lookup table used by the MINDIST function for an alphabet size a = 4.*

	a	b	c	d
a	0	0	0,67	1,34
b	0	0	0	0,67
c	0,67	0	0	0
d	1,34	0,67	0	0

3.4 The face image Time series

As reported above, the basic concept of our approach is to use time series to estimate the pose of the face in an input image. To reach this goal, it is required to represent the face image in a mapped one-dimensional vector. So that dimensionality is reduced and consequently, the complexity of the learning steps becomes weak.

An image is a multidimensional representation in which each dimension represents a height, width, depth, or level of resolution. A 2D face image is a two-dimensional function, $f(x,y)$ (equation 3.10), where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x,y) is called the intensity of that image at that point. An image can be represented as a matrix (M×N), such that the elements of this matrix are the pixels. The number of points or pixels increase rapidly with the image resolution (each pixel represents one of the colour or greyscale information).

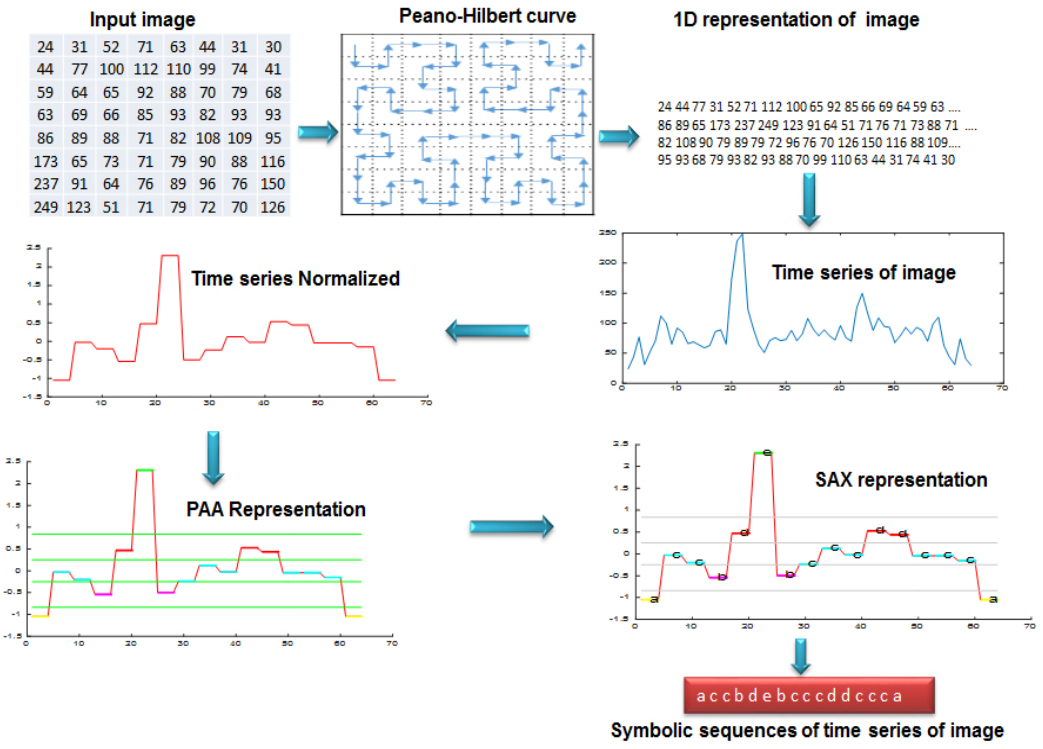
$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{3.10}$$

To convert an image from a multidimensional representation into a one-dimensional representation to generate the face image time series, we used the space-filling curve to scan and intelligently traverse the image. Afterward, the SAX symbolization technique is applied to each time series, which encodes the numeric vectors into a string sequence as shown in figure 3.8.

3.4.1 Space-Filling Curve SFC

In Scientific Computing, space-filling curves (SFCs) are interesting and useful tools to improve certain properties of data structures or algorithms, or even to carry out dimension reduction, or for fast optimization of complex problems. The SFCs are encountered in various areas of computer science, particularly if it is important to linearize multidimensional data, such as compression, pattern recognition, or texture analysis etc....

Space-filling curves (SFCs) are a manner of mapping the higher-dimensional problems to a one-dimensional problem. They are special types of curves that can cover and pass through all the points in space once and only once in a



24	31	52	71	63	44	31	30
44	77	100	112	110	99	74	41
59	64	65	92	88	70	79	68
63	69	66	85	93	82	93	93
86	89	88	71	82	108	109	95
173	65	73	71	79	90	88	116
237	91	64	76	89	96	76	150
249	123	51	71	79	72	70	126

```
24 44 77 31 52 71 112 100 65 92 85 66 69 64 59 63 ....
86 89 65 173 237 249 123 91 64 51 71 76 71 73 88 71 ....
82 108 90 79 89 79 72 96 76 70 126 150 116 88 109....
95 93 68 79 93 82 93 88 70 99 110 63 44 31 74 41 30
```

a c c b d e b c c c d d c c c a

Figure 3.8 – *Example of conversion of an image 8×8 into a SAX sequence.*

particular order. Thus, we can consider it as an orderliness of these points on a line while preserving the locality, i.e. the close points are closely ordered on the line. Numerous well-known curves can completely cover an area and order points (see figure 3.9).

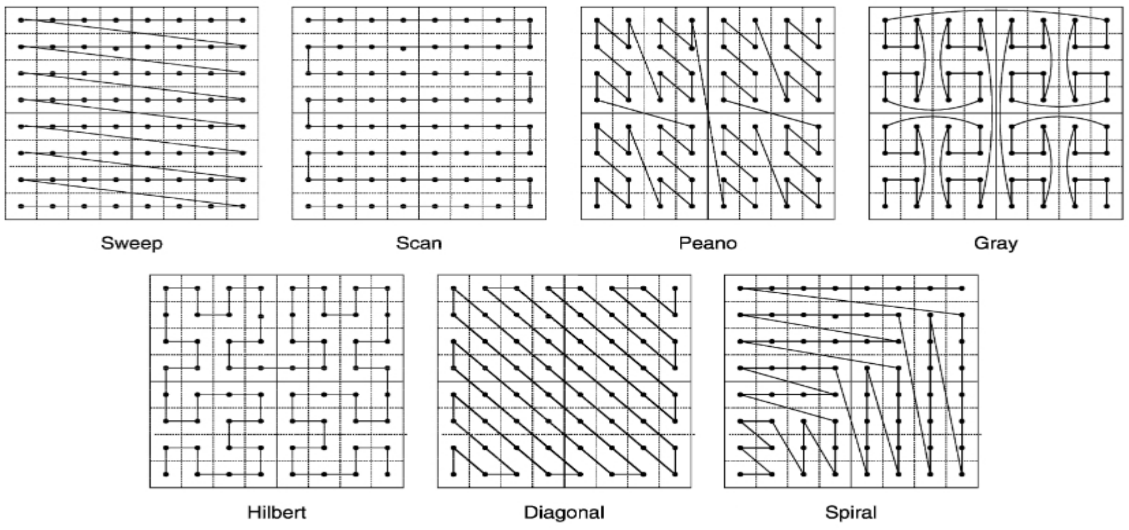


Figure 3.9 – *Some examples of two-dimensional space-filling curves.*

SFCs generate and affect to each point a unique identifier, using the coordinates of this point. This identifier characterizes the position of the point in a one-dimensional space. Each space-filling curve uses a strategy to determine a visiting order of the points of the multidimensional space. The space-filling curve thus requires a linear order of the points. This linearization allows a one-dimensional indexing structure to index the objects: they are indexed by their identifier. Figure 3.10 illustrates an example of a one-dimensional indexing of a 2D space

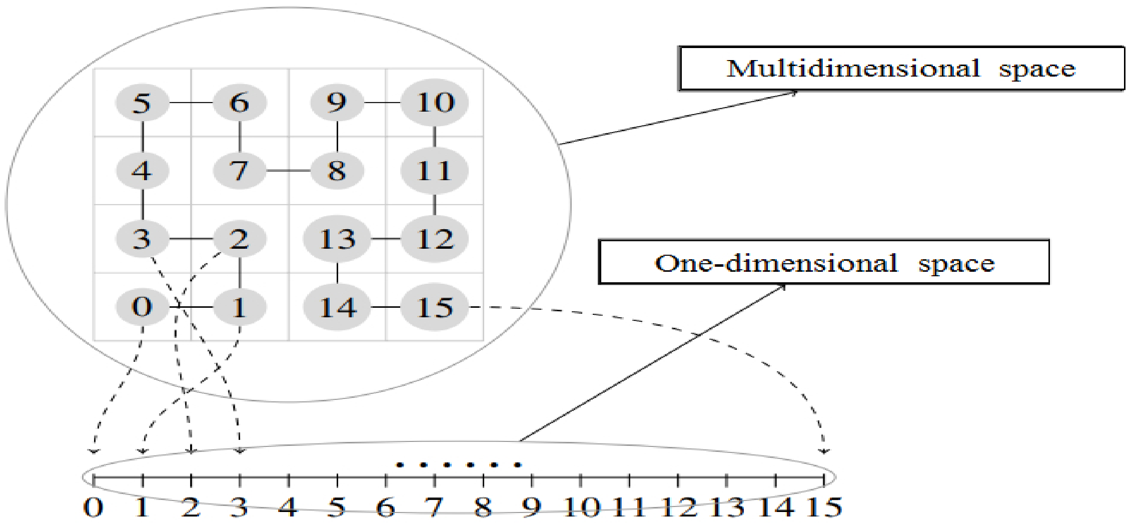


Figure 3.10 – *Examples of conversion of a 2D space into a 1D.*

The locality preserving and continuity are principal criteria for selecting the most suitable space-filling curve. The Peano-Hilbert curve is a diacritical among the space-filling curves, which is characterized by its simplicity, neighbourhood preservation, and continuity (figure 3.12).

The Peano-Hilbert curve is a recursive function that operates on the $\mathcal{S}$ area, which is constructed via an iterative process to fill the square $[0,1]^2$. It is defined by the following algorithm:

1. Divide the initial square (image matrix) into four squares congruent and connected the centre of each point in a clockwise direction without return back to the first point as in figure 3.11-a.

2. Each square is divided again to form four groups of four squares. Similarly, it connects the central points so that the latter group 1 is connected with the first point in group 2, and so on as in figure 3.11-b.

3. Repeat this process until we reach infinity covering the original with a square curve.

An image of resolution $M \times M$ (2D) can be represented as a square of M^2 area. Thus, the scanned image by SFC curve can be regarded as a curve in one dimension space with a length equal to the square of M.

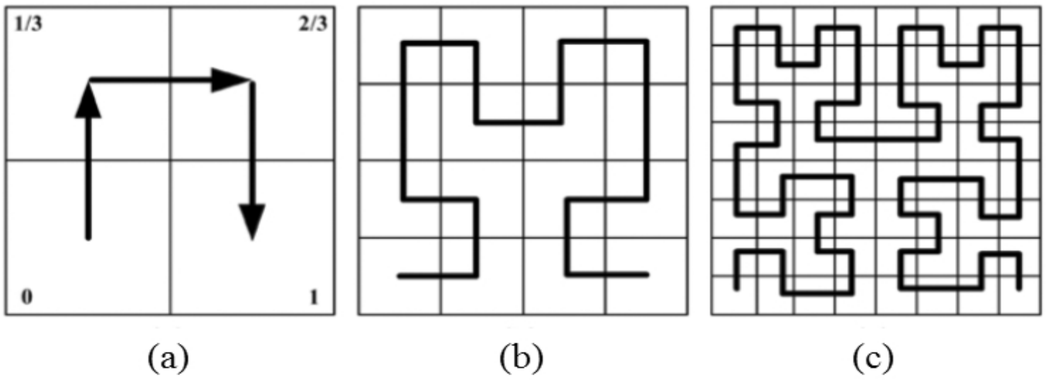


Figure 3.11 – *Example of the Peano–Hilbert's construction curve: (a) the unit interval to unit square mapping; (b,c) the second and third-order curves [Bauman, 2006].*

Although the construction of the Peano Hilbert curve is difficult relative to the Scan and Sweep curves (figure3.9), but it presents the best conservation of the locality of pixels. Where the image pixels are ordered in such a way that the locality of these pixels will be quasi sequentially, which is less true to the other curves. This property permits to keep the order of the information given by pixels. Therefore, it allows preserving the similarity and the dissimilarity between the original images.

Note that in this thesis, to generate the face image time series, we used two space-filling curves: Sweep-curve and Peano-Hilbert curve. The Sweep-curve is constructed in a simple way where the image is scanned row by row from left to right starting from the top left as figure 3.12 shows.

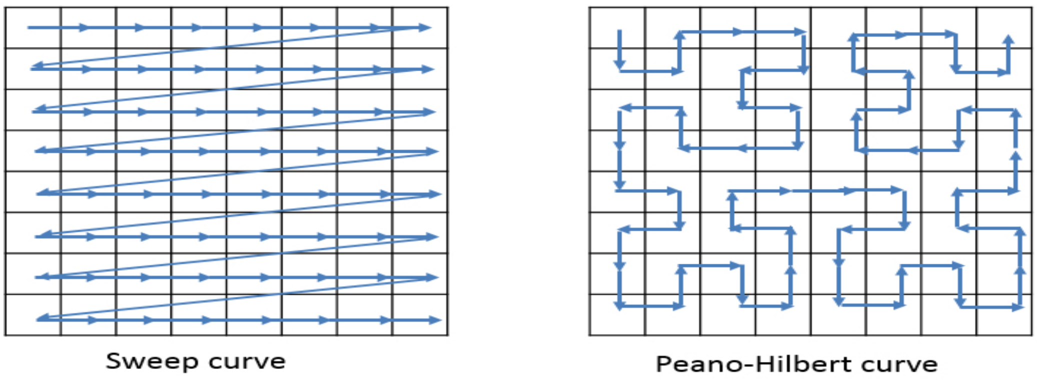


Figure 3.12 – *Examples of conversion of an image into a 1D.*

It is worth pointing out that before transforming any face image into a vector representation using a space-filling curve as we mentioned above, we have effected some modifications, or rather, we preprocessed each face image. We performed some filters to enhance the image quality in an attempt to benefit from all the information in the input image. Especially, the captured images are not always acquired in controlled illumination environments.

3.4.2 Image filtering

As in the real cases, the images are with different illumination, so, for the sake of smooth workflow, it is required to preprocess the face images first to increase visual quality in an image and to obtain the necessary information from an image, then we map them afterward into 1D.

Hence, we use different categories of images during the learning of the model with three categories of images. Firstly, we use the images of the database as they are without processing. Secondly, we apply a gradient filter on images. Then, we use the LBP transformation of the image quotient filtered by a dynamic morphological filter $(DMQI - LBP)$. All these categories of images are mapped into one-dimension with the sweep curve. Meanwhile, the face images scanned with Peano-Hilbert curve are filtered by Gaussian or Laplacian filter. So, each face image has six representations into one-dimension space, which are:

1. Time series generated through mapping the Sweep curve over the face image without processing.
2. Time series generated through mapping the Peano-Hilbert curve over the face image without processing.
3. Time series generated through mapping the Sweep curve over the face image was performed with the gradient filter.
4. Time series generated through mapping the Sweep curve over the face image that was processed with the $(DMQI - LBP)$ operator.
5. Time series generated through mapping the Peano-Hilbert curve over the face image that was performed with the Gaussian filter.
6. Time series generated through mapping the Peano-Hilbert curve over the face image that was performed with the Laplacian filter.

3.4.2.1 Gradient filter

The gradient of an image measures the changes in the intensity of the same point in the original image in the horizontal and vertical directions. Mathematically, the gradient of a function of two variables is a vector in two dimensions. The modulus of the vector is the magnitude of the gradient which tells us how quickly the image is changing, while the direction of the vector tells us the direction in which the image is changing most rapidly. The gradient at an image location is computed by combining the partial derivative of the image in x direction and the y direction. We can write this as:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{3.11}$$

and the magnitude of the gradient is:

$$|\nabla f(x,y)| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \tag{3.12}$$

The gradient operator is not very sensitive to illumination changes as well as invariant under image rotation. These properties are important, and motivate us to use the gradient filter to improve the image quality in low illumination, therefore allow us to increase the classification rate.

Because the gradient acts as a high-pass filter which renders it sensitive to noise, the image gradient is filtered by Gaussian filter with Unsharp contrast enhancement filter which sharpened edges of the elements without increasing noise or blemish.

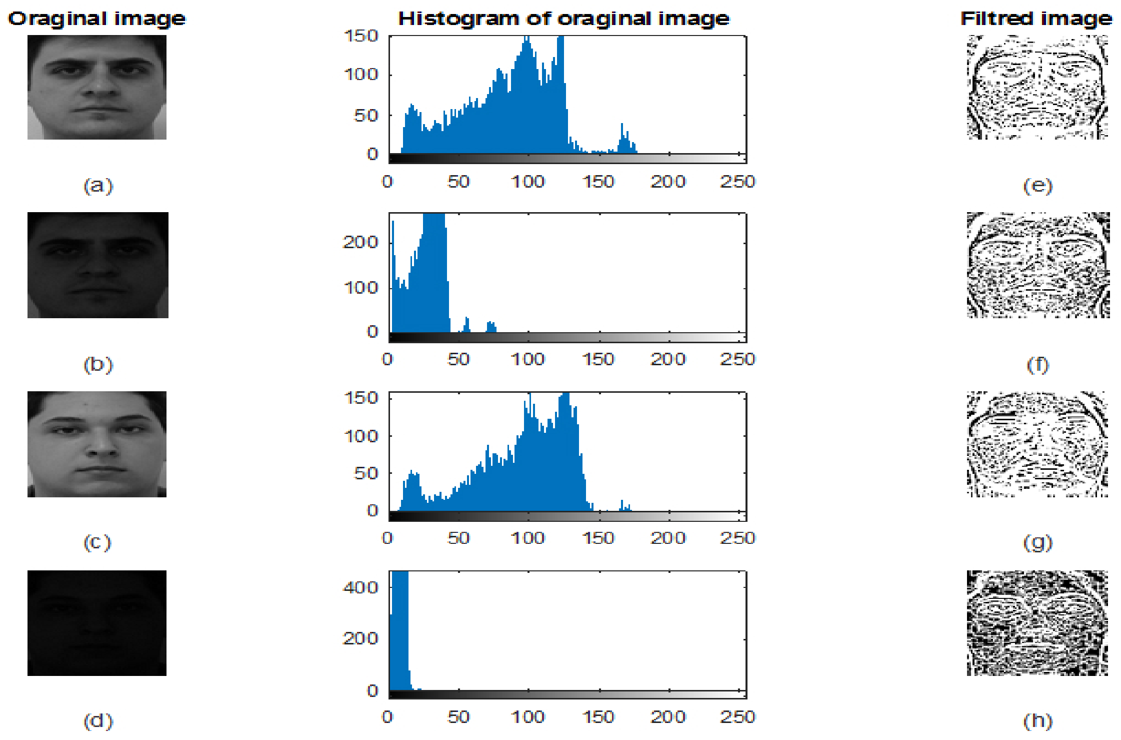


Figure 3.13 – *Example the gradient of the original image in low and natural illumination.*

Figure 3.13 shows that the gradient of the original image (figure 3.13-f,-h) in low lighting (figure 3.13-b,-d) is almost the same as the image in natural lighting (figure 3.13-a,-c).

3.4.2.2 Gaussian filter

Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function. Gaussian filter is a low pass filter that is widely used for smoothing and noise removal. The Gaussian function for calculating the transformation in each pixel of an image $I(i,j)$ is given as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{3.13}$$

where (x, y) denotes the Cartesian coordinates of the image which show the dimensions of the window. And σ is the standard deviation of the Gaussian distribution, that determines the filter response. The image smoothing effect will be higher if σ has a large value.

Convolution of the image $I(i,j)$ by a kernel $H(x,y)$, yields a new image $I'(i',j')$ which is defined as:

$$I'(i',j') = \sum_{(x,y)\in R_H} I(i-x, j-y) \times H(x,y) \tag{3.14}$$

where $H(x,y) = G(x,y)$.

The Gaussian filter is isotropic if the kernel windows size is large enough for a sufficient approximation (at least 5×5).

3.4.2.3 Unsharp Masking

Unsharp masking (USM) is an old method of image sharpening, which was introduced by Schreiber [Schreiber, 1970] in 1970 for the purpose of improving the quality of wire photo pictures for newspapers. The USM is one of the most ubiquitous techniques for edge sharpening. The process of unsharp masking emphasizes the high-frequency components of an image, i.e. the edge regions where there is an explicit change in image intensity. So the first step in the USM filter is to subtract a blurred (low-pass filtered) version of the image from the original to generate the mask.

let $\bar{f}(x,y)$ indicates to the blurred image, the unsharp mask can be expressed as:

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y) \tag{3.15}$$

Subsequently, the mask is added again to the original weighted by the factor λ, which controls the amount of sharpening. Thus, the sharpened image is expressed in equation form as follows

$$g(x,y) = f(x,y) + \lambda * g_{mask}(x,y) \tag{3.16}$$

For $\lambda = 1$, we have unsharp masking. For $\lambda > 1$ the process is referred to as high boost filter.

3.4.2.4 Laplacian filter

The Laplacian of an image strengthens regions of rapid intensity change and attempts to deemphasize regions with slowly varying. Therefore, the edges on images and the contrast of edges can be noticed more clearly. The Laplacian filter is also a derivative operator as the gradient operator, which can be accomplished by combining of the second derivatives in the horizontal and vertical directions. The Laplacian of an image $f(x,y)$ denoted $\nabla_2 f(x,y)$, is defined as:

$$\nabla_2 f(x,y) = \left(\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2} \right) \tag{3.17}$$

We can see the Laplace filter effect on the image in the figure 3.14. We mention that the original images in natural illumination are showed in figures 3.14-a,-c, and which are in low illumination are showed in figures 3.14-b,-d.

3.4.2.5 DMIQ-LBP image

The DMIQ-LBP image is obtained after the application of Dynamic Morphological Quotient Image [Wang et al., 2007], combined with Local binary pattern (LBP) [Huang et al., 2011a], as shown in figure 3.15. We noted that figures 3.15-a,-c show the original images in natural illumination, while figures 3.15-b,-d are in low illumination.

