AGE ESTIMATION IN FACIAL IMAGES

A Thesis in
Computer Science and Engineering
by
Venkata Soumya Pisupati

© 2013 Venkata Soumya Pisupati

Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science

December 2013
The thesis of Venkata Soumya Pisupati was reviewed and approved* by the following:

Robert T Collins  
Associate Professor of Computer Science and Engineering  
Thesis Advisor

William Higgins  
Distinguished Professor of Electrical Engineering

Raj Acharya  
Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School
Automatic age estimation in human facial images is important because people’s behavioral patterns, lifestyles and preferences vary along with their age, and preferred modes of human-computer interaction are different for people of different age groups. An approach for automatic age estimation from facial images therefore has the potential for a number of practical applications. Human beings develop the ability to accurately estimate age early in life, but automatic age estimation is very sensitive to factors such as race, gender, facial attachments such as glasses, beards, facial piercings, and other visually perceptive factors like attractiveness, thus making it one of the challenging problems in computer vision. In this thesis we develop computer vision and machine learning based approaches for extracting visual features from a human facial image and mapping those to an output label estimating the approximate age (in years) or age group (range of years) of the individual.

We develop an automatic eye localization algorithm which finds the coordinates of both the eye centers. These co-ordinates are used to register all the facial images into a common frame of reference. Such a localization algorithm is very useful in real world applications where information about eye-center is usually not available. We then extract feature descriptors that represent the facial image in a low dimensional space. In this thesis we have experimented with three different kinds of age relevant feature descriptors in order to capture the shape and texture information of facial images: Discrete Cosine Transform (DCT), Local Binary Pattern (LBP) and Gradient Orientation Pyramids (GOP).

We then propose two machine learning pipelines for age estimation. First, we develop a hierarchical approach to estimate the age of the test image using two steps. In the first step we perform soft classification by separating the training data into age groups and using a kNN/SVM classifier to compute the probability of a new test image belonging to each of the age three groups. We then use Gaussian Process (GP) regression for the final age estimation. Gaussian process regression is a Bayesian approach that computes the posterior density for the age of a test image given both the training and test data. GP produces confidence intervals for the age, thereby providing the user with more information. Second, we develop expectation maximization (EM) framework to jointly address the issues of categorizing the training data points into groups and learning the hyper parameters of the Gaussian Process regression model corresponding to each group. Unlike the earlier approaches where we manually categorize the training data into three groups based on age, this approach automatically groups the data.

We test our methods on the publicly available FGNET aging database, a very popular and challenging database for testing facial aging based algorithms. We evaluate our algorithms using the Leave One Person Out (LOPO) evaluation scheme. In LOPO we train the regressor on images of all the persons in the database except one, and test it on the images of that one person which have not been used in the training. We repeat this for all the people in our database and compute the error by averaging across errors from the individual testing rounds. This metric is known as the Mean Absolute Error (MAE) and has been used as the performance metric to evaluate performance of various features and our estimation algorithms.
<table>
<thead>
<tr>
<th>Chapter Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>viii</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Automatic Age Estimation</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Challenges in Automatic Age Estimation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Problem Statement</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Literature Review</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1 Feature Extraction</td>
<td>5</td>
</tr>
<tr>
<td>1.3.2 Feature Classification</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Overview and Contributions of the Thesis</td>
<td>9</td>
</tr>
<tr>
<td>1.4.1 Overview</td>
<td>9</td>
</tr>
<tr>
<td>1.4.2 Contributions</td>
<td>11</td>
</tr>
<tr>
<td>1.5 Organization of the Thesis</td>
<td>12</td>
</tr>
<tr>
<td>Chapter 2 Preprocessing and Feature Extraction</td>
<td>13</td>
</tr>
<tr>
<td>2.1 Eye localization</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Registration based on Eye locations</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Feature Extraction Methods and Techniques</td>
<td>20</td>
</tr>
<tr>
<td>2.3.1 Local Binary Patterns</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2 Discrete Cosine Transform</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3 Gradient Orientation Pyramids</td>
<td>24</td>
</tr>
<tr>
<td>2.4 Feature Fusion</td>
<td>26</td>
</tr>
<tr>
<td>Chapter 3 Hierarchical Age Estimation using Manual Age Grouping</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Classification Algorithms</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Support Vector Machines</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2 K-Nearest Neighbor Classification</td>
<td>31</td>
</tr>
<tr>
<td>3.2 Regression Algorithms</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1 Support Vector Regression</td>
<td>32</td>
</tr>
<tr>
<td>3.2.2 Gaussian Processes Regression</td>
<td>33</td>
</tr>
<tr>
<td>3.2.2.1 The Multivariate Gaussian Distribution</td>
<td>33</td>
</tr>
<tr>
<td>3.2.2.2 Bayesian Linear Regression</td>
<td>34</td>
</tr>
<tr>
<td>3.2.2.3 Gaussian Processes</td>
<td>37</td>
</tr>
<tr>
<td>3.2.2.4 Kernel Selection and Hyper-Parameter Estimation for Gaussian Process Regression</td>
<td>40</td>
</tr>
<tr>
<td>3.3 Hierarchical Estimation</td>
<td>42</td>
</tr>
</tbody>
</table>
Chapter 4 Hierarchical Age Estimation via Automatic Grouping using an Expectation Maximization based Approach ................................................................. 46

4.1 The Expectation Maximization Algorithm ........................................... 47
4.2 EM Algorithm based Framework for Hierarchical Age Estimation .......... 48

Chapter 5 Experiments ........................................................................... 53

5.1 FGNET aging Database ....................................................................... 53
5.2 Results ............................................................................................... 54

Chapter 6 Conclusions and Future Work ................................................ 66

Bibliography .......................................................................................... 68
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of an Automatic Age Estimation System.</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>An example of a facial image with scarring. For this case, the scar appears</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>on all the images taken after the age of 35.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Stages in Age Estimation</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Flow chart of the proposed age estimation method</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Original and the Binary Thresholded Image</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>(a) Example Image from the data base (b) Magnitude of ( x )-Gradient (c)</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Magnitude of ( y )-Gradient (d) Cumulative columns sum for each column of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the horizontal gradient image (e) Cumulative row sum for each row of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vertical gradient image (f) Image with bounding box</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Examples of Eye Localization</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>Block Diagram for Image Preprocessing</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>Basic LBP Operation</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>The zigzag ordering of the DCT coefficients (source: Wikipedia).</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>Computation from an Input Image (Source: Ling et. al)</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>Supervised Machine Learning Approaches</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>Example of an Optimal Hyper-plane</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>Transformation to a Higher Dimensional Space (source: Steiner10).</td>
<td>29</td>
</tr>
<tr>
<td>15 (a)</td>
<td>Univariate and (b) Multivariate Gaussian Distribution</td>
<td>34</td>
</tr>
<tr>
<td>16</td>
<td>A Flow Chart of our proposed Hierarchical Age Estimator</td>
<td>45</td>
</tr>
<tr>
<td>17</td>
<td>Block Diagram of the EM Framework</td>
<td>52</td>
</tr>
<tr>
<td>18</td>
<td>Data distribution according to age</td>
<td>54</td>
</tr>
<tr>
<td>19</td>
<td>Example of an image where the localization algorithm could not detect the</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>eyes</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Histogram of normalized error of eye localization</td>
<td>56</td>
</tr>
<tr>
<td>21</td>
<td>Cumulative Scores for the Hierarchical and EM based approaches</td>
<td>65</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1: Existing Global Feature Extraction Methods 6
Table 2: MAE for Single Level Age Estimation using Regression 57
Table 3 MAE for Hierarchical Age Estimation with Manual Classification and Gaussian Process Regression 58
Table 4 MAE of hierarchicaal age estimation using kNN classification and GP regression 59
Table 5 : MAE of hierarchical Age estimation using SVM classification and GP regression 60
Table 6 Classification accuracies for kNN and SVM classifiers using different feature sets 61
Table 7 MAE for hierarchical estimation using feature fusion 61
Table 8 MAE for 3 groups using GOP and combined features 62
Table 9 MAE obtained using the proposed EM Approach with 3 Groups 63
Table 10 MAE obtained for the hierarchical and the EM methods with automatically preprocessed images 63
Table 11 MAE per group for various algorithms 64
ACKNOWLEDGEMENTS

First and foremost, I want to thank my advisor Dr. Robert Collins for consenting to be my mentor and patiently guiding me through all my delays and setbacks. Dr. Collins has always inspired me both as a researcher and as a personality and I count myself incredibly fortunate to be one of his students. I shall always be grateful to him for generating that interest and curiosity towards the areas of Computer Vision in me.

Next, I would sincerely like to thank my committee member Dr. William Higgins for his time and for consenting to be on my committee. The insights and knowledge gained through his Image Processing classes were instrumental in shaping up this thesis.

I would like thank the IST department, Professor David Hall, and the NC2IF laboratory for giving me the opportunity to work with them during the course of my stay at Penn State.

I would like to thank all my Professors at Penn State and DA-IICT. It is only because your foundations I could reach where I am today.

I would like to thank my family – my uncle, aunt, cousins, grandparents, my parents in-law, and my friends for the amazing support system they provide.

I want to thank my siblings Deepthi and Pranav. You guys are the best!

The road to this thesis was a long and challenging one and I am eternally grateful to my best friend and husband Phani for being there with and holding my hand at each and every step. You made this journey not just possible but a very memorable one.

Finally this thesis is a tribute to all the unconditional love, support, prayers and sacrifices of my parents Vijay and Manibala.
Chapter 1

Introduction

Human beings possess an innate ability to not only recognize faces, but also to distinguish one face from another, whereas machines have only recently started displaying these skills. Scientists first began working towards developing such systems in the mid 1960’s. Face recognition has come a long way since then. Today’s facial recognition systems deploy state of the art algorithms and are being used for a variety of commercial and government applications such as law enforcement, voting, user authentication and security purposes at various venues like airports and banks.

The latest models of facial recognition systems have been extended to include solutions to other relevant and challenging problems such as face verification, identity recognition, gender detection, gender based classification and expression recognition. One such extension to the face recognition problem is **automatic age estimation from facial data**. Compared to the above mentioned problems, automatic age estimation from faces is a topic that has been relatively less explored despite being a very interesting and a challenging sub-field in facial recognition [Han13].

1.1 Automatic Age Estimation

Automatic age estimation in human facial images is important for many reasons. People’s behavioral patterns, lifestyles and preferences vary along with their age. Also preferred modes of human-computer interaction are different for people of different age groups. Thus a good understanding of age can lead to more smooth and successful communication between people and
machines, making automatic age estimation from facial recognition an area with a lot of potential for a number of practical applications \([\text{Geng07, Steiner10}]\) related to law enforcement, multi-cue identification, age specific access control and age specific HCI. Human beings develop the ability to estimate age early in life and can be fairly accurate in their estimation. Human age estimation is however very sensitive to factors like race, gender, facial attachments such as glasses, beards, facial piercings, and other visually perceptive factors like attractiveness. Hence it would be very useful to have an automated Age Estimation System that takes a human facial image as its input and assigns an output label to the image, where this label is the exact age (in years) or the age group (year range) of the individual face as shown below in Fig 1. \([\text{Geng07}]\).

![Figure 1: Example of an Automatic Age Estimation System.](image-url)

Additionally, an ideal automatic age estimation system must also be capable of doing Age Synthesis, which means that given an input facial image and a target older age, the system should be capable of producing a synthesized image of the individual at the target age by re-rendering the face image aesthetically with natural aging and rejuvenating effects \([\text{Geng07}]\).