An image in certain lighting conditions can be represented by the production of the illumination L and reflectance R. Such a module can be expressed as follows:

$$I(x,y) = L(x,y) * R(x,y) \tag{3.18}$$

Where *I(x,y)* is a value of each pixel in an image, *L(x,y)* is dependent on the lighting source, while *R(x,y)* is determined by the characteristics of the surface

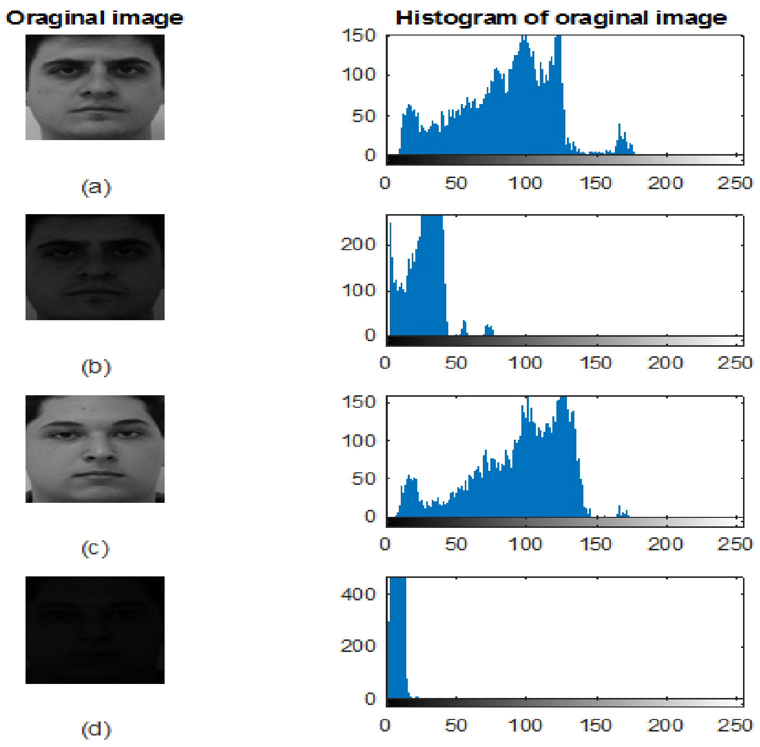


Figure 3.14 – *Example of the original image in low and natural illumination over the Laplacian filter.*

of an object. Using equation (3.18), the reflectance can be expressed as the quotient of the base image and the illumination *L* [Shashua and Riklin-Raviv, 2001]. The filtering of the illumination will lead to the invariance reflectance. A convenient filter can reach this aim. Motivated by the low complexity and the good performance of the morphological quotient image (MQI), the estimation of the illumination *L(x,y)* is done by using a morphological close operator, which is a non-linear operator defined by a dilation followed by an erosion.

$$R(x,y) = \frac{I(x,y)}{L(x,y)} = \frac{I(x,y)}{Close(x,y)} \tag{3.19}$$

The dilatation effect is to expand the image where the pixels of the expanded image are the sum of the pixels of the original image and the structuring element. This transformation tends to eliminate dark objects. Contrariwise, erosion is the effect of shrinking the image while the pixels of the eroded image are the difference pixels between the original image and the structuring element. Erosion allows darkening and spreading the edges of the darkest objects. Therefore, with a suitable size of a structuring element, the close operation can preserve some particulate patterns while it attenuating others. The close eliminates the dark areas that are smaller than the structuring element, keeps the edges of the

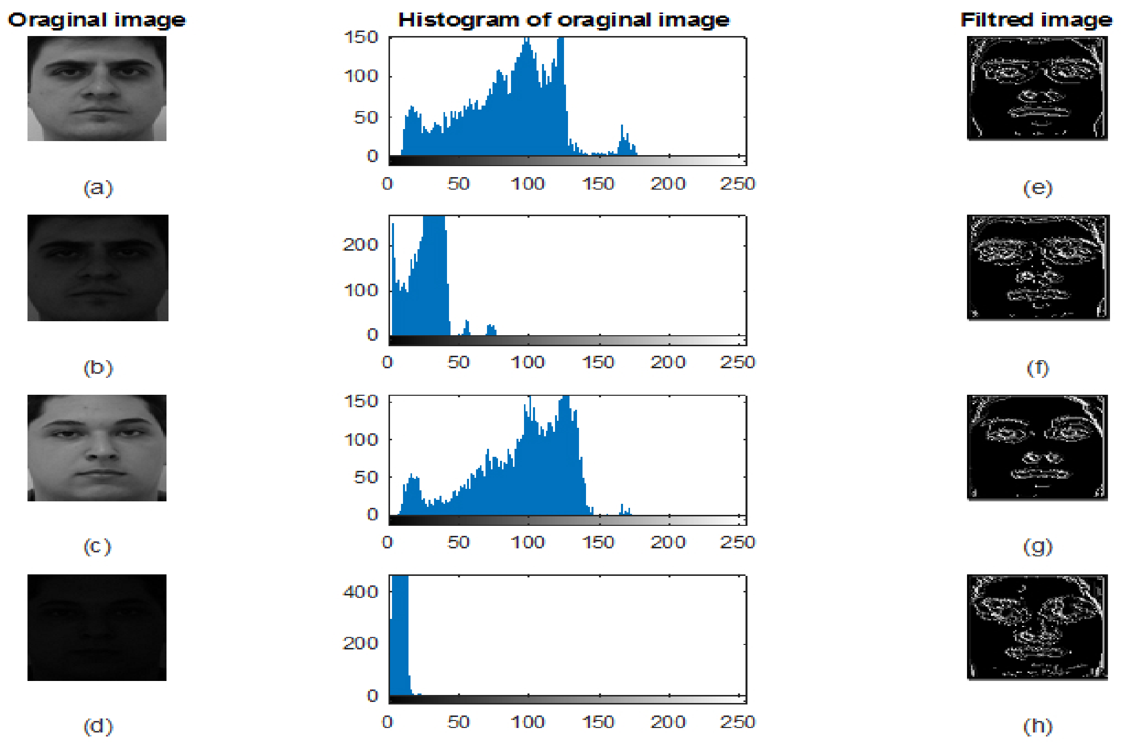


Figure 3.15 – *Example DMQI-LBP image of the original image in low and natural illumination.*

object, and connects the areas of the same light intensity. The way to make the illumination invariant is to use the close operator, which lead to a smooth version of the input image, especially for images with low lighting.

The size of the structuring element plays an important role in making a good morphological filter. Wang et al. [Wang et al., 2007] have indicated that with a large structuring element, the close operator keeps only the large-scale features, but poor performance to compensate on local illumination especially in the case of images under the dark zone. On the other hand, with a small size, it results in good local illumination normalization, but simultaneously misses large scale features. To overcome this problem, Zhang et al. [Zhang et al., 2007] proposed Dynamic Morphological Quotient Image (DMQI) using a structuring element with dynamic size according to the formula (3.20). DMQI is expressed by the equation (3.21).

$$DClose(x,y) = \begin{cases} Close^{l}(x,y), & \alpha * Close^{s}(x,y) < Close^{l}(x,y) \\ Close^{m}(x,y), & \beta * Close^{s}(x,y) < Close^{l}(x,y) < \alpha * Close^{s}(x,y) \\ Close^{s}(x,y), & Close^{l}(x,y) < \beta * Close^{s}(x,y) \end{cases} \tag{3.20}$$

$$DMQI(x,y) = \frac{I(x,y)}{L(x,y)} = \frac{I(x,y)}{DClose(x,y)} \tag{3.21}$$

Where α and β are the parameters of the feature scales, while $\alpha > \beta > 1.0$. l, m, and s are the optional sizes of templates, while $l > m > s > 1$ [Zhang et al., 2007].
In the regions of brow, eye, nose, mouth, or the boundary of changing light intensity, the greyscale is changing significantly. In this case, the choice of the close operator with a large size is better to keep the features of these regions. So, the DMQI image is calculated using the condition $\alpha * Close^s(x, y) < Close^l(x, y)$, which shows that pixels of the close operator with a large size is very different than the pixels of image obtained by the close operator with small size. However, if the regions are under illumination or in a smooth region, such as cheek and forehead, the change of grey values in these regions is weak. So, in this case, the use of the close operator with a small size is sufficient. Thus, the DMQI image is calculated using the condition $Close^l(x, y) < \beta * Close^s(x, y)$ Zhang et al. [2007].

3.5 The used machine learning algorithms

The ultimate goal of our thesis is the classification of the face poses. After the representation of each face image with time series that are coded with SAX representation and measure the pairwise similarity matrix between all symbolic time series, the last step is to classify the sequence (therefore the face image) to assign each one to an adequate class (frontal or profile view classes).

3.5.1 K-Means Clustering

The k-means is an unsupervised classification algorithm. It is the most popular type of partition clustering method, which provides clusters of data. The data are iteratively partitioned into a subset of k groups. These clusters are formed iteratively to determine the intrinsic group among the unlabelled database. The k-means split the data according to the similarity, where the similar elements share the same cluster [Hartigan, 1975].

First, the k-means algorithm selects randomly the k cluster centroids $c_1, \dots, c_k$ in the data-points. Each centroid will define a cluster. Then two steps are executed:

- Classification: each datapoint is assigned to the closed centroid in terms of similarity, i.e. by measuring the Euclidean distance between each point in the cluster and the mean of their centroid.

- Minimization: in this step, centroids are repositioned in a position that minimizes the sum of the distance of all datapoints assigned to it.

The process is repeated until all the points are assigned to their respective clusters. In other words, each datapoint is labelled with the name of the closed cluster.

The figure 3.16 explains the k-means Clustering Algorithm procedure.

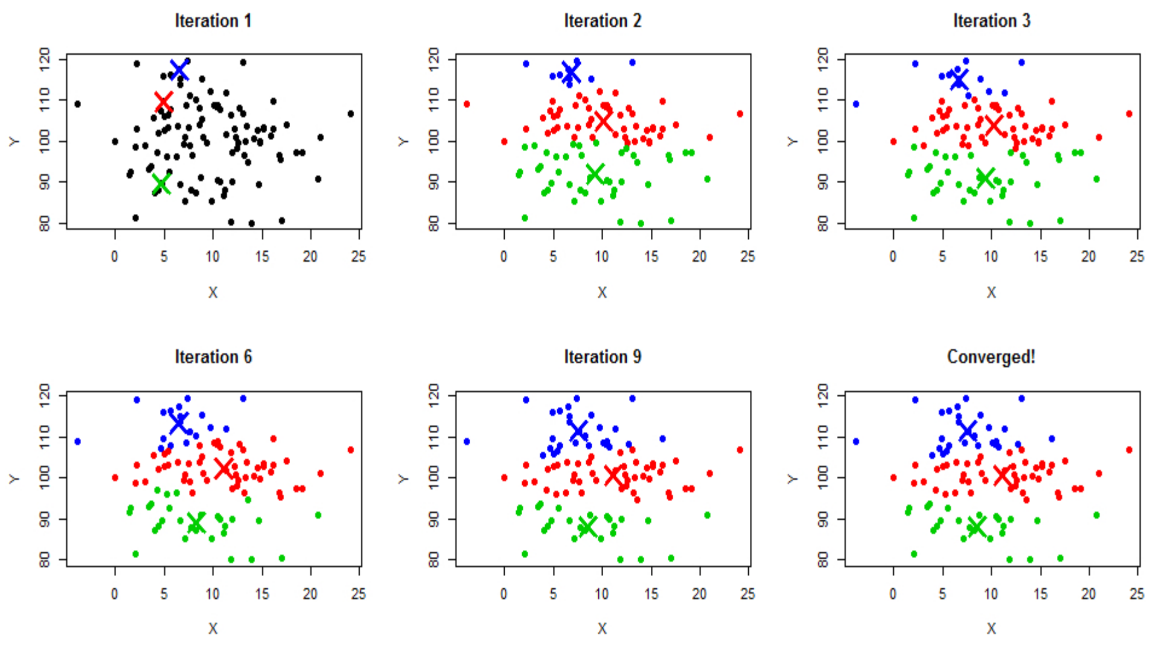


Figure 3.16 – *Illustration of k-means algorithm.*

The K-means algorithm [Ertel, 2018] is described as follows:

Algorithm 1 K-means algorithm

```
Input: A database of point $X = \{x_1, \dots, x_N\}$
        k: Number of desired clusters
Initialization:
Choose randomly k initial centres $C = \{c_1, ...., c_k\}$
  repeat
    classify instance $x_1, \dots, x_N$ to each nearest $c_i$;
    Recalculate $c_i, \dots, c_k$.
  until No change in $c_i, \dots, c_k$
  return ($c_i, \dots, c_k$)
```

The cluster centroids C for points $x_1, \dots, x_N$ is calculated by:

$$C = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.22}$$

where N is the number of all datapoints.

Thus, the objective of k-Means clustering is to optimize total intra-cluster variance, or, the squared error function:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{k} ||x_i^{(j)} - c_j||^2 \tag{3.23}$$

Where J is the objective function of the centroid of the cluster. k is the number of clusters and N is the number of cases. C is the number of centroids. X is the given data-point from which we have to determine the Euclidean Distance to the centroid.

3.5.2 K-Nearest Neighbours (*KNN*)

The *KNN* or *K*-nearest neighbours is one of the topmost supervised machine learning algorithms. It is non-parametric, in which any assumption for underlying data distribution is required. As well as, it is lazy learning classification method. An algorithm is denoted a lazy learning when it classifies the new observation based on stored, and labelled instances rather than it learns a discriminative function from the training data [Thirumuruganathan, 2010]. In other words, the lazy algorithm stores merely the training database, and are all used to measure similarity with a new instance.

The idea behind the *KNN* is to label an unlabelled instance with the class of the closest instances in the training set. Put another way, the *KNN* computes the *KNN* closest nearest class to the query or test point using distance measures such as Euclidean distance, Hamming distance, Manhattan distance or Minkowski distance. The label assigned to this query point is the label of the class of the most votes among those nearest neighbours.

The *KNN* steps process can be outlined as follows

Algorithm 2 *KNN* algorithm

```
Input: X: training data,Y: class labels of X
        query x , K Nearest Neighbour
  for $i = 1$ To $N$ do
      Compute distance $d(X_i, x)$
  end for
    Compute set $I$ containing indices for the $k$ smallest distances $d(X_i, x)$.
  return Majority label for $\{Y_i$ where $i \in I\}$
```

An example of classification using *KNN* with $K = 3$ is shown in figure 3.17

In this illustration, the first class is represented by a star and the second by a triangle. The new instance to be classified is in the form of a square with a question mark inside.

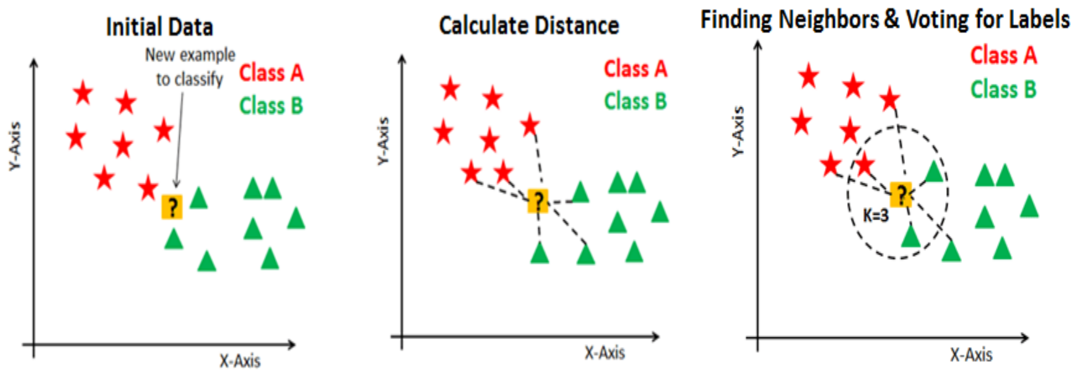


Figure 3.17 – *Illustration of KNN algorithm for* $K = 3$.

The choice of K number of nearest neighbours and distance metric for computing the nearest distance is impacting essentially on the performance. Generally, the lowest values of K will lead the system to be sensitive to noise, while larger values of K reduce the effect of this noise on the classification but will lead to over-smoothed boundaries [Thirumuruganathan, 2010]. When $K = 1$, the algorithm is known as the nearest neighbour algorithm.

3.5.3 Support Vector Machines (SVMs)

Support Vector Machines are one of the most popular and widely used supervised machine learning algorithms pioneered by Vapnik [Vapnik, 1999]. Their principle is to find a separation that can split the data into classes using a boundary as most simple as possible, such that the gap or the distance between the different groups of data and the boundary which separates them is the maximum. This distance is also known as "margin" and the "support vectors" are the data closest to the boundary. The boundary can be determined either by a hyperplane when the data are linearly separable or through a kernel function when a problem is not linearly separable.

Linear SVMs are the simplest form of this algorithm. It is applicable in the case where the data are linearly separable. It seeks to find a hyperplane creating the optimal separation between classes and providing a maximum margin. This hyperplane is the decision boundary as illustrated in figure (3.18-a).

In the case of binary classification (or two class), the optimal separation

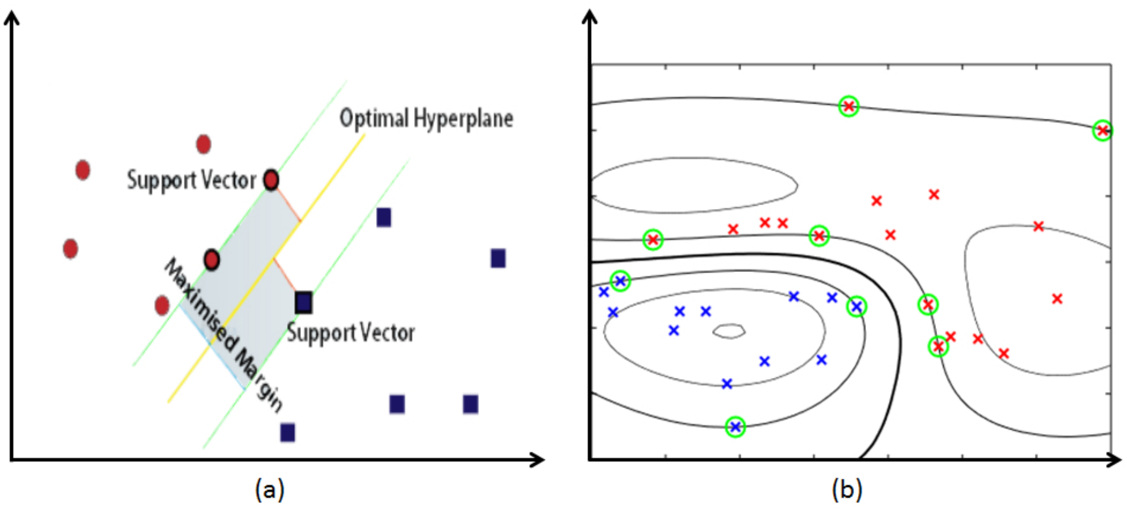


Figure 3.18 – *Illustration of Support Vector Machine.(a) Linearly separable case, (b) Non Linearly separable case.*

hyperplane can be expressed by the following equation:

$$f(x) = w^t x + b \tag{3.24}$$

where w is the weight vector and b is the threshold value known as the bias.

The SVM learns to find w, b, which allow the function f to has the value 1 for the nearest data points belonging to the first class, and –1 for the nearest ones of the second class. While hyperplane will be satisfying the following constraints:

$$\begin{aligned} w^t x + b &\geqslant +1 \\ w^t x + b &\leqslant -1 \end{aligned} \tag{3.25}$$

The distance between the two hyperplanes of the equation $w^t x + b = +1$ and $w^t x + b = -1$ is:

$$\frac{2}{\| w \|} \tag{3.26}$$

That is also referred to as the margin. Therefore, the optimal hyperplane can be yield by maximizing the margin, in other words by minimizing the norm of the separated hyperplane. Thus, the problem becomes an optimization problem that minimizes the objective function [Awad and Khanna, 2015].

$$J(w) = \frac{1}{2} \| w \|^2 \tag{3.27}$$

subject to the constraint

$$y_i(w^t x_i + b \geqslant +1), i = 1, 2, \cdots, N \tag{3.28}$$

where, y_i is the label of input x_i and N is the number of training points.

In many cases, we cannot deal with classification problems in which the decision boundary is not linear. For that, an alternative manner of SVM used kernel function to separate and cluster the classes. The process to separate the different classes involved in the classification task, in this case, the process is performed in two steps. Firstly, the kernel maps the data from the input space into a higher dimensional referred to as kernel space, where data will be linearly separable. Next, the SVM algorithm is used to find the optimal hyperplane that separates the new data vectors. This is an attractive method because the difficulty related to passing to kernel space is more lessen compared to learning a non-linear surface.

The kernel function can be used to characterize the distance between two patterns x_i and x_j. We can cite some kernel functions:

The linear kernel would be the simplest one, it is just the dot product of the features.

$$K(x_i, x_j) = x_i^t x_j \tag{3.29}$$

The polynomial kernel of degree d transforms the data by adding a simple non-linear transformation of the data.

$$K(x_i, x_j) = (x_i^t x_j + c)^d \tag{3.30}$$

The hyperbolic tangent (sigmoid) uses a sigmoid activation function.

$$K(x_i, x_j) = tanh(k x_i^t x_j + c) \tag{3.31}$$

Gaussian radial basis function (RBF) is one of the most often used kernels in practice, that can map an input space into infinite-dimensional space(figure 3.18-b-).

$$K(x_i, x_j) = exp(\frac{\| x_i x_j \|^2}{2\sigma}) \tag{3.32}$$

The value of the RBF kernel depends on the Euclidean distance between x and y. Either x or y will be the centre of the radial basis function and σ will quantify the spread around the centre and determine the area of influence over the data space [Murty and Devi, 2011].

Although SVMs were originally designed for binary classification, they can handle a multi-class problem. The methods suggested by Weston et al.[Weston et al., 1999] and Platt et al. [Platt et al., 1999] are among the early extensions of the SVM binary classification to the multiclass. The conventional manner to reach this purpose is to decompose multiclass classification into several binary problems using either the one-versus-all or one-versus-one method.

One-versus-all strategy also known as the one-against-all (OAA), it is the simplest and earliest decomposition method. This strategy reduced N multiclass into an N binary classifier [Vapnik, 1998]. Each classifier aims to distinguish the class of index n from all the others. In other words, the n^{th} classifier is training while regarding the samples in the n^{th} class as positive examples and all the remaining as negative examples. To assign an example to the corresponding class label, the N classifiers are run, and the decision is obtained by applying the "winner-takes-all" principle. Therefore, the final label selected is the one associated with the classifier that returned the highest value.

One-versus-one strategy also called one-against-one (OAO), this algorithm is proposed by Knerr et al.[Knerr et al., 1990], while Friedman [Friedman, 1996] and Kressel[Kressel, 1999] are the first to adopt it in SVM. The strategy of this algorithm is to build $N(N-1)/2$ binary classifiers to enable the use of the basic SVM model for N multiclass classification, using all the binary pair-wise classification of the N classes. Each classifier is trained by taking the examples of two of the N classes only, and requires evaluation of $(N-1)$ SVM classifiers. An example is assigned to the corresponding class through the decision that is usually obtained by carrying out the majority vote "max-wins voting" (MWV).

3.6 EXPERIMENTAL VALIDATION

In this section we, evaluate the performance of the methods presented in previous sections. The experiments are executed in Matlab 2015, on a PC with Intel Core (TM) i5 CPU M 480 @2.67GHz 2.66GHz with 6 GB memory.

We carried out these experiments according to the SAX2FACE algorithm (algorithm 3). The algorithm of our approach is outlined as follows:

As we described earlier in subsection (3.4.2), we have used two strategies to map face image into 1D: the sweep curve and the Peano-Hilbert space-filling curve.

In the first part, we conduct experiments using the first strategy, i.e. we

Algorithm 3 SAX2FACE algorithm

```
Input: Training database
Output: A class of a testing database
Step:
    1 Scan each input image with space-filling curve.
    2 Represent each curve as times series and perform a symbolic
      transformation using the SAX method.
    3 Compute all pairwise similarities between symbolic sequences.
    4 Run a classification to estimate the poses classes.
```

performed our approach on the time series generated by using the sweep curve, which we will be detailing in subsection (3.6.2).

In the second part, the Peano-Hilbert curve is employed to generate the face image time series. We used the same algorithm 3, while the difference between the followed protocols is the input images and a distance measure metric. The procedures of this process will be presented in the next subsection (3.6.3).

The experiments are conducted on three publicly available databases for head-pose estimation, which are reported for each database independently.

3.6.1 The databases

To evaluate the performance of our approach, we tested it on three databases, namely GTAV database [Tarrés and Rama, 2012], FEI database [Thomaz and Giraldi, 2010], and FERET database [Phillips et al., 1998].

GTAV database: includes a set of 44 persons with 27 pictures per person which correspond to different pose views (0°, ±30 °, ±45 °, ±60 ° and ±90 °) under three different illuminations (environment or natural light, strong light source from an angle of 45°, and finally an almost frontal mid-strong light source (figure 3.19).

FEI database: contains 2800 images of 100 men and 100 women, each individual has 14 images in an upright frontal position with profile rotation of up to about 180 degrees (figure 3.20). It has 400 frontal images in natural light and 400 images in low lighting. In the FEI database, each person has 12 images in a different pose with natural light and two images in frontal view with weak light.

FERET database consists of images that are collected in a semi-controlled environment, of different age, race, and sex distribution. With poses fa, fb for the frontal pose, and ql, qr for the left and right respectively quarter pose (±22.5 °),

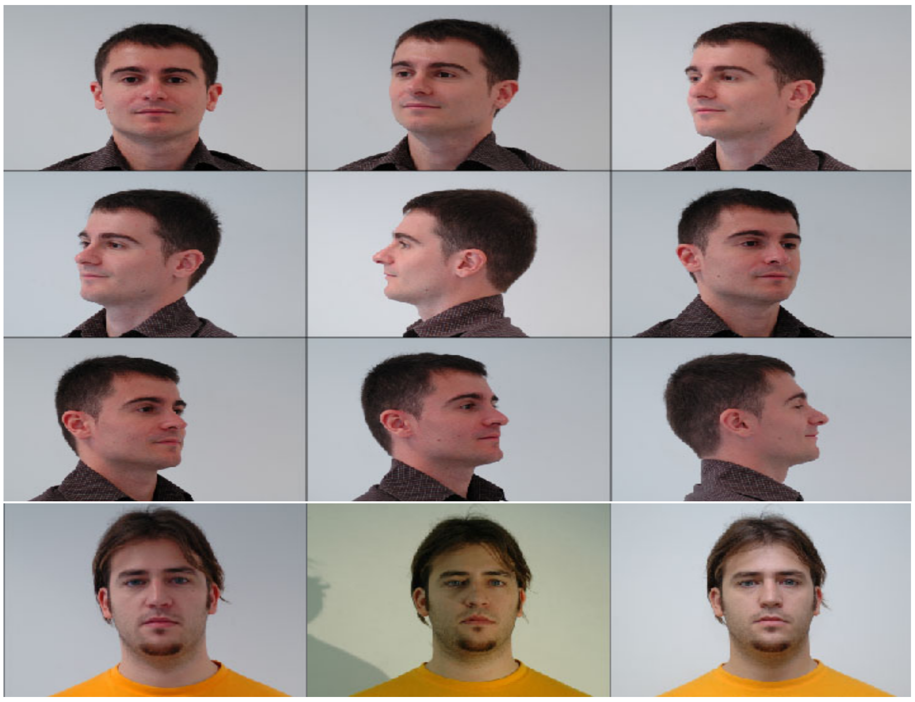


Figure 3.19 – *Example of a face from the GTAV Face database (different poses and illuminations) [Tarrés and Rama, 2012].*

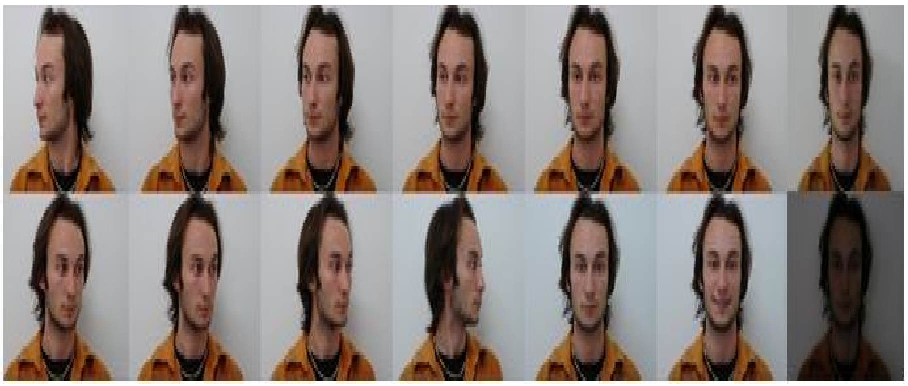


Figure 3.20 – *Some examples of face images from the FEI face database in different pose and illumination [Thomaz and Giraldi, 2010].*

hl, hr are the poses mid-left and mid-right respectively ($\pm 67.5\ ^\circ$), and pl, pr are profile poses left and right respectively ($\pm 90\ ^\circ$) as in figure 3.21. The global total of the used images is around 9180 images.

3.6.2 The Protocol of the First Strategy

In this first strategy, the experiences are carried out with the three face image time series generated through mapping the Sweep curve over the face image: without processing, that was performed with the gradient filter, and that was processed with the (DMQI-LBP) operator.

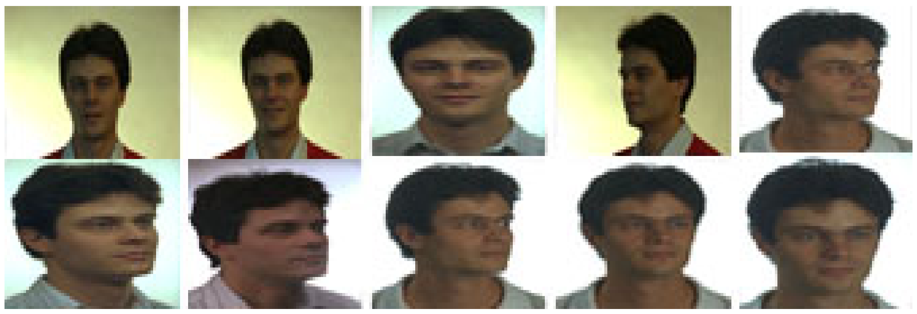


Figure 3.21 – *Example of a person from the FERET Face database [Phillips et al., 1998].*

Note that the input images are cropped manually for each facial image with different poses, and are resized to size 100×100 then converted to greyscale. Thus, the length of the numerical time series is 10000 (the total number of the pixels in the image 100×100).

3.6.2.1 Optimal SAX parameters

We recall that at the learning step, we encode each image in a SAX symbolic time series. The SAX requires two parameters, namely the size of the windows frame "w", and the length of the alphabet "a", which are the two main parameters that control the quality of the SAX.

We performed as a first experiment using the GTAV database to find out the best parameters w and a of SAX, which boost the classification rate. We have conducted the experiments using the images as they are without any modification or processing, and we have applied our algorithm to this database. We calculated the classification rates with different values of w and a.

We classified the poses using the k-means algorithm into three main classes: class of frontal pose, class of the left view, and a class of the right view that group poses in quarter profile (left or right) to full profile. As we have interested in determining the optimal SAX parameters that permit the classification of the input image into a suitable class, the k-means algorithm can be used for this purpose. The k-means allows investigating of the structure and the distribution of the data by grouping the elements into homogeneous subgroups. Further, it is easy to implement, and it does not require many parameters. It needs only the k number of clusters that is already known, in our case k=3.

In Table 3.3, we show the F-Score of classification for each frame size (w=5,6,7,8,9,10,15,20,25, 35, 45,55 and 64) and with different alphabet size (a=5, 6, 7, 8, 64, 128).

F-Score [Sokolova et al., 2006], or F-measure is an alternative to evaluate classification accuracy, that combined precision (P) and recall (R) metrics.

$$F = \frac{2PR}{P+R} \tag{3.33}$$

Where

$$P = \frac{TP}{TP+FP}; R = \frac{TP}{TP+FN} \tag{3.34}$$

- TP or True Positives: the number of data points correctly classified from the positive class.
- FP or False Positives: the number of data points predicted to be in the positive class but belonging to the negative class.
- FN or False Negatives: the number of data points predicted to be in the negative class but belonging to the positive class.

Table 3.3 – *F-Score of the classification of face poses on the GTAV database using K-means algorithm to evaluate the parameter w (frame size).*

Frame size	Alphabet size	Left view %	Frontal view %	Right view %
5	5	81.76	45.70	87.24
	6	82.92	48.02	87.95
	7	82.02	47.22	88.55
	8	81.15	46.76	88.90
	64	86.32	54.11	90.60
	128	86.92	54.70	90.60
10	5	81.80	45.22	86.69
	6	82.19	46.18	87.40
	7	82.17	45.54	87.51
	8	81.33	46.30	88.21
	64	82.25	48.53	89.14
	128	82.25	48.72	89.37
15	5	81.70	44.76	88.16
	6	81.84	46.22	88.34
	7	81.67	46.52	88.57
	8	80.75	46.15	88.91
	64	80.31	46.99	89.18
	128	80.97	47.45	89.18

Continued on next page

Table 3.3 – continued from previous page

Frame size	Alphabet size	Left view %	Frontal view %	Right view %
20	5	79.70	39.14	85.86
	6	80.90	40.08	86.19
	7	81.33	41.20	86.28
	8	80.39	42.44	87.24
	64	80.62	45.53	88.80
	128	81.72	46.55	88.87
25	5	79.96	40.94	86.78
	6	81.63	42.51	87.10
	7	81.74	44.00	87.68
	8	80.84	44.01	87.62
	64	81.55	46.85	89.09
	128	81.55	46.55	89.00
35	5	78.63	40.39	87.70
	6	78.62	42.39	87.82
	7	78.62	42.39	87.82
	8	76.96	42.13	88.44
	64	77.92	43.43	88.64
	128	77.97	43.59	88.64
45	5	79.57	40.74	88.26
	6	77.49	41.35	88.18
	7	78.44	41.49	87.89
	8	78.68	39.45	87.00
	64	77.70	41.92	87.83
	128	77.29	41.68	87.83
55	5	78.38	40.00	87.65
	6	77.17	41.50	88.29
	7	78.44	41.49	87.89
	8	76.40	39.15	87.45
	64	76.35	39.23	87.02
	128	76.44	39.69	86.84
64	5	78.23	40.98	88.31
	6	76.27	39.84	87.98
	7	78.19	39.28	86.93
	8	76.42	39.36	87.75

Continued on next page

Table 3.3 – continued from previous page

Frame size	Alphabet size	Left view %	Frontal view %	Right view %
	64	76.01	38.30	86.29
	128	76.01	38.30	86.29

We can observe that with SAX, and when maximizing the size of the window, the classification rate decreases. This is totally normal since the SAX symbolic encoding is lossless with great values of codeword w.

To ensure the best classification rates, we choose the smallest frame size (w=5, to avoid loss of information), and applied it at the rest of the evaluations to represent the time series in the next experiments.

After determining the frame size, we should determine the best alphabet size. We have fixed w at 5, while the alphabet size a varies in 5,10,15,20,64, and 128. To evaluate the classification rate, we have performed experiments using K nearest neighbour (KNN) and support vector machine (SVM) with Gaussian kernel function for each database.

These classification algorithms were repeated with the three categories of images: without filter (noted OUTPRS in the tables), with Gradient filter (noted GRAD in the tables), and with DMQ-LBP (noted DMQLBP in the tables).

3.6.2.2 Experiment on the GTAV database

The classification results of the GTAV database are listed in Tables 3.4 and 3.5.

Table 3.4 – *Classification rate of the GTAV database for each approach using KNN algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	99.72	97.4	99.43	O
	10	99.34	96.32	98.96	U
5	15	99.24	95.88	99.34	T
	20	99.62	98.15	99.90	P
	64	99.62	98.89	99.53	R
	128	99.43	96.63	99.34	S
	5	99.81	99.62	99.90	
	10	**100**	**100**	**100**	G
5	15	100	100	100	R
	20	99.91	100	99.91	A
	64	100	100	100	D
	128	99.91	99.63	100	
	5	100	100	100	D
	10	99.53	98.50	99.15	M
5	15	99.53	99.26	99.71	Q
	20	99.71	99.63	99.81	L
	64	99.62	99.62	99.72	B
	128	99.81	100	99.81	P

Table 3.5 – *Classification rate of the GTAV database for each approach using SVM algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	99.72	100	99.72	O
	10	99.81	100	99.81	U
5	15	99.81	99.63	99.90	T
	20	99.25	99.62	99.34	P
	64	99.24	99.26	99.43	R
	128	99.81	99.81	99.63	S
	5	99.90	100	99.90	
	10	100	100	100	G
5	15	99.90	100	99.90	R
	20	99.90	98.89	99.81	A
	64	100	100	100	D
	128	99.90	100	99.90	
	5	99.90	100	99.90	D
	10	99.81	100	99.81	M
5	15	99.81	100	99.81	Q
	20	99.72	99.25	99.90	L
	64	99.90	99.26	99.72	B
	128	100	100	100	P

We can resume from Table 3.4 and 3.5:

- Frontal poses have reached a classification rate of 100% (w = 5, a = 10 . . . 64 with gradient, a=5,128 with DMQ-LBP using KNN, and a=5,10 without processing, a=10,15, 64, 128 with Gradient, a=5,10,15,128 with DMQ-LPB using SVM)
- For left and right classes, almost all poses have been well classified by the three approaches.
- The classification with the SVM algorithm allows us to achieve the best classification rate comparing to the KNN algorithm.
- Using images without processing, all poses in frontal view were classified correctly (with SVM).
- Using Gradient and DMQ-LBP, all poses were nearly classified successfully for each alphabet size.

3.6.2.3 Experiment on the FEI database

In Tables 3.6 and 3.7 we illustrate similar results using KNN and SVM on the FEI database.

- The frontal poses were classified with a rate between 97% and 98%.The results obtained with gradient and DMQ-LBP images are powerful than the images without processing. The classification rate of 98.24% is yield for $a = 15$ using KNN, while a classification rate of 98.45% using SVM for the gradient images. The best classification rate are achieved with DMQ-LBP for both KNN and SVM, where we have reached a classification rate nearly 99% (98.81% using KNN for $a = 20$, and 98.91% using SVM for $a = 10$).
- We can notice that nearly all the poses from left or right have been classified with the appropriate label for the three types of images (without processing, gradient or DMQ-LBP).
- The best classification rate for all views are achieved with DMQ-LBP using SVM for $a = 10$).

It is worth pointing out that the frontal poses in the FEI database were captured in different illuminations, certain frontal pose images are in a dark light as shown in figure 3.20 and figure 3.22.

Figure 3.22 – *Example of the input images in FEI database with low lighting.*

Table 3.6 – *Classification rate of the FEI database for each approach using KNN algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	98,44	97.99	98.56	O
	10	98.32	97.91	98.55	U
5	15	98.19	97.60	98.26	T
	20	98.31	97.96	98.62	P
	64	98.26	97.67	98.22	R
	128	97.70	97.36	98.12	S
	5	98.32	97.74	98.31	
	10	98.31	97.92	98.56	G
5	15	98.94	98.24	98.45	R
	20	98.01	97.61	98.44	A
	64	98.56	97.91	98.32	D
	128	98.31	97.99	98.31	
	5	98.25	98.04	98.81	D
	10	98.11	97.68	98.23	M
5	15	98.00	98.08	98.93	Q
	20	98.81	98.81	98.38	L
	64	98.05	97.92	98.81	B
	128	98.69	98.28	98.75	P

3.6.2.4 Experiment on the FERET database

The results shown in Tables 3.8 and 3.9 indicate the classification accuracy achieved in the FERET database. As shown in figure 3.21, the face images are acquired at a different distance relative to the camera.

Table 3.7 – *Classification rate of the FEI database for each approach using SVM algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	98.38	97.83	98.37	O
	10	97.46	97.19	98.37	U
5	15	98.70	97.80	98.00	T
	20	97.71	97.01	97.88	P
	64	98.38	97.73	98.38	R
	128	97.89	97.31	98.13	S
	5	97.94	97.83	98.68	
	10	97.53	97.35	98.43	G
5	15	98.38	98.21	98.93	R
	20	97.33	96.85	98.01	A
	64	97.20	97.11	98.38	D
	128	98.39	98.45	99.06	
	5	98.75	97.83	97.75	D
	10	98.82	98.91	99.31	M
5	15	98.76	98.66	98.75	Q
	20	98.75	98.79	98.81	L
	64	98.57	97.66	97.93	B
	128	98.88	98.33	99.12	P

Table 3.8 – *Classification rate of the FERT database for each approach using KNN algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	99.39	98.87	99.41	O
	10	98.33	97.15	98.38	U
5	15	98.41	97.29	98.28	T
	20	98.71	97.63	98.50	P
	64	98.71	97.46	98.52	R
	128	98.72	97.29	98.52	S
	5	86.83	90.59	90.40	
	10	86.86	91.00	90.20	G
5	15	86.86	91.00	90.20	R
	20	86.12	90.78	89.60	A
	64	87.73	91.74	90.38	D
	128	86.43	91.75	89.91	
	5	99.39	98.89	99.19	D
	10	99.24	98.87	99.26	M
5	15	99.16	98.70	99.23	Q
	20	99.36	98.83	99.38	L
	64	99.30	98.76	99.17	B
	128	99.19	98.74	99.35	P

For the FERET database, we can see that the frontal poses have attained a classification rate of 98.87% using KNN for alphabet size $a = 5$ with the images without processing, and 98.89% with images processed by DMQ-LBP. And a classification rate of close to 100% using SVM, where we have achieved 99.15% for $a = 5$ with the images without processing, and 99.42%, 99.28% and 99.41% for alphabet size $a = 5, 10, 64$ respectively with images processed by DMQ-LBP.

Table 3.9 – *Classification rate of the FERT database for each approach using SVM algorithm to determine the best alphabet size.*

Frame size	Alphabet size	Left view%	Frontal view %	Right view %	
	5	99.42	99.15	99.46	O
	10	99.13	98.61	99.18	U
5	15	99.36	98.85	99.22	T
	20	99.21	98.48	99.00	P
	64	99.22	98.85	99.38	R
	128	99.16	98.24	98.76	S
	5	84.21	89.12	92.74	
	10	75.61	87.38	86.04	G
5	15	81.74	87.37	91.13	R
	20	81.71	87.40	91.16	A
	64	77.44	87.37	87.46	D
	128	82.38	87.69	91.69	
	5	99.38	99.42	99.48	D
	10	99.35	99.28	99.34	M
5	15	99.33	98.89	99.28	Q
	20	99.27	98.98	99.11	L
	64	99.36	99.41	99.46	B
	128	99.16	98.87	99.25	P

It can be deduced from these results on all database, that if the images are under a natural environment, it is sufficient to apply SAX on time series of images with any treatment (FERT case), even in the case of images with different lighting (GTAV case). In the case where the images are in weak or dark light (FEI case), it is preferable to use the images processed using the second processing (Gradient filter), or the third processing (DMQ-LBP) with a high alphabet size. Therefore, the conditions under which the images were taken are used to determine the necessary parameters to use the SAX encoding process.

3.6.3 The Protocol of the Second Strategy

In this subsection, we present the experiment results that are realized using the Peano-Hilbert curve technique. We proceeded in the same way as the first protocol, where we supported the same algorithm described previously (algorithm 3), whereas we changed the types of input images and the way to transform these images into time series. Further, we used another method to measure the similarity between the SAX sequences, by computing the similarity with *MINDIST* introduced by SAX and the edit distance, respectively. We also used three categories of face images, where the input images in the experiences are sampled using with and without pre-processing, where we used a Gaussian and Laplace filters as shown in figure 3.23. In these experiences the images are mapped to one dimension using the Peano-Hilbert curve. Since the Peano-Hilbert technique requires a resolution of $2^n \times 2^n$, the input images are resized to 128×128, so the length of each time series N is 16384 (the total number of the pixels in image 128×128).

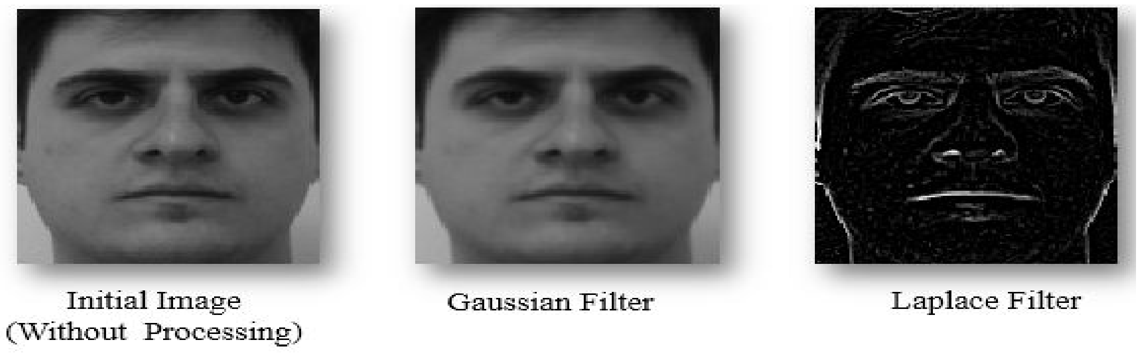


Figure 3.23 – *Example of the input images in the experiments after applying filters.*

After the similarity measure is calculated by the distance introduced by the SAX (*MIDIST*), we apply the KNN and SVM algorithms to classify and estimate the pose. Similar to the first protocol, the poses are split into three main classes: class of frontal pose, class of the left views, and a class of the right views.

Results are shown in Table 3.10 and Table 3.11.

Table 3.10 – *Results of KNN classification rate with MIDIST.*

		without preprocessing%	Gaussian filter%	Laplacian filter%
FEI Database	Left	98.57	97.95	98.62
	Frontal	98.20	97.70	98.00
	Right	98.75	98.62	98.25
GTAV Database	Left	99.34	99.34	99.18
	Frontal	97.44	97.36	99.25
	Right	99.44	99.44	99.91
FERET Database	Left	96.36	98.82	99.32
	Frontal	96.66	97.82	98.93
	Right	96.80	98.67	99.44

Table 3.11 – *Results of SVM classification rate with MIDIST.*

		without preprocessing%	Gaussian filter%	Laplacian filter%
FEI Database	Left	97.63	95.49	96.33
	Frontal	97.22	95.64	96.27
	Right	98.26	97.52	98.00
GTAV Database	Left	99.06	98.88	99.43
	Frontal	97.78	99.25	95.71
	Right	98.50	98.68	99.05
FERET Database	Left	94.48	98.40	98.12
	Frontal	91.63	99.00	99.15
	Right	92.74	98.95	98.94

In the second experiment, we measure the similarity with the edit distance (equation 3.3) for the three bases, also with and without filtering pretreatment. Results are illustrated in Table 3.12 and Table 3.13.

Table 3.12 – *Results of KNN classification rate with Edit distance.*

		without preprocessing%	Gaussian filter%	Laplacian filter%
FEI Database	Left	98.08	98.37	97.88
	Frontal	97.35	97.54	97.44
	Right	98.00	97.95	98.31
GTAV Database	Left	99.91	100	100
	Frontal	98.88	100	99.26
	Right	99.81	100	99.81
FERET Database	Left	98.33	98.33	99.03
	Frontal	96.83	96.88	98.04
	Right	98.14	98.18	99.10

Table 3.13 – *Results of SVM classification rate with Edit distance.*

		without preprocessing%	Gaussian filter%	Laplacian filter%
FEI Database	Left	96.16	96.20	96.46
	Frontal	95.90	95.00	95.54
	Right	97.74	96.52	96.50
GTAV Database	Left	98.87	100	99.43
	Frontal	96.24	100	98.53
	Right	98.31	100	99.11
FERET Database	Left	98.40	98.20	98.11
	Frontal	99.10	99.01	99.20
	Right	98.95	98.95	99.00

We can summarize from the Table 3.10, 3.11, 3.12, and 3.13:

- **FEI database**:
  - The highest classification rates are obtained using the KNN for both *MINDIST* and the Edit distance.
  - The results for all poses are almost similar with the images without processing and Laplace filter.
  - *MINDIST* measure proved better classification rates than the Edit distance for KNN and SVM.
  - The left and the right views are mostly classified with the correct class compared to the frontal poses.
- **GTAV database**:

  - We can note that the results achieved using the Edit distance are better than *MINDIST* for both KNN and SVM.
  - The most poses have been correctly classified, especially when we used preprocessed images.

- **FERET database**:
  - The frontal poses have reached the best classification rate using SVM for both the Edit distance and *MINDIST*.
  - Mostly for all views, the best results were obtained when using the Laplace filter with rates the range of 98% and 99%.

Overall, in the Edit and *MINDIST*-based that are used with both KNN and SVM classifications, the best results were obtained using the Laplace filter with rates exceeding 98%, and the classifications rate that are achieved using Edit distance are better than what were achieved by *MINDIST* in the most cases.

However, we should notice that even without preprocessing filters application, results still extremely important. Moreover, we should put into consideration that the FEI database contains images with low lighting (figure 3.22), and GTAV database contains images with different lighting.

3.6.4 Processing Time of SAX representation

Firstly, we compare the computational time in terms of feature extraction, for our proposed method based on SAX representation, and Histogram of oriented gradients (HOG). HOG feature is widely used to estimate the head orientation [Diaz-Chito et al., 2018, Alioua et al., 2016, Wang et al., 2013a, Vo et al., 2019, Saeed et al., 2015]. To extract the feature using HOG, the image is split into cells or small regions, and the orientation gradients are calculated for each pixel in the cell. Then, a 1D histogram of the orientation feature is formed from each cell.

In Table 3.14, we compare the feature extraction speed. We provide the time required to create the SAX, and HOG with different sizes of HOG cells. The size of the face image has been normalized to 100×100. The SAX representation is performed with frame size $w = 5$, and alphabet size $a = 5$.

It seems that the SAX takes a long time to generate a feature vector than the HOG vectors but, this is the time needed to map the image to a numeric vector and to convert this vector to SAX sequence, while the generation of the HOG vector is not enough. The use of HOG vector is required feature reduction methods before using the final feature vector in classification or regression task to predict the head orientation[Diaz-Chito et al., 2018, Vo et al., 2019]. Therefore, using HOG necessitates additional time compared to the SAX representation due to the high dimension of the feature vector (Table 3.14). And often, the HOG vector is combined with other descriptors to provide en excellent feature vector representation. [Alioua et al., 2016, Saeed et al., 2015].

Table 3.14 – *The computational time to generate the SAX and the HOG vector.*

	SAX 5,5	HOG 2x2	HOG 4x4	HOG 5x5	HOG 8x8	HOG 9x9
Time (s)	0.0969	0.0185	0.0153	0.0151	0.0149	0.0142
Length of feature Vector	2000	86436	20736	12996	4356	3600

Secondly, we compare the computational time consumed to calculate *MIDIST* and Edit distance. Obviously from Table 3.15, we notice that the Edit distance is more time-consuming than *MIDIST*.

Table 3.15 – *The computational time to measure the similarity of two time series.*

		w=45	w=10	w=5
Time(s)	*MIDIST*	0.0710	0.1619	0.2986
	Edit distance	0.0426	0.6405	2.3206

Although the Edit distance provides good results on the major case, it consumes excessive execution time, which increases with the number of images in the database.

Thus, using SAX representation allows us to decrease the time of feature extraction, as well as the time-consuming to measure the similarity.

3.6.5 Peano-Hilbert vs Sweep techniques

In this section, we compare the classification rate of head pose in terms of the way to generate the face time series. We have carried out the experiment using Peano-Hilbert and Sweep to map the face images, which are resized to 128×128 on the three database. Then, we have converted each face time series to SAX sequences with frame size $w = 45$, and alphabet size $a = 5$.

In Table 3.16, we report the classification rates reached using KNN, and SVM.