1.1.1 Challenges in Automatic Age Estimation

There are a number of challenges associated with building automated age estimation systems some of which are listed below \([\text{Geng07}]\).
1) **Aging Patterns are incomplete**: Aging is a slow, irreversible and an uncontrollable process and is something that cannot happen at will and this makes the collection of training data for the process of age estimation an extremely laborious process. The available datasets today contain a very limited number of age separated images for each person and the images at higher ages are rare.

2) **Aging patterns are personal**: Different people age differently depending on their genes, gender, lifestyles, working environment, weather conditions, and health factors such as drinking and smoking. Thus it is not easy to find a unique mapping from facial images (features) to ages (class labels).

3) **Aging patterns are temporal**: The face status at a particular age will affect all the older faces but not the younger ones, which means that, any particular change to a face at a certain age will affect the subsequent training and testing phases. For example consider a case where an individual develops a scar across his forehead at a particular age as shown below in Fig 2. For this case, the scar appears on all the images taken after the age of 35. If we train the aging model generically, the estimation error for this individual case will be very high whereas if we train the model to incorporate the scarring after a particular age, the model will be too specific.

![Figure 2](image-url)  
*Figure 2: An example of a facial image with scarring. For this case, the scar appears on all the images taken after the age of 35.*
1.2 Problem Statement

This thesis has the following objectives

- Provide a detailed survey in the area of age estimation using facial images.
- Propose new techniques based on the ideas already explored and to come up with our own solution to solve the age estimation problem that leverages recent advances in machine learning.

1.3 Literature Review

Recent years have seen a number of researchers investigating how to accurately estimate ages from facial images. A good starting point to explore age estimation from facial images is the survey paper on age progression in human faces by Narayanan Ramanathan et al [Narayanan09]. In this paper the authors give an in depth understanding regarding the basics of the human aging process by helping us understand the morphological changes associated with human growth. They then try to analyze answers to questions like how humans perceive aging, which part of the human face constitutes an age invariant signature, and how does aging impact facial recognition performance. D’Arcy Thompson’s study on morphogenesis [Darcy92] was the first step in this area and played a crucial role in understanding the morphological changes associated with growth in human faces. Many subsequent research works stemmed from this study. Shaw et al. [Shaw74] observed that cardioordial strain plays an important part in the way humans perceived aging and came up with an affine shear transform to describe facial growth. Kwon et al [Kwon94] used 47 high resolution images of a face to classify the images into one of three age groups: babies, young adults or seniors. Their approach was based on geometric ratios of key facial features and wrinkle analysis. Fu et al [Fu10] have provided a comprehensive review of the many existing methods
for age estimation till 2010 and more recently Han et al [Han13] have provided a comparison of the of all the published methods for automatic age estimation.

In general, most Age-Estimation systems are designed to have two stages as shown below in Fig 3.

- **Feature Extraction**: Extraction of the aging features
- **Feature Classification/Regression**: Classification/Association of these features to age-groups or specific ages.

![Figure 3: Stages in Age Estimation](image)

Most age estimation methods are classified based on the feature extraction and the feature classification techniques they employ. A review of some prior studies on both of these stages is given in the subsections below.

### 1.3.1 Feature Extraction

The facial features used in previous works can be divided into three categories: global features, local features and hybrid features.

**Global Features**: Global features contain the overall individual characteristics of a person such as identity, expression, gender and ethnicity along with aging characteristics. They thus offer
better information about individual characteristics such as the appearance and the shape of a face than features like skin and wrinkles. *Global features are generally used to do detailed age estimation*. Most approaches that use global features try to generate face aging models and build aging functions to simulate or compensate for the aging process. The global feature extraction methods that have been used in the literature along with their descriptions are listed in Table 1.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Used In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Appearance Models (AAM)</td>
<td>Represents faces with statistical shape and appearance models using PCA. AAM parameters are extracted from face landmark points. Optimal regressors (aging functions) are built on these parameters.</td>
<td>First proposed by Lanitis et al. [Lantis02]. Geng et al. [Geng06]. Lanitis et al. [Lantis04]. Geng et al. [Geng07]. Yan et al. [Yan07].</td>
</tr>
<tr>
<td>Aging Pattern Subspaces (AGES)</td>
<td>Takes a sequence of individual aging facial images as the input and constructs a subspace representation of the aging patterns for these images using PCA. Uses AAM to construct the feature vector. When an unseen face comes, it is projected onto a subspace which reconstructs the face with minimum error.</td>
<td>First proposed by Geng et al. [Geng07]. Guo et al. [Guo08]. Fu et al. [Yun08]. Fukai et al. [Fukai07]</td>
</tr>
</tbody>
</table>

Table 1: Existing Global Feature Extraction Methods
**Local Features:** Local features consist of features such as wrinkles, skin, hair, and geometric features like distance ratios of various facial components. Local features encode age features and are often more specific to an age-group rather than an individual. They are thus used to classify people into age groups. Local features are also more robust to factors like illumination due to the sufficiency of local patches to overcome the shortcomings of sensitivity to illumination variations and image occlusions [Narayanan09]. Kwon and Lobo [Kwon94] classified images into three groups: babies, young adults and senior adults based on wrinkles and distance ratios. They first extracted the distance ratios using facial anthropometry and used these distances to distinguish babies from adults. They then used snakelets in pre-designated facial regions to extract wrinkles which they then subsequently used to differentiate between young and senior adults. Horng et al [Horng01] used geometric features and wrinkles to classify the facial images into four groups. They used the Sobel Filter to compute the depth and degree of the wrinkle. Hayashi et al. [Hayashi01] used skin and wrinkle features to address the age estimation problem. They extracted skin using a skin color and then extracted the wrinkles using a Digital Template Hough Transform (DTHT). Takimoto et al, [Takimoto08] used a combination of Sobel filters and Gabor jets to distinguish a deep wrinkle from a fine one. Jun-Da Xia and Chung-Lin Huang [Txia09] proposed to use wrinkle and hair features for age classification. They also used the Sobel filter to extract wrinkles, and combined them with the hair color features. The regions for feature extraction were based on facial landmarks detected by an AAM.

**Hybrid Features:** Hybrid features are a combination of global and local features. They were proposed by Suo et al. [Suo08] who constructed a set of sparse features consisting of AAM, wrinkle, skin, hair and face configuration features using a hierarchical face model. For each facial component four kinds of features are extracted: photometric, geometric, topological, and configuration features. The feature extraction described here is computationally very elaborate
compared to other methods. Matthew Steiner [Steiner10] in his study on facial age estimation used a combination of age relevant shape (AAM) and texture (DCT) features to learn a model of the human aging process. Choi et al. [Choi11] proposed a hierarchical age estimation scheme using a hybrid of AAM, wrinkle and skin features. They extract the wrinkle features using a set of region specific Gabor filters and the skin feature is extracted using a Local Binary Pattern.

1.3.2 Feature Classification

Age estimation techniques can be classified into three categories from the point of view of feature classification: age-group classification, single level age estimation and hierarchical age estimation. Age group classification is used to classify the images into one of multiple age-groups. In the past, classification algorithms such as LDA [Gao09], Support Vector Machine (SVM) [Lian05], distance measure [Gunay08], and neural networks [Horng01] have been used. Single level age estimation and hierarchical age estimation are both used to estimate the detailed age of the individual image. Regression techniques such as quadratic regression [Lantis02], Support Vector Regression (SVR) [Guo08], the MLP algorithm - Multi-Layer Perceptron [Lantis04, Suo08] and the RUN algorithm [Yan07] have been used for single stage age estimation.

Hierarchical age estimation was implemented by Lanitis et al. [Lantis04] who showed that hierarchical age estimation performs better than single level age estimation. This is so because, since facial aging is perceived differently for different age-groups, and thus age estimation for a specific age group gives a more accurate result. Guo et al. [Guo09] proposed a classification scheme based on small groups divided by gender and age rather than on all ages. Luu et al. [Luu09] proposed hierarchical classification based on SVM for age group classification
followed by SVR for specific age estimation. The best reported hierarchical classifier was a combination of an age specific and an appearance specific classification [Choi11, Lanitis04].

1.4 Overview and Contributions of the Thesis

In this section we first give an overview of our proposed method. We briefly mention the algorithms that we have used along with a flow chart which describes the architecture of our proposed age estimation system. The detailed descriptions of each of these algorithms and techniques are given in the subsequent chapters.

1.4.1 Overview

The goal of this thesis as stated in section 1.2 is to construct our own unique solution to the problem of age estimation by building upon the already existing ones. We use images from the FG-NET database for this work which is described in further detail in Chapter 5. The proposed age estimation method consists of global and local feature extraction. This is followed by feature fusion and hierarchical age estimation including a hybrid age + appearance based group classification and a specific age estimation algorithm as shown in Figure 4. In the global feature extraction step, we use the Gradient Orientation Pyramid (GOP) [Ling10] algorithm to extract the shape parameters. In the local feature extraction step, the texture and skin features are extracted using Discrete Cosine Transform (DCT) and the Local Binary Patterns (LBP) algorithms respectively. We then perform a feature level fusion to combine the global and the local features into a feature vector. For the age estimation step, we propose a hierarchical approach where we first use a support vector machine to classify the new test image into one of the three age groups. We then propose an Expectation - Maximization (EM) algorithm based method to automatically
categorize the training data into different groups based on aging and appearance. Finally the
detailed age is estimated within each selected group using Gaussian process regression. We learn
the hyper-parameters of the GP for each group within the EM framework.

Figure 4: Flow chart of the proposed age estimation method
1.4.2 Contributions

The primary contributions of this thesis are

- **Automatic Extraction of Eye Center Co-ordinates**: An automatic eye localization algorithm for facial images has been implemented and its performance has been evaluated. We find eye centers by using a Hough transform for detecting circle hypotheses and then performing a series of geometrical tests exploiting the symmetrical properties of facial images in order to remove the false positives. These co-ordinates are then used to register all facial images into a common frame of reference. Such a localization algorithm is very useful in real world applications where information about eye-center or facial landmark points is usually not available.

- **Hierarchical Age Estimation using Soft Labeling**: We develop a hierarchical approach to estimate the age of the test image using a soft classification-regression algorithm. In the first step we perform soft classification by manually separating the training data into age groups and using a SVM classifier to compute the probability of the test image belonging to each of these groups. In the second step we use Gaussian Process regression for the estimation. Gaussian Process regression is a Bayesian approach that computes the posterior density for the age of a test image given both the training and test data. Gaussian process can generate confidence intervals for the age, thereby providing the user with more information. Additionally, in Gaussian Process’, the kernel hyper-parameters (length-scale, noise level, etc.) can be learnt via evidence maximization. The estimated age of the individual is then found by taking a weighted average of the ages computed from the regressors of each group. Such a soft classification technique is advantageous over the traditional hard classification since the age is computed taking all the groups into account which mitigates the effects of a wrong classification thereby reducing the overall error.
**Automatic Data Grouping for Soft Labeling**: Finally we develop an expectation maximization (EM) framework to jointly address the issues of categorizing the training data points into groups and learning the hyper-parameters of the Gaussian Process regression model corresponding to each group. Unlike the earlier approach where we manually categorized the training data into groups based on age, the EM method will automatically group the data based on appearance. This automatic grouping will result in a better performance for the system.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows – Chapter 2 gives a detailed description of the various preprocessing techniques that have been implemented. We then develop an automatic eye localization algorithm to compute eye coordinates used for facial image registration. We also describe the various feature extraction algorithms that we have implemented in order to generate our image descriptors. In Chapter 3 we first provide a mathematical model for regression, covering Linear Bayesian, Support Vector Machines (SVM) and Gaussian Process (GP) regression techniques. We provide detailed insights into how SVM and GP regressions work. We then combine the above methods with a classifier to propose our hierarchical algorithm for age estimation. In Chapter 4 we propose an Expectation Maximization (EM) based framework to group the data automatically and also learn the hyper parameters of the Gaussian process regressor corresponding to each group. Chapter 5 presents the experiments performed and discusses the results obtained. Chapter 6 concludes the thesis.
Chapter 2

Preprocessing and Feature Extraction

In this chapter, we describe the preprocessing techniques that we applied on the images in the FGNET database. The goal of these preprocessing methods is to extract just the facial regions and remove the background from every image, while making sure that the illumination and pose are consistent across all the images. First we describe the eye localization method that we use to register and scale all the images in the data base.