Table 3.16 – *The KNN and SVM Classification rate using Peano-Hilbert and Sweep techniques.*

		KNN		SVM	
		Peano-Hilbert	Sweep	Peano-Hilbert	Sweep
FEI Database	Left	98.61	94.02	96.33	91.40
	Frontal	98.00	93.44	96.27	89.90
	Right	98.25	93.69	98.00	92.31
GTAV Database	Left	99.18	96.71	99.43	95,21
	Frontal	99.25	91.80	95.71	86.96
	Right	99.91	96.84	99.05	93.84
FERET Database	Left	99.32	94.53	98.12	94.82
	Frontal	98.93	89.31	99.15	90.70
	Right	99.44	95.187	98.94	94.11

It is clear to see that using the Peano-Hilbert curve to generate the face time series is much better than using the Sweep curve. On all databases, the frontal view is extremely labelled with the correct pose for both KNN and SVM using the Peano-Hilbert curve. While the frontal pose did not exceed a 93% when we used the Sweep curve. The same note for the left and the right views, where almost all poses have been well classified in the case of face time series create by the Peano-Hilbert curve, and in the case of Sweep curve the high classification rate did not exceed a 96%. We can conclude from all these results that our proposed method still very competitive in terms of high classification rate and low time computation even when the acquisition conditions (lighting, resolution) are not controlled. As well as, the Peano-Hilbert technique preforms better than Sweep curve.

3.7 Conclusion

Aiming to estimate the head orientation, we have presented a new approach in this chapter. The proposed method is simple, easy to implement, robust, accurate, and computationally efficient. The method presents another way to extract the facial features by representing a face image as time series. The matrix representation of the face image is mapped into a 1D vector of a time series using two kinds of space-filling curves, while each position in the time series would represent the intensity of the corresponding pixel in the input 2D image. Besides, each time series is encoded with SAX symbolic representation to convert the numerical series to a symbolic sequence. Several classification methods throughout the generated big data sets of similarity matrices were used to create classifiers of frontal vs. profile faces' poses. As well, we investigated the influence of SAX parameters (frame size w, alphabet size a) on the classification rates of facial poses. The experimental results have shown that our approach is robust and allows us to separately classify the poses even in degraded conditions.

In this chapter, we have classified the facial pose in heavily three-class only (frontal, left/right views) using simple classifiers (KNN, SVM). Therefore, we will seek in the next chapter to improve the performance of the classification task. We will suggest performing a deep learning model to classify the head pose in a large range of angles instead of three classes.

4 Neural networks and deep learning for head pose estimation

Contents

4.1 Introduction

In the previous chapter, we have dealt with the problem of head pose classification as a problem of time series classification, as well as, we have shown the usefulness of SAX representation in performing the head pose estimation model. However, we used classical algorithms to select the frontal pose among the other poses. In the same context of this chapter, we cast the head orientation problem into a natural language processing problem. Inspired by the advances in machine translation, we introduced a novel technique called SAX Recurrent Encoder-Decoder (SAX-RED) as shown in figure 4.1, which is based on sequence-to-sequence neural network (Seq2Seq) [Sutskever et al., 2014]. The model takes the SAX sequence as an input of the encoder which learns the representation of this sequence, and from this representation, the decoder generates the outputs which describe the head pose in different views (yaw, pitch).

The fact that the key insight of the Seq2Seq model is the encoder-decoder, which is a recurrent neural network, we start this chapter with an overview of neural networks and deep learning architecture, especially the recurrent neural network. Subsequently, we provide a general idea for understanding the concept of Sequence-to-Sequence model, and how it works. As well as how it can leverage and use this model to build a model to estimate the head pose orientation.

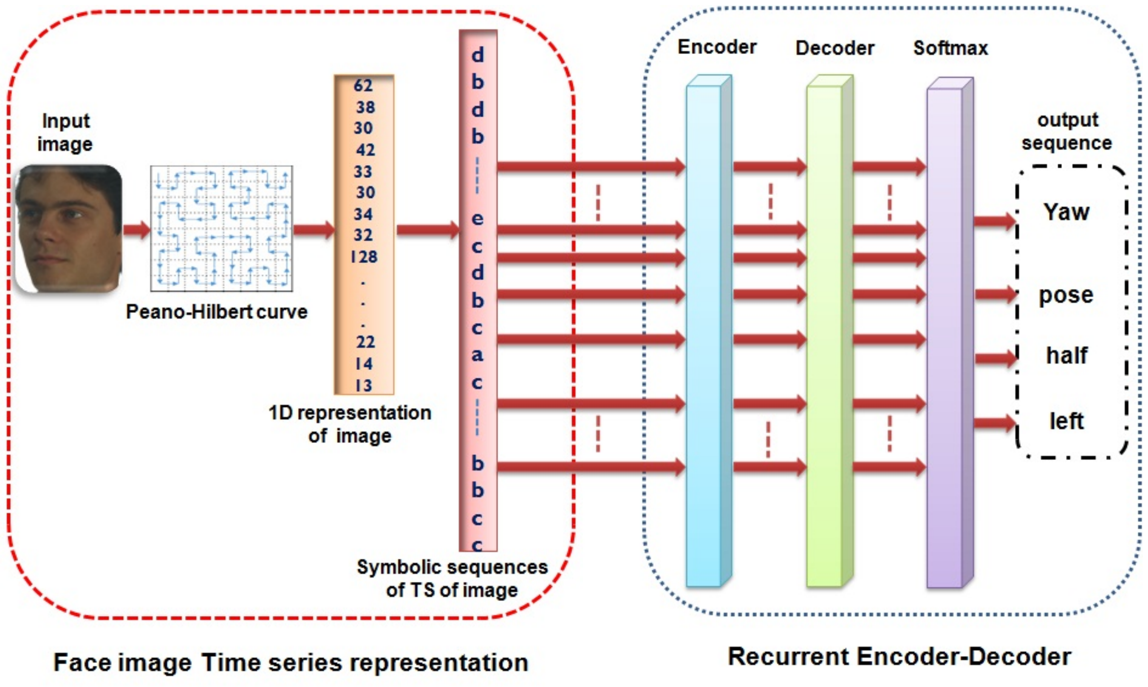

Input image
Peano-Hilbert curve
62
38
30
42
33
30
34
32
128
22
14
13
1D representation of image
d b d b e c d b c a c b b c c
Symbolic sequences of TS of image
Encoder
Decoder
Softmax
output sequence
Yaw
pose
half
left
Face image Time series representation
Recurrent Encoder-Decoder

Figure 4.1 – *An overview of SAX-RED for face pose estimation.*

4.2 Neural Networks and Deep Learning

Neural networks are a type of machine learning that mimic the behaviour of the human brain, and allow computer programs to recognize patterns and solve common problems in the areas of artificial intelligence (AI), machine learning, and deep learning (figure 4.2).

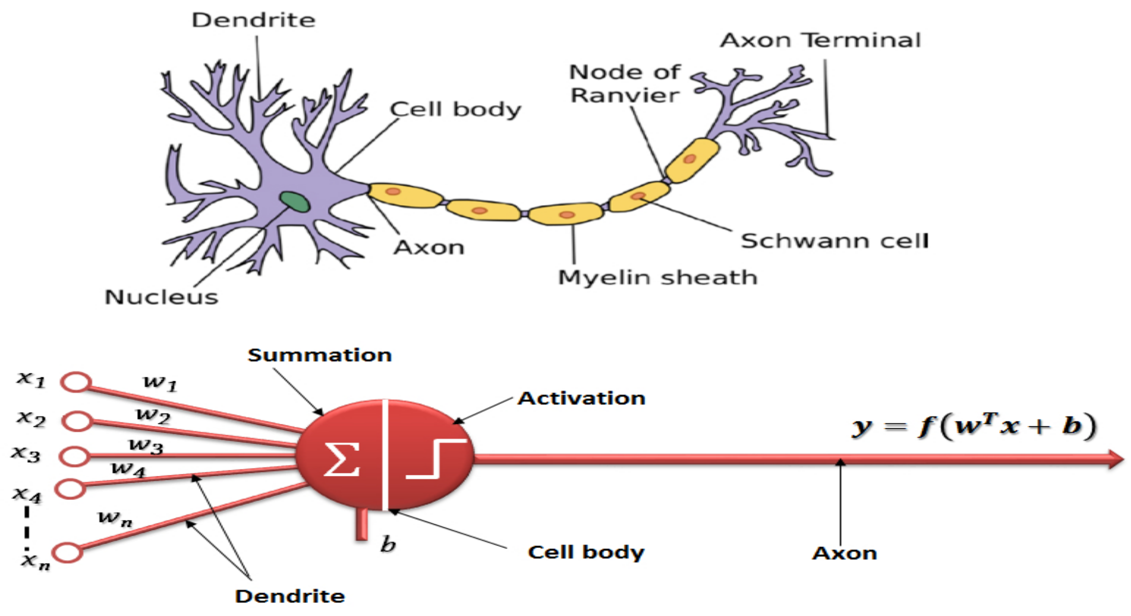


Figure 4.2 – *Illustration comparing a biological neuron to an artificial neuron.*

Neural networks arose in the 1940s, but figuring out how to train them has been a mystery for 20 years. The first step towards neural networks took place in 1943 when McCulloch and Pitts [McCulloch and Pitts, 1943] developed a mathematical model of a neuron that imitates the human brain called an MCP model. It is regarded as the first foundation of an advanced future of artificial neural networks. In 1958, there was the second event that is considered as the birth of scalable network models which is susceptible to learning Rosenblatt's innovation [Rosenblatt, 1957]. He has created the first operational model of a learning network: the Perceptron that demonstrates its ability to solve classification problems. This model was the first to perform pattern recognition. Three years later, Bernard Widrow and Marcian Hoff [Widrow and Hoff, 1960] developed both the ADALINE (ADAptive LINear) and MADALINE (Multiple ADAptive LINear Elements) models. MADALINE was the first neural network to be applied to a real-world problem.

Progress on neural network research roughly halted for a decade, until the 1980s. This interruption was directly due to the work of Minsky and Papert

[Minsky and Papert, 1969] published in 1969. The authors highlighted the intrinsic limits of perception, which related to linear separability and associated representation problems. In other words, the perceptron cannot deal with non-linearly-separable data sets. Or that they relate to the complexity of algorithms. Despite the emergence of the backpropagation concept by P.Werbos [Werbos, 1974] in 1974, the research in the field of artificial neural networks has remained drops sharply.

After a period of oblivion, neural networks are receiving a lot of attention again from the 1980s to the current day. Hopfield [Hopfield, 1982] presented an addressable memory neural network in 1982, which is an example of recurrent neural networks. Twelve years after the suggestion of backpropagation to learn the weighted in neural networks, David Rumelhart et al.[Rumelhart et al., 1986] reinvented it. They have used backpropagation for training Multilayer Perceptron (MLP). This model has been able to overcome the limitation of the old neural networks that can solve the nonlinear problems. In 1989, LeCun et al.[LeCun et al., 1989] trained a Convolutional Neural Network with the backpropagation algorithm to learn handwritten digits. Further, LeCun et al.[LeCun et al., 1998] combined the Stochastic Gradient Descent algorithm (SGD) with the backpropagation to build LeNet-5 architecture. LeNet-5, which consists of 7 layers, was learned for classifying hand-written numbers on checks. These works are regarded as emanating the era of Convolutional Neural Networks. Until 2006, this field was known as the Artificial Neural Network, which was renamed deep learning when Hinton et al.[Hinton et al., 2006] have been learned Deep Belief Networks using a greedy layer-wise unsupervised strategy. And from that time until now, research projects are continuing.

The major contributions in artificial neural networks area are presented in Table 4.1.

Table 4.1 – *Important events in the history of artificial neural networks.*

Contributor	Contribution	Year
McCulloch & Pitts [McCulloch and Pitts, 1943]	MCP model	1943
Hebb [Hebb, 1949]	Hebbian learning rule	1949
Rosenblatt [Rosenblatt, 1957]	Perceptron model	1958
Widrow [Widrow and Hoff, 1960]	ADALINE/MADALINE models	1959
Werbos [Werbos, 1974]	Backpropagation	1974
Ackley [Ackley et al., 1985]	Boltzmann Machine	1985
Smolensky [Smolensky, 1986]	Restricted Boltzmann Machine (RBM)	1986
Jordan [Jordan, 1997]	Recurrent Neural Network	1986
Rumelhart [Rumelhart et al., 1986]	Multilayer Perceptron (MLP)	1986
Hochreiter [Hochreiter and Schmidhuber, 1997]	LSTM	1997
LeCun [LeCun et al., 1998]	LeNet-5	1989
Hinton [Hinton et al., 2006]	Deep Belief Network (DBN)	2006
Bengio [Bengio et al., 2006]	Deep Autoencoder based networks	2006
Salakhutdinov & Hinton	Deep Boltzmann machines	2009
Krizhevsky [Krizhevsky et al., 2012]	AlexNet	2012
Bengio [Bengio et al., 2013]	Deep Transfer Learning (DTL)	2013
Fukushima [Fukushima, 2013]	Feedforward Neural Networks (D-FFNN)	2013
Szegedy [Szegedy et al., 2015]	GoogLeNet	2014
Simonyan [Simonyan and Zisserman, 2014]	VGGNet	2014
Goodfellow [Goodfellow et al., 2014]	Generative Adversarial Networks (GAN)	2014
He et al. [He et al., 2015]	Residual Network (ResNet)	2015
Huang [Huang et al., 2017]	Dense Convolutional Network (DenseNet)	2016
Arjovsky [Arjovsky et al., 2017]	Wasserstein GAN (WGAN)	2017
He [He et al., 2017]	Mask R-CNN	2017
SMA Eslami [Eslami et al., 2018]	Generative Query Network (GQN)	2018

Deep learning or hierarchical learning is a new technology introduced in machine learning research, which achieved great success in several fields such as pattern and speech recognition, computer vision, autonomous driving, image classification, natural language processing, and machine translation. Deep learning models are built by successive layers (neural network layers), where the number of these layers determines the depth of the model, which means the word "deep" in deep learning. Unlike shallow learning where the model cares to learn one or two layers only to represent the data, deep learning has the ability to provide a representation of data automatically using multiple layers and learn hierarchically the abstractions of data. The features are extracted in each level (layer), from the low level (input data) to the high level (output), and the information in each level is combined to create a useful representation as shown in figure 4.3.

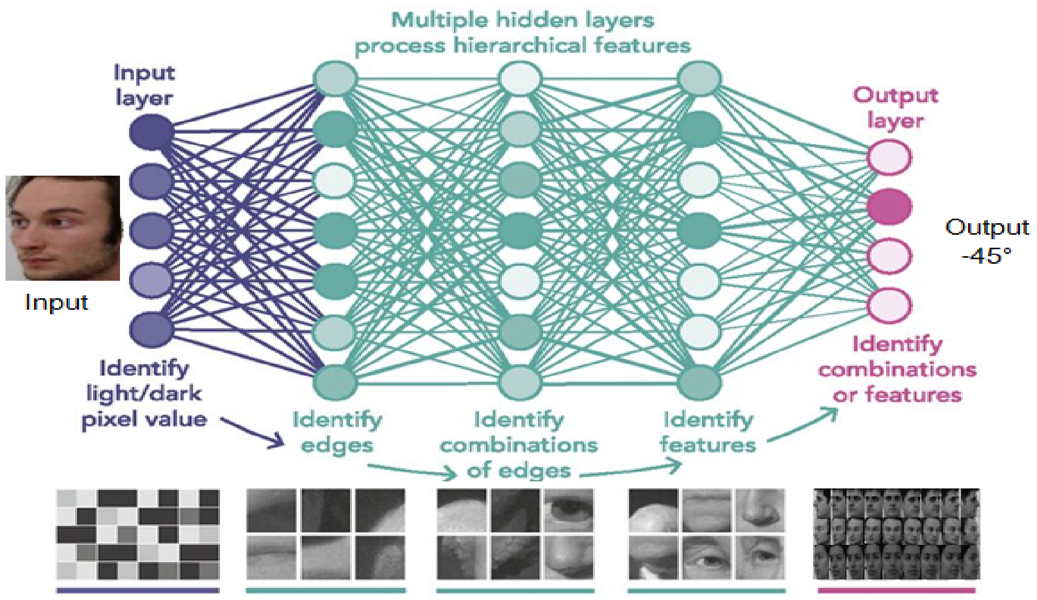


Figure 4.3 – *Illustration of the deep neural network model.*

Various types of neural networks with very different properties have been developed. The most popular types of deep neural networks are known as Convolutional Neural Networks (CNNs) [Gu et al., 2018], and Recurrent neural networks (RNNs) [Salehinejad et al., 2017] .Due to our approach that relies mainly on the Seq2Seq model, we will provide more details on the RNNs in this thesis.

4.3 Recurrent Neural Networks (RNNs)

RNN is one of the most popular deep neural networks that widely used to deal with the ordinal or temporal problems according to its internal memory, such as time series, speech recognition, image captioning, signal processing, or natural language processing (NLP). The architecture of a simple RNN is composed of three layers, which are input, recurrent hidden, and output layers, as presented in figure 4.4. The basic idea behind the RNN is the feedback connection between hidden units over time that provides a recurrence mechanism. This means that the output of recurrent neural networks depends on the previous elements within the sequence. In other words, the current input is the output of the prior time.

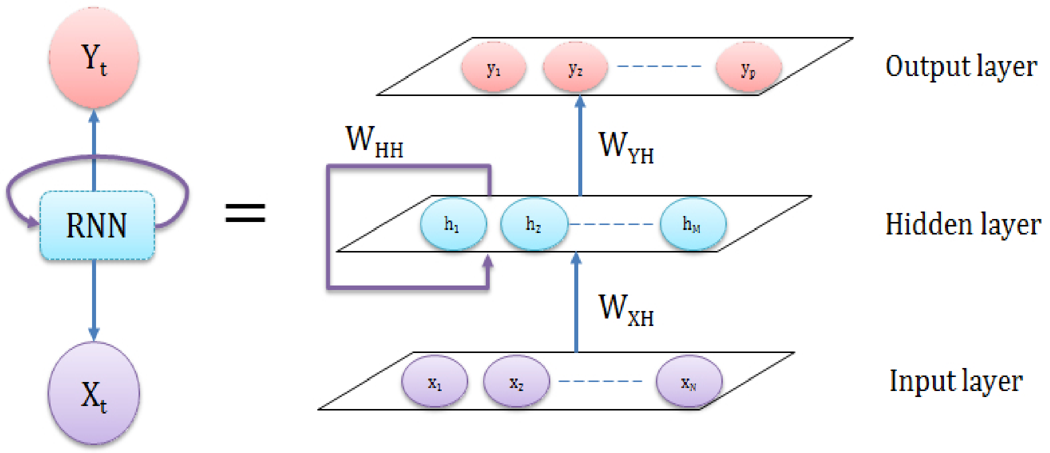


Figure 4.4 – *Example of a simple RNN.*

The input layer receives a sequence of vectors $x_t = (x_1, x_2, ..., x_N)$ and processes them one at a time. At each time step t, the recurrent neural network is based on the input vector x_t and previous hidden state h_{t-1} to update its memory and produces a hidden state h_t as in equation 4.1.

$$h_t = f(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{4.1}$$

where f is the hidden layer activation function, W_{ih} is the weight matrix from the input-to-hidden, W_{hh} is the matrix of recurrent weights between the hidden layer and itself, and b_h is the bias vector of the hidden units. These weight matrices are known as parameters of RNN, which are calculated and optimized to obtain a powerful model.

The output layer is computed as:

$$O_t = g(W_{ho}h_t + b_o) \tag{4.2}$$

where g is the activation functions, W_{ho} is the weight matrix from the hidden-to-output, and b_o is the bias vector in the output layer.

Recall that f and g are activation functions applied to hidden and output layers, respectively. The activation functions are one of an imperative component of a deep neural network that can control the output of a deep learning model, its accuracy, and the computational efficiency of training a model. Moreover, they have a considerable impact on the neural network's ability to converge and the convergence speed. They play the role of a gate between the input feeding the current neuron and its output going to the next layer, where they take a decision, whether a neuron should be activated or not. In other words, the activation functions turn the neuron output as a classifier, that classifies incoming information as useful or less-useful depending on a rule or threshold.

The original and more granular activation function used for RNN modules are: Sigmoid, TanH (Hyperbolic Tangent), and ReLU (Rectified Linear Unit).

4.3.0.1 The sigmoid function

The sigmoid is a nonlinear function, its curve is S-shaped (figure 4.5). It is mathematically defined by equation

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4.3}$$

The sigmoid transforms and normalizes the output of each neuron between the range 0 and 1. More to the point, this function returns a value :

- $y \to 1$ if the input to a sigmoid function is a positive large number ($e^{-x} \to 0$),
- $y \to 0$ if the input to a sigmoid function is a negative large number ($e^{-x} \to \infty$),
- $y \to \frac{1}{2}$ if the input to a sigmoid function is 0 ($e^{-x} = 1$)

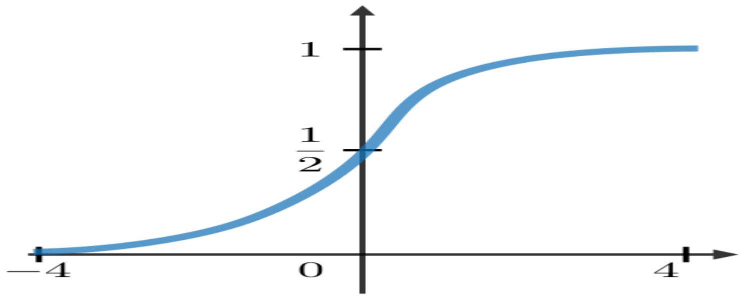


Figure 4.5 – *The Sigmoid function.*

4.3.0.2 The Tanh function

The *tanh* function is similar to the sigmoid function. The only difference is that it is symmetric around the origin. Mathematically, *tanh* is a shifted version of the sigmoid function. The *tanh* activation function is expressed as

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.4}$$

In this case, the function returns a value :

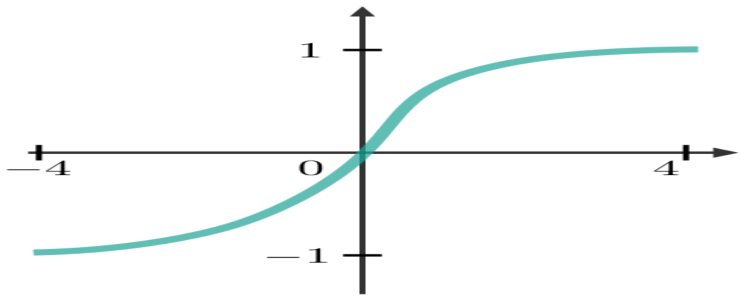


Figure 4.6 – *The Tanh function.*

- $y \rightarrow 1$ if the input to a sigmoid function is a positive large number ($e^{-x} \rightarrow 0$),
- $y \rightarrow -1$ if the input to a sigmoid function is a negative large number ($e^{x} \rightarrow 0$),
- $y \rightarrow 0$ if the input to a sigmoid function is 0 ($e^{-x} = 1$)

Thus, the *tanh* activation functions can output values between -1 and +1.

4.3.0.3 The ReLu function

The rectified linear unit (ReLu) is one of the key elements that has revolutionized deep learning. The use of the ReLu non-linearity in the hidden layers can significantly speed up the training time. In essence, the function returns 0 if it receives a negative input, and will output the input directly if it receives a positive value. The function is defined as:

$$f(x) = max(0, x) \tag{4.5}$$

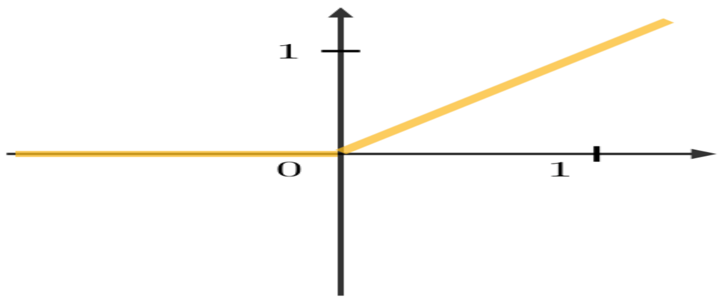


Figure 4.7 – *The ReLu function.*

4.3.0.4 The Softmax function

The $softmax$ function is useful for neural networks, which is widely used in multiple classifications. Whereas it can normalize an input value into a vector of values that sum to 1. Whatever the value of the input is positive, negative or zero, the $softmax$ transforms them into values in the range (0,1).

The $softmax$ function is defined as:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.6}$$

The soft function is emphasizing to the large values in the input and drops the smaller ones, that is why it is called by this name. The property that the sum of all the values equals 1 makes it a great function to formulate the probability distribution.

4.3.1 Training recurrent neural networks

The purpose of learning a neural network is usually to address supervised learning problems i.e. problems seeking to find a function $y = f(x)$, that can present the relationship between a given labelled database $(Y; X)$ (frequently known as "training set"), and generalize the results to unlabelled data (referred to as the "test set").