2.1 Eye localization

Eye localization is a very important first step towards pose invariance in image preprocessing for facial images. Most of the eye localization algorithms found in literature use some sort of training to localize the eye coordinates. Our method does not use any training data, which makes our system more ubiquitous since in many real life scenarios the availability of training data is almost limited to none. Our database consists of images that have already been passed through a face detection system that extracts the facial region. Thus the eye localization system developed here assumes that the face has been roughly localized and already cropped using face detection algorithm.

We first localize the eye region in an image by constructing a bounding box around the eyes. In order to do this, we first threshold the image and convert it into a binary image. The idea is to choose a low threshold, so that the pupil region turns dark and the facial skin region turns bright. Choosing a universal threshold that works for all images in the data base is hard in practice. Hence, we used an adaptive threshold approach, where we start with a low threshold, and iteratively keep increasing the threshold if the eyes (pupils) are not detected (the tests used for
pupil detection are described subsequently). An original image and its thresholded binary image are shown in Fig 5.

![Original and Binary Thresholded Image](image)

**Figure 5: Original and the Binary Thresholded Image**

Next, we find the horizontal and vertical gradients of the binary thresholded image for all images in the database, as shown in Fig 6. Finding the horizontal gradient helps us to get an idea of the exact position of the facial region in the image. This is because, in the horizontal gradient image of the face, since gradient values are very high at the boundary of the face and the background, the two vertical lines between which the face is contained are preserved. We find the two vertical lines of the face between which the face region is contained. In order to find these two vertical lines, for each column of the image we find a cumulative sum of gradient magnitudes over all the rows. The two columns which have the largest cumulative sums are considered to be the estimates of the two columns (vertical lines) between which the face region is contained.

Similarly, using the vertical gradient image of the face and the cumulative sum of gradient magnitude over all columns for each row, we find the two horizontal lines between which the face region is contained. Here, for most of the cases, the first line corresponds to the
transition between the hair and forehead, and the second line can correspond to either the transition from nose to lips or chin to the background.

To find the two vertical boundary lines, we search for the two columns with maximum cumulative sum separately in the left half and the right half of the image. Similarly, to find the two vertical horizontal boundary lines, we search for the two rows with maximum cumulative sum separately in the top half and the bottom half of the image.

![Figure 6](image)

Once the face region is more finely localized, we want to find the candidate pupil locations. To do this, we apply a circle detection algorithm on the image to find all the possible circles in the face region. In this work, we used the circle detection algorithm given in [Peng10], using the
This method uses the Hough Transform to detect circles. The transform is applied to the gradient magnitude image and results in several circles being detected. So in order to find the potential candidates that correspond to the eyes we performed the following geometric tests:

1. The circle pair should be within the bounding box.
2. The y-coordinates corresponding to the centers of the pair should be almost the same (we have allowed a maximum leeway of around 40 pixels).
3. The x-coordinates corresponding to the centers of the pair should be at least $d_e$ pixels apart.
4. The y-coordinates of the pair should be at least at least $d_t$ pixels from the y-coordinate corresponding to the top bounding box line and at least $d_b$ pixels from the y-coordinate corresponding to the bottom bounding box line.
5. The x-coordinate of the left eye is at least $d_l$ pixels from the x-coordinate corresponding to the left box line. Similarly, x-coordinate of the right eye is at least $d_r$ pixels from the x-coordinate corresponding to the right bounding box line.

**In this work we chose:** $d_e = 75, d_t = 25, d_b = 50, d_l = 25$ and $d_r = 25$

Note that this choice of the parameters depends on the scale of image. In our work, we scaled all the images to be of size 400 x 500 before we start any of the preprocessing. If the scale of the images changes, the values have to be changed appropriately. After performing all the geometric tests, if only one pair remains, we declare it as the eye pair location. If there is more than one pair, we consider a small square region of side 8 pixels around each circle and choose the circle pair which has the lowest mean intensity inside the square region - the idea is that if the pair actually corresponds to the eyes, then it should have the darkest intensity since the pupil region is dark. If we can’t find any pair that satisfies the geometric conditions given above, we repeat the whole
process by increasing the threshold used for converting the original image into a binary image, and repeat the process. If there is no pair detected after the threshold reaches a value of 255 (the highest intensity possible), we will not consider the image for processing. Fig 7 shows some results obtained using the eye localization algorithm described. We also include some cases corresponding to wrong detections.

![Figure 7: Examples of Eye Localization](image)

### 2.2 Registration based on Eye locations

In this section, we describe the image registration techniques that we used in this work. The goal of the image registration is three fold:

- Rotate the image so that the y-coordinates of the eyes are on the same line.
- Scale the image so that the distance between the eyes is same for all the images
• Translate the image so that the line connecting the centers of the eyes is at the same position in all the images.

Below, we describe how we accomplish each of these tasks.

Let \((x_l, x_r)\) and \((y_l, y_r)\) denote the x- and the y- coordinates of the left and the right eye of the image, respectively. The angle corresponding to the rotation when \(y_r > y_l\) is given as

\[ \theta = \text{atan2}\left( \frac{y_r - y_l}{x_r - x_l} \right) \]  

(2.1)

Similarly when \(y_l > y_r\) the equation is obtained as

\[ \theta = -\text{atan2}\left( \frac{y_r - y_l}{x_r - x_l} \right) \]  

(2.2)

The rotation matrix about the origin is then obtained as

\[
R = \begin{bmatrix}
\cos(\theta) & \sin(\theta) \\
-\sin(\theta) & \cos(\theta)
\end{bmatrix}
\]  

(2.3)

In addition to the rotation, we want to perform a scaling so that the distance between the eyes is the same in all the images. Let \(d\) denote the desired distance between the eyes (in pixels). We then want to scale all the images at a scale \(s\) defined as

\[ s = d / (x_r - x_l) \]  

(2.4)

With the scaling incorporated, the rotation matrix can be modified as

\[
R = \begin{bmatrix}
s \cdot \cos(\theta) & s \cdot \sin(\theta) \\
-s \cdot \sin(\theta) & s \cdot \cos(\theta)
\end{bmatrix}
\]  

(2.5)

Note that this definition of the rotation matrix considers that the rotation is about the origin, if we want to rotate about a point \((x_{cen}, y_{cen})\) this transformation will be modified to first move the center to the origin and then translate it back after the rotation. The transformation that defines this operation is obtained as

\[
\begin{bmatrix}
x_{\text{new}} \\
y_{\text{new}}
\end{bmatrix} = \begin{bmatrix}
s \cdot \cos(\theta) & s \cdot \sin(\theta) \\
-s \cdot \sin(\theta) & s \cdot \cos(\theta)
\end{bmatrix} \begin{bmatrix}
x_{\text{cen}} \\
y_{\text{cen}}
\end{bmatrix} + \begin{bmatrix}
x_{\text{cen}} \\
y_{\text{cen}}
\end{bmatrix}
\]  

(2.6)

In the above equation, \((x_{\text{new}}, y_{\text{new}})\) represent the coordinates after transforming the image. Using this operation we can transform the image such that the y-coordinates of the transformed
image are on the same line and the distance between the eyes is the same across all the images. While applying this transformation, we used nearest neighbor interpolation to map the non-integer pixel locations. In order to further translate the image so that the line connecting the centers is at a fixed location in all the images, we used the following method. With the centers of the eye as a starting point, we only keep a fraction of the image in all the directions. We have chosen coefficients \((k_l, k_r, k_t, k_b)\) to crop the facial region. In the y-direction we select a region from \(y_l - k_t \cdot d\) (pixels) to \(y_l + k_b \cdot d\) (pixels) where \(y_l\) denotes the y-coordinate of the left eye in the transformed image (note that the right eye has the same y-coordinate after transformation). In a similar manner we select the region \(\left(\frac{x_l + x_r}{2}\right) - k_l \cdot d\) to \(\left(\frac{x_l + x_r}{2}\right) + k_r \cdot d\) where \((x_l, x_r)\) are the x-coordinates of the left and the right eye in the transformed image respectively. In this manner, we crop the facial regions from all the images. After performing all the above preprocessing steps, the remaining facial image is scaled to a size 100 x 100. Histogram equalization is applied to this image prior to extracting features described below. The block diagram of our preprocessing pipeline is shown below in Fig 8.

Figure 8 : Block Diagram for Image Preprocessing
2.3 Feature Extraction Methods and Techniques

A key issue in facial analysis is finding efficient descriptors to represent the facial images. Feature extraction is the process of representing high dimensional image data with a low dimensional feature vector that retains useful information from that image. The performance of an age estimation algorithm is highly dependent on this step and hence it is very important to select good features. Ideally facial image feature descriptors must have the following characteristics [Ahonen06]

1) They must have high inter-class variance.
2) They must have low intra-class variance
3) They must be easy to compute

In this work, we use 3 kinds of features:

- LBP (Local Binary Pattern) Features
- DCT (Discrete Cosine Transform) Features
- GOP (Gradient Orientation Pyramids) Features

2.3.1 Local Binary Patterns

The Local Binary Pattern (LBP) operator is a texture descriptor that has been used in a wide variety of facial recognition applications. The basic idea of the LBP is that for every pixel in an image, the LBP operator returns a discrete value that characterizes the local texture information in such a way that it is robust to luminance changes. The basic LBP operator follows the following steps to assign a label to each pixel in the image:
a) Centered at a pixel in the image, the operator compares that pixel’s intensity to the intensities of its surrounding 8 neighboring pixels, assigning a 0 or a 1 to each neighbor depending on whether they are less than or greater than the center value [Prince12].

b) These binary values are then concatenated in a predetermined order to form an 8 bit binary pattern, as shown below in Fig 8. This binary pattern is then converted to a single decimal value. This decimal value serves as the label of the center pixel.

A histogram of these labels across all pixels can then be used as a texture descriptor for the image.

The LBP operator can be extended in many ways. We have used three extensions of the LBP operator for our feature extraction namely: LBP with Uniform Patterns, Rotation Invariant LBP and Spatially Enhanced Histograms all of which are described below.

**Uniform Patterns:** An LBP is called uniform if the binary pattern contains at most two bitwise transitions from 0 to 1 or vice versa when the bit pattern is considered circular [Ahonen]. For example the patterns 01110000 and 11001111 are uniform whereas the patterns 11001001 and 01010011 are not. For an 8 bit binary pattern, there are total of 256 possible labels, out of which only 58 labels correspond to uniform patterns. It has been noted by Ojala et al [Ojala02] in their experiments with texture images that when using a neighborhood of 8 pixels, uniform patterns account for almost 90% of all patterns. Thus, in the computation of the LBP labels, uniform patterns are preferred so that there is a separate label for each uniform pattern and all the non-uniform patterns are labeled with a single label. We use LBP with uniform patterns in our work.
because it has been noted Choi et al. (Choi et al., 2011) that for a facial image uniform patterns describe various texture information such as bright spots, flat areas, dark spots and edges whereas non-uniform patterns do not contain enough information to these textures.