In fact, the role of training of RNN is to provide an approximated function $\hat{y} = f(x)$, and to minimize the loss function definer by the difference between the RNN output "$\hat{y}$" and the desired output "y". In RNN, the loss function is computing for each time step, while the total error was simply the sum of the losses across all time steps (equation 4.7).

Generally, The training of RNN is regarded as a challenging task due to the difficulty to initialize the weights in the network, and the complexity of selecting the optimization algorithm that ensures minimal training loss. In other words, the most straightforward manner to training RNN is to determine the optimal parameters i.e. weights matrices and biases, that can decrease this loss function.

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T} \mathcal{L}_t(\hat{y}_t, y_t) \tag{4.7}$$

Where $\hat{y}_t$ and y_t are the predicted output and the desired label at timestamp t respectively.

Typically, the training of RNNs requires the reiteration of two steps, the forward pass, and the backward pass. The forward pass is described in equations (4.1), (4.2). As we have mentioned above, at the time "t", the vector of the input data "x_t" is multiplied by the input weight matrix W_{ih}, then connects the input layer to the hidden layer and added to the weight hidden unit of the previous step "$t-1$" define with z_t, as shown in equation (4.8). b_t is the bias vector at time step t.

Then this result is transformed by the hidden layer activation function $f()$ to produce the hidden state h_t (equation (4.9))

In the same manner, the hidden state h_t produced is multiplied by the weight W_{ho} which connects this layer to the output layer which define v_t (equation 4.10), and the output of neural network O_t is obtained from the output of activation function $g()$ as shown in equation (4.11). Then, the output of neural network O_t is compared to the desired target and calculates the error between them.

$$z_t = W_{ih}x_t + W_{hh}h_{t-1} + b_t \tag{4.8}$$

$$h_t = f(z_t) = f(W_{ih}x_t + W_{hh}h_{t-1} + b_t) \tag{4.9}$$

$$v_t = W_{ho}h_t + b_o \tag{4.10}$$

$$O_t = g(v_t) = g(W_{0h}h_t + b_o) \tag{4.11}$$

The backward phase is executed after the forward phase. This phase is accomplished using Backpropagation-through-time (BPTT) [Werbos, 1990, Sutskever, 2013], which is the application of the backpropagation training algorithm to learn the sequence data. The ultimate goal of the BPTT algorithm is to adjust the weights to minimize the error (loss function) of the network outputs. Mathematically, BPTT calculates the partial derivatives of the loss function $\mathcal{L}$ with respect to the synaptic weights.

Basically, BPTT propagated backward in time the gradient descent algorithm to compute the weight change of each layer from the output towards the input layer based on the chain rule. Subsequently, the gradient is used by an optimization algorithm to update each parameter in the RNN (typically the weights). Therefore, the gradient of the RNN parameters is computes for each layer from the last layer to the input layer i.e. $t = T \; to \; t = 1$.

The error e_T at the output layer i.e. at $t = T$ is:

$$\frac{\partial \mathcal{L}}{\partial v_T} = \frac{\partial \mathcal{L}}{\partial g(v_T)} \frac{\partial g(v_T)}{\partial v_T} = e_T \tag{4.12}$$

and for each time steps t is:

$$\frac{\partial \mathcal{L}}{\partial v_t} = \frac{\partial \mathcal{L}}{\partial g(v_t)} \frac{\partial g(v_t)}{\partial v_t} = e_t \tag{4.13}$$

It is worth pointing out that, the loss function $\mathcal{L}_T$ at time step t is impacted only by the output layer at this current time step. While computing the gradient at the hidden layer, an intermediate-term is added. This term presents the recurrence of the hidden layer. In other words, the change in the hidden layer at the time step t affects the loss function at the actual time step and the loss at the next step.

At the final time step T, the error from the hidden layer δ_T, affects only the final step:

$$\begin{aligned}
\delta_T = \frac{\partial \mathcal{L}}{\partial z_T} &= \frac{\partial \mathcal{L}}{\partial f(z_T)} \frac{\partial f(z_T)}{\partial z_T} \\
&= \frac{\partial \mathcal{L}}{\partial v_T} \frac{\partial v_T}{\partial f(z_T)} \frac{\partial f(z_T)}{\partial z_T} \\
\implies \delta_T &= W_{ho}^T e_T \odot f'(z_T) \qquad (4.14)
\end{aligned}$$

Where $W_{ho}^T = \frac{\partial v_T}{\partial f(z_T)}$, $f'(z_T) = \frac{\partial f(z_T)}{\partial z_T}$, and $'' \odot ''$ is the element wise multiplication operator.

Thus, the error in the hidden layer δ_t, at the time step t is computed as follows:

$$\begin{aligned}
\delta_t = \frac{\partial \mathcal{L}}{\partial z_t} &= \frac{\partial \mathcal{L}}{\partial v_t} \frac{\partial v_t}{\partial z_t} + \frac{\partial \mathcal{L}}{\partial z_{t+1}} \frac{\partial z_{t+1}}{\partial z_t} \qquad (4.15) \\
&= W_{ho}^T e_t \odot f'(z_t) + W_{hh}^T \delta_{t+1} \odot f'(z_t) \\
\implies \delta_t &= (W_{ho}^T e_t + W_{hh}^T \delta_{t+1}) \odot f'(z_t) \qquad (4.16)
\end{aligned}$$

It can notice from equation (4.15), that to calculate the error of the hidden layer at time step t, it is required to identify the error at time step $t+1$ previously. Consequently, it is necessary to know the error at time step $t+2$ to calculate the error at time step $t+1$ and so forth. For this reason, it should be commenced from the last time step T, and proceed backward through time to calculate the previous time steps.

To minimize the loss function $\mathcal{L}$, the derivation of the error should be calculated with respect to all the weight matrices across the time.

$$\begin{aligned} dW_{ho} &= \sum_t \frac{\partial \mathcal{L}}{\partial W_{ho}} = \sum_t \frac{\partial \mathcal{L}}{\partial v_t} \frac{\partial v_t}{\partial W_{ho}} \\ dW_{ih} &= \sum_t \frac{\partial \mathcal{L}}{\partial W_{ih}} = \sum_t \frac{\partial \mathcal{L}}{\partial z_t} \frac{\partial z_t}{\partial W_{ih}} \\ dW_{hh} &= \sum_t \frac{\partial \mathcal{L}}{\partial W_{hh}} = \sum_t \frac{\partial \mathcal{L}}{\partial z_t} \frac{\partial z_t}{\partial W_{hh}} \end{aligned} \tag{4.17}$$

By applying the formulas from equations (4.12), (4.13), (4.15) in equation (4.16), the gradient of the loss function with respect to the network's parameters can be written as follows:

$$\begin{aligned} dW_{ho} &= \sum_t e_t h_t^T \\ dW_{ih} &= \sum_t \delta_t x_t^T \\ dW_{hh} &= \sum_t \delta_t h_{t-1}^T \end{aligned} \tag{4.18}$$

An important step to learn an RNN is the update of weights via an optimization algorithm. The mini-batch Stochastic Gradient Descent (SGD) [Ruder, 2016] is often used to carry out this step. It is the preferred way to optimize neural networks' parameters.The parameters are updated at each optimization iteration, according to equation (4.19).

$$\theta_{t+1} = \theta_t - \eta \sum_{i=Bt}^{Bt+B} \frac{\partial \mathcal{L}_t}{\partial \theta} \tag{4.19}$$

Where $\theta = \{W_{hh}, W_{ih}, W_{ho}, b_h, b_i, b_o\}$, η is the learning rate and B is the size of the mini-batch. The training database is split into small batch sizes and performs updates on each of these batches rather than computing the gradient for the whole database. This algorithm is computationally efficient and fast. In our model, we used RMSprop [Hinton, 2012] which is one common variation of Stochastic Gradient Descent (SGD).

Despite RNN is exhibiting a higher ability to model the sequence data, it suffers from some problems. For long periods of time, RNN could be incapable to learn causal relationships between very long sequences. Moreover, during the BPTT process, RNN may be vulnerable to gradients issues which are known as vanishing, or exploding gradients. These phenomena occur when the gradients rapidly converge to zero (vanishing gradients), or diverge rapidly to infinity (exploding gradients) because RNN is unrolled backward through time.

4.3.2 Long short-term memory (LSTM)

Hochreiter and Schmidhuber [Hochreiter and Schmidhuber, 1997] introduced a new neural for the RNN family called Long Short Term Memory (LSTM), which designed a special memory cell, thereby avoiding the vanishing gradient problem. The LSTM architecture consists of a set of multiplicative gates, the Input (I), Output(O), and Forget(F) gates as illustrated in figure 4.8.

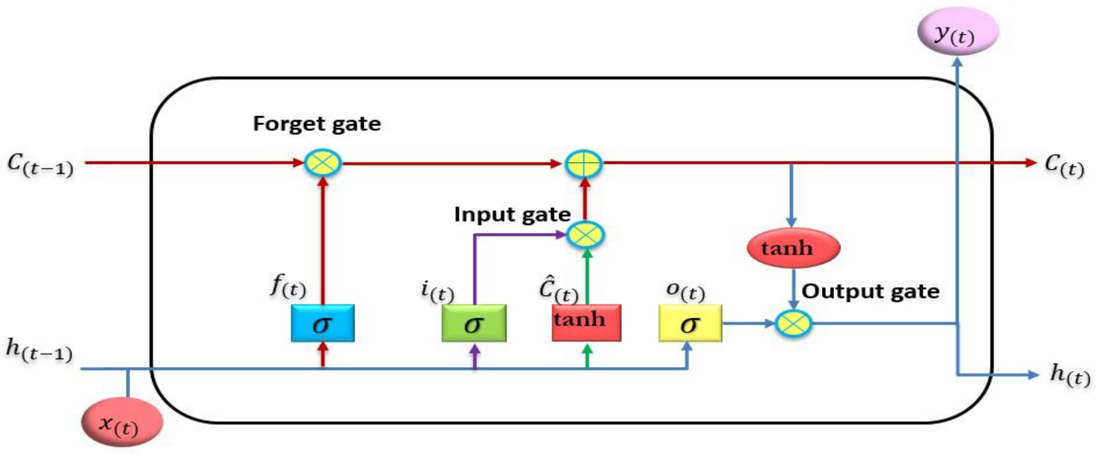


Figure 4.8 – *Example of LSTM memory cell.*

These gates control the circulation of the information that traverses the LSTM cell. They have the ability to keep, add, or remove information to the cell state, represented with vectors h_t, (short term state) and C_t (long term state). The forget gate dropped a part of memory, which is derived from the previous long term state C_{t-1}, short term state h_{t-1} and the input data x_t. Then it adds to a new memory selected by the input gate to update the long term state C_t using the following equations:

$$C_t = f_t \odot C_{t-1} + i_t. \odot \tilde{C}_t \tag{4.20}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4.21}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{4.22}$$

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_i) \tag{4.23}$$

For the short term state $''h''$, the "C" state is copied and run through $''tanh''$ and multiply it by the output of the sigmoid gate, following these equations:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4.24}$$

$$y_t = h_t = o_t \odot tanh(C_t) \tag{4.25}$$

where σ is the logistic sigmoid function, and $''i, f, o''$ and $\tilde{C}$ are respectively the input gate, forget gate, output gate, and cell activation vectors.

4.3.3 Gated Recurrent Units (GRU)

Gated Recurrent Units is another model of RNNs, introduced by Cho et al. [Cho et al., 2014] in 2014, to address the vanishing gradient problem. The GRU can be considered as a simplified version of LSTM which reduces the gates into two rather than three. Conceptually, a GRU combines the input and forget gates into one *"update gate"* and even it has a *"reset gate"* (figure 4.9). As the LSTM, the gates in GRU regulate the flow of information into this cell following these equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{4.26}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{4.27}$$

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{4.28}$$

$$h_t = (1 - z_t) \odot \tilde{h}_{t-1} + z_t \odot h_t \tag{4.29}$$

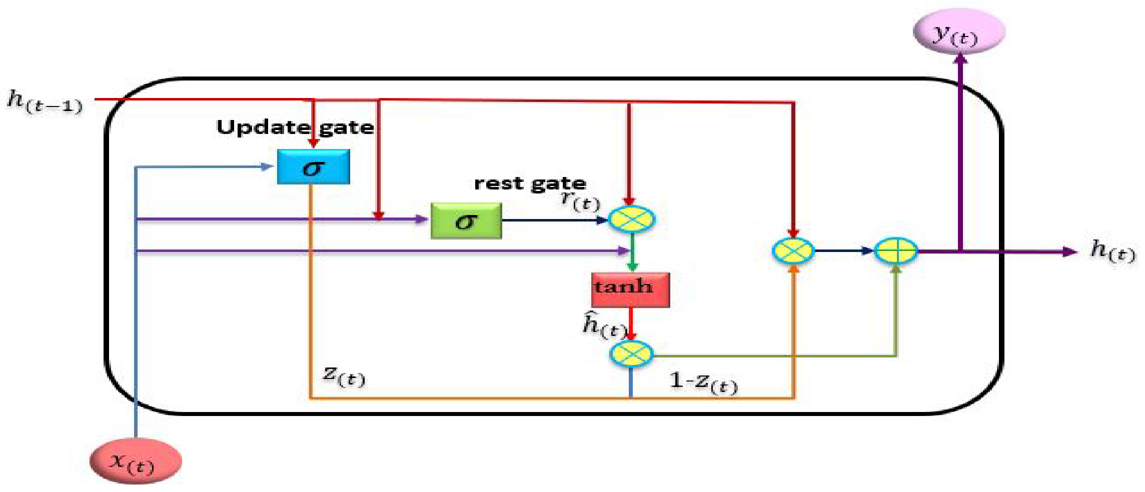


Figure 4.9 – *Example of GRU memory cell.*

4.3.4 Bidirectional RNNs (Bi-RNN)

Schuster et al. [Schuster and Paliwal, 1997] suggested a new type of RNNs intending to introduce a new structure to be bidirectional for processing data in a recurrent neural network. However, when this invention is applied to a sequence not only the information can go through the natural temporal sequences, but even further information can transform knowledge to previous time steps. The Bi-RNN functions as two RNNs, one handles the sequence from the first time step to the end in a forward direction in part, and the other for the backward direction in another part. The Bi-RNN is formulated as following:

$$h_F^t = \sigma(W_F x_t + U_F h_F^{t-1} + b_F) \tag{4.30}$$

$$h_B^t = \sigma(W_B x_t + U_B h_F^{t-1} + b_B) \tag{4.31}$$

$$y_t = \sigma(W_o h_f + W_o h_B + b_o) \tag{4.32}$$

Where W_F, U_F are Weights matrix in a forward direction and W_B, U_B are Weights matrix in a backward direction.
In addition to that, the combination of Bi-RNN with LSTM makes Bi-LSTM, and it can be possible to combine Bi-RNN with GRU to get Bi-GRU.

4.4 Sequence-to-Sequence model

Although the deep learning models are composed of CNN structure or RNN showed promising results for the problem in which their output is in a fixed-size either it is one or multi-classes, one or multi-values. However, these models are absolutely powerless to deal with the problem that has sequences data in variable size at the input and output, this issue spurred Sutskever et al. [Sutskever et al., 2014, Cho et al., 2014] to pioneer the sequence-to-sequence model. Sequence-to-Sequence is a deep learning model that takes sequence data as an input and generates another sequence at the output. It is the dominant architecture for many Natural Language Processing (NLP) models, such as neural machine translation, text summarization, speech recognition, Chatbots models, etc. In late 2016, Google used such a model in their Google Translate service [Wu et al., 2016].

To understand the process of such a sequence-to-sequence model, we provide an example of a neural machine translation model. This model is training to translate a source sentence "Je suis étudiant" into the target sentence "I am a student". As shown in figure 4.10, the core of the sequence-to-sequence model is the encoder and decoder used together in tandem to create a translation model.

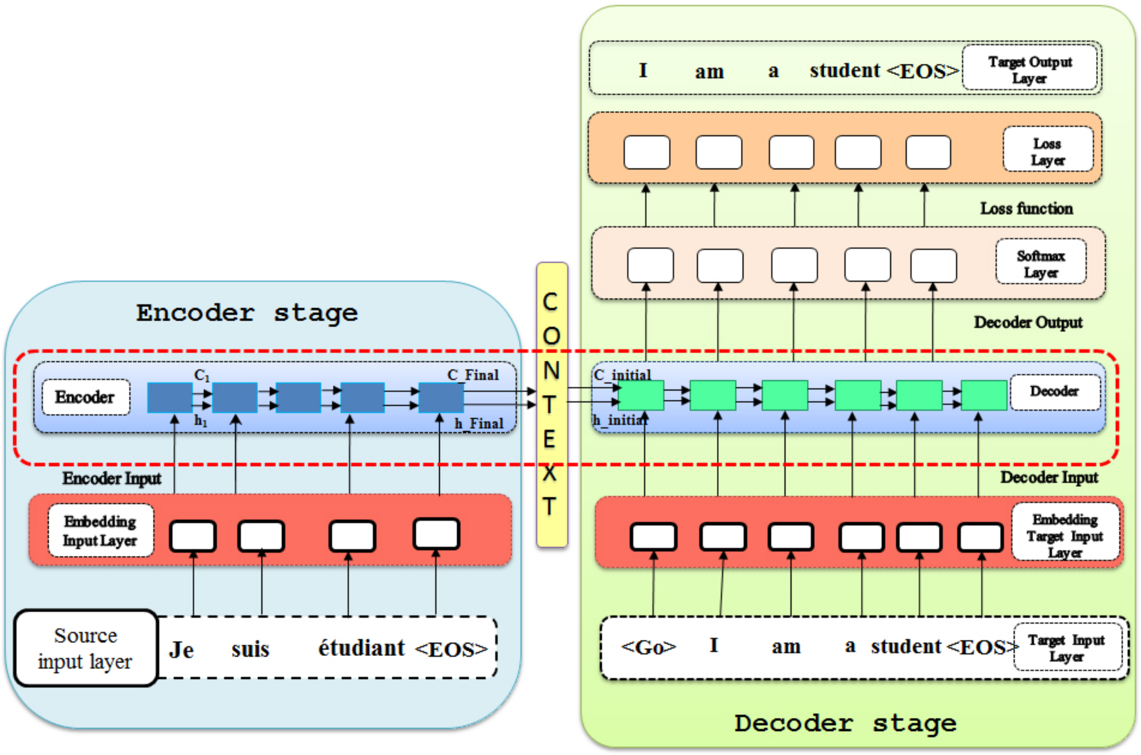


Figure 4.10 – *Illustration of neural machine translation model.*

The encoder is an RNN model that takes source sentences as input which learns to map a variable-length sequence into a fixed vector as a representation of this sentence. Then this representation is used to decode it into another variable-length sequence by the decoder.

Since the deep learning model can't deal with the sequence of the word directly, it requires converting this sentence into numeric form. For this reason, it is necessary to effect a pre-process to data before feeding the source sentence in the encoder and the target sentence in the decoder. The first step is tokenization. Tokenization is the task that segments the sentence into lists of words or vocabulary to generate a word index dictionary in which the word is the key and the corresponding integer is the value.In other words, the sentence is tokenized by assigning each unique word to an integer value. Moreover, it adds the $< EOS >$ token at the last of each sentence to identify its end. Besides, it indicates to the decoder to start the decoding process, the start of sequence such as $< Go >$ or $< SOS >$ token is inserted at the start of the target sentence.

In practice, the neural machine translation model is learning to translate numerous sentences (e.g. using a mini-batch training approach). Certainly, the length of sentences can vary. The filling way allows the sentences to be a fixed length. This action helps to promote the RNN (LSTM or GRU) execution because it expects the input instances to have the same length. While the short input sentences are appended with a $< Pad >$ token for the sake of the same length of the longest sentence in the source sentence. Similarly, the short target sentences are filled till becoming in the same length as the longest target sentence.

Subsequently, the embedding layers map each token to a fixed-length embedding vector for both the input and target sentences. Consequently, these embedding layers permit creating a dictionary where the word token is the key and the corresponding vector is the value. Further, They can learn the difference and relationship between the words.

For example, if we have these two sentences as input:

- "Je suis étudiant",
- "J'ai vu un chat sur tapis"

And the target sentences:

- I am a student
- I saw a cat on a mat

Assume that they are tokenized as sequences:

Source Sentence										
Sentence 1	*Word*	Je	suis	étudiant	Pad	Pad	Pad	Pad	Pad	EOS
	Token	70	189	68	00	00	00	00	00	99
Sentence 2	*Word*	Je	ai	vu	un	chat	sur	un	tapis	EOS
	Token	70	15	115	90	62	78	90	180	99
Target Sentence										
Sentence 1	*Word*	Go	I	am	a	student	Pad	Pad	Pad	EOS
	Token	01	45	115	80	162	00	00	00	99
Sentence 2	*Word*	Go	I	saw	a	cat	on	a	mat	EOS
	Token	01	45	158	80	50	120	80	156	99

Table 4.2 – *Example of sentences tokenization.*

In this example, the longest length of the source sentence is 8, and the longest length of the target sentence is 7, so the short sentences are appended with $< Pad >$ token that coded with zero. The $< EOS >$ added at the end of the sentences. As well as, the $< Go >$ token is inserted at the outset of the target sentences. An integer is assigned to each word. Therefore, the sentences are converted to vectors as follows:

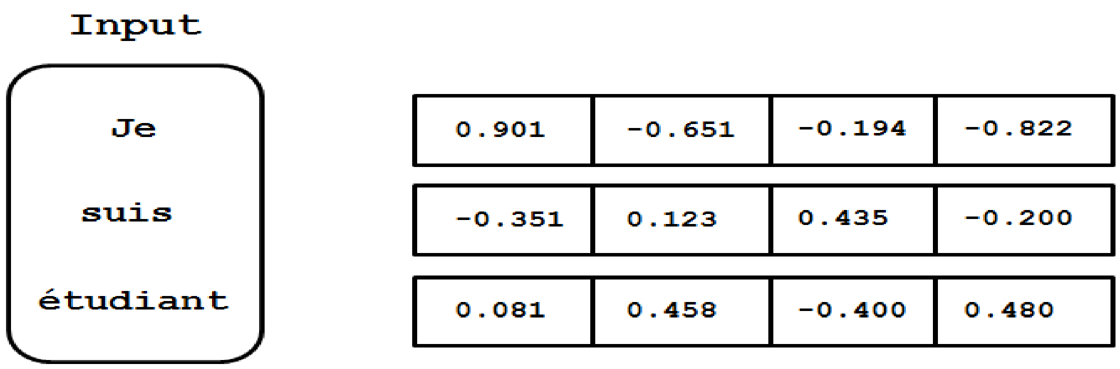


Figure 4.11 – *Example of vector embedding representation.*

- The Source sentence
  - $S1 = [70, 189, 68, 0, 0, 0, 0, 0, 99]$,
  - $S2 = [70, 15, 115, 90, 62, 78, 90, 180, 99]$.
- The Target sentence
  - $S1 = [1, 45, 115, 80, 162, 0, 0, 99]$
  - $S2 = [1, 45, 158, 80, 50, 120, 80, 156, 99]$

After that, the mode seeks to retrieve the corresponding word embedding representations for both the source and the target. The embedded words are represented as an array, where the line number is the token of words and the number column is corresponding to the size of the embedding word or vocabulary $''V''$, which usually corresponds to the most frequent words for each language. The embedding weights, one set per language, are usually learned during the training, while it can use "*word2vec*" [Le and Mikolov, 2014], or "*Glove*" [Pennington et al., 2014] embedding techniques to produce word vectors.

Once the data preparation is accomplished, the word embeddings are fed into the main network which is composed of two RNNs, an encoder for the source language and a decoder for the target language.

Due to RNN design requires at each time step two inputs, so in the case of the encoder, each word in the source is taken as input, which is inputting separately, and the hidden states of the previous time-step. Wherefore at $timestep = 0$, a zero vector is used as an initial hidden state since there is a lack of prior information. The hidden state is updated relying upon the information that comes from the inputted words. This process is repeated until the end of the sentence marked by the $< EOS >$ symbol. The hidden state at the final step is usually referred to as the context vector. While at this last step, the meaning of the whole input sentence is summarized in the hidden vector.

When the end of the source sentence is reached, the $< GO >$ symbol is sent to the decoder to indicate the inception of the decoding process to predict the output sentence item by item. Moreover, the decoder receives the context vector as the initial hidden state. The decoder used the $< GO >$ as input at $timestep = 1$ to update the hidden states and predict the output. The decoder provides a probability distribution over all the words in the output vocabulary which is inferred from a softmax activation layer. In this fashion, the word with the highest probability should be selected as the first word in the predicted output sentence. Then, this output will be the input of the decoder in the next step. And so on, until we reach the end of the sentence i.e. the decoder outputs the $< EOS >$ token. Thus, the decoder produces the predicted translated sentence which is compared to the target sentence by computing the loss function. The Cross-entropy loss is communally used since the translation task is regarded as a classification problem.