**Spatially Enhanced Histograms:** The basic LBP histogram—which is the histogram representation of all the labels in an image, can be extended to a spatially enhanced histogram in order to encode appearance as well as spatial relationships of face regions. A facial image can be broken down into $m$ facial regions as shown in the image below. After the $m$ regions $R_0 \ldots R_{m-1}$ are determined, we compute a histogram independently for each of these $m$ facial regions. The resulting $m$ histograms are then concatenated to form a spatially enhanced histogram of size $m \times n$, where $n$ is the length of a single LBP histogram. This enhanced histogram represents a feature vector describing the image.

**Rotation Invariant LBP:** We also generate rotationally invariant LBP codes by circularly rotating the original LBP code until its minimum value is reached. An example of a rotation invariant code is given below in Fig. The motivation for using rotation invariant LBP codes is that rotation invariance makes the code more robust to order we start from.

![Rotation Invariant LBP Example](image)

**LBP Feature Extraction:** We computed the local binary pattern (LBP) histograms on 100 non-overlapping patches of the image, each of size 10 x 10. For each patch, we get a 10 dimensional feature vector. This is because there are 9 unique labels that correspond to the uniform and rotationally invariant LBPs and 1 label that corresponds to non-uniform patterns. Thus the LBP
feature vector for an image is obtained by concatenating the feature vectors corresponding to each of the 100 patches and has a dimension of 1000 (100 patches * 10 features per patch).

2.3.2 Discrete Cosine Transform

We also developed a feature set based on the discrete cosine transform of the image. The top-left corner entry of the transformed image will have a large magnitude. This is the DC or the low frequency coefficient. The DC coefficient represents the average intensity of the image. The remaining 63 coefficients are called the AC coefficients, and they correspond to higher frequencies.

In this work, we applied a 2D-DCT to non-overlapping patches of size 10 x 10, resulting in a 10 x 10 matrix. Within each block of the image, the pixels differ from a low variation to a high variation in a zig-zag pattern [Wang10], as shown below in Fig. 10.

Note that the block shown here is 8 x 8, whereas we used a block size of 10 x 10 in our work.

Figure 10: The zigzag ordering of the DCT coefficients (source: Wikipedia).
We have retained 10 DCT coefficients in each block, following the order above. Since each image is of size 100*100, there will be 100 such non-overlapping blocks, leading to a 1000 dimensional feature vector to represent the image.

2.3.3 Gradient Orientation Pyramids

Gradient Orientation Pyramids (GOP) is a feature extraction technique first proposed by Li et al in [Li11]. Image Gradient based descriptors have been used to represent image information extensively in the past. One such gradient based descriptor that has been used a lot in the context of facial recognition is the famous Histogram of Oriented Gradients (HOG) descriptor [Dalal05]. The HOG descriptor describes a local object’s appearance and shape by using the distribution of unsigned intensity gradients or edge orientations. It is easy to understand and implement, and it is computationally efficient. However, the one disadvantage in such gradient based techniques is that intensity gradient magnitude information is very sensitive to changes in illumination. Unfortunately, in most histogram approaches, the gradient orientations are weighted by these gradient magnitudes.

GOP is an extension of the popular HOG algorithm. GOP also uses gradient information as a feature descriptor. However, in contrast to the traditional gradient based methods, GOP discards magnitude information and uses only unit signed orientations (directions) while computing the feature sets. This demonstrates a significant improvement in the performance. Furthermore, the gradient directions at different scales are combined to make a hierarchical representation.

Given an Image $I(p)$, where $p = (x, y)$ indicates the pixel locations, the pyramid of $I$ is defined as $P(I) = \{I(p; \sigma)\}_{\sigma=0}^{s}$ as

\[
I(p; 0) = I(p)
\]
Here \( \phi(p) \) is a Gaussian kernel. We use a 3*3 Gaussian kernel with standard deviation of 0.5. \( \downarrow_2 \) denotes half size down-sampling and \( s \) is the number of pyramid layers. We have set \( s = 4 \) in our experiments, thus building a 5 layered image pyramid for each input image.

The signed gradient orientation (direction) at each scale \( \sigma \) is then defined by its normalized gradient vector at each pixel.

\[
g(I(p; \sigma)) = \frac{\nabla(I(p, \sigma))}{|\nabla(I(p, \sigma))|}
\]

The Gradient Orientation Pyramid (GOP) of \( I \) is then defined as

\[
G(I) = \{g(I(p, \sigma))\}_{\sigma=0}^{s}
\]

Fig.11 illustrates GOP computation from an input image.

The advantage of using GOP features is that while computing a GOP feature, the GO’s for the entire image are computed at various scales unlike the previous two algorithms where the LBP and DCT features are extracted patch wise. Thus GOP features retain sufficient spatial and appearance based information to reflect the individual aging characteristics for an image. Another advantage of using GOP is that GOP features are more efficient compared to other commonly used global features such as AAM, AGES etc.
**GOP Feature Extraction:** In our work, we used 5 pyramid scales resulting in images of sizes 100x100, 50x50, 25x25, 13x13, and 7x7 respectively. Every image produces both x and y gradients for every pixel. Therefore the overall feature vector will be of the size 2(10,000+2500+625+169+49) = 26686. Once we obtain the feature vector for all the images, we run a PCA on the resulting 1002*26686 matrix and retain the first 150 columns.

### 2.4 Feature Fusion

Once the global and local features are extracted, we perform a feature level fusion to combine them. Given a facial image, we have three sets of feature descriptors; the shape features \( f_1 = [g_1, \ldots, g_k] \) extracted using GOP, the skin features \( f_2 = [l_1, \ldots, l_m] \) extracted using LBP and the texture features \( f_3 = [d_1, \ldots, d_n] \) extracted using DCT. In feature normalization each feature set is normalized using z score normalization defined below.

\[
    f_i' = \frac{f_i - \mu_i}{\sigma_i} \quad i = 1, 2, 3
\]

Here \( f_i \) is the \( i^{th} \) feature vector and \( \mu_i \) and \( \sigma_i \) are the corresponding mean and standard deviations of the feature vector. \( f_i' \) is the normalized \( i^{th} \) feature vector and \( i \) is the feature index. The fused feature is created by concatenating the normalized features as given below.

\[
    f_{new} = [f_1', f_2', f_3']
\]

Concatenation results in an increase in the dimension of features which increases the computation time. We thus do a dimension reduction using the Principal Component Analysis (PCA) algorithm.

\[
    f_{pca} = [f_1 \ldots f_p] \quad p < k + m + n
\]

\( f_{pca} \) is then used for age estimation.
Chapter 3
Hierarchical Age Estimation using Manual Age Grouping

In this chapter, we describe the proposed two-stage hierarchical approach towards age estimation. In the first stage, we develop a classifier that divides the images into different groups based on their ages. We call this method a Soft Classifier/Soft Labeling. We then train regression models on each of these image groups to perform detailed age estimation. We use supervised Machine Learning methods to accomplish these tasks. Supervised learning (machine learning) takes a known set of input data (training/train data) and known responses to the data (training/train labels), and seeks to build a predictor model that generates reasonable predictions (outputs/test labels) for the response to new data (test data). Fig. 12 illustrates the supervised approach.

1
Known Data
Known Responses

Model

2
Model
New Data

Predicted Responses

Figure 12: Supervised Machine Learning Approaches

Supervised learning is applicable to both classification and regression algorithms. We first provide a background on machine learning classification and regression algorithms that we have used in this work. We then describe how we apply these algorithms to construct our hierarchical system to address the issue of age estimation.
3.1 Classification Algorithms

In this section we give a detailed description of the two of the classification algorithms that have been used in our work, namely

1) Support Vector Machines (SVM)
2) K Nearest Neighbor Classification (KNN)

3.1.1 Support Vector Machines

A Support Vector Machine (SVM) is a supervised learning method that uses support vectors to build a model for classification or regression. The basic algorithm is described in [Vapnik63]. In a simple two class linear classification problem, the goal of SVM is to find an optimal hyper-plane to separate two classes as shown below in Fig. 13.

In addition to separating the two classes, the hyper-plane should have largest margin (in the sense that it should have a greatest distance from the closest point on either side). Support vector machines with large margin are preferred since they provide the best generalization. For example, in Fig. 13, the line \( H_1 \) separates the two classes with no error but the margin between the data
points and the hyper-plane is very small. Conversely, $H_2$ in addition to separating the two classes with no error also provides best (maximum) possible margin thereby giving the best generalization. An SVM finds the optimal hyper-plane by constructing an optimization problem from the data such that the margin between the two classes is maximized without any misclassification in the two classes.

In general, for practical applications it is hard to find a hyper-plane that can classify the data with full accuracy (i.e. having no misclassifications). Therefore slack variables are introduced into the SVM model. These slack variables quantify the misclassification error. In addition, another parameter $C$ called the cost parameter is also introduced to let us control the degree of misclassification we can tolerate.

The basic version of SVM only allows linear classification. This can be generalized by applying the kernel trick. The kernel trick states that data which is not linearly separable in a lower dimensional space can be made to be linearly separable in a suitable higher dimension. To achieve this we use a mapping function to transform the lower dimensional data into the higher dimension as shown in Fig.14 [Steiner10].

![Figure 14: Transformation to a Higher Dimensional Space (source: Steiner10).](image-url)
In Fig. 14, separation of the data points in the 2-D space (on the left) is quite hard and is made linearly separable only after projecting the data points to a higher dimensional space.

Because the new points can be very high-dimensional, the calculation of their coefficients and of their scalar products can be expensive. The “Kernel trick” is based on the observation that the transformed vectors only appear in the form of inner products and can thus be replaced by a function \( k(x_1, x_2) \) which behaves like a scalar product between \( x_1 \) and \( x_2 \) in the higher dimensional space while keeping the computational costs low. Such a function \( k(\ldots) \) is called a kernel function. There are four basic kernel functions that are popularly used in SVMs:

- Linear: \( k(x_i, x_j) = x_i^T x_j \)
- Polynomial: \( k(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \)
- Radial Basis Function: \( k(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2), \gamma > 0 \)
- Sigmoid: \( k(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \)

For the above mentioned kernels \( \gamma, r, d \) are the parameters to be estimated. For many real world applications RBF kernels have proven to be a reasonable choice and are often used \([Hsu03]\). The reasons why the RBF is popular is because

a) It is a non-linear kernel, i.e. it maps data points into a higher dimensional space non-linearly. Thus it can be effectively suited for most real world data where the relationship between the data and its labels is non-linear.

b) The number of hyper-parameters to be selected determines the complexity of an SVM. The RBF kernel has relatively fewer hyper parameters to be determined.

c) RBF has fewer numerical difficulties compared to the other kernels.

Generally, for most datasets the RBF kernel is a reasonable choice, however for datasets with an abnormally large number of features, typically linear kernels are used.
**Cross Validation:** There are two parameters for an RBF kernel which are not known beforehand and need to be selected by the user: the cost parameter $C$ and the hyper-parameter $\gamma$. It is not known beforehand which $C$ and $\gamma$ are best suited for a given dataset. Thus some kind of model selection (parameter search) must be done. The purpose of such a parameter selection is to identify a good $(C, \gamma)$ pair so that the classifier accurately predicts the label of the test data. We want to avoid model over fitting, thus the focus here is on achieving high test data accuracy rather than a very high training data accuracy. We achieve this goal of parameter selection using cross validation. In *k* fold cross validation, the training data is first divided into $k$ sets of equal size. The classification algorithm is iteratively run for $k$ rounds where, in each round, one set is tested using the model built on the remaining $k-1$ sets. In each of the $k$ rounds we do a grid search on $C$ and $\gamma$. Various pairs of $(C, \gamma)$ values are tried and the one with the best cross-validation accuracy is picked.

### 3.1.2 K-Nearest Neighbor Classification

The **K-Nearest Neighbor** ($k$-NN) algorithm is a non-parametric method for classifying objects based on the closest training examples in feature space. The $k$-nearest neighbor algorithm is one of the simplest of machine learning algorithms where an object is classified by a majority vote of its neighbors, with the object being assigned to the class that is most common amongst its $k$ nearest neighbors, where $k$ is typically a small positive integer.

**Algorithm Description:** The training phase for $k$-NN consists of storing all known training data points and their class labels. If we want to tune the value of '$k'$ and/or perform feature selection, the cross-validation technique described above can be used on the training dataset. The testing phase for a new instance, given the training data is as follows.

- Give a test point $x_0$, find the $k$ training samples $x_r, r = 1 \ldots k$ closest in distance to $x_0$
Classify \( x_0 \) by computing and assigning the label of the most frequent class among its \( k \) nearest neighbors to the test point.

The best choice of \( k \) depends upon the data. Generally, larger values of \( k \) reduce the effect of noise on the classification, but make boundaries between classes less distinct. In our work we have chosen the value of \( k \) to be 50.

### 3.2 Regression Algorithms

The above section described the classification algorithms that have been used in our age estimation approach. The objective of these classifiers is to first divide the training images into different groups based on their ages and then classify a test image into one of the different groups. The next step is to perform regression where, given the age group of the test data, we use a regressor corresponding to that group to perform detailed age estimation. We now take a look at the regression algorithms we have used:

3) Support Vector Regression (SVR)

4) Gaussian Process Regression (GP)

#### 3.2.1 Support Vector Regression

Support Vector Regression (SVR) was introduced by V. Vapnik et al. [Vapnik63] shortly after they introduced the notion of slack variables in SVMs. The objective of SVR is similar to SVM. In SVM, we need to find a hyper-plane that provides the best separation between data points of various classes while providing the most optimal margin for generalization. Support Vector Regression has a similar objective. The only difference is that since the goal of regression of to fit
a model to the given data points as opposed to separating them. For this reason, in SVR all training samples with an error less than $\epsilon$ are ignored thereby reducing the size of the training set.

### 3.2.2 Gaussian Processes Regression

We start this section by first discussing the standard multivariate Gaussian distribution. We then discuss Bayesian linear regression. We then introduce Gaussian Processes (GP), and then describe how GP can be used in the context of Bayesian regression for nonlinear models in the subsection on Gaussian Process Regression.

#### 3.2.2.1 The Multivariate Gaussian Distribution

The standard multivariate Gaussian (shown in Fig 15(b)) distribution is defined as follows

$$p(x | \mu, \Sigma) = N(\mu, \Sigma) = (2\pi)^{-D/2} |\Sigma|^{-1/2} \exp\left(\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$$

where $\mu$ is the mean vector and $\Sigma$ is the covariance matrix. The multivariate Gaussian distribution has the following properties

a) **Marginalization Property:** The marginal probability densities of a multivariate Gaussian follow a Gaussian distribution.

$$p(x_A) = \int p(x_A, x_B | \mu, \Sigma) \, dx_B \quad \text{and} \quad p(x_B) = \int p(x_A, x_B | \mu, \Sigma) \, dx_A$$

both are Gaussian

b) **Conditioning Property:** The conditional probability densities of a multivariate Gaussian also follow a Gaussian distribution.

$$p(x_A | x_B) = \frac{p(x_A, x_B | \mu, \Sigma)}{\int_{x_A} p(x_A, x_B | \mu, \Sigma) \, dx_A} \quad (3.2)$$

$$p(x_B | x_A) = \frac{p(x_A, x_B | \mu, \Sigma)}{\int_{x_B} p(x_A, x_B | \mu, \Sigma) \, dx_B} \quad (3.3)$$
3.2.2.2 Bayesian Linear Regression

Consider the training data $T_D = \{x_i, y_i\}_{i=1}^N$, $x_i \in R^D$, $y_i \in R$ where $x_i$ is a D-dimensional input vector of the form $\begin{bmatrix} x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix}$ and $y_i$ is the corresponding observed target value/label. The standard linear regression model is of the form

$$f(x) = x^T w, \ y_i = f(x_i) + \epsilon_i \quad i = 1 \ldots N$$  

Here $w$ corresponds to the vector of weights (parameters) of the linear model and $f$ is the function value. We assume that the observed values differ from the function values $f(x)$ by an additive noise $\epsilon$ which follows an independent identically distributed Gaussian distribution with a zero mean and variance $\sigma_n^2$

$$\epsilon \sim N(0, \sigma_n^2)$$

We first use the training data $T_D$ to find the weight vector. In a non-Bayesian approach, $w$ is computed using the maximum likelihood estimate approach. Then these weights are used to find
the label $y$, corresponding to a new test input $x$, using the equation $y_* = x^T w$. Standard linear regression gives a point estimate for the new test label.

In our thesis, we use a Bayesian framework in which we find a probability distribution for the new test label averaged over all the possible weights. This distribution is of the form $p(y_* | x_*, T_D)$ and is known as a predictive distribution. In Bayesian linear regression we first compute the posterior probability distribution for the model parameters (the weights vector) using Bayes theorem. We then average the data model of the new test point over the distribution of the weights to find the predictive distribution.

**Step I**

Compute the posterior distribution of weights using the formula

$$p(w|T_D) = \frac{p(T_D|w)p(w)}{p(T_D)}$$

(3.5)

where $p(T_D) = \int w p(T_D|w)*p(w)dw$, is the marginal likelihood, $p(w)$ is the prior and $p(T_D|w)$ is the likelihood. This is a direct application of Bayes theorem. The likelihood can be computed using the data model i.e., $p(T_D|w) = p(y|x,w)$.

Since $y = Xw + \epsilon$, where $X = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$ and $w = \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix}$, we get

$$p(y|X,w) = N(Xw, \Sigma_n),$$

(3.6)

where $\Sigma_n = \sigma_n^2 I$. We assign a Gaussian prior $p(w) = N(0, \Sigma_w)$ on the weights. In practice we don’t know $\Sigma_w$ and need to estimate it from the training data. Thus after substitution we compute the posterior as

$$p(w) = N\left(\frac{1}{\sigma_n^2}A^{-1}Xy, A^{-1}\right)$$

(3.7)

where $A = \frac{1}{\sigma_n^2}XX^T + \Sigma_w^{-1}$

**Step II**
Now we can compute the predictive probability distribution $p(y_{*}, x_{*} | T_{D})$ of a new test label by averaging the data model for the new test point over all possible values of $w$ as follows

$$p(y_{*}, x_{*} | T_{D}) = \int_{w} p(y_{*}, | x_{*}, T_{D}, w) \ast p(w | T_{D})dw$$  \hspace{1cm} (3.8)

Upon simplification and substitution, we get

$$p(y_{*}, x_{*} | T_{D}) = N \left( \frac{1}{\sigma_{n}^{2}} x_{*}^{T} A^{-1} X y, x_{*}^{T} A^{-1} x_{*} \right)$$  \hspace{1cm} (3.9)

The regression model described above is a linear model. Linear models are intuitively easy to understand and implement, however not all data can be modeled using linear models. If the relationship between the input data and its output cannot be approximated by a linear function, these models give a poor prediction performance and thus linear models have a very limited flexibility.

One way to overcome this issue is by using the principle that separation of data is easier in higher dimensions. This is similar to the kernel trick that we explained earlier. The data points are transformed into a new higher dimensional space and then the dot products are replaced with a kernel function $k(x_{1}, x_{2})$. Applying this method to Bayesian regression is very involved. However, Gaussian Processes (GP) [Rasmussen04] provide an alternative way to approach this issue. Gaussian processes provide a general framework to work with nonlinear models. Using GP, we can arrive at the exact same solution as we would get using a Bayesian linear regression with the kernel trick but in a much more elegant way. It should be noted that unlike Bayesian Linear Regression, Gaussian Process Regression is a non-parametric regression and hence can model any arbitrary of the input points.
3.2.2.3 Gaussian Processes

We first discuss the concept of a probability distribution over functions using the following example. Using this concept, we then explain what a Gaussian Process is and how it is represented. We then use the Gaussian Process to address the regression problem.

**Example:** Consider a bag $Z$ of $N$ random vectors $Z_1, \ldots, Z_N$ where each vector is a $D$-dimensional vector of random variables of the form $Z_i = \begin{bmatrix} Z_{i1} \\ \vdots \\ Z_{iD} \end{bmatrix}$. Assume that $Z$ follows a multivariate Gaussian distribution with a mean vector $\mu$ and a covariance matrix $\Sigma$. The probability distribution of $Z$ can therefore be given as

$$p(Z = z) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2} (z-\mu)^T \Sigma^{-1} (z-\mu)} \tag{3.10}$$

In the above scenario, each random vector has the same dimension which makes it possible for us to model the probability distribution of $Z$ using a multivariate Gaussian. We now extend the above bag of vectors framework to the case where each random vector $Z_i$ has a different (possibly even infinite) dimension. In such cases, since we cannot model the probability distribution of $Z$ using the above framework, we make use of Gaussian Processes to solve the problem.

A **Gaussian Process** is a collection of random variables such that any finite sub collection of random variables has a joint Gaussian distribution [Rasmussen04].

It is characterized by a mean function $m(x)$ and a covariance function $k(x; x_*)$. This is also known as the **covariance kernel**. Using these two functions we can compute the probability distribution.

We now go back to the example of drawing a vector from a bag of vectors. Note that functions with a finite domain can be interpreted as vectors, where the vector represents the functional
value of the various points in the domain. In this manner, the bag of vectors can be extended to a
bag of functions, \( F \) where \( F \) is a set of all possible mappings from a finite domain \( X \) to set of real
numbers \( R \). Assuming that \( X \) takes the form = \( \{ x_1 ... x_M \} \), we can represent \( F \) as a set of all
possible mappings \( h(.) \), and \( h(.) \) will be an \( M \) dimensional vector of the form \( h = \{ h(x_1),
h(x_2) .... h(x_M) \}^T \). Now if we specify that \( h \sim N(\mu; \sigma^2 I) \), we can compute a probability
distribution for \( F \) such that its probability density function is given by

\[
p(h) = \prod_{i=1}^{M} \frac{1}{\sqrt{2 \pi \sigma}} e^{-\frac{(h(x_i)-\mu_i)^2}{2\sigma^2}} \quad \text{(3. 11)}
\]

Similar to the bag of functions of finite domain, we can consider an example of a bag of functions
with infinite domains. For such cases, a Gaussian Process can be used to specify the distribution
over functions. The mean and the variance of the Gaussian Process can be computed by using the
mean function \( m(.) \) and \( k(.,.) \) respectively. Thus for a given a finite subset of the domain
\( x_1,x_2 .... x_M \in X \) the associated finite set of random variables (which represent the functional
values for the input points \( x_1,x_2 .... x_M \in X \) ) have the distribution

\[
\begin{bmatrix}
 h(x_1) \\
 \vdots \\
 h(x_M)
\end{bmatrix}
\sim N(\begin{bmatrix}
 m(x_1) \\
 \vdots \\
 m(x_M)
\end{bmatrix}, \begin{bmatrix}
 k(x_1, x_1) & \cdots & k(x_1, x_M) \\
 \vdots & \ddots & \vdots \\
 k(x_M, x_1) & \cdots & k(x_M, x_M)
\end{bmatrix}) \quad \text{(3. 12)}
\]

This is denoted using the notation \( h(.) \sim GP(m(.), k(.,.)) \). We now apply the concept of
probability distribution over functions in a Bayesian Regression framework to define the
Gaussian Process Regression (GPR) model.