4.5 Sequence-to-Sequence architecture for head pose estimation

As long as the result of SAX is a symbolic sequence and since the recurrent neural networks (RNNs) are designed specifically for sequential data processing, we propose to learn a model based on RNNs architecture in order to classify the head poses. We performed a sequence-to-sequence model which is similar to the language translation introduced by Sutskever et al. [Sutskever et al., 2014].

The overall outline for training is illustrated in figure 4.12. In our model, the encoder recurrent neural network encodes and learns the relationship between the symbolic sequence to present it into a vector and the decoder decodes it into a sequence of words in the desired language. We had taken the symbolic sequences of SAX as an input of the encoder, which are all at the same length. While the output sequences are words of different vocabulary and length. All the words of the output sequences are tokenized and used as an index to present their location in vocabulary. We append a word $< Pad >$ to the short sequence. Moreover, we add the word $< Go >$, which prompts the decoder to start the decoding process, and word $< Eos >$ (end-of-sentence), which signals the end of the process. Also, the token $< Unk >$ is inserted to symbolize the unknown words in the output

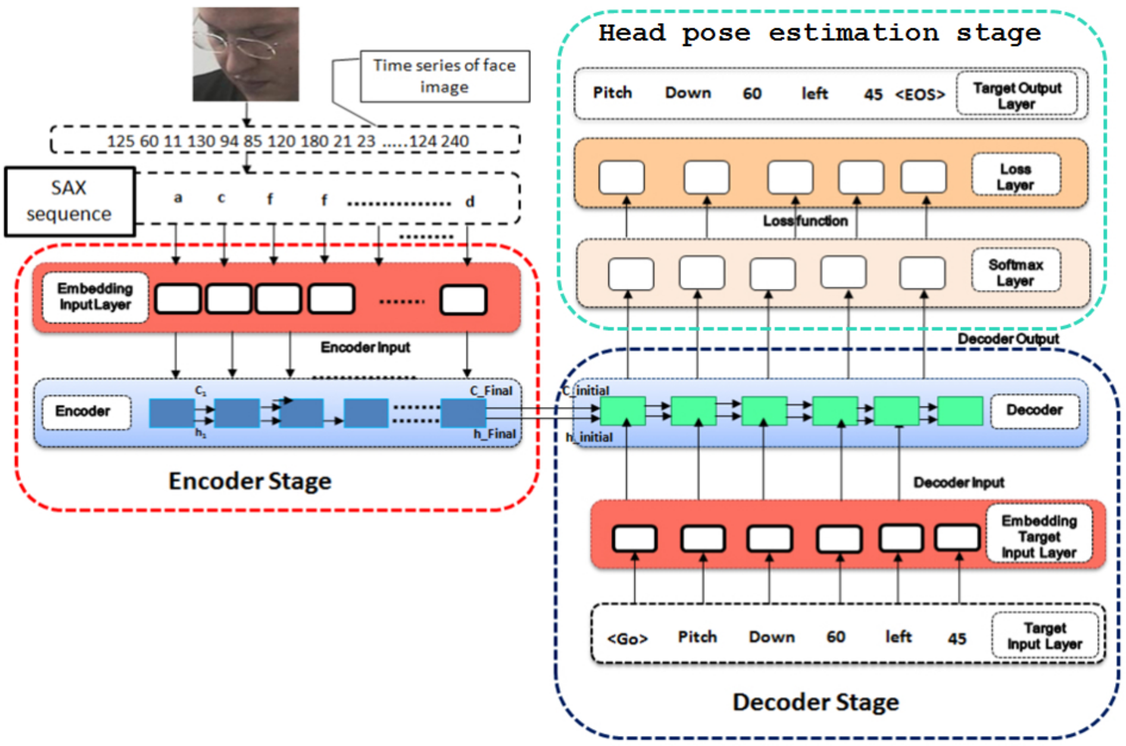


Figure 4.12 – *The proposed Encoder-Decoder training model for head pose estimation.*

vocabulary. We notice that the SAX symbolic sequences are more lengthy than the words of output sequences. Here the SAX symbol is treated as words of a first language and the labels of head poses are presented as translation into another language. To insert these words into both the encoder and the decoder, the embedding layer is used to transform each word into a vector representation. These embedding layers can learn the difference and relationship between the words. As well, the embedding weights are learned jointly with the recurrent neural network layers (Encode-decoder) during the training phase.

For the recurrent neural network in our model, we have used firstly a Bi-LSTM for both the encoder and the decoder, Then a Bi-GRU also in the encoder and decoder.

4.6 EXPERIMENTAL VALIDATION

In order to evaluate the robustness of our approach, we realized experiments on three databases, FERET [Phillips et al., 1998], CAS-PEAL[Gao et al., 2008], and Pointing'04 database [Gourier et al., 2004].

The experiments are executed in Python using the TensorFlow Library [Abadi et al., 2016], on a PC with Intel Core i7-6500U CPU @2.50GHz×4 with 8 GB memory.

The experiences are run with a Bi-LSTM Encoder-Decoder and the model with Bi-GRU with 32 hidden units. The training step is effectuated with the batch size that equals to 32, the Root Mean Square Propagation (*RMSProp*) is applied as an optimizer and the learning rate is selected as 0.005. As reported above, the input sequences of our model are the sequences generated using SAX. To create the SAX symbolic sequences, the input face images are resized to size 128×128 and scanned by the Peano-Hilbert curve to generate the time series which are transformed into SAX symbolic representation. We applied three resolutions -(*w*,*a*)- of SAX (16,8), (16,16), (16,32). So, the length of a SAX sequence is 1024 (128*128/16), which corresponds to the time steps of the RNN encoder. Once these sequences are generated, we fed them into our model. The output sequences have different lengths in each database, so we need some preprocessing before inputting it into the decoder. We have followed the plan for training our model as illustrated in figure 4.12.

4.6.1 Experiment on the FERET database

FERET database consists of images that are collected in a semi-controlled environment, of different age, race, and sex distribution. With 2645 images in poses "fa", "fb" for the frontal pose, and 726 images in "ql", 722 images in "qr" for the left and right quarter pose respectively ($\pm 22,5°$),1219 images in "hl", 1278 images in "hr" are the poses mid-left and mid-right respectively ($\pm 67,5°$), and 1271 images in "pl", 1319 images "pr" are profile poses on left and right respectively ($\pm 90°$). The total number of images in this database is 9180.

Since this database is large enough in terms of the number of training samples and the pose classes are not complex (7 yaw poses only), we have handled it as a baseline for fine-tuning the model parameters.

For the FERET, 7 output sequences are corresponding to the 7 yaw pose. Each pose is presented by a sequence of 4 words. So, the sequences have a maximum length of size 4. Table 4.3 shows the output sequences used to represent the poses on this database. All poses sequences have a length of 4, only the sequence which is represents the frontal pose has a length of size 3. Thus, the input sequence length is 1024, and the output sequence length is 4.

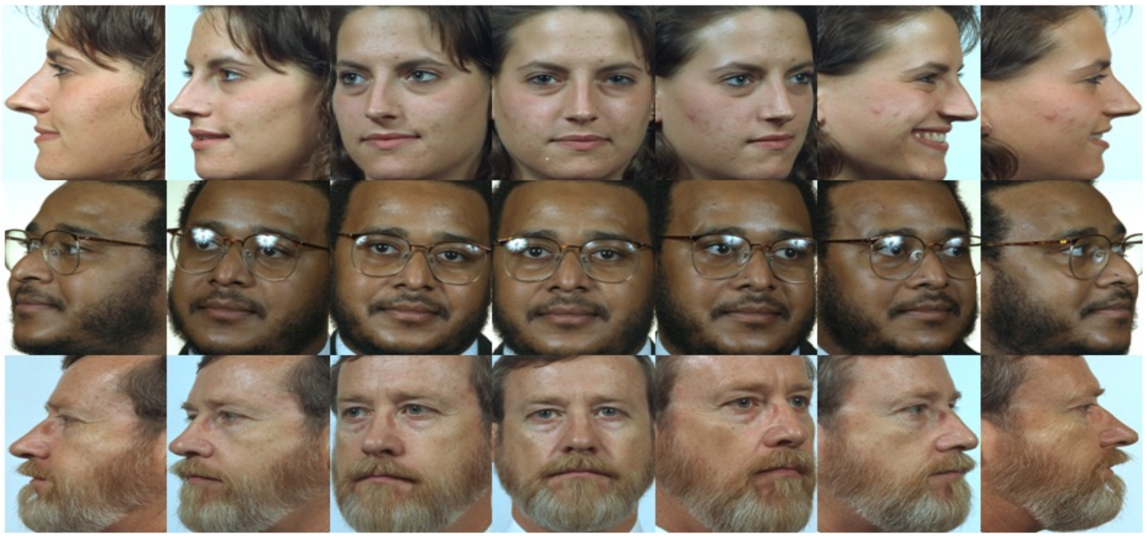


Figure 4.13 – *Examples of a face from the FERET database [Phillips et al., 1998].*

Table 4.3 – *The output sequences of FERET database, the angle is equal to* $22,5,^\circ$, 67.5° *or* 90°

1er word	2sd word	3th word	4th word
Yaw	pose	left	Angle
Yaw	pose	frontal	
Yaw	pose	right	Angle

The results of our experiences on FERET are shown in Table 4.4.

Table 4.4 – *Classification Accuracy(%) with Bidirectional Encode-Decoder in different SAX Resolution on the FERET database.*

Resolution		(16,8)		(16,16)		(16,32)	
RNN		Bi-LSTM	Bi-GRU	Bi-LSTM	Bi-GRU	Bi-LSTM	Bi-GRU
Epochs	150	94.54	95.914	94.585	96.193	94.343	96.210
	200	93.526	96.022	95.241	96.264	95.366	96.426
	300	95.519	96.381	95.519	96.390	95.996	96.667

We noticed from Table 4.4, and figure 4.14 that the accuracy values are improved greatly when the epochs number is increased, this observation was checked for different SAX resolutions.

Furthermore, the Encoder-Decoder with Bi-GRU performs well compared to the Encoder-Decoder that used the Bi-LSTM. Which is seems more clear in figure 4.15, where the accuracy achieved using the Bi-GRU is better for all resolution of SAX.

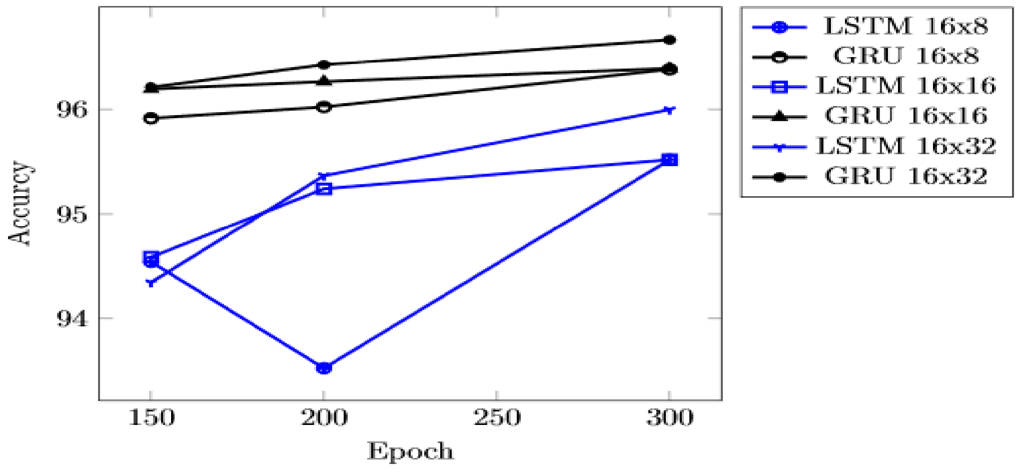


Figure 4.14 – *Variation of the accuracy values with epochs number for Bi-LSTM and Bi-GRU (FERET)*

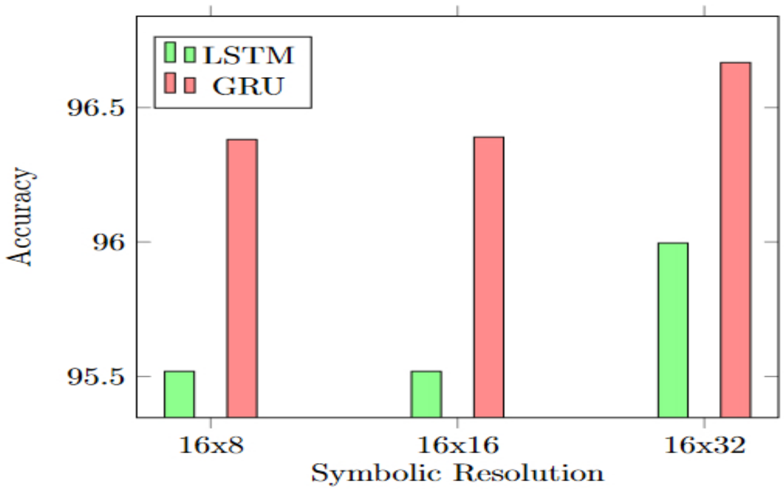


Figure 4.15 – *Average accuracy values of each resolution of SAX with Bi-LSTM and Bi-GRU (FERET)*

4.6.2 Experiment on the CAS-PEAL database

CAS-PEA contains images of 1039 individuals, with 21 poses for each person, 7 yaw poses([-45°, 45°] with intervals of 15° and 3 pitch poses (-30°, 0°, and 30°). Notice that 101 individuals have poses varying in the range [-67°, 67°] with intervals of 22°.

As we have obtained good results on FERET through the best combination of parameters that have been mentioned above, we evaluated our model on the CAS-PEAL database using these same hyper-parameters.

Since it is recommended to use large training samples in deep learning, besides, a pose estimation model requires specific labels, we have employed all databases with all poses. Thus, we have organized the poses into 21 classes, plus

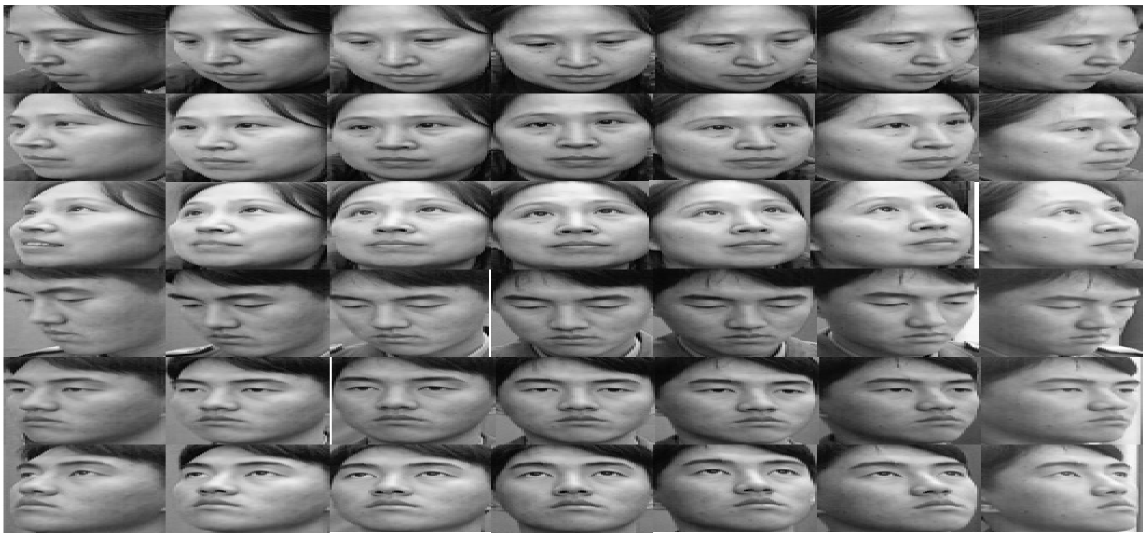


Figure 4.16 – *Examples of a face from the CAS-PEAL database [Gao et al., 2008].*

12 classes which indicated poses with $\pm 67°$, $\pm 22°$ (8 in pitch and 4 in yaw), so the total is 33 classes. Therefore, the labels of poses are presented with 33 output sequences, which have been illustrated in Table 4.5. The pitch poses are presented with 4 words. The yaw poses are characterized by 3 words, except the frontal pose, which has been denoted with 2 words only. The last word (4th word) in the output sequence presents the angle that varies according to the angles of the poses in this database. Thus, the short sequences have been appended with $< Pad >$ token.

Table 4.5 – *The output sequences of CAS-PEAL database.*

1er word	2^{sd} word	3th word	4th word
Pitch	up	left	Angle
Pitch	up	right	Angle
Pitch	down	left	Angle
Pitch	down	right	Angle
Yaw	left	Angle	
Yaw	right	Angle	
Yaw	frontal		

The results of our experiences on CAS-PEAL database with different *SAX* resolution are reported in Table 4.6. As we have shown in figure 4.17 the variation of accuracy in terms of epoch number, and the influence of the SAX resolution on the results in figure 4.18.

Table 4.6 – *Classification Accuracy (%) with Bidirectional Encoder-Decoder in different SAX Resolution on CAS-PEAL.*

Resolution		(16,8)		(16,16)		(16,32)	
RNN		Bi-LSTM	Bi-GRU	Bi-LSTM	Bi-GRU	Bi-LSTM	Bi-GRU
Epochs	150	92.346	93.931	91.613	92.481	89.726	89.915
	200	93.372	93.214	93.36	93.927	88.302	90.926
	300	94.715	93.939	94.333	**95.328**	90.529	91.304

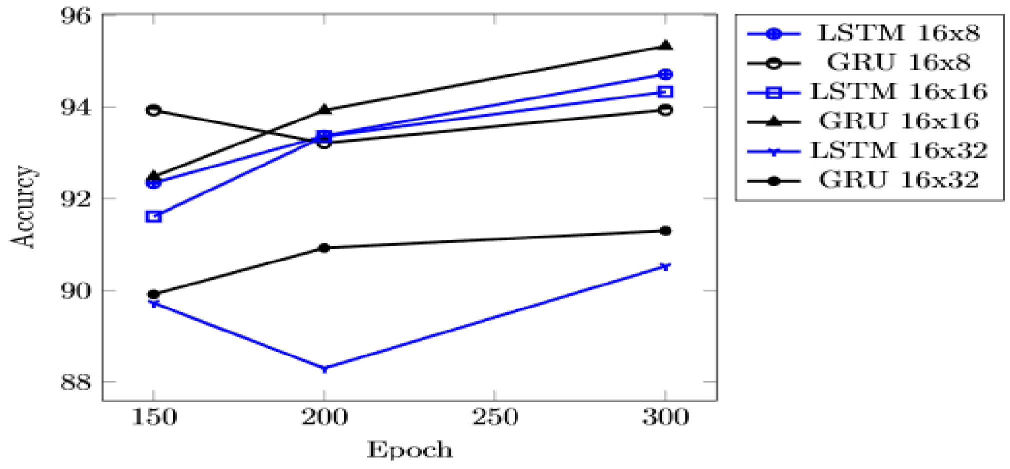


Figure 4.17 – *Variation of the accuracy values with epochs number for Bi-LSTM and Bi-GRU (CAS-PEAL)*

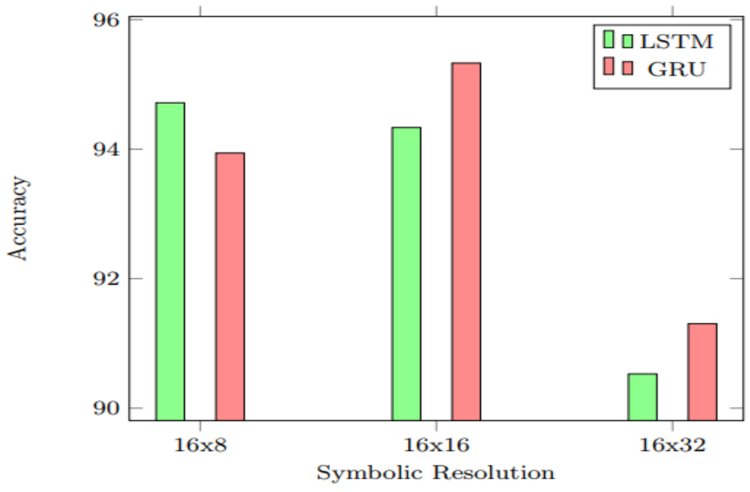


Figure 4.18 – *Average accuracy values of each resolution of SAX with Bi-LSTM and Bi-GRU (CAS-PEAL).*

Similarly to the FERET database, the classification accuracy in this database is varied proportionally with the number of epochs. Besides, we can note that the SAX-RED with the Bi-GRU is powerful than the Bi-LSTM for resolution

(16,16), and (16,32), and the higher accuracy is reached using Bi-GRU with SAX resolution (16,16).

In Table 4.7, we compared our results on CAS-PEAL with recent methods that have been evaluated on this database: KCovGa [Ma et al., 2014], kVoD+NC [Ma et al., 2015], VRF+LDA Huang et al. [2010], DCNN [Cai et al., 2015a], and SLRCNN (Soft Label Regularized CNN)[Xu et al., 2019].

Table 4.7 – *Head pose estimation comparison of various methods on the CAS-PEAL database.*

Method	Classes	Accuracy (%)	Number of training samples	Number of test samples
KCovGa[Ma et al., 2014]	7	94.2	934	466
kVoD+NC[Ma et al., 2015]	7	94.2	934	466
RF+LDA[Huang et al., 2010]	7	97.23	934	466
DCNN[Cai et al., 2015a]	9	98.29	196 950	876
SLRCNN [Xu et al., 2019]	21	99.19	2800	1400
Our approach	33	95.328	17247	4312

It is worth pointing out that Ma et al.[Ma et al., 2014; 2015; 2008] and Huang et al.[Huang et al., 2010] have used just 200 subjects of the CAS-PEAL database which were classified into seven yaw poses only. Cai et al. [Cai et al., 2015a] used only pitch poses which are identified with $0°$ and yaw poses labelled with $\pm 45°$, $\pm 30°$,$\pm 15°$,$0°$, 8754 images in all. They have estimated 9 poses for each subject via a deep convolutional neural network.

Ma et al. in [Ma et al., 2014; 2015], proposed two descriptors named Covariance Descriptor of Gabor filters (CovGa) and fisher Vector of local Descriptors (VoD), which are improved by combining them with a metric learning method named Keep It Simple and Straightforward Metric Learning (KISSME). So they have produced other descriptors K-CovGa and K-VoD. These descriptors extracted the head pose features. The Nearest Centroid classifier (NC) was used to classify the yaw pose. The accuracy of these methods is about 94.2%. Also, Huang et al. [Huang et al., 2010] proposed a classifier with Random Forests (RF) combined with Linear Discriminative Analysis (LDA) for estimating the yaw pose only, which achieved an accuracy of 97.32%. A Deep Convolutional Neural Network (DCNN) at 8 layers proposed by Cai et al. [Cai et al., 2015a] to classify the head pose with a very large number of training samples which produced from shift and the scale original images, therefore, there was 196 950 images for training (which is more 11 times larger than our training samples), the accuracy provided is 98.29% for 9 poses. Xu et al.[Xu et al., 2019] achieved excellent accuracy of the head rotation of 21 poses.

The best accuracy that we have achieved is 95.33%, which can be considered as a powerful one because we estimated 33 poses of the head with fewer training samples compared with the number of training samples used in [Cai et al., 2015a].

4.6.3 Experiment on the Prima head-pose database (Pointing'04)

Pointing'04: It is comparatively one of the old head pose databases. However, it is still used for research related to this field due to its challenging nature and its great diversity with consecutive poses. It consists of 2790 images for 15 subjects into 2 sets. Each set of each subject has 93 different poses, 13 yaw poses, 40 images in pitch up, and 40 images in pitch down. The range of pitch and yaw angles is in $[-90°, +90°]$, whereas the difference between two consecutive poses is $30°$ for the pitch view, and $15°$ between two adjacent yaw poses as shown in figure 4.19.

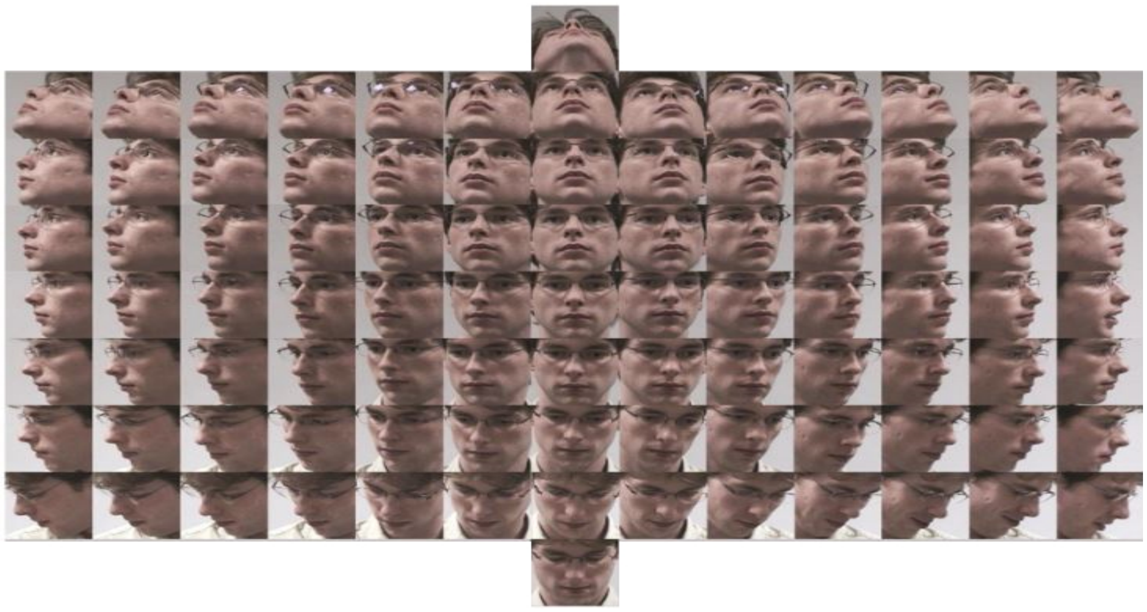


Figure 4.19 – *Pointing'04 database images of a single subject in all 93 poses [Gourier et al., 2004].*

We have evaluated our method in the same manner on the Pointing'04 database. In the same manner, input sequences are the sequences generated by SAX from the face images time series. At this time, the poses labels have been described with a sequence of 5 words as shown in Table 4.8. Five words are used to discriminate each pitch pose. All yaw poses sequences have a size length of 4, only the sequence which presented the frontal pose has the length of size 3. The pitch angle indicated the variation of the pitch pose related to pitch angles in this database. So, the input sequence length is 1024, and the output sequence length is 5 (the short sequences have appended with $< Pad >$ token).

Table 4.8 – *The output sequences of Pointing'04 database.*

1er word	2sd word	3th word	4th word	5th word
Pitch	up	pitch angle	left	Angle
Pitch	up	pitch angle	right	Angle
Pitch	down	pitch angle	left	Angle
Pitch	down	pitch angle	right	Angle
Yaw	pose	left	Angle	
Yaw	pose	right	Angle	
Yaw	frontal			

As the number of samples in this database is not large enough, we have applied our method on this database following the similar measurement protocol that was used in [Gao et al., 2017]. We have tested our approach with 5-fold cross-validation, and 300 epochs. We have calculated the same measurements values, the accuracy (ACC), also the Mean Absolute Error (MAE) for pitch, yaw, and pitch+yaw (estimate pitch and yaw angles together). The Mean Absolute Error (MAE) is calculated according to this formula:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\hat{Y}_n - Y_n| \tag{4.33}$$

Where $\hat{Y}_n$ and Y_n are the estimated and real label of N tested images respectively.

Tables 4.9 and 4.10 show the results on Pointing'04. We observe that the encoder-decoder with Bi-GRU has much higher accuracy than Bi-LSTM especially for pitch pose. This result can be explained that 86% of the images in this database (2400 images) are in pitch pose. The figure 4.20 demonstrated the effect of the SAX resolution on the classification rate.

Table 4.9 – *Classification Accuracy(%) with Bi-LSTM Encode-Decoder in different SAX Resolution on Pointing'04.*

Resolution	(16,8)	(16,16)	(16,32)	
RNN	Bi-LSTM	Bi-LSTM	Bi-LSTM	
Pitch+Yaw	80.88 ± 0.04	81.63 ± 0.02	81.92 ± 0.04	A
Pitch	86.11 ± 0.04	88.97 ± 0.02	85.08 ± 0.04	C
Yaw	50.37 ± 0.05	37.04 ± 0.03	63.16 ± 0.10	C
Pitch+Yaw	1.28 ± 0.21	1.09 ± 0.09	1.06 ± 0.23	M
Pitch	0.86 ± 0.34	0.50 ± 0.17	0.80 ± 0.29	A
Yaw	3.71 ± 0.67	4.74 ± 0.5	2.56 ± 1.29	E

Table 4.10 – *Classification Accuracy(%) and MEA with Bi-GRU Encode-Decoder in different SAX Resolution on Pointing'04.*

Resolution	(16,8)	(16,16)	(16,32)	
RNN	Bi-GRU	Bi-GRU	Bi-GRU	
Pitch+Yaw	84.76 ± 0.03	82.91 ± 0.04	84.37 ± 0.04	A
Pitch	86.88 ± 0.02	84.87 ± 0.04	86.04 ± 0.03	C
Yaw	71.65 ± 0.08	71.96 ± 0.06	73.94 ± 0.09	C
Pitch+Yaw	0.94 ± 0.13	1.12 ± 0.23	0.96 ± 0.24	M
Pitch	0.77 ± 0.21	0.98 ± 0.38	0.82 ± 0.32	A
Yaw	1.96 ± 1.22	1.91 ± 1.17	1.78 ± 1.36	E

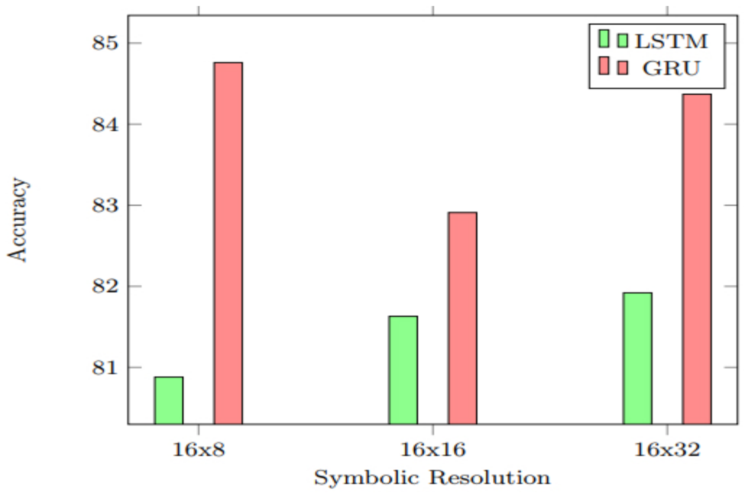


Figure 4.20 – *Average accuracy values of each resolution of SAX with Bi-LSTM and Bi-GRU for Pitch+Yaw poses (Pointing'04).*

A comparative study between our approach and the state-of-the-art techniques is given in Table 4.11 and 4.12, using Pointing'04. We have compared our results with Multivariate Label Distribution (MLD) [Geng and Xia, 2014], random forest (RF+LDA) [Huang et al., 2010], Dirichlet-tree distribution enhanced random forest (D-RF) [Liu et al., 2016], and multi-level structured hybrid forest (MSHF) [Liu et al., 2017], besides the classifiers have been proposed by Gao et al [Gao et al., 2017]. Thus, the stack autoencoder model with Extreme Gradient Boostin (SAE-XGB) [Vo et al., 2019], and SLRCNN [Xu et al., 2019].

Table 4.11 – *Head pose estimation comparison of various methods according to Accuracy on Pointing'04 database.*

Method	Accuracy (%)		
	Pitch	Yaw	Pitch+Yaw
MLD [Geng and Xia, 2014]	86.24 ± 0.97	73.30 ± 1.36	64.27 ± 1.82
RF+LDA [Huang et al., 2010, Liu et al., 2017]	68.73	78.40	62.23
D-RF [Liu et al., 2016]	86.94	83.52	71.83
MSHF [Liu et al., 2017]	90.7	92.3	84.0
DLDL [Gao et al., 2017]	91.65 ± 1.13	79.57 ± 0.57	73.15 ± 0.72
C-ConvNet [Gao et al., 2017]	73.15 ± 2.74	62.90 ± 1.81	42.97 ± 1.67
ConvNet+LS (KL) [Gao et al., 2017]	72.62 ± 1.01	62.90 ± 2.76	41.83 ± 2.20
ConvNet+LD [Gao et al., 2017]	90.00 ± 0.77	76.27 ± 0.82	69.00 ± 0.89
SLRCNN [Xu et al., 2019]	$\mathbf{96.09 \pm 1.21}$	$\mathbf{88.71 \pm 2.27}$	$\mathbf{85.77 \pm 3.06}$
SAE-XGB [Vo et al., 2019]	68.99 ± 1.46	57.24 ± 3.56	-
Our approach	$\mathbf{86.04 \pm 0.03}$	$\mathbf{73.94 \pm 0.09}$	$\mathbf{84.37 \pm 0.04}$

Table 4.12 – *Head pose estimation comparison of various methods according to MAE on Pointing'04 database.*

Method	MAE		
	Pitch	Yaw	Pitch+Yaw
MLD [Geng and Xia, 2014]	2.69 ± 0.15	4.24 ± 0.17	6.45 ± 0.29
DLDL [Gao et al., 2017]	1.69 ± 0.32	3.167 ± 0.07	4.64 ± 0.24
C-ConvNet [Gao et al., 2017]	5.28 ± 0.65	6.02 ± 0.44	10.56 ± 0.74
ConvNet+LS (KL)[Gao et al., 2017]	5.23 ± 0.39	5.87 ± 0.53	10.42 ± 0.66
ConvNet+LD [Gao et al., 2017]	1.94 ± 0.20	3.68 ± 0.16	5.34 ± 0.17
SLRCNN [Xu et al., 2019]	0.76 ± 0.22	1.74 ± 0.37	1.25 ± 0.28
Our approach	$\mathbf{0.82 \pm 0.32}$	$\mathbf{1.78 \pm 1.36}$	$\mathbf{0.96 \pm 0.24}$

The accuracy achieved with MLD [Geng and Xia, 2014]is about 64.27%. In [Geng and Xia, 2014], the authors suggested using a multivariate label instead of a single label to describe the head poses. For each face image, MLD is generated via a discretized bivariate Gaussian distribution to estimate the head poses in yaw and pitch rotation. The classification accuracy obtained by the random forest (RF+LDA) [Huang et al., 2010] is 62.23%. Liu et al. [Liu et al., 2016; 2017] proposed enhanced versions of random forest using Dirichlet-tree distribution enhanced random forest (D-RF) and multi-level structured hybrid forest (MSHF). The accuracy increased from 71.83% by D-RF to 84% via MSHF. Gao et al.[Gao et al., 2017] combined Label Distribution Learning (LDL)[Geng, 2016], with deep Convolutional Neural Networks to learn a model using the label ambiguity in both feature learning and classifier learning, which allows the model to overcome the problem of over-fitting. Gao et al. [Gao et al., 2017] have used different Convolutional Neural Networks (ConvNets) models based on ZF-Net model [Zeiler and Fergus, 2014]: Classification ConvNets (C-ConvNet) with *softmax* as a loss function. They have combined a label smoothing (LS) [Szegedy et al., 2016] with ConvNet and Kullback-Leibler KL divergence(relative entropy) applied as a loss function (ConvNet+LS (KL)), and ConvNet+LD classifier joined label distribution (LD)[Geng, 2016] with ConvNet. The accuracies achieved by these models are low. The authors interpreted that the inability to learn these models because of the few numbers of training samples. However, they have proposed a Deep Label Distribution Learning DLDL, a framework to surmount this limitation which enhanced the accuracy to 73.15%. Compared with that, our approach allows us to reach a powerful accuracy of 84.37%. Furthermore, our model has the ability to learn with a small number of training samples better than the DLDL model. We can see that the accuracy achieved by Xu et al. [Xu et al., 2019] is slightly higher of 1.4% compared to our result but with a standard deviation of ± 3.06, while with our model was only ± 0.04 and we can notice that MAE in our approach is lower for Pitch+Yaw poses.

Finally, we can summarize the results of our approach, based on what is displayed in the figure 4.21. The figure 4.21 illustrates the accuracy variation by relevance to the SAX resolution using the Bi-GRU on the three databases.

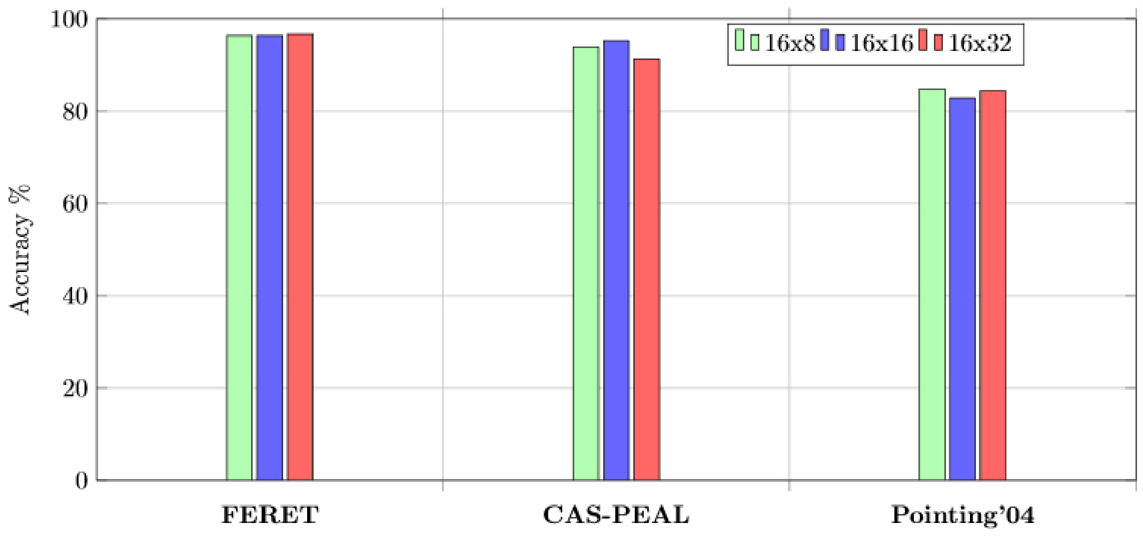


Figure 4.21 – *Average accuracy values of each resolution of SAX with Bi-GRU for all database.*

- **FERET database**: the classification accuracy achieved for the various SAX resolutions is roughly the same. This can be explained that the pose angles variation on this database varied only in yaw angles. The FERRET is less complex than the two other databases.
- **CAS-PEAL database**: for this database the SAX resolution (16,16) allow us to reach the best classification accuracy.

- **Pointing 04′ database**: we can see that the accuracy (Pitch+Yaw) that have reached with both SAX resolutions (16,8) and (16,32) is almost equal, but with resolution (16,32) the result for Yaw pose is better as shown in figure 4.22. A noteworthy, that database has some heavily challenges. The number of images is not enough to train a deep learning model, as well the images in yaw pose present only 14% of the total number of images (390 images), while the number of pitch pose is 86% samples. Moreover, it has a large angle variation, since there are 93 poses, 40 in pitch up, 40 in pitch down, and 13 images in yaw pose.

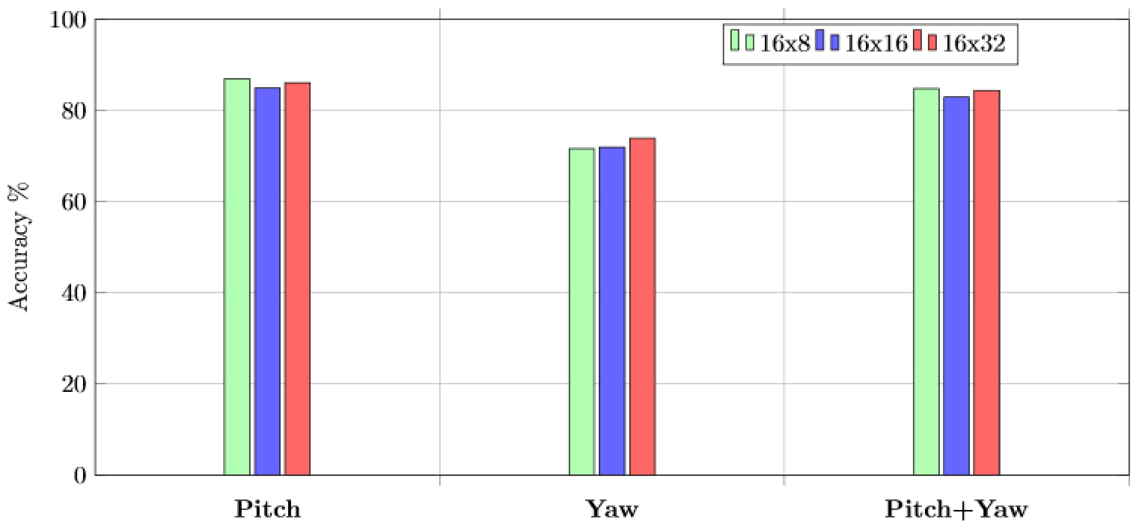


Figure 4.22 – *Average accuracy values of each resolution of SAX with Bi-GRU according to angle rotation on Pointing 04′ database.*

As already outlined, we can infer that the effect of SAX resolution on the performance of head pose estimation relies on the type of data or the samples images. For databases that have one axis of angle variation, it is enough to choose smaller values for the alphabet size as the case of the FERET database. While in the case of databases that have samples with a large range of angles rotation, it would rather increase the alphabet size to proved more details, which permit to discrimination of more difference between the poses, as the images of CAS-PEAL and Pointing 04′ databases.

4.7 Conclusion

In this chapter, we showed the capabilities of encoder-decoder models based on bidirectional RNN (Bi-LSTM, Bi-GRU) layers for learning the sequence-to-sequence head pose estimation model.

In the previous chapter, we have demonstrated the efficiency of exploiting time series representation to extracted features and build a model to classify the head pose. Whereas, we have used classic classifiers to determine the head orientation in three-classes only. While in this chapter, we have suggested an approach to benefit the dimensionality reduction achieved through SAX representation, and the privilege of the deep learning model.

The sequence-to-sequence model is heavily designed to handle natural language processing issues. Due to that, the SAX representation is a string sequence, and the head pose labels can be described with words sequence, we have spurred to carry out the SAX-RED model to estimate the head rotation. Similar to neural machine translation, our model used the encoder to map the SAX sequence as the sentence of the first language, and the decoder generates the head pose label as the sentence of the translated language.

In the experimental phase, we have investigated the influence of SAX resolution, as well as the RNN type. We have implemented two architectures of encoder-decoder, one with bidirectional LSTM, the other with bidirectional GRU. We have evaluated these models with different SAX resolutions in a different database. The experimental results have shown that our model enhances significantly the detection rates and reaches out high accuracy in yaw and pitch rotation with a large range of poses, and even with small numbers of training samples.

Conclusion and perspectives

This dissertation has mainly investigated approaches for improving the efficiency and accuracy of the head pose estimation task. The head pose estimation has been widely investigated, and several approaches are suggested to overcome this problem. These approaches can be either local (features-based), relied upon the position of the principal face features which are simple, run fast, and yield great accuracy. However, they require high precision of facial key-point detection which are very sensitive to the acquisition of image conditions, and more heavily with occluded face regions. The global methods (appearance-approaches), which used features extracted from the whole face to learn relationships between the training samples and their labels with regression or classification algorithms. These techniques have received considerable attention in this domain, especially dimensionality reduction methods. Dimensionality reduction methods map a high-dimensional feature space with few dimensions that describe the pose changes. These methods are highly efficient in terms of accuracy and implementation speed, but it's still a challenge to attain low dimensions that represent the pose variations without other changes in the image.

Out of the lack of these approaches in this thesis, we presented a new technique for facial pose classification characterized by its simplicity, robustness, and computationally economic. The method uses dimensionality reduction through time-series representation of the learning images.

This thesis provides two major contributions. The first major contribution is dealing with the head pose estimation as a time-series classification task. We have successfully developed a new method that ensures dimensionality reduction, as well as, extracts facial features that discriminate the pose variations. We have explored the time series representation to extract these features, and carry out the head pose classifiers model. The face images are mapped to vectors representation as time series using the space-filling curves. Then, these vectors are encoded as string sequences via SAX. After that, we calculated the pairwise similarity matrices between the SAX sequences of the images from different databases using a different distance metric. The similarity measure was deemed as the decision criterion for classifying the poses. To classify the sequences (therefore the images), we used KNN and SVM classifications to affect each one to the adequate class (frontal or profile view classes). Thus the obtained results showed the robustness and efficiency of the proposed approach even in degraded conditions.

The second major contribution is to cast the head orientation problem into a natural language processing problem. We present a principled method for learning Seq2Seq models to predict the head poses. Seq2Seq or encoder-decoder neural networks are usually used in natural language processing to translate words in different languages depending on the context of the words in the sentence. The SAX sequences allow us to use Seq2Seq for performing a similar model to neural machine translation. While in our model, the encoder reads the SAX symbolic sequence and encodes it into a fixed-length vector to learn the relationship between the symbols. Then, the decoder is used for decoding the preprocessed vectors to predict the sequences representing the labels of the poses. As a result, our model is more powerful and leads to enhance the detection rates significantly and reaches out high accuracy in yaw and pitch rotation with a large range of poses, and even with small numbers of training samples.

In terms of directions for future work, we would be interested in exploring other symbolic transformation techniques such as vector quantization (VQ), and subsequently other numerical and/or semantic distance measures. Further, we would like to evaluate our method in real time using video streams. As well as, we envisage evaluating the method with the generative adversarial networks for getting enriched pictures for dealing with occlusion problems.

My scientific contributions

internationals Journals

1. H.Mekami, A.Bounoua, S.Benabderrahmane, Leveraging deep learning with symbolic sequences for robust head poses estimation. Pattern Anal Applic 23, 1391–1406 (2020). https://doi.org/10.1007/s10044-019-00857-5.
2. H.Mekami, S.Benabderrahmane, A.Bounoua, A. Ahmed Taleb. Local Patterns and Big Time Series Data for Facial Poses Classification. JCP, 2018, vol. 13, no 1, p. 18-34.

internationals Conferences

1. H.Mekami, S.Benabderrahmane. SAX2FACE: Estimating Facial Poses with Peano-Hilbert Curves and SAX Symbolic Time Series. Procedia Computer Science 109 (2017): 217-224.
2. H. Mekami, A Bounoua, S.Benabderrahmane, A.Ahmed Taleb. Facial Pose Classification using Hilbert Space Filling Curve and Multidimensional Scaling. International conference pattern analysis and intelligent systems PAIS 2015, Tebessa (Algerie) 26-27 oct. 2015
3. H.Mekami, S.Benabderrahmane. Towards a new approach for real time face detection and normalization. 2010 International Conference on Machine and Web Intelligence. IEEE, 2010.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Andrea F Abate, Paola Barra, Chiara Pero, and Maurizio Tucci. Head pose estimation by regression algorithm. *Pattern Recognition Letters*, 2020.

Stephen Ackland, Francisco Chiclana, Howell Istance, and Simon Coupland. Real-time 3d head pose tracking through 2.5 d constrained local models with local neural fields. *International Journal of Computer Vision*, 127(6-7):579–598, 2019.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer, 1993.

Murad Al Haj, Jordi Gonzalez, and Larry S Davis. On partial least squares in head pose estimation: How to simultaneously deal with misalignment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2602–2609. IEEE, 2012.

Nawal Alioua, Aouatif Amine, Alexandrina Rogozan, Abdelaziz Bensrhair, and Mohammed Rziza. Driver head pose estimation using efficient descriptor fusion. *EURASIP Journal on Image and Video Processing*, 2016(1):2, 2016.

Mikel Ariz, José J Bengoechea, Arantxa Villanueva, and Rafael Cabeza. A novel 2d/3d database with automatic face annotation for head tracking and pose estimation. *Computer Vision and Image Understanding*, 148:201–210, 2016.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223, 2017.

Mariette Awad and Rahul Khanna. Support vector machines for classification. In *Efficient Learning Machines*, pages 39–66. Springer, 2015.

Kevin Bailly and Maurice Milgram. Boosting feature selection for neural network based regression. *Neural Networks*, 22(5):748–756, 2009.

Vineeth Nallure Balasubramanian, Jieping Ye, and Sethuraman Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007.

Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 3d constrained local model for rigid and non-rigid facial tracking. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2610–2617. IEEE, 2012.

Paola Barra, Silvio Barra, Carmen Bisogni, Maria De Marsico, and Michele Nappi. Web-shaped model for head pose estimation: An approach for best exemplar selection. *IEEE Transactions on Image Processing*, 29:5457–5468, 2020.

Jilliam Maria Diaz Barros, Bruno Mirbach, Frederic Garcia, Kiran Varanasi, and Didier Stricker. Fusion of keypoint tracking and facial landmark detection for real-time head pose estimation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2028–2037. IEEE, 2018.

Konstantin Evgen'evich Bauman. The dilation factor of the peano-hilbert curve. *Mathematical Notes*, 80(5-6):609–620, 2006.

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

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Chiraz BenAbdelkader. Robust head pose estimation using supervised manifold learning. In *European conference on computer vision*, pages 518–531. Springer, 2010.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *NIPS*, pages 153–160. MIT Press, 2006.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

D Beymer. Face recognition under varying pose,". In *Proceedings of 23rd Image Understanding Workshop*, volume 2, pages 837–842, 1994.

Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.

Woodrow Wilson Bledsoe et al. Facial recognition project report. Technical report, Technical report PRI 10, Panoramic Research, Inc, 1964.

Guido Borghi, Riccardo Gasparini, Roberto Vezzani, and Rita Cucchiara. Embedded recurrent network for head pose estimation in car. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1503–1508. IEEE, 2017.

Ying Cai, Meng-long Yang, and Jun Li. Multiclass classification based on a deep convolutional network for head pose estimation. *Frontiers of Information Technology & Electronic Engineering*, 16(11):930–939, 2015a.

Ying Cai, Menglong Yang, and Ziqiang Li. Robust head pose estimation using a 3d morphable model. *Mathematical Problems in Engineering*, 2015, 2015b.

Yu Cai, Yanjin Huang, and Shugong Zhang. A method for nose tip location and head pose estimation in 3d face data. In *International Conference on Automatic Control and Artificial Intelligence (ACAI 2012)*. IET, 2012.

Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188–228, 2002.

Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 126–133. IEEE, 1999.

Ju-Chin Chen and Jenn-Jier James Lien. A view-based statistical system for multi-view face detection and pose estimation. *Image and Vision Computing*, 27(9): 1252–1271, 2009.

Ke Chen, Kui Jia, Heikki Huttunen, Jiri Matas, and Joni-Kristian Kämäräinen. Cumulative attribute space regression for head pose estimation and color constancy. *Pattern Recognition*, 87:29–37, 2019.

Longbin Chen, Lei Zhang, Yuxiao Hu, Mingjing Li, and Hongjiang Zhang. Head pose estimation using fisher manifold learning. In *2003 IEEE International SOI Conference. Proceedings (Cat. No. 03CH37443)*, pages 203–207. IEEE, 2003.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

TF Cootes, MG Roberts, KO Babalola, and CJ Taylor. Active shape and appearance models. In *Handbook of Biomedical Imaging*, pages 105–122. Springer, 2015.

Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 23(6):681–685, 2001.

Ciprian Adrian Corneanu, Marc Oliu Simón, Jeffrey F Cohn, and Sergio Escalera Guerrero. Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1548–1568, 2016.

David Cristinacce and Tim Cootes. Automatic feature localisation with constrained local models. *Pattern Recognition*, 41(10):3054–3067, 2008.

Donggen Dai, Wangkit Wong, and Zhuojun Chen. Rankpose: Learning generalised feature with rank supervision for head pose estimation. *arXiv preprint arXiv:2005.10984*, 2020.

Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding similar time series. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 88–100. Springer, 1997.

Dmytro Derkach, Adria Ruiz, and Federico Sukno. 3d head pose estimation using tensor decomposition and non-linear manifold modeling. In *2018 International Conference on 3D Vision (3DV)*, pages 505–513. IEEE, 2018.

Dmytro Derkach, Adrià Ruiz, and Federico M Sukno. Tensor decomposition and non-linear manifold modeling for 3d head pose estimation. *International Journal of Computer Vision*, 127(10):1565–1585, 2019.

Katerine Diaz-Chito, Aura Hernández-Sabaté, and Antonio M López. A reduced feature set for driver head pose estimation. *Applied Soft Computing*, 45:98–107, 2016.

Katerine Diaz-Chito, Jesús Martínez Del Rincón, Aura Hernández-Sabaté, and Debora Gil. Continuous head pose estimation using manifold subspace embedding and multivariate regression. *IEEE Access*, 6:18325–18334, 2018.

Vincent Drouard, Sileye Ba, Georgios Evangelidis, Antoine Deleforge, and Radu Horaud. Head pose estimation via probabilistic high-dimensional regression. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4624–4628. IEEE, 2015.

Vincent Drouard, Sileye Ba, and Radu Horaud. Switching linear inverse-regression model for tracking head pose. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1232–1240. IEEE, 2017a.

Vincent Drouard, Radu Horaud, Antoine Deleforge, Sileye Ba, and Georgios Evangelidis. Robust head-pose estimation based on partially-latent mixture of linear regressions. *IEEE Transactions on Image Processing*, 26(3):1428–1440, 2017b.

Bernhard Egger, William AP Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhoefer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, et al. 3d morphable face models—past, present, and future. *ACM Transactions on Graphics (TOG)*, 39(5):1–38, 2020.

Egor Elagin, Johannes Steffens, and Hartmut Neven. Automatic pose estimation system for human faces based on bunch graph matching technology. In

Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, pages 136–141. IEEE, 1998.

Wolfgang Ertel. *Introduction to artificial intelligence*. Springer Publishing Company, Incorporated, 2018.

SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360 (6394):1204–1210, 2018.

Gabriele Fanelli, Juergen Gall, and Luc Van Gool. Real time head pose estimation with random regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 617–624. IEEE, 2011.

Michele Fenzi, Laura Leal-Taixé, Bodo Rosenhahn, and Jorn Ostermann. Class generative models based on feature regression for pose estimation of object categories. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 755–762, 2013.

Joachim Flocon-Cholet. *Classification audio sous contrainte de faible latence*. PhD thesis, Université Rennes 1, 2016.

Jacob Foytik and Vijayan K Asari. A two-layer framework for piecewise linear manifold-based head pose estimation. *International journal of computer vision*, 101(2):270–287, 2013.

J Friedman. Another approach to polychotomous classification. dept. *Statistics, Stanford Univ., Tech. Rep*, 1996.

Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

Yun Fu and Thomas S Huang. Graph embedded analysis for head pose estimation. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 6–pp. IEEE, 2006.

Kunihiko Fukushima. Training multi-layered neural network neocognitron. *Neural Networks*, 40:18–31, 2013.

Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.

Fei Gao, Shuai Li, and Shufang Lu. How frontal is a face? quantitative estimation of face pose based on cnn and geometric projection. *Neural Computing and Applications*, pages 1–17, 2020.

Wen Gao, Bo Cao, Shiguang Shan, Xilin Chen, Delong Zhou, Xiaohua Zhang, and Debin Zhao. The cas-peal large-scale chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(1):149–161, 2008.

Andrew Gee and Roberto Cipolla. Determining the gaze of faces in images. *Image and Vision Computing*, 12(10):639–647, 1994.

Xin Geng. Label distribution learning. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1734–1748, 2016.

Xin Geng and Yu Xia. Head pose estimation based on multivariate label distribution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1837–1842, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Georgios Goudelis, Anastasios Tefas, and Ioannis Pitas. Automated facial pose extraction from video sequences based on mutual information. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(3):418–424, 2008.

Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial features. In *ICPR International Workshop on Visual Observation of Deictic Gestures*. Citeseer, 2004.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

Aryaman Gupta, Kalpit Thakkar, Vineet Gandhi, and PJ Narayanan. Nose, eyes and ears: Head pose estimation by locating facial keypoints. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1977–1981. IEEE, 2019.

Sabri Gurbuz, Erhan Oztop, and Naomi Inoue. Model free head pose estimation using stereovision. *Pattern Recognition*, 45(1):33–42, 2012.

Yaron Gurovich, Yair Hanani, Omri Bar, Guy Nadav, Nicole Fleischer, Dekel Gelbman, Lina Basel-Salmon, Peter M Krawitz, Susanne B Kamphausen, Martin Zenker, et al. Identifying facial phenotypes of genetic disorders using deep learning. *Nature medicine*, 25(1):60, 2019.

ByungOk Han, Suwon Lee, and Hyun S Yang. Head pose estimation using image abstraction and local directional quaternary patterns for multiclass classification. *Pattern Recognition Letters*, 45:145–153, 2014.

Kota Hara and Rama Chellappa. Growing regression tree forests by classification for continuous object pose estimation. *International Journal of Computer Vision*, 122(2):292–312, 2017.

CV Hari and Praveen Sankaran. A clustered locally linear approach on face manifolds for pose estimation. *Pattern Analysis and Applications*, 20(4):1169–1178, 2017.

John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Hiyam Hatem, Zou Beiji, Raed Majeed, Jumana Waleed, and Mohammed Lutf. head pose estimation based on detecting facial features. *International journal of multimedia and ubiquitous engineering*, 10(3):311–322, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

Geoffrey Hinton. Lecture 6d: a separate, adaptive learning rate for each connection. slides of lecture neural networks for machine learning. Technical report, Technical report, Slides of Lecture Neural Networks for Machine Learning, 2012.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Heng-Wei Hsu, Tung-Yu Wu, Sheng Wan, Wing Hung Wong, and Chen-Yi Lee. Quatnet: Quaternion-based head pose estimation with multiregression loss. *IEEE Transactions on Multimedia*, 21(4):1035–1046, 2018.

Bin Huang, Renwen Chen, Wang Xu, and Qinbang Zhou. Improving head pose estimation using two-stage ensembles with top-k regression. *Image and Vision Computing*, 93:103827, 2020.

Chen Huang, Xiaoqing Ding, and Chi Fang. Head pose estimation based on random forests for multiclass classification. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 934–937. IEEE, 2010.

Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, and Liming Chen. Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(6):765–781, 2011a.

Dong Huang, Markus Storer, Fernando De la Torre, and Horst Bischof. Supervised local subspace learning for continuous head pose estimation. In *CVPR 2011*, pages 2921–2928. IEEE, 2011b.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Jeffrey Huang, Xuhui Shao, and Harry Wechsler. Face pose discrimination using support vector machines (svm). In *Proceedings. fourteenth international conference on pattern recognition (Cat. No. 98EX170)*, volume 1, pages 154–156. IEEE, 1998.

Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3(14):2, 2003.

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings 2001 IEEE international conference on data mining*, pages 273–280. IEEE, 2001.

Min-Joo Kang, Jung-Kyung Lee, and Je-Won Kang. Combining random forest with multi-block local binary pattern feature selection for multiclass head pose estimation. *PloS one*, 12(7), 2017.

Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

Josef Kittler, Patrik Huber, Zhen-Hua Feng, Guosheng Hu, and William Christmas. 3d morphable face models and their applications. In *International Conference on Articulated Motion and Deformable Objects*, pages 185–206. Springer, 2016.

Caroline Kleist. Time series data mining method: A review. *Humboldt-Universitat zu Berlin*, 2015.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

Martin Koestinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 2144–2151. IEEE, 2011.

Flip Korn, Hosagrahar V Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *Acm Sigmod Record*, 26(2): 289–300, 1997.

U. Kressel. Pairwise classification and support vector machines. In *Advances in Kernel Methods – Support Vector Learning*, page 255–268. MIT Press, 1999.

A Krizhevsky, I Sutskever, and GE Hinton. 2012 alexnet. *Adv. Neural Inf. Process. Syst.*, pages 1–9, 2012.

Stéphane Lathuiliere, Rémi Juge, Pablo Mesejo, Rafael Munoz-Salinas, and Radu Horaud. Deep mixture of linear inverse regressions applied to head-pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 3, page 7, 2017.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chenglong Li, Fan Zhong, Qian Zhang, and Xueying Qin. Accurate and fast 3d head pose estimation with noisy rgbd images. *Multimedia Tools and Applications*, 77(12):14605–14624, 2018a.

Deqiang Li and Witold Pedrycz. A central profile-based 3d face pose estimation. *Pattern Recognition*, 47(2):525–534, 2014.

Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *arXiv preprint arXiv:1804.08348*, 2018.

Wei Li, Yan Huang, and Jingliang Peng. Automatic and robust head pose estimation by block energy map. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3357–3361. IEEE, 2014.

Ye Li, YingHui Wang, Jing Liu, Wen Hao, and Liangyi Huang. A novel feature-based pose estimation method for 3d faces. In *International Conference on E-Learning and Games*, pages 361–369. Springer, 2018b.

Haibin Liao, Shejie Lu, and Dianhua Wang. Tied factor analysis for unconstrained face pose classification. *Optik*, 127(23):11553–11566, 2016.

Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

Yong Liu, Qicong Wang, Yi Jiang, and Yunqi Lei. Supervised locality discriminant manifold learning for head pose estimation. *Knowledge-Based Systems*, 66:126–135, 2014.

Yuanyuan Liu, Jingying Chen, Zhiming Su, Zhenzhen Luo, Nan Luo, Leyuan Liu, and Kun Zhang. Robust head pose estimation using dirichlet-tree distribution enhanced random forests. *Neurocomputing*, 173:42–53, 2016.

Yuanyuan Liu, Zhong Xie, Xiaohui Yuan, Jingying Chen, and Wu Song. Multi-level structured hybrid forest for joint head detection and pose estimation. *Neurocomputing*, 266:206–215, 2017.

Changwei Luo, Juyong Zhang, Jun Yu, Chang Wen Chen, and Shengjin Wang. Real-time head pose estimation and face modeling from a depth image. *IEEE Transactions on Multimedia*, 21(10):2473–2481, 2019.

Iiris Lüsi, Sergio Escarela, and Gholamreza Anbarjafari. Sase: Rgb-depth database for human head pose estimation. In *European conference on computer vision*, pages 325–336. Springer, 2016.

Bingpeng Ma, Shiguang Shan, Xilin Chen, and Wen Gao. Head yaw estimation from asymmetry of facial appearance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(6):1501–1512, 2008.

Bingpeng Ma, Xiujuan Chai, and Tianjiang Wang. A novel feature descriptor based on biologically inspired feature for head pose estimation. *Neurocomputing*, 115:1–10, 2013.

Bingpeng Ma, Annan Li, Xiujuan Chai, and Shiguang Shan. Covga: A novel descriptor based on symmetry of regions for head pose estimation. *Neurocomputing*, 143:97–108, 2014.

Bingpeng Ma, Rui Huang, and Lei Qin. Vod: a novel image representation for head yaw estimation. *Neurocomputing*, 148:455–466, 2015.

Francisco Madrigal and Frederic Lerasle. Robust head pose estimation based on key frames for human-machine interaction. *EURASIP Journal on Image and Video Processing*, 2020(1):1–19, 2020.

Gian Luca Marcialis, Fabio Roli, and Gianluca Fadda. A novel method for head pose estimation based on the "vitruvian man". *International Journal of Machine Learning and Cybernetics*, 5(1):111–124, 2014.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Hayet Mekami and Sidahmed Benabderrahmane. Towards a new approach for real time face detection and normalization. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 455–459. IEEE, 2010.

Gregory P Meyer, Shalini Gupta, Iuri Frosio, Dikpal Reddy, and Jan Kautz. Robust model-based 3d head pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3649–3657, 2015.

Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

Erik Murphy-Chutorian and Mohan Manubhai Trivedi. Head pose estimation in computer vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):607–626, 2009.

M Narasimha Murty and V Susheela Devi. *Pattern recognition: An algorithmic approach*. Springer Science & Business Media, 2011.

Alex Nanopoulos, Rob Alcock, and Yannis Manolopoulos. Feature-based classification of time-series data. *International Journal of Computer Research*, 10 (3):49–61, 2001.

Athi Narayanan, Ramachandra Mathava Kaimal, and Kamal Bijlani. Yaw estimation using cylindrical and ellipsoidal face models. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2308–2320, 2014.

Athi Narayanan, Ramachandra Mathava Kaimal, and Kamal Bijlani. Estimation of driver head yaw angle using a generic geometric model. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3446–3460, 2016.

Sourabh Niyogi and William T Freeman. Example-based head tracking. In *Proceedings of the second international conference on automatic face and gesture recognition*, pages 374–378. IEEE, 1996.

Miguel A Ochoa-Villegas, Juan A Nolazco-Flores, Olivia Barron-Cano, and Ioannis A Kakadiaris. Addressing the illumination challenge in two-dimensional face recognition: a survey. *IET Computer Vision*, 9(6):978–992, 2015.

Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden markov model-based approach to sequential data clustering. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 734–743. Springer, 2002.

Massimiliano Patacchiola and Angelo Cangelosi. Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods. *Pattern Recognition*, 71:132–143, 2017.

Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301. Ieee, 2009.

Xi Peng, Junzhou Huang, Qiong Hu, Shaoting Zhang, and Dimitris N Metaxas. Head pose estimation by instance parameterization. In *2014 22nd International Conference on Pattern Recognition*, pages 1800–1805. IEEE, 2014.

Xi Peng, Junzhou Huang, Qiong Hu, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. From circle to 3-sphere: Head pose estimation by instance parameterization. *Computer Vision and Image Understanding*, 136:92–102, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

P Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J Rauss. The feret database and evaluation procedure for face-recognition algorithms. *Image and vision computing*, 16(5):295–306, 1998.

John C Platt, Nello Cristianini, John Shawe-Taylor, et al. Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553, 1999.

Chunmei Qing, Jianmin Jiang, and Zhijing Yang. Normalized co-occurrence mutual information for facial pose detection inside videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(12):1898–1902, 2010.

Georgia Rajamanoharan and Timothy F Cootes. Multi-view constrained local models for large head angle facial tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 18–25, 2015.

Chotirat Ann Ratanamahatana and Eamonn Keogh. Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 11–22. SIAM, 2004.

Bisser Raytchev, Ikushi Yoda, and Katsuhiko Sakaue. Head pose estimation by nonlinear manifold learning. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 4, pages 462–466. IEEE, 2004.

Andreas Riener and Andreas Sippl. Head-pose-based attention recognition on large public displays. *IEEE computer graphics and applications*, 34(1):32–41, 2014.

Eric Sven Ristad and Peter N Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Rotation invariant neural network-based face detection. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 38–44. IEEE, 1998.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Nataniel Ruiz, Eunji Chong, and James M Rehg. Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2074–2083, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Anwar Saeed, Ayoub Al-Hamadi, and Ahmed Ghoneim. Head pose estimation on top of haar-like face detection: A study using the kinect sensor. *Sensors*, 15 (9):20945–20966, 2015.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

Georgia Sandbach, Stefanos Zafeiriou, Maja Pantic, and Lijun Yin. Static and dynamic 3d facial expression recognition: A comprehensive survey. *Image and Vision Computing*, 30(10):683–697, 2012.

William F Schreiber. Wirephoto quality improvement by unsharp masking. *Pattern recognition*, 2(2):117–121, 1970.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Amnon Shashua and Tammy Riklin-Raviv. The quotient image: Class-based re-rendering and recognition with varying illuminations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):129–139, 2001.

Reza Shoja Ghiass, Ognjen Arandjelović, and Denis Laurendeau. Highly accurate and fully automatic 3d head pose estimation and eye gaze estimation using rgb-d sensors and 3d morphable models. *Sensors*, 18(12):4280, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

Sujith Srinivasan and Kim L Boyer. Head pose estimation using view based eigenspaces. In *Object recognition supported by user interaction for service robots*, volume 4, pages 302–305. IEEE, 2002.

Jie Sun and Shengli Lu. An improved single shot multibox for video-rate head pose prediction. *IEEE Sensors Journal*, 20(20):12326–12333, 2020.

Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

David Joseph Tan, Federico Tombari, and Nassir Navab. Real-time accurate 3d head tracking and pose estimation with consumer rgb-d cameras. *International Journal of Computer Vision*, 126(2-4):158–183, 2018.

F Tarrés and A Rama. Gtav face database. from http. *gpstsc. upc. es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase. htm*, 2012.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Saravanan Thirumuruganathan. A detailed introduction to k-nearest neighbor (knn) algorithm. *Retrieved March*, 20:2012, 2010.

Carlos Eduardo Thomaz and Gilson Antonio Giraldi. A new ranking method for principal components analysis and its application to face image analysis. *Image and vision computing*, 28(6):902–913, 2010.

V Vapnik. Statistical learning theory new york. *NY: Wiley*, 1998.

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

Marco Venturelli, Guido Borghi, Roberto Vezzani, and Rita Cucchiara. From depth data to head pose estimation: a siamese approach. *arXiv preprint arXiv:1703.03624*, 2017.

Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

Minh Thanh Vo, Trang Nguyen, and Tuong Le. Robust head pose estimation using extreme gradient boosting machine on stacked autoencoders neural network. *IEEE Access*, 8:3687–3694, 2019.

Bingjie Wang, Wei Liang, Yucheng Wang, and Yan Liang. Head pose estimation with combined 2d sift and 3d hog features. In *Image and Graphics (ICIG), 2013 Seventh International Conference on*, pages 650–655. IEEE, 2013a.

Chao Wang and Xubo Song. Robust head pose estimation using supervised manifold projection. In *2012 19th IEEE International Conference on Image Processing*, pages 161–164. IEEE, 2012.

Chao Wang and Xubo Song. Robust head pose estimation via supervised manifold learning. *Neural Networks*, 53:15–25, 2014.

Chao Wang, Yuanhao Guo, and Xubo Song. Head pose estimation via manifold learning. In *Manifolds—Current Research Areas*. InTech, 2017.

Haofan Wang, Zhenghua Chen, and Yi Zhou. Hybrid coarse-fine classification for head pose estimation. *arXiv preprint arXiv:1901.06778*, 2019a.

Jian-Gang Wang and Eric Sung. Em enhancement of 3d head pose estimated by point at infinity. *Image and Vision Computing*, 25(12):1864–1874, 2007.

Juanjuan Wang, Lifang Wu, Xiaoguang He, and Jie Tian. A new method of illumination invariant face recognition. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, pages 139–139. IEEE, 2007.

Mei Wang and Weihong Deng. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*, 2018.

Qicong Wang, Yuxiang Wu, Yehu Shen, Yong Liu, and Yunqi Lei. Supervised sparse manifold regression for head pose estimation in 3d space. *Signal Processing*, 112:34–42, 2015.

Xianwang Wang, Xinyu Huang, Jizhou Gao, and Ruigang Yang. Illumination and person-insensitive head pose estimation using distance metric learning. In *European conference on computer vision*, pages 624–637. Springer, 2008.

Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013b.

Yujia Wang, Wei Liang, Jianbing Shen, Yunde Jia, and Lap-Fai Yu. A deep coarse-to-fine network for head pose estimation from synthetic data. *Pattern Recognition*, 94:196–206, 2019b.

Paul Werbos. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

Laurenz Wiskott, Rolf P Würtz, and Günter Westphal. Elastic bunch graph matching. *Scholarpedia*, 9(3):10587, 2014.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Yue Wu and Qiang Ji. Facial landmark detection: A literature survey. *International Journal of Computer Vision*, 127(2):115–142, 2019.

Jiahao Xia, Libo Cao, Guanjun Zhang, and Jiacai Liao. Head pose estimation in the wild assisted by facial landmarks based on convolutional neural networks. *IEEE Access*, 7:48470–48483, 2019.

Luhui Xu, Jingying Chen, and Yanling Gan. Head pose estimation with soft labels using regularized convolutional neural network. *Neurocomputing*, 337:339–353, 2019.

Ying Ying and Han Wang. Dynamic random regression forests for real-time head pose estimation. *Machine vision and applications*, 24(8):1705–1719, 2013.

Jongmin Yoon and Daijin Kim. An accurate and real-time multi-view face detector using orfs and doubly domain-partitioning classifier. *Journal of Real-Time Image Processing*, 16(6):2425–2440, 2019.

Yu Yu, Kenneth Alberto Funes Mora, and Jean-Marc Odobez. Robust and accurate 3d head pose estimation through 3dmm and online head model reconstruction. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 711–718. Ieee, 2017.

Yu Yu, Kenneth Alberto Funes Mora, and Jean-Marc Odobez. Headfusion: 360° head pose tracking combining 3d morphable model and 3d reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2653–2667, 2018.

Hui Yuan, Mengyu Li, Junhui Hou, and Jimin Xiao. Single image-based head pose estimation with spherical parametrization and 3d morphing. *Pattern Recognition*, page 107316, 2020.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Yaoyao Zhang, Jie Tian, Xiaoguang He, and Xin Yang. Mqi based face recognition under uneven illumination. In *International Conference on Biometrics*, pages 290–298. Springer, 2007.

Yuyao Zhang, Khalid Idrissi, and Christophe Garcia. A dictionary-learning sparse representation framework for pose classification. In *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2013.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):918–930, 2016.

Zhenqiu Zhang, Yuxiao Hu, Ming Liu, and Thomas Huang. Head pose estimation in seminar room using multi view face detectors. In *International Evaluation Workshop on Classification of Events, Activities and Relationships*, pages 299–304. Springer, 2006.

Zhiqiang Zhao, Qiaoli Zheng, Yan Zhang, and Xin Shi. A head pose estimation method based on multi-feature fusion. In *2019 IEEE 7th International Conference on Bioinformatics and Computational Biology (ICBCB)*, pages 150–155. IEEE, 2019.

Ronghang Zhu, Gaoli Sang, Ying Cai, Jian You, and Qijun Zhao. Head pose estimation with improved random regression forests. In *Biometric Recognition*, pages 457–465. Springer, 2013.

Yulian Zhu, Zhimei Xue, and Chunyan Li. Automatic head pose estimation with synchronized sub manifold embedding and random regression forests. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(3):123–134, 2014.

Xuan Zou, Josef Kittler, and Kieron Messer. Illumination invariant face recognition: A survey. In *2007 first IEEE international conference on biometrics: theory, applications, and systems*, pages 1–8. IEEE, 2007.