**Gaussian Process Regression Model:** Once again, we consider the training data \( T_D = \{ x_i, y_i \}_{i=1}^{N} \), \( x_i \in R^D \); \( y_i \in R \) where \( x_i \) is a D-dimensional input vector of the form \( \begin{bmatrix} x_{i1} \\
 \vdots \\
 x_{iD} \end{bmatrix} \)

and \( y_i \) is the corresponding observed target value/label. In this case, the training data can be
either the LBP feature set, the DCT feature set, the GOP feature set or the fusion feature set. The
target value/label \( y_i \) is the age of the image \( x_i \). The regression model is of the form
where \( \varepsilon_i \) are \( i.i.d \) noise variables with independent \( N(0, \sigma^2) \). Analogous to Bayesian Linear Regression, here too, we assume a prior distribution over functions \( f(\cdot) \). In particular, we assume that \( f(\cdot) \) has a zero mean Gaussian Process prior, \( f(\cdot) \sim \text{GP}(0, k(\cdot, \cdot)) \) for some valid covariance function \( k(\cdot, \cdot) \). We also define \( y_* \) as the label corresponding to a new test input \( x_* \). Therefore, similar to the Bayesian Linear Regression scenario, given the training set \( T_d \), the prior \( p(f) \), and the new input point \( x_* \), the problem here is to compute the predictive distribution for the new test label \( p(y_* | x_*, T_d) \). In Bayesian Linear Regression, we first used Bayes’ rule in order to compute the parameter posterior \( p(w | T_d) \) which we then use to compute the predictive distribution for a new test point. The procedure to compute the predictive distribution in a Gaussian Process is simpler than the Bayesian Linear Regression process.

The Marginalization property for Multivariate Gaussian in section 3.2.1 states that for any function \( f(\cdot) \) drawn from a zero mean Gaussian Process prior with a covariance function \( k(\cdot, \cdot) \) the marginal distribution over any set of input points must have a joint multivariate Gaussian distribution. Applying this to the training and test points, we get

\[
\begin{bmatrix}
  [f] \\
  [f_*]
\end{bmatrix} | X, X_* \sim N(0, 
\begin{bmatrix}
  K(X, X) & K(X, X_*) \\
  K(X_*, X) & K(X_*, X_*)
\end{bmatrix})
\] (3.14)

where

- \( f \in \mathbb{R}^N \) such that \( f = [f(x_1), f(x_2), \ldots, f(x_N)]^T \)
- \( f_* \in \mathbb{R}^{N_*} \) such that \( f_* = [f((x_1_*), f(x_2_*), \ldots, f(x_{N_*})] \)
- \( K(X, X) \in \mathbb{R}^{N \times N} \) such that \( (K(X, X))_{ij} = k(x_i, x_j) \)
- \( K(X, X_*) \in \mathbb{R}^{N \times N_*} \) such that \( (K(X, X_*)_{ij} = k(x_i, x_{j*}) \) and so on.

Now from Equation 3.13 and the property that the sums of independent Gaussian Random variables is also a Gaussian, we have
Now using the Conditioning property for multivariate Gaussians, we get

$$\left[ \begin{array}{c} y \\ f_* \end{array} \right] | X, X_* \sim N(0, \left[ \begin{array}{cc} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{array} \right])$$ (3.15)

where

$$p(y_* | y, X, X_*) = N(\mu_*, \Sigma_*)$$ (3.16)

Thus, in this way we can compute the predictive distribution of the new test image. If we want to compute a point estimate for the age from this distribution, the mean $\mu_*$ serves as the best estimate.

### 3.2.2.4 Kernel Selection and Hyper-Parameter Estimation for Gaussian Process Regression

From the above sections it is clearly shown that Gaussian Process regression is a non-parametric regression method. Unlike the Linear Bayesian regression model, where we must first compute a posterior distribution for its parameter $w$ in order to find the predictive distribution for the test label, in Gaussian Process regression we compute the predictive distribution by directly evaluating the mean $\mu_*$ and covariance $\Sigma_*$ of the Gaussian distribution. In order to compute these it is important to select the correct covariance kernel. The correct covariance kernel is a function/kernel which gives rise to a valid Gaussian Process.

In general any real valued function $m(\cdot)$ is acceptable but for $k(\cdot, \cdot)$ the rule is that for any set of elements $x_1, x_2, \ldots, x_M \in X$, the resulting matrix $K = \left[ \begin{array}{ccc} k(x_1, x_1) & \cdots & k(x_1, x_M) \\ \vdots & \ddots & \vdots \\ k(x_M, x_1) & \cdots & k(x_M, x_M) \end{array} \right]$ must be a valid covariance matrix corresponding to some multivariate Gaussian distribution.
According to probability theory this property holds true if $K$ is positive semi-definite.

We have chosen the kernel function $k(.,.)$ to be the **Squared-Exponential kernel** which is of the form

$$k(x,x_*) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}||x-x_*||^2\right) \quad (3.19)$$

This is a very commonly used kernel in many applications. Here $\sigma_f^2, l$ are known as the hyperparameters of the Gaussian Process since in order to evaluate the covariance kernel, we need to determine these values. It is very important to interpret the meaning of these hyper parameters in order to understand how the data behaves. Here $\sigma_f^2$ is the maximum allowable covariance. Intuitively this parameter is high for functions whose observed labels show a high amount of variation. The hyper-parameter $l$ is the bandwidth parameter and determines the amount of correlation between two input points $x, x_*$. Input points that are farther apart from each other will have a higher correlation for higher values of $l$ than for lower values of $l$.

The set of hyper-parameters whose values need to be determined in order to solve the Gaussian Process regression problem are

$$\theta = \{\sigma_n^2, \sigma_f^2, l\}$$

In order to make inferences about all the hyper parameters, we compute the probability of the data given the hyper-parameters.

$$L = \log p(y|x, \theta)$$

Due to our assumption that the data follows a Gaussian distribution, we can simplify the above equation to

$$L = -\frac{1}{2} \log |\Sigma| - \frac{1}{2} \log (y - \mu)^T \Sigma^{-1} (y - \mu) - \frac{n}{2} \log(2\pi) \quad (3.20)$$

In our case $\mu = \bar{0}$ and $\Sigma = K(X,X) + \sigma_n^2 I$. This quantity is called log marginal likelihood [Rasmussen04]. Now, the hyper-parameter values are those values that optimize the log
marginal likelihood based on its partial derivatives. This process is known as evidence maximization.

3.3 Hierarchical Estimation

We now describe the proposed hierarchical age estimation algorithm for age estimation using the classification and regression techniques we described in the above sections. The principle behind our age estimation algorithm is to partition the training data into several groups, and build one regressor for each group. The motivation for this approach is that the aging pattern in humans is different during different ages. In younger years, shape is a crucial factor (geometric features such as distance ratios between facial features change noticeably during childhood than adulthood) whereas in older years, texture becomes important (wrinkles are found in adulthood rather than childhood). Therefore if facial features are trained on the whole dataset, these characteristics are not incorporated into the training model when estimating the age of a new image. Keeping these points in mind, the designed hierarchical age estimation system uses a two stage coarse-to-fine estimation technique.

We first divide the training into different groups based on ages. In our work, we experimented with two types of age group partitions:

1) Partition the data into three groups:
   a. Group A = \{Image I \mid \text{age (I) <= 10 years}\}
   b. Group B = \{ Image I \mid 11 \text{ years < = age (I) <= 20 years}\}
   c. Group C = \{ Image I \mid \text{age (I) > 20 years}\}

2) Partition the data into four groups.
   a. Group A = \{Image I \mid \text{age (I) <= 10 years}\}
   b. Group B = \{ Image I \mid 11 \text{ years <= age (I) <= 20 years}\}
c. Group C = \{ \text{Image } I \mid 20 \text{ years} \leq \text{age (I)} \leq 40 \text{ years} \}

d. Group C = \{ \text{Image } I \mid \text{age (I)} > 40 \text{ years} \}

Once we partition the images into groups, we build a regression model for each group using only the images in that group. Now, when a new test image comes in, the hierarchical age estimator uses the following steps to compute its estimated age:

- Use the classification algorithm (SVM/k-NN) to classify the test image into one of the age groups.
- Apply the regressor (SVR/GPR) corresponding to the group of the test image to determine its actual age.

Typically such a classification system is known as **Hard Classification**. Although the hierarchical hard-classification – regression system can improve the performance compared to single stage age estimation system, it has the major **drawback** that errors occurring in the age group classification step are propagated in the regression step. This means that if an input test image is misclassified during the age group classification, there will be a large increase in the estimation error. Such misclassifications are especially possible for boundary images, e.g. an image of a 19 year old person is likely to be classified into age group 21-40 instead of the 11-20. Thus to reduce the effect of such a misclassification, we propose a new classification method called **Soft Classification** / Soft Label Assignment. In this method we assign soft labels to each test image and then estimate the age by applying all three regression models on the image and combining them together according to the soft labels produced. The Soft Classification approach is described in detail in the section below.

### 3.3.1 Soft Label Assignment

We saw in the above section that in hard classification, the test image is first assigned to one of the age groups and once its age group has been determined, its age is computed using the
regressor for only that specific group. Soft Classification, in contrast, computes the likelihood/probability of the test image belonging to a particular group, for all the age groups. We call this the **group probability**. Next it computes the ages of the test image as computed by each age group. The estimated age of the image is computed by taking a weighted average over all of the computed ages, where the group probabilities serve as the weights. Soft classification can be mathematically represented as

\[
Age\ (TI) = \sum_{k=1}^{N} \pi_k * Age_k(TI)
\]

where \(TI\): Test Image ; \(k\): group number ; \(\pi_k\): probability of the test image belonging to the \(k^{th}\) group and \(Age_k(TI)\): Age of the test image computed by the \(k^{th}\) group.

We show an example of computing the group probabilities for the kNN algorithm. When a new test image is considered, we compute the distances between the test image and all the training images. We then select k nearest neighbors corresponding to the test image. We compute the soft labels for this image as follows

\[
\pi_i = \frac{\#\ Images\ belonging\ to\ Group\ i}{k}
\]

Similar computations can be done within the SVM framework too to obtain the soft probabilities for each group. The flow chart for our Hierarchical age estimation system with soft classification is shown below in Fig 16.
To summarize this chapter, we first discussed at length the classification and regression algorithms that were implemented. We then built a two stage age estimation system consisting of age-group determination followed by a group specific regression. To avoid the drawbacks of a hard classification system we proposed a soft labeling approach to computing the estimated age.

The important things that we would like to highlight here are

1) The regression hyper parameters can be learned in a more systematic way for GP regression (evidence maximization) than for SVR (cross validation)

2) The classification step in our hierarchical system groups the training images solely based on their ages and does not take their appearance into account.

As mentioned in the previous chapters, studies have shown that similarity in appearance plays an important role in the aging process. Thus taking into consideration these two points, we also have developed a hybrid age + appearance based grouping system, which we describe in the next chapter.
Chapter 4

Hierarchical Age Estimation via Automatic Grouping using an Expectation
Maximization based Approach

In the previous chapter, we described a hierarchical soft labeling approach for age estimation where we first used a suitable classification algorithm to compute the group probabilities for each test image for a set of predefined age groups. We then used regression to compute the ages of the test image inside each group and estimated the overall age for the test image by combining these ages with the group soft probabilities obtained in previous step. We will show in Chapter 5 that such a soft labeling approach outperforms the traditional hard classification that has been used in the literature so far. We can think of this approach as an age–based grouping, where we group the training data according to their ages and develop models for each age-group.

Studies have shown that people who look similar will tend to age in a similar way [Lanitis02]. The age-group based grouping described above, while doing an overall good job, does not take this observation into account. Therefore, one idea to incorporate this into our system is to do an appearance based grouping to group training images that look similar or that tend to exhibit similar aging patterns, and train individual classifiers for these different image clusters. We developed an Expectation-Maximization (EM) algorithm based framework to address this issue of grouping based on appearance. We would like to highlight the fact this EM based approach automatically groups the training data by learning the underlying image similarities, unlike the previous approach where we have to group the training data manually into groups based on the subject ages. Another point to note is that the hyper-parameters used in Gaussian Process Regression can be learned for each group within the proposed EM framework.
We now provide a description of the general EM algorithm and then show how we use it to address our problem of age estimation.

4.1 The Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a popular algorithm in machine learning, computer vision and statistics, used for maximum likelihood parameter estimation when some of the random values involved are not observed. The EM algorithm formalizes an intuitive idea for obtaining the estimates of the parameters when some of the data is missing by iterating through the following steps:

- Replace missing values with estimated values
- Estimate the parameters to maximize a likelihood function.
- Repeat until convergence is obtained
  - Step (a) using the estimated parameter values as the true values.
  - Step (b) using estimated values as the ‘observed’ values.

Thus in the context of finding the maximum likelihood parameter estimate, the way EM works is as follows

**E-Step:** In this step, given the observed variables and the current parameter set, compute the expectation of the log-likelihood function of missing values with respect to the conditional distribution of the hidden variables given the observed variables and current parameter set.

**M-Step:** Given the expected log-likelihood, this step computes a new set of parameters that will maximize this likelihood.

To provide a mathematical description the algorithm, consider a set of observed data $X$, missing (hidden) values $Z$ and the unknown parameter values $\theta$. The likelihood function is given by

$$L(\theta ; X, Z) = p(X, Z | \theta)$$  \hspace{1cm} (4.1)
The maximum likelihood estimate (MLE) of the unknown parameters is then given by

\[
\text{MLE} = \arg_{\theta} \max(\log(p(X, Z | \theta)))
\]  \hspace{1cm} (4.2)

Finding the MLE is hard because we don’t know the missing values $Z$. Therefore we use the EM algorithm to iteratively maximize the log likelihood function through a step of Expectation (E) and Maximization (M) steps.

First the algorithm starts with an initial assumption of parameter values $\theta^0$ for $\theta$.

**Expectation step (E-step):** The E-Step is performed to calculate the *expected value of the log-likelihood function* under the conditional distribution of the hidden variables $Z$ given the observations $X$ and the current set of parameters $\theta^t$ as shown below in equation

\[
Q(\theta | \theta^t) = E_{Z|X, \theta^t}[\log L(\theta ; X, Z)]
\]  \hspace{1cm} (4.3)

**Maximization step (M-step):** Find the new set of parameter values $(\theta^{t+1})$ that maximize the quantity $Q(\theta | \theta^t)$

\[
\theta^{t+1} = \arg \max( Q(\theta | \theta^t))
\]  \hspace{1cm} (4.4)

It has been shown that using this algorithm guarantees an increase of the likelihood function of the observed data with each successive iteration. The algorithm is typically run until convergence.

### 4.2 EM Algorithm based Framework for Hierarchical Age Estimation

As we have stated in our introduction, our goal is to categorize the training data into different groups based on their appearance and learn the hyper-parameters for each group. Assume that we want to divide the data into $M$ groups, where $M$ is known. Here our training data \( \{x_n, n = 1..N\} \) along with the training labels \( \{y_n, n = 1..N\} \) correspond to the known observations $X$
and the hyper parameters that have to be learned are our unknown parameter values $\theta$. In this case, the training data can be either the LBP feature set, the DCT feature set, the GOP feature set or the fusion feature set. The training values/labels $y_i$ are the ages of the images $x_i$. The hyper parameters are the hyper-parameters corresponding to the GP regressor for each group.

We introduce a new variable $z_{nm}$ that is defined as

$$z_{nm} = \begin{cases} 1, & x_n \in \text{group } m \\ 0, & x_n \notin \text{group } m \end{cases}$$

The variable $z_{nm}$ indicates whether or not the $nth$ point belongs to group $m$. For example, if $z_{11} = 1$ and $z_{22} = 1$, it means that the first training point belongs to group one, and the second training point belongs to group 2, respectively. Here the indicator variables $z_{nm}$ correspond to the missing values $Z$ that we defined while describing the EM framework, since for every point in the training data we don’t know to which group it belongs.

Since each training point can belong to only one group, it can be seen that

$$\sum_{m=1}^{M} z_{nm} = 1 \quad \forall \ n = 1,2, \ldots, N$$

If we know the variables $z_{nm}$ then we can find the hyper parameters of the $mth$ group using the evidence maximization approach we described in Chapter 3. Since the variables $z_{nm}$ are not known we use EM iterations:

**E Step:**

In the E-Step, in order to compute the expected value of the log likelihood function, we should first compute the conditional distribution of the hidden variables given the training data and the current parameter set $p(Z | \theta, X)$. Let $\theta_m = \{ \sigma_n^2, \sigma_f^2, l \}$ be the set of hyper parameters for the $mth$ group, and let $\theta = \{ \theta_1, \theta_2, \ldots, \theta_M \}$ be the set of all the parameters. If we represent the current parameter set using $\theta^t$, then the desired conditional distribution is given as
Using Bayes theorem the condition distribution in equation 4.5 can be re-written as

$$ p(z_{nm} = 1 \mid x_n, y_n, \theta^t_m) . $$

(4.5)

Using Bayes theorem the condition distribution in equation 4.5 can be re-written as

$$ p(z_{nm} = 1 \mid x_n, y_n, \theta^t_m) = \frac{p(y_n \mid z_{nm} = 1, \theta^t_m, x_n) p(z_{nm} = 1)}{\sum_{k=0}^{M} p(y_n \mid z_{nm} = k, \theta^t_m, x_n) p(z_{nm} = k)} $$

(4.6)

Let $p(z_{nm} = 1 \mid x_n, y_n, \theta^t_m)$ be denoted by $w_{nm}$. Here $[w_{nm}]$ for all $n = 1, ..., N$ and $m = 1, 2, ..., M$, is known as a membership weights matrix and each point $w_{nm}$ denotes the probability of the $nth$ data point belonging to the $mth$ group. Given the membership weights, using equation 4.3 the Q-function (expectation of the log likelihood function) $Q(\theta \mid \theta^t)$, can be evaluated as

$$ Q(\theta \mid \theta^t) = E_{(x,y,\theta^t)}[\log p(x, y, z \mid \theta)] $$

(4.7)

Since the training variables $y_n$, $n = 1, 2, ..., N$ are not independent of each other, the number of terms in the summation of the expectation is exponential ($M^N$) and therefore evaluating the expectation becomes computationally very complex and hard for this case. Therefore we used an approximation approach proposed in [Yang11]. In the E-Step, after obtaining memberships weights for all the training points, we assign each training point to the group with the largest membership weight. In the M-step only the assigned points are used for finding the parameters corresponding to each group.

**M Step**

In the M-step, using the evidence maximization method for finding the hyper-parameters of a GP, we evaluate the likelihood function for each of the M groups as follows

$$ L_m = -\frac{1}{2} \log |\Sigma_m| - \frac{1}{2} (y_m - \mu_m)^T \Sigma^{-1}_m (y_m - \mu_m) - \frac{n}{2} \log (2\pi) $$

(4.10)

Where $y_m$ denotes the set of all the training labels belonging to the $m^{th}$ group $\mu_m$ is 0 and $\Sigma_m$ is the covariance function evaluated using only the points from the $m^{th}$ group. After finding the hyper-parameters we re-compute the membership weights. The overall algorithm is summarized below.
**Initialization**

Parameter initialization plays a very important role in the EM algorithm. In our work we initialize the prior \( p(z_{nm}) \) to be a uniform distribution \( \forall \ m \). In order to initialize the hyper-parameters, we first group the training data into groups based on their ages as described in chapter 3. We then compute the hyper-parameter set for each group which we input to the EM algorithm as the initial set of hyper-parameters. Such an initialization of the training data helps us ensure that the model performs both age based and appearance based grouping.

**Algorithm**

1) Initialize \( \theta^0 \) and the group priors \( p(z_{nm}) \)

2) Repeat until convergence

   a. Compute \( w_{nm}^t \) \( n = 1 \ldots N; m = 1 \ldots M \)

   b. Find the groups by assigning the point \( n \) to the group which has the highest membership weight \( w_{nm}^t \)

   c. Find the current parameter set \( \theta^t \) for each of the \( M \) groups.

Thus upon convergence, the training data have been grouped into clusters. These are adapted in such a way that they take into account both the age as well as the appearance of the image. *When a new test point comes, its age is calculated using the SVM based soft classification into one of the clusters followed by Gaussian Process regression we described in the previous chapter.* To recap, the hyper-parameters of the regressors for each cluster are already learned in the EM framework.
The block diagram of our EM framework is shown below in Fig 17.
Chapter 5

Experiments

In this chapter, we describe our experiments and the results obtained using various machine learning approaches described earlier. As mentioned earlier, face registration using automatically extracted eye locations during the pre-processing is an important step for the estimation task. In order to evaluate this step, we have also provided results obtained using manually annotated eye locations. First, we provide a description of the database we used for the analysis.

5.1 FGNET aging Database

The FG-NET Aging Database (Face and Gesture Recognition Network) is an aging database that has 1002 images corresponding to 82 different persons (6-18 images per person) in the age range of 0-69. The database also provides 68 landmark features, which are manually identified, for each image. Of these landmark points we have used the eye co-ordinate locations to quantify the performance of our automatic eye coordinate detection algorithm for face alignment. Other meta-information that has been provided for each image include: image size, age, gender, spectacles, hat, mustache, beard, horizontal pose and vertical pose. In our work we use only the age information. The FGNET database has a lot of variations in head pose, illumination and facial expression since the images are taken from real-life albums of different people. The data distributions of the FG-NET database according to age are shown below.
5.2 Results

5.2.1 Eye Localization Performance

In this section we compare the results obtained using our eye localization method with the ground truth provided along with the FGNET data base. Overall, our localization method was able to locate the eyes for 996 images (out of 1002) in the data base. We consider that the algorithm has localized eye locations if it can find at least one pair that satisfies all the geometrical conditions that we described in Chapter 2. For some images, the resolution was very poor and hence the algorithm could not locate the eyes. An example of one such image is shown below.
For the images in which our algorithm found the eye locations, we used the Euclidean distance between the actual and the estimated eye co-ordinate locations for both the eyes, and we used the average of these two values as an error metric. Here we normalized the distance measures with respect to the image sizes.

\[
\text{Error} = \frac{[(\text{error})_{\text{left-eye}} + (\text{error})_{\text{right-eye}}]}{2}
\]

\[
(\text{error})_{\text{left-eye}} = \text{norm}\left(\frac{\text{error}_{x-left}}{\text{imsize}_x}, \frac{\text{error}_{y-left}}{\text{imsize}_y}\right)
\]

\[
(\text{error})_{\text{right-eye}} = \text{norm}\left(\frac{\text{error}_{x-right}}{\text{imsize}_x}, \frac{\text{error}_{y-right}}{\text{imsize}_y}\right)
\]

Using the error metric defined above, we plot the histogram of normalized mean error for eye-coordinate extraction using our automatic eye location method below.
As it can be seen, our algorithm could find the eye locations within an average error of 10% for 70% of the images and within an average error of 20% for 80% of the images.

5.2.2 Age Estimation Performance

We used ‘Leave one person out’ (LOPO) testing where we train the models on the images of all the persons but one, and test the model on that one person. This process is repeated for all the persons in the database and the absolute error in the age obtained for each person is averaged. This metric is called as the Mean Absolute Error (MAE).

Below we report the MAE for several scenarios. We consider two kinds of preprocessed images:

1) Manually pre-processed: Face registration is done using the eye-coordinate locations given in the landmark points.
2) **Automatically preprocessing:** Face registration is done by using the eye coordinates detected by our algorithm.

We first manually preprocess the images in the databases to test all our algorithms. We then apply automatic preprocessing to the images and run the algorithms that performed the best on the manually preprocessed images.

In Table 2, we provide the MAE obtained using single stage estimation for different feature sets. The results are provided for both SVM and GP regression.

<table>
<thead>
<tr>
<th>Feature-Set</th>
<th>GPR</th>
<th>SVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT</td>
<td>6.56</td>
<td>6.34</td>
</tr>
<tr>
<td>LBP</td>
<td>7.83</td>
<td>7.74</td>
</tr>
<tr>
<td>GOP</td>
<td>5.92</td>
<td>5.85</td>
</tr>
</tbody>
</table>

Table 2: MAE for single level age estimation using GP regression

In Table 3, we report the MAE obtained using hierarchical age estimation with manual classification, which means that the test point is first manually classified into one of the groups. We have reported the MAE’s for two cases: a) when the training data is split into 3 groups of ages 0-10, 11-20, 21 and above and b) when the training data is split into 4 groups of ages 0-10, 11-20, 21-40 as explained in Chapter 3. For each of these two cases, we fist develop the regressors for each group using GP regression. Once the group of the test point is determined, the age is estimated using corresponding within group regressor.
Table 3 MAE for hierarchical age estimation with manual classification and GP regression

<table>
<thead>
<tr>
<th>Feature-Set</th>
<th>Number of Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>DCT</td>
<td>3.59</td>
</tr>
<tr>
<td>LBP</td>
<td>3.68</td>
</tr>
<tr>
<td>GOP</td>
<td>3.53</td>
</tr>
</tbody>
</table>

From the above results it is clearly seen that hierarchical age estimation outperforms single level age estimation. This is because of its ability to consider the difference in age features for different age groups, which is lacking in a single level estimation. Thus if we have a powerful classifier to classify the test images into one of the groups accurately, hierarchical age estimation will give a very good performance.

Another important point to note is that, for our data set, the performance of GP regression almost matched SVR. Apart from giving an almost similar MAE, GP is superior to SVR for the following reasons.

1) With GP regression, the kernel hyper-parameters (length-scale, noise level, etc.) can be learned via evidence maximization which is a more systematic approach than the cross validation technique used for SVMs. Also cross validation is extremely laborious and computationally intensive whereas the optimized evidence maximization is computationally more efficient than cross validation.
2) Unlike SVM, GP provides a full probabilistic prediction (predictive distribution) for the age by providing the standard deviation along with the mean estimate of the age, thereby giving an estimate of uncertainty in the prediction.

3) In addition, GPs can be easily extended and incorporated into a hierarchical Bayesian model. This property of GP is particularly useful to us and lets us nicely integrate hyper-parameter learning into the EM framework for automatic training.

For these reasons, we choose GP regression as our regressor and henceforth evaluate our further set of experiments using only GP for regression.

Next we provide the results for hierarchical age estimation using SVM and kNN classifiers to classify test cases into 4 age groups followed by GP regression for subsequent age estimation. We provide the MAE using different feature sets. We also provide the MAE obtained for both hard and soft classification.

In Table 4, the MAE obtained using a kNN classifier and a GP regressor is tabulated, and in Table 5, the MAE obtained using SVM classifier and a GP regressor is tabulated.

<table>
<thead>
<tr>
<th>Feature -Set</th>
<th>MAE (years)</th>
<th>Soft Classification</th>
<th>Hard Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCT</td>
<td>8.03</td>
<td>8.25</td>
<td></td>
</tr>
<tr>
<td>LBP</td>
<td>8.65</td>
<td>7.36</td>
<td></td>
</tr>
<tr>
<td>GOP</td>
<td>6.79</td>
<td>7.24</td>
<td></td>
</tr>
</tbody>
</table>

Table 4 MAE of hierarchical age estimation using kNN classification and GP regression
It can be seen from Tables 4 and 5 that SVM provides a better performance compared to that of kNN. Though k-NN is a relatively simple algorithm to understand implement, it is unable to accurately classify the images in the FG-NET database based solely on the Euclidean distance between features due to lack of sufficient training data, thereby leading to lower classification accuracy as compared to that of SVM.

We tabulated the classification accuracies in Table 6. From now on we use SVM classifier to classify the test point.
It can also be seen from the above results that GOP features give the best performance among the three feature sets. Thus local features by themselves are not robust enough to discriminate among individual ages and need to be combined with global features to estimate age at an individual level.

Next, in Table 7, we show the MAE obtained using hierarchical age estimation with various feature combinations.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Classification Accuracy (Percentage)</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>kNN</td>
<td>SVM</td>
</tr>
<tr>
<td>DCT</td>
<td>52.8</td>
<td>62.7</td>
</tr>
<tr>
<td>LBP</td>
<td>57.9</td>
<td>62.6</td>
</tr>
<tr>
<td>GOP</td>
<td>56.7</td>
<td>65.2</td>
</tr>
</tbody>
</table>

Table 6 Classification accuracies for kNN and SVM classifiers using different feature sets

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Classification</th>
<th>Regression</th>
<th>Soft</th>
<th>Hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP</td>
<td></td>
<td>GOP</td>
<td>6.32</td>
<td>6.56</td>
</tr>
<tr>
<td>DCT</td>
<td></td>
<td>GOP</td>
<td>6.16</td>
<td>6.23</td>
</tr>
<tr>
<td>LBP + DCT</td>
<td></td>
<td>GOP</td>
<td>5.59</td>
<td>5.90</td>
</tr>
<tr>
<td>LBP + DCT + GOP</td>
<td></td>
<td>LBP + DCT + GOP</td>
<td>5.35</td>
<td>5.76</td>
</tr>
</tbody>
</table>

Table 7 MAE for hierarchical estimation using feature fusion
It can be seen from the table above that feature fusion improved the performance compared to using just a single feature. Thus the most robust feature sets are a combination of both local and global features.

In Table 8, we tabulate MAE obtained using 3 groups for both GOP feature set and also the combined feature sets.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Classification (SVM)</th>
<th>Regression (GP)</th>
<th>MAE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>GOP</strong></td>
<td><strong>GOP</strong></td>
<td>5.42</td>
<td>5.92</td>
</tr>
<tr>
<td><strong>LBP + DCT+GOP</strong></td>
<td><strong>LBP + DCT+GOP</strong></td>
<td><strong>5.30</strong></td>
<td></td>
<td><strong>5.77</strong></td>
</tr>
</tbody>
</table>

Table 8 MAE for 3 groups using GOP and combined features

It can be seen that using 3 groups produced a lower MAE compared to the MAE obtained using 4 groups. This is because when we are using 4 groups, there was not enough training data for the classifier to accurately classify a test point between the third and the fourth group. This affects the performance of the classifier and hence the MAE increased when we used four groups.

In Table 9, we provide MAE obtained using the proposed EM algorithm for automatically grouping the training data, followed by SVM classification and GP regression for the test data. We tabulated the MAE obtained using GOP features and the combined features for 3 groups.
It can be seen that the proposed EM algorithm performed better than the hierarchical approach. This is because in EM we are grouping the data based on both age and appearance, and thus the performance improved.

Finally, in Table 10, we provide the MAE obtained for both the hierarchical method and the EM with 3 groups for the automatically preprocessed images using combined features.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Soft</td>
</tr>
<tr>
<td>Hierarchical Approach</td>
<td>7.69</td>
</tr>
<tr>
<td>EM - Framework</td>
<td>7.51</td>
</tr>
</tbody>
</table>

Table 10 MAE obtained for the hierarchical and the EM methods with automatically preprocessed images
It can be seen that using Automatic Eye Detection deteriorated the performance of the estimation algorithm. This degradation is understandable since the eye localization we used was not perfect.

In Table 11, we provide the MAE per each group for both the hierarchical and the EM based estimation algorithm using manually and automatically preprocessed images. The features set used here is the fusion feature set (LBP + DCT + GOP) is used, and the classification and the regression are performed using SVM and GP, respectively.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Group 1 (&lt;= 10)</th>
<th>Group 2 (&gt;10 &amp; &lt;=20)</th>
<th>Group 3 (&gt;= 20)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Soft</td>
<td>Hard</td>
<td>Soft</td>
</tr>
<tr>
<td>Hierarchical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Manual)</td>
<td>3.08</td>
<td>2.67</td>
<td>4.06</td>
</tr>
<tr>
<td>EM (Manual)</td>
<td>2.45</td>
<td>2.60</td>
<td>4.53</td>
</tr>
<tr>
<td>Hierarchical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Automatic)</td>
<td>6.4</td>
<td>4.99</td>
<td>4.61</td>
</tr>
<tr>
<td>EM (Automatic)</td>
<td>6.08</td>
<td>5.78</td>
<td>4.54</td>
</tr>
</tbody>
</table>

Table 11 MAE per group for various algorithms

It can be seen that the performance of the estimation algorithm was worse for the third group which consists of images of people with age greater than 20. One way to improve the estimation
performance in this group is to collect more images of people belonging to this group and use them for training.

In Fig. 21, we plot the cumulative scores of the proposed EM and the hierarchical approaches for age estimation for 3 three groups for fusion features. Cumulative score at a level $l$ represents the number of images in the data base for which the MAE is less than $l$, expressed as a percentage of the total number of images in the database. It is defined by the following equation:

$$CS(l) = 100 \times \frac{N_{\text{MAE} < l}}{N}$$

We plot the graphs for both manually preprocessed and automatically preprocessed images. It can be seen that automatic grouping of training data using EM framework outperformed the other methods.
Chapter 6

Conclusions and Future Work

In this thesis, we developed algorithms for automatic age estimation from facial images. Following are the major contributions and the summary of the thesis.

- We developed a simple eye localization algorithm that can be useful for finding the eye coordinates in facial images. The eye localization algorithm can be used to scale and rotate all the images. We have shown later on that eye coordinate location is an important step in the preprocessing since the performance of the estimation methods depend on the accuracy of the localization.

- We proposed the use of both local and global features for the estimation by combining the features after z-score normalization. For the local features, we used the DCT and the LBP. For global features, we used GOP. The best performance was achieved when a combination of these features were used together.

- We proposed a hierarchical age estimation approach in which we first obtain an estimate of the age group of a new test image, and then estimate the accurate age by using only the training images belonging to the estimated age group. Further, we showed that instead of hard classifying the test image into one of the groups, the performance of the estimation can be improved by first computing the probability that a test image belongs to one of groups, and then combining the estimates obtained using images from individual groups according to these probabilities. In this work, we used Gaussian Process regression algorithm.
• We then proposed an Expectation Maximization framework to categorize the data into groups based on both age and appearance, and to learn the hyper parameters of the Gaussian Processes to be used on these groups. This approach, when used with the combination of local and global features provided the best mean absolute error in age estimation.

**Future Work:** The work in this thesis can be extended in several ways. First, as we showed, the performance of the age estimation algorithms is very sensitive to the preprocessing. Thus, there is a need to build better eye localization system to implement the image preprocessing. Second, the MAE obtained was significantly higher in the group with images of older people. This is because the number of images for training the classifier and the regressor for this group are low. Therefore, there is a need to collect training data of such images. Third, the performance of the age estimation for older people can potentially improve by considering a different set of features such as Gabor wavelet coefficients that can represent texture information for wrinkles.
Bibliography


Wang10 Yao Wang, “DCT and Transform Coding,” *Class Notes for Multimedia and Communication System – I (EE3414)*, Polytechnic University, Available Online:

