FACIAL FEATURE POINT TRACKING BASED ON A GRAPHICAL MODEL FRAMEWORK

by

Serhan COŞAR

Submitted to the Graduate School of Engineering and Natural Sciences in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
January 2008

FACIAL FEATURE POINT TRACKING BASED ON A GRAPHICAL MODEL FRAMEWORK

APPROVED BY:

Assistant Prof. Dr. Müjdat Çetin (Thesis Supervisor)	______________
Prof. Dr. Aytül Erçil (Thesis Co-Supervisor)	______________
Associate Prof. Dr. Berrin Yanıkoğlu	______________
Associate Prof. Dr. Mustafa Ünel	______________
Assistant Prof. Dr. Hakan Erdoğan	______________

DATE OF APPROVAL: ______________

ACKNOWLEDGEMENTS

I would like to thank my advisors Assist. Prof. Müjdat Çetin and Prof. Aytül Erçil for their guidance, encouragement, understanding throughout this study and also for providing the opportunity, the motivation and the resources for this research to be done. I also owe many thanks to Assoc. Prof. Mustafa Ünel for his endless help and kindness.

I am also grateful to Assist. Prof. Hakan Erdoğan and Assoc. Prof. Berrin Yanıkoğlu for their participation in my thesis committee.

I would like to thank my friends Batu Akan, Ali Özgür Argunşah, Ali Baran Çürüklü and the other members of the VPALAB for listening me kindly any time and for helping me to tackle the problems I had during this thesis.

Special thanks to Pınar Karagöz and Emrecan Çökelek for all the encouragement, support, and the motivation they has provided throughout the thesis.

Finally, I would like to thank to my father and mother, Orhan Coşar and Aysun Coşar, for giving birth to me, for raising me and encouraging me to go on in the academic life. Although they do not understand technically what I am doing, I would like to thank them for giving me support and motivation they gave throughout this thesis.

January 2008 Serhan COŞAR

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

FACIAL FEATURE POINT TRACKING BASED ON A GRAPHICAL MODEL FRAMEWORK

Serhan COŞAR

Electronics Engineering and Computer Science, MS Thesis, 2008

Thesis Supervisor: Assistant Prof. Dr. Müjdat Çetin

Thesis Co-Supervisor: Prof. Dr. Aytül Erçil

Keywords: Facial feature tracking, human-computer interaction, facial expression analysis, graphical models

In this thesis a facial feature point tracker that can be used in applications such as human-computer interfaces, facial expression analysis systems, driver fatigue detection systems, etc. is proposed. The proposed tracker is based on a graphical model framework. The position of the facial features are tracked through video streams by incorporating statistical relations in time and the spatial relations between feature points. In many application areas, including those mentioned above, tracking is a key intermediate step that has a significant effect on the overall system performance. For this reason, a good practical tracking algorithm should take into account real-world phenomena such as arbitrary head movements and occlusions. Many existing algorithms track each feature point independently, and do not properly handle occlusions. This causes drifts in the case of arbitrary head movements and occlusions. By exploiting the spatial relationships between feature points, the proposed method provides robustness in a number of scenarios, including e.g. various head movements. To prevent drifts because of occlusions, a Gabor feature based occlusion detector is developed and used in the proposed method.

The performance of the proposed tracker has been evaluated on real video data under various conditions. These conditions include occluded facial gestures, low video resolution, illumination changes in the scene, in-plane head motion, and out-of-plane head motion. The proposed method has also been tested on videos recorded in a vehicle environment, in order to evaluate its performance in a practical setting. Given these results it can be concluded that the proposed method provides a general promising framework for facial feature tracking. It is a robust tracker for facial expression sequences in which there are occlusions and arbitrary head movements. The results in the vehicle environment suggest that the proposed method has the potential to be useful for tasks such as driver behavior analysis or driver fatigue detection.

ÖZET

GRAFİKSEL MODEL TABANLI YÜZ ÖZNİTELİK NOKTA TAKİBİ

Serhan COŞAR

Elektronik Mühendisliği ve Bilgisayar Bilimleri, Yüksek Lisans Tezi, 2008

Tez Danışmanı: Yard. Doç. Dr. Müjdat Çetin

Yardımcı Tez Danışmanı: Prof. Dr. Aytül Erçil

Anahtar Sözcükler: Yüz öznitelik takibi, insan-bilgisayar arabirimi, yüz ifade analizi, grafiksel modeller

Bu tezde önerilen yöntem, insan-bilgisayar arayüzü, mimik analiz sistemleri, sürücü yorgunluğu algımalara sistemlerinde kullanılabilecek bir yüz öznitelik takip yöntemidir. Önerilen yöntemin tabanı grafiksel modellere dayanmaktadır. Zamandaki istatistiksel ilişkiler ve öznitelikler arasındaki uzamsal ilişkiler kullanılarak öznitelik noktalarının yerleri takip edilmektedir. Yukarıda bahsedilen bir çok uygulama alanında takip, bütün sistemin başarımını etkileyebilecek kadar önemli bir role sahiptir. Bu nedenle iyi bir takip algoritması gerçekte meydana gelebilecek şartları (keyfi kafa hareketleri ve yüz üzerinde oluşabilecek engeller) göz önüne almalıdır. Varolan bir çok yöntemde öznitelik noktaları ayrı ayrı takip edilmekte ve engel olma durumu düzgün şekilde ele alınmamaktadır. Bu, keyfi kafa hareketlerinde ve engel durumlarında kaymalara sebep olmaktadır. Burada önerilen yöntem öznitelikler arasındaki uzamsal bilgiyi de hesaba katarak bu tarz durumlarda gürbüzlük sağlamaktadır. Engeller yüzünden oluşabilecek kaymalar Gabor öznitelikleri üzerine kurulu bir engel algılayıcı ile düzgün bir şekilde bertaraf edilebilmektedir.

Önerilen yöntemin başarımı bir çok farklı şartlar altında kaydedilmiş videolar üzerinde değerlendirilmiştir. Bu farklı şartlar; özniteliklerin engellenerek sahnede görülmediği durumları, düşük çözünürlüklü verileri, sahnede aydınlanma değişimi olan durumları, düzlemsel kafa hareketi ve düzlem dışı kafa hareketlerinin olduğu durumları içermektedir. Yöntem ayrıca otomobil ortamında kaydedilen videolarda da denenerek, yöntemin pratik bir uygulamadaki başarımı değerlendirilmiştir. Bu sonuçlardan yola çıkarak, önerilen yöntemin yüz öznitelik takibi için yaygın kullanılabilecek, gelecek vaadeden bir yöntem olduğu sonucuna varılabilmekte ayrıca engel ve kafa hareketi içeren mimik video dizilerindeki başarımına göre de gürbüz bir yöntem olduğu sonucuna varılabilmektedir. Otomobilde sürüş şartlarında kaydedilen videolardaki başarıma bakılırsa yöntemin, sürücü davranışı analiz sistemlerinde veya sürücü yorgunluk algılama sistemlerinde kullanılabilecek kuvvette bir yöntem olduğu kanısına varılabilir.

CHAPTER 1

INTRODUCTION

The aim of this thesis is to develop a novel method for facial feature tracking which is robust under real-world conditions such as occlusions, arbitrary head movements.

1.1 Motivation

Beginning from the time of the first invention of computers, they made their way easily to everyone's homes and offices and they became a part of human's life. Now, people spend much of their time in a day working in front of their computers. People communicate with their computers more than they communicate with their wives/husbands, friends, parents, etc. The communication between the human and the computer has become very important and crucial. Since the beginning of this communication it has been done by using interface devices, such as the mouse and the keyboard. Although working with a mouse and a keyboard seems comfortable in most situations, it is not natural to humans at all. There have been an increasing amount of diseases people suffer because of working in front of computers in uncomfortable positions. Consider how people pass their thoughts, opinions, and information to each other. Using non-verbal signals such as body language and facial expressions, humans can signal need, fear, or pain without words. So one question can be asked at this point: Why can people not just communicate with computers in the same manner?

Consider a computer that understands a human based on facial gestures, body movements, voice, etc. This can be a dream, but as in most cases, it was not in film industry. Stanley Kubrick directed a film in 1968 called "2001 : *A Space Odyssey*" which is based on the first book of a four novel series by Arthur C. Clarke. In the film there was a computer called "*HAL* 9000" which was the main computer of the spaceship. But this was not the main thing about it. HAL could hear, speak, plan, recognize faces, see, and judge facial expressions. It could even read lips! Considering the current state of the technology, it can be concluded that humans are not yet close to building a computer with the full intelligence or visual ability of HAL.

Building such a human-computer interface (HCI) requires a variety of computer vision-based analysis methods such as face analysis, gesture analysis, body analysis, etc. Considering all these methods, analyzing face can give very effective results because face has a very important role in the human body that can communicate lots of information. Face is a natural means by which people recognize each other. One of the first visual patterns a baby learns to recognize is the face. Such a face analysis system is a tool that can be used in a broad area of applications that relate to both academic research and commercial research topics. Such topics are: automatic surveillance systems, the classification and retrieval of images and videos, smart environments (smart vehicles, automatic driver fatigue detection, etc.), video-phone and video conferencing, model-based facial image coding (for example MPEG-4), face-based biometric person authentication systems, virtual reality and games, disabled aid, even in experimental behavioral psychology.

1.2 Problem Definition & Current State of the Art

A face analysis system, mentioned in the previous section, concerns some sub-topics such as the detection, tracking, recognition and modeling of faces or facial expressions in still images or image sequences. By the developments in the last decade, it can be concluded that face recognition methods have come to a certain level. There are many algorithms that can robustly work under various conditions. But in facial expression analysis systems there are still problems that have not been solved yet. Facial expression analysis is currently one of the most challenging problems in pattern analysis research community.

A facial analysis system can be divided into three parts, illustrated in Figure 1.1. The feature detection involves detecting some distinguishable points that can define the movement of facial components such as eyes, eye brows, mouth. The expression recognition part uses the information from the tracking part and outputs results such as happy, sad, surprised, etc. The tracking part is the tracking phase of the detected features. It can be defined as a bridge between the detection and recognition part for this reason it is the most important part of a facial analysis system. Beyond its importance, facial feature tracking is a very challenging problem because each facial expression is generated by non-rigid object deformations and these deformations are person-dependent. The complex nature of generating facial expressions makes it a hard problem starting from the representation phase of facial

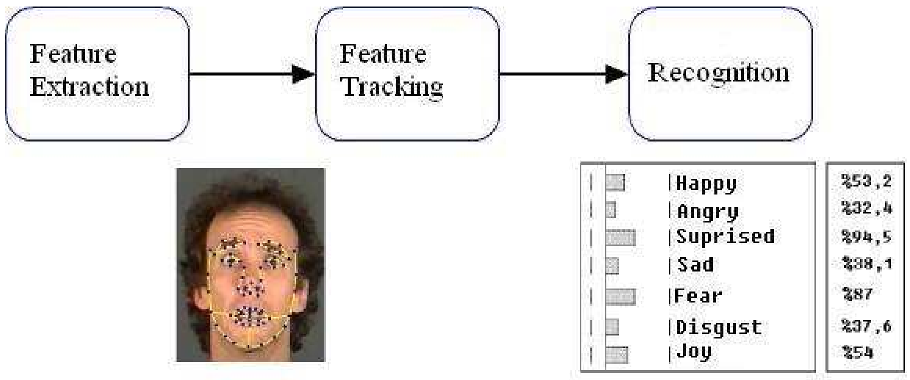


Figure 1.1: A facial analysis system.

components. Other than the complex movement of these facial components, there is the movement of head that makes the problem even more complex. In addition to these "inner" problems there are "outer" problems reasoned by external effects such as external occlusions, low resolution data, etc.

1.3 Contributions of this Thesis

To cope with such problems mentioned in the previous section, in this thesis a video based facial feature point tracking method that is based on graphical model framework is proposed. The proposed method is built on a statistical tool that uses the temporal relations in time and spatial relations between facial features. These statistical connections provide tracking to continue in the case of arbitrary head movements and uncertain data. In real-world scenarios facial features can be occluded and data may become useless. The proposed work can also handle occlusion and prevent the tracker to lose feature point positions.

1.4 Outline

The outline of the thesis is as follows: in chapter 2, the mathematical tools that this thesis is based on are explained in detail and some recent works are examined. In chapter 3, in the light of the mathematical tools, the proposed method based on graphical models is explained in detail. The performance evaluation results for the proposed tracker under various conditions are presented in chapter 4. Finally in chapter 5, conclusions are made and some possible extensions and future research directions are discussed.

CHAPTER 2

BACKGROUND

In this chapter firstly general tracking, covering all kinds of tracking problems, is briefly introduced and existing methods in this area are examined. In sections 2.2 and 2.3, Kalman filtering and Graphical Models are explained in detail since they are the tools this thesis is based on. In section 2.4, one of the subtopics of general tracking and the main topic of this thesis, facial feature tracking is explained and a detailed literature overview is given. Finally in section 2.5 a discussion in the light of previous methods is provided; open research areas are stated and the motivation of this thesis is explained.

2.1 Tracking

Tracking is the process of locating a moving object (or several objects) in image over time using a camera. This process typically consists of the analysis of video frames and outputting the location of the moving target within frames as an output. There are three main cases in tracking problems: moving object, moving camera, and the case when both of them are moving. Because of the complexity of other cases most of the current work is focused on the static-camera, moving-object case. Although this case seems simpler compared with other cases, one of the main difficulties here is the association of target locations in consecutive frames when the objects are moving fast relative to the frame rate. To cope with such problems, different motion models are used to describe the movement of the objects.

Usually tracking is used as the mid-part of a behavior analysis system, as illustrated in Figure 2.1. A typical behavior, motion analysis system consists of three parts: detection, tracking, and activity recognition. Tracking plays an important role between detection and activity recognition. It uses the information from detection part, detected regions or points, and outputs tracking results which are to be used by the activity recognition part. In recent years because of the improvement in these kinds of systems, the moving object definition is also extended and other tracking problems like human tracking, hand tracking, face/head tracking, facial

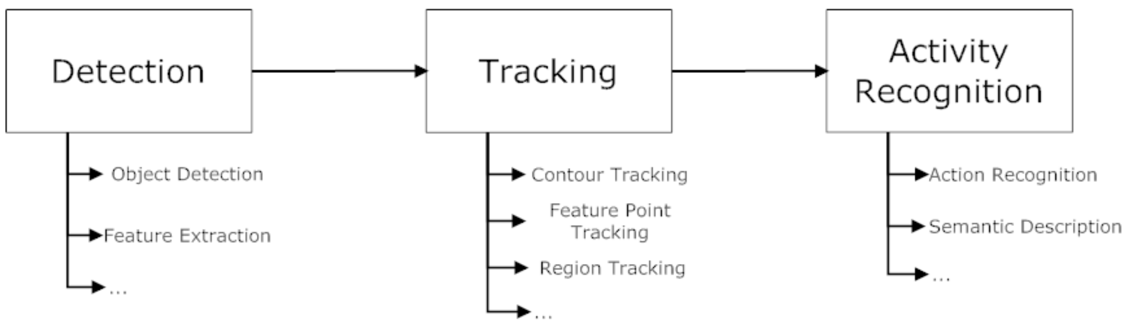


Figure 2.1: A general diagram of a behavior understanding system.

component tracking have arisen. In this section these problems will be briefly introduced and the approaches that have emerged as the solution of these problem will be briefly explained.

As explained above, object tracking is the problem of estimating the trajectory of an object in the image plane as it moves around a scene [55]. There are lots of methods that differ from each other based on the way they approach object representation, the way they use image features, and the way they model motion, appearance, and shape of objects. For example, points, geometric shapes and object contours can be used for representation [55, 23, 12]. There are some methods that use probability densities of object features, templates or active appearance models for representation [55, 15]. Different color spaces, like RGB, HSV or YCrCb, edge properties, optical flow or texture can be used as image features [55, 11]. For point tracking there are deterministic methods in which a cost associating each object in frame t - 1 to a single object in frame t using a set of motion constraints is defined. There are also statistical methods such as Kalman Filters, Particle Filters that use the state space approach to model the object properties such as position, velocity, and acceleration. In methods where objects are represented as templates there are methods like Mean-Shift, Kanade-Lucas-Tomasi (KLT) Tracker that use density based appearance models. In contour or shape based methods, some works use state space models to define the shape and the motion parameters of the contours or some other works evolves contours using energy functionals to define the shapes. In addition there are methods that use Hough transform or histograms for object shape matching.

The methods in human tracking, hand tracking or face tracking which are other emerging application areas can also be divided into categories similar to the ones explained above. Because this similar categorization and because these topics are out of the scope of this thesis, no more information will be given further. For more information please refer to [55, 48, 47]. In section 2.4, facial feature tracking as the main scope of this thesis is examined and a literature overview is provided.

Since the statistical tracking methods constitute the main scope of this thesis, the theoretical background of these methods is explained in this paragraph. For such methods, tracking, if described as a statistical problem, is the processing of measurements obtained from an object in order to maintain an estimate of its current state, which typically consists of kinematic components (position, velocity, etc.) and other components (signal strength, image intensity, etc.). Let x_t denote the state to be estimated, and let $y_{1:t}$ be the measurement vector (observations up to t: $\{y_0, y_1, ..., y_t\}$) where the subscript t denotes a discrete time index. Both x_t and $y_{1:t}$ are in general random quantities. Let $p(x_t)$ be the distribution of x_t; then the joint distribution is $p(x_t, y_{1:t}) = p(x_t)p(y_{1:t}|x_t)$, where $p(y_{1:t}|x_t)$ is the likelihood function. Using Bayes theorem,

$$p(x_t|y_{1:t}) = \frac{p(x_t)p(y_{1:t}|x_t)}{\int p(x_t)p(y_{1:t}|x_t)dx_t} \tag{2.1}$$

which is the *posterior* distribution of x_t, and is what Bayesian inference attempts to estimate. Assuming $p(x_t|y_{1:t})$ is obtained, tracking is solved; knowing, by definition, everything about the current state of the object, including its location and other dynamics in the state vector. Thus tracking can be formulated as a Bayesian inference problem, with $p(x_t|y_{1:t})$ as the objective. Note that in this formulation, the posterior $p(x_t|y_{1:t})$ is a time-varying quantity; in a tracking problem, $p(x_t|y_{1:t})$ at time t is evolved from $p(x_{t-1}|y_{1:t-1})$ at time $t-1$. In this sense, tracking is also a density propagation problem.

In reality, instead of obtaining the posterior density itself, a Bayesian inference task may focus on only estimating some properties of the density, such as moments, quantiles, highest posterior density regions, etc. All these quantities can be expressed in terms of posterior expectations of functions of x_t. The posterior expectation of a function $f(x_t)$is

$$E[f(x_t)|y_t] = \frac{\int f(x_t)p(x_t)p(y_t|x_t)dx_t}{\int p(x_t)p(y_t|x_t)dx_t} \tag{2.2}$$

The integration in this expression has until recently been the source of most of the practical difficulties in Bayesian inference, especially in high dimensions. In most applications, analytic evaluation of this integration is impossible. The alternative is numerical approximation. Most of the statistical tracking literature focus on the development of this kind of approximate numerical techniques for this computation problem.

In the next two sections, two statistical methods based on the Bayesian tracking scheme explained above are examined in detail.

2.2 Tracking Using Kalman Filtering

A Kalman filter is an optimal recursive estimator that infers parameters of an interest from indirect, inaccurate and uncertain observations of this interest. In the *state-space* representation, this interest becomes the unknown state variable, x_t, and uncertain observations become measurements, y_t, for a sequence $t = \{0, ..., T\}$. This is illustrated in Figure 2.2. *State-space* models can be defined as a notational convenience for estimation and control problems. It is developed to make the problems tractable for analysis. In Kalman filter the process noise is assumed as a Gaussian distribution. Because there is a linear relation in the model, this assumption provides the unknown state, x_t, to be also represented by a Gaussian. This ensures the optimality of the estimator that comes from minimizing the mean square error of the estimated unknown states.

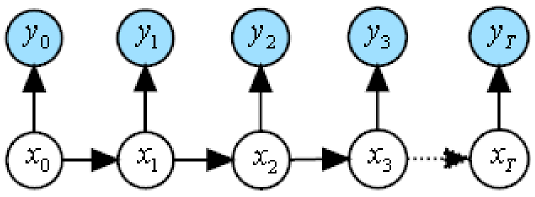


Figure 2.2: *State-space* representation of Kalman filter.

Consider a dynamic process described by an n-th order difference equation as a linear system

$$x_{i+1} = a_{0,i}x_i + a_{1,i}x_{i-1} + ... + a_{n-1,i}x_{i-n+1} + w_i \qquad i \geq 0 \tag{2.3}$$

where $\{w_i\}$ is the $zero-mean$ (statistically) and *white* (spectrally) Gaussian noise with a covariance matrix Q. Considering the initial random variables of x as $\{x_{-n+1}, ..., x_{-1}, x_0\}$, these are also zero-mean Gaussian distributions with covariance matrix Σ_0, the initial covariance matrix of the dynamic process in equation 2.3. Assuming $\{w_i\}$ and $\{x_i\}$ are statistically independent, the linear dynamic system can have the following form

$$\vec{X}_{i+1} = \begin{bmatrix} x_{i+1} \\ x_i \\ x_{i-1} \\ \vdots \\ x_{i-n+2} \end{bmatrix} = \underbrace{\begin{bmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_i \\ x_{i-1} \\ x_{i-2} \\ \vdots \\ x_{i-n+1} \end{bmatrix}}_{\vec{X}_i} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{D} w_i \quad (2.4)$$

Thinking of the observation term, the above expression leads to a general form as

$$\vec{X}_{i+1} = A\vec{X}_i + Bu_{i+1} + Dw_i \quad (2.5)$$

$$\vec{Y}_i = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \vec{X}_i + Ev_i = C_i\vec{X}_i + Ev_i \quad (2.6)$$

where $\{\vec{Y}_i\}$ represents the observations and $\{w_i\}$, $\{v_i\}$ represents the statistically independent Gaussian noises for the process (transition) model and measurement (observation) model with covariance matrices Q and R respectively. Thus, the equations 2.5 and 2.6 are the transition equation and measurement equation respectively. Here the $\{u_i\}$ term represents the input to the model which is a need for control problems but it can be neglected for most of the estimation problems.

The Kalman filter is a tool for estimating *state-space* variables that have a linear-Gaussian relation between each other. The reason it is called a "filter" is because it finds the best estimate from noisy data, filtering out the noise, and it uses only the previous data, not the future data which would produce an off-line algorithm.

Defining $\hat{X}_k^-$ to be *a priori* state estimate at step k given knowledge of the process prior to step k, and $\hat{X}_k$ to be *a posteriori* state estimate at step k given measurement Y_k, *a priori* and *a posteriori* estimate errors can be defined as

$$e_k^- = X_k - \hat{X}_k^- \quad (2.7)$$

$$e_k = X_k - \hat{X}_k \quad (2.8)$$

The *a priori* and *a posteriori* estimate error covariances are

$$\Sigma_k^- = E[e_k^- e_k^{-T}] \quad (2.9)$$

$$\Sigma_k = E[e_k e_k^T] \quad (2.10)$$

Considering the actual measurement, the equations of the Kalman filter begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{X}_k$ as a linear combination of an *a priori* estimate $\hat{X}_k^-$ and a weighted difference between an actual measurement Y_k and a measurement prediction $C\hat{X}_k^-$ as

$$\hat{X}_k = \hat{X}_k^- + K(Y_k - C\hat{X}_k^-) \tag{2.11}$$

The difference $(Y_k - C\hat{X}_k^-)$ in equation 2.11 is called the residual that reflects the disagreement between the predicted measurement $C\hat{X}_k^-$ and the actual measurement Y_k.

The matrix K is called the *gain* of Kalman filter that is a minimum mean-square error (MMSE) estimator for the *a posteriori* error covariance in equation 2.10. For the detailed derivation of this minimization please refer to [8]. One form of the resulting K is

$$K_k = \Sigma_k^- C^T (C\Sigma_k^- C^T + R)^{-1} \tag{2.12}$$

After finding the Kalman gain, the *a posteriori* (optimal) estimate, $\hat{X}_k$, and the error covariance matrix of the *a posteriori* estimate, $\hat{\Sigma}_k$ can be found easily. Considering the terms (*a priori* and *a posteriori*) in the derivations up to now, Kalman filter has two steps: prediction and the correction of this prediction by the use of measurements. In the light of the derivations these two steps are

$$\hat{X}_k^- = A\hat{X}_{k-1} \tag{2.13}$$

$$\Sigma_k^- = A\Sigma_{k-1}A^T + Q \tag{2.14}$$

and

$$K_k = \Sigma_k^- C^T (C\Sigma_k^- C^T + R)^{-1} \tag{2.15}$$

$$\hat{X}_k = \hat{X}_k^- + K(Y_k - C\hat{X}_k^-) \tag{2.16}$$

$$\Sigma_k = (I - K_k C)\Sigma_k^- \tag{2.17}$$

respectively.

As these equations are recursive equations, Kalman filter is a recursion algorithm where *a priori* estimate and *a posteriori* estimate contribute to each other as time increments, illustrated in Figure 2.3.

The two critical assumptions of Kalman filters, the linearity and Gaussian distribution, may not be satisfied in some cases. For example, there can be some cases where the relations between the unknown variable and its observations can not be modeled as a linear-system. To deal with such problems, the non-linear version of Kalman filter, *extended kalman filter* (Ekf), can be used. When the

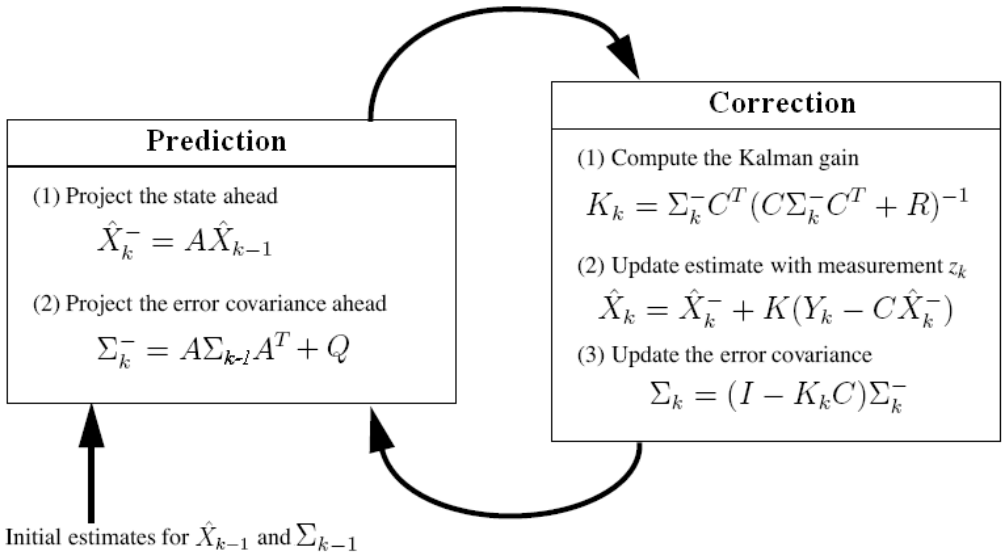

Prediction
(1) Project the state ahead
$\hat{X}_k^- = A\hat{X}_{k-1}$
(2) Project the error covariance ahead
$\Sigma_k^- = A\Sigma_{k-1}A^T + Q$
Correction
(1) Compute the Kalman gain
$K_k = \Sigma_k^- C^T(C\Sigma_k^- C^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{X}_k = \hat{X}_k^- + K(Y_k - C\hat{X}_k^-)$
(3) Update the error covariance
$\Sigma_k = (I - K_k C)\Sigma_k^-$
Initial estimates for $\hat{X}_{k-1}$ and Σ_{k-1}

Figure 2.3: Recursive nature of Kalman filter equations.

Gaussian assumption of Kalman filter is not enough, *particle filter* that is based on non-Gaussian distributions can be used.

Kalman filter is also a tool for many control problems. To see such an explanation of the Kalman filter from a control problem aspect, please see [27, 29, 30, 26].

As it can be concluded from the above explanations, Kalman filter is an algorithm that can be used for state estimation problems. In these kinds of problems, typically the states represent each parameters of interest individually. For instance; in a tracking problem, each state represent an individual object of interest. This can not be enough in some cases and there can be need of statistical relations between each state. This can be done by augmenting the states and defining a new state that includes all the parameters of interests. Thus, the relations between the objects can be represented in the covariance matrix of the joint probability. In this case the corresponding covariance matrix will be a full matrix and this will be a load in computations. To deal with this problem, according to the relations between states, the inverse covariance matrices should be constructed as sparse matrices. But in most cases this construction is not easy. Graphical models provides an convenient framework in which the inverse covariance matrices can be constructed as sparse matrices easily by using the visualization property of graph theory. In the next section this framework will explained in detail.

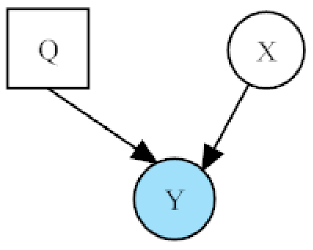
2.3 Tracking Using General Graphical Models

2.3.1 General Description

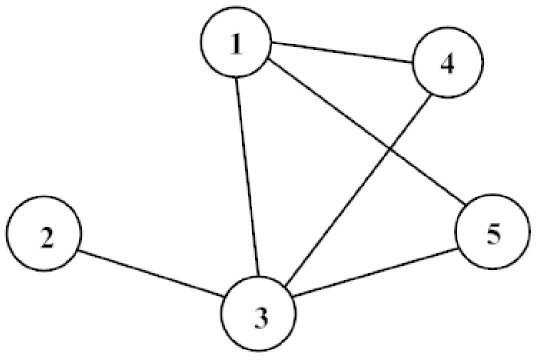
Graphical models can be defined as a marriage of graph theory and probability theory. As the fruits of this marriage, they provide a tool for the two major problems of engineering: uncertainty and complexity. The visualization property and the modular structure of graph theory makes complex probabilistic relations clear and understandable. In addition, inference algorithms make this structure to be computationally more powerful and robust.

Most of the classical probabilistic methods like Kalman filters, hidden Markov models, factor analysis are special cases of this general graphical framework. Graphical models have an increasingly important role in the design and analysis of machine learning algorithms and provide a powerful basis, a common language for many applications with probabilistic descriptions. This gives an opportunity for their extensive use in fields such as artificial intelligence, error correcting codes, speech processing, statistical physics, image processing, remote sensing, and computer vision.

Generally a graph G is defined by a set of nodes V, and a corresponding set of edges E. Each node $s \in V$ is associated with a random vector x_s which can be drawn from a wide range of probability distributions. But only Gaussian distributions are in the scope of this thesis. Usually there is an observation node y_s that is associated with every node x_s. If there is an observation on a node, that node is shown as shaded. Some examples are given in Figure 2.4-a.



(a)



(b)

Figure 2.4: (a) Some examples of different variable types (square is for discrete random variable, circle is for continuous random variable) (b) an undirected graph.

Each edge $(s, t) \in E$ connects two nodes $\{s, t\} \in V$ where $s \neq t$. Consider there are a sequence of nodes $s_0, s_1, ..., s_T$ a *path* between nodes s_0 and s_T is defined to be in such a way that $(s_{i-1}, s_i) \in E$ for $i = 1, ..., T$, and $(s_{i-1}, s_i) \neq (s_{j-1}, s_j)$ for $i \neq j$. If there is a path between every pair of nodes then G is said to be a connected graph. A *cycle*, or loop, is defined as a path which starts and ends with the same node. If a graph has no cycles, it is said to be *tree-structured.* Note that in a connected *tree-structured* graph, there is a unique path between each pair of nodes. A *clique* is defined to be a set of nodes in which every node is directly connected to every other node in the clique. For example, in Figure 2.4-b the sets $\{x_1\}$, $\{x_1, x_3\}$, $\{x_1, x_3, x_4\}$, and $\{x_1, x_3, x_5\}$ are all cliques. However, $\{x_1, x_3, x_4, x_5\}$ is not a clique because there is no edge between x_4 and x_5.

The (lack of) edges represent conditional independence assumptions. They provide a compact representation of joint probability distributions. For example, consider the case of N binary random variables. A representation of the joint, $P(X_1, ..., X_N)$, needs $O(2^N)$ parameters, whereas a graphical model may need exponentially fewer, depending on which conditional independence assumptions are made. The neighborhood of a node $s \in V$ is defined as $N(s) = \{t|(s, t) \in E\}$. The models are divided into two main categories: directed and undirected graphs. Directed graphs are graphs in which there is a causal relation between random variables. In undirected graphs the relation is bidirectional.

2.3.2 Directed Graphs

Directed models are the models where there is a one way relation between nodes. For instance, if there is an edge from node s to node t this means that s '*causes*' t. So, this disallows directed cycles, loops. Directed graphical models, also known as Bayesian networks (BNs), belief networks, etc. are popular with the artificial intelligence (AI) and machine learning communities.

Consider the example in Figure 2.5. Here, nodes represent binary random variables. The event "*grass is wet*" ($W = true$) has two possible causes: either the water sprinkler is on ($S = true$) or it is raining ($R = true$). The strength of this relationship is shown in the table below W; this is called W's conditional probability table (CPT). For example, $P(W = true|S = true, R = false) = 0.9$ (second entry of second row), and hence, $P(W = false|S = true, R = false) = 1 - 0.9 = 0.1$, since each row must sum to one. Since the C node has no parents, its CPT specifies the prior probability that it is cloudy (in this case, 0.5).

The simplest statement of the conditional independence relationships encoded

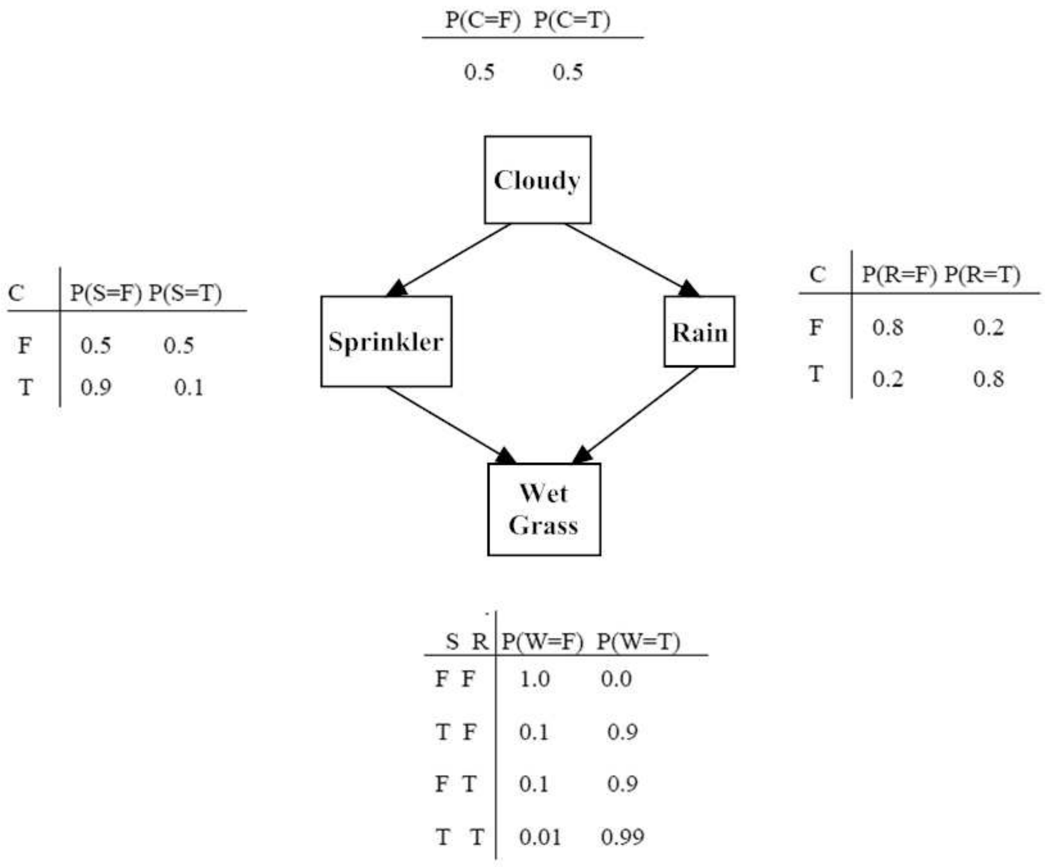


Figure 2.5: A example for directed graphical models (Taken from [31]).

in a Bayesian network, like the example given in Figure 2.5, can be stated as follows: a node is independent of its grandparents given its parents, where the grandparent/parent relationship is with respect to some fixed topological ordering of the nodes. This fact can be used to specify the joint distribution more compactly.

By the chain rule of probability, the joint probability of all the nodes in Figure 2.5 is

$$P(C, S, R, W) = P(C) \times P(S|C) \times P(R|C, S) \times P(W|C, S, R) \tag{2.18}$$

By using conditional independence relationships, it can be rewritten as

$$P(C, S, R, W) = P(C) \times P(S|C) \times P(R|C) \times P(W|S, R) \tag{2.19}$$

where this allows the third term to be simplified because R is independent of S given its parent C (written $R \perp S|C$), and the last term because W is independent of its grandparent C given its parents S, R ($W \perp C|S, R$).

The conditional independence relationships allows to represent the joint more compactly. Here the savings seem minimal, but in general, if there are n binary nodes, the full joint would require $O(2^n)$ parameters, but the factored form would only require $O(n2^k)$ parameters, where k is the maximum fan-in of a node.

As mentioned in section 2.3.1, most of the classical statistical methods are the special cases of general graphical models. For example, PCA, ICA, HMM, Kalman filters, etc. can be described as directed graphical models. This description for time-series statistical models like Kalman filters, HMM, etc will be given in this section. For other explanations please refer to [31].

For time series statistical models, consider the model in Figure 2.6-a, which represents a hidden Markov model (HMM). This makes the joint distribution

$$P(Q,Y) = P(Q_1)P(Y_1|Q_1)\prod_{t=2}^{4} P(Q_t|Q_{t-1})P(Y_t|Q_t) \tag{2.20}$$

For a sequence of length T, it will be for T time steps. In general, such a dynamic Bayesian network (DBN) can be specified by just drawing two time slices–the structure (and parameters) are assumed to repeat.

The Markov property states that the future is independent of the past given the present, i.e., $Q_{t+1} \perp Q_{t-1}|Q_t$. This Markov chain can be parameterized by using a transition matrix, $M_{ij} = P(Q_{t+1} = j|Q_t = i)$, and a prior distribution, $P(Q_1 = i)$.

An HMM is a hidden Markov model because the states, Q_t, of the Markov chain cannot be seen, instead what is observed is just a function of them, namely Y_t. For example, if Y_t is a vector, then $P(Y_t = y|Q_t = i) = N(y; \mu_i, \Sigma_i)$. A richer model, widely used in speech recognition, is to model the output (conditioned on the hidden state) as a mixture of Gaussians.

A linear dynamical system (LDS), like in Kalman filter, has the same topology as the model in Figure 2.6-b, except that the hidden nodes have linear-Gaussian CPDs. Replacing Q_t with X_t, the model becomes

$$P(X_1 = x) = N(x; x_0, \Sigma_0) \tag{2.21}$$

$$P(X_{t+1} = x_{t+1}|U_t = u; X_t = x) = N(x_{t+1}; Ax + Bu, Q) \tag{2.22}$$

$$P(Y_t = y|X_t = x; U_t = u) = N(y; Cx + Du; R) \tag{2.23}$$

As it can be seen here, the description of the linear system in Kalman filter is same

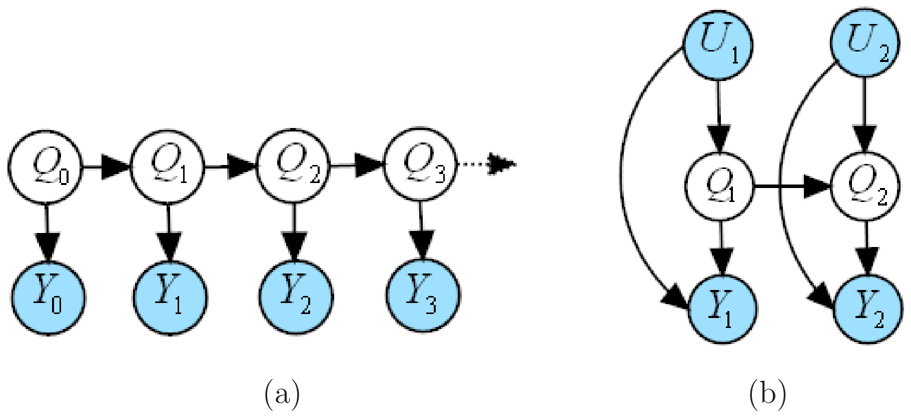


Figure 2.6: (a)A hidden Markov model (HMM) shown for 4 time slices (b)An input-output HMM.

with the description in section 2.2. Notice that typical equations of Kalman filter can also be derived from this aspect.

2.3.3 Undirected Graphs

These models are also known as Markov random fields (MRFs). In Markov random fields, the structural properties of the graphs play an important role. The conditional independence in MRF is defined with *graph separation*. Assume that A, B, and C are subsets of V. Then B is said to separate A and C if and only if there is one path between sets A and C which only pass through set B. It can be said that a graphical model has Markov property if x_A and x_C are conditionally independent given x_B when A and C are separated by B. According to Figure 2.4-b, the sets $\{x_4\}$, $\{x_5\}$, and $\{x_2\}$ are separated by the set $\{x_1, x_3\}$. So random variables x_4, x_5, and x_2 are conditionally independent given x_1 and x_3 as

$$p(x_2, x_4, x_5|x_1, x_3) = p(x_2|x_1, x_3)p(x_4|x_1, x_3)p(x_5|x_1, x_3) \tag{2.24}$$

In general the above property is as follows

$$p(x_s|x_{V\setminus s}) = p(x_s|x_{N(s)}) \tag{2.25}$$

where the *neighborhood* of a node s is defined as $N(s) = \{t|(s,t) \in E\}$. This means that the probability distribution of a random variable, conditioned on its nearest neighbors, at any given node is independent of the rest of the model.

The joint distribution of MRF is defined by

$$p(x) = \frac{1}{Z}\prod_{c\in C}\psi_c(x_c) \tag{2.26}$$

where C is the set of maximal cliques in the graph, $\psi_c(x_c)$ is a potential function (a positive, but otherwise arbitrary, real-valued function) on the clique x_c, and Z is the normalization factor:

$$Z = \sum_{x} \prod_{c \in C} \psi_c(x_c) \tag{2.27}$$

Consider the example in Figure 2.4-b, with the observations nodes. In this case, the joint distribution is

$$P(x, y) \; \alpha \; \psi(x_1, x_3, x_5)\psi(x_1, x_3, x_4)\psi(x_2, x_3) \prod_{i=1}^{4} \psi(x_i, y_i) \tag{2.28}$$

This structure is first used in a low-level vision problem [20] where the x_i's are usually hidden, and each x_i node has its own 'private' observation node y_i, as in Figure 2.4-b. The potential $\psi(x_i, y_i) = P(y_i|x_i)$ encodes the local likelihood; this is often a conditional Gaussian, where y_i is the image intensity of pixel i, and x_i is the underlying scene 'label'.

2.3.4 Inference

The main goal of inference is to estimate the values of hidden nodes, given the values of the observed nodes.

For a directed graph, if there is an observation on the "leaves" of the model, and the hidden variables that causes this are trying to be inferred, this is called *diagnosis*, or $bottom - up\ reasoning$; if there is an observation on the "roots" of the model, and the effects are trying to be predicted, this is called *prediction*, or $top - down\ reasoning$. In undirected graphs, since there is no causal relation between nodes, such a distinction is not required, but the main goal still remains.

The inference is divided into two categories: exact inference and approximate inference. In the following two sections these two categories are explained.

2.3.4.1 Exact Inference

Consider the water-sprinkler network in Figure 2.5, and suppose there is an observation that shows the grass is wet, which is denoted by $W = 1$ (1 represents true, 0 represents false). There are two possible causes for this: either it is raining, or the sprinkler is on. Which is more likely? Bayes' rule can be used to compute the posterior probability of each explanation. Bayes' rule states that

$$P(X|y) = \frac{P(y|X)P(X)}{P(y)} \tag{2.29}$$

where X are the hidden nodes and y is the observed evidence. In words, this formula becomes

$$posterior = \frac{conditional\ likelihood \times prior}{likelihood} \tag{2.30}$$

In the current example,

$$P(S = 1|W = 1) = \frac{P(S = 1, W = 1)}{P(W = 1)} = \frac{\sum_{c,r} P(C = c, S = 1, R = r, W = 1)}{P(W = 1)} = \frac{0.2781}{0.6471} = 0.430 \tag{2.31}$$

and

$$P(R = 1|W = 1) = \frac{P(R = 1, W = 1)}{P(W = 1)} = \frac{\sum_{c,s} P(C = c, S = s, R = 1, W = 1)}{P(W = 1)} = \frac{0.4581}{0.6471} = 0.708 \tag{2.32}$$

where

$$P(W = 1) = \sum_{c,s,r} P(C = c; S = s; R = r; W = 1) = 0.6471 \tag{2.33}$$

is a normalizing constant, equal to the probability (likelihood) of the data. So, it is more likely that the grass is wet because it is raining than because of the sprinkler: the likelihood ratio is 0.708/0.430 = 1.647. (Note that this computation has considered both scenarios, in which it is cloudy and not cloudy.)

In general, computing posterior estimates using Bayes' rule is computationally intractable. One way to see this is just to consider the normalizing constant, Z: in general, this involves a sum over an exponential number of terms. (For continuous random variables, the sum becomes an integral, which, except for certain notable cases like Gaussians, is not analytically tractable). Lots of methods are developed to solve this problem. They have usually used the conditional independence assumptions encoded in the graph to speed up exact inference. In the following sections these methods are explained.

Elimination

Consider the problem of computing the normalizing constant P(W = 1) for the water–sprinkler model. Using the factored representation of the joint implied by the graph,

$$P(W = w) = \sum_c \sum_s \sum_r P(C = c; S = s; R = r; W = w)$$
$$= \sum_c \sum_s \sum_r P(C = c) \times P(S = s|C = c) \times P(R = r|C = c) \times P(W = w|S = s, R = r) \tag{2.34}$$

The key idea of many inference algorithms is to "push" the sums in as far as possible, thus:

$$P(W=w) = \sum_c P(C=c) \sum_s P(S=s|C=c) \sum_r P(R=r|C=c) \times P(W=w|S=s,R=r) \tag{2.35}$$

becomes

$$P(W=w) = \sum_c P(C=c) \sum_s P(S=s|C=c) \times \phi_1(c,w,s) \tag{2.36}$$

where

$$\phi_1(c,w,s) = \sum_r P(R=r|C=c) \times P(W=w|S=s,R=r) \tag{2.37}$$

Performing the second sum,

$$P(W=w) = \sum_c P(C=c) \times \phi_2(c,w) \tag{2.38}$$

where

$$\phi_2(c,w) = \sum_s P(S=s|C=c) \times \phi_1(c,w,s) \tag{2.39}$$

By performing the above principle of distributing sums over products, the exact inference becomes computationally tractable. The amount of work done here is bounded by the size of the largest term that is created, like ϕ_1, ϕ_2. The key thing here is to choose the proper summation (elimination) order, which is practically a hard problem.

This method can be generalized greatly to apply to any commutative model that does not have loops. For example in the undirected case, the above conditional probabilities will be edge potentials. The basis of many common algorithms, such as Viterbi decoding [31], is based on this approach.

Junction-tree

Another way to avoid computational load, because of the calculation of common terms, is to convert the graphical model into a tree by clustering nodes together. This clustering is the graphical representation of what is done in the elimination process. The algorithm involves graph-theoretic operations in which nodes are removed in an order from the graph, where, when a node is removed, its remaining neighbors are linked. For instance, the clustered version of Figure 2.5 is given in Figure 2.7. The resulting graph in Figure 2.7 is called a *triangulated graph*.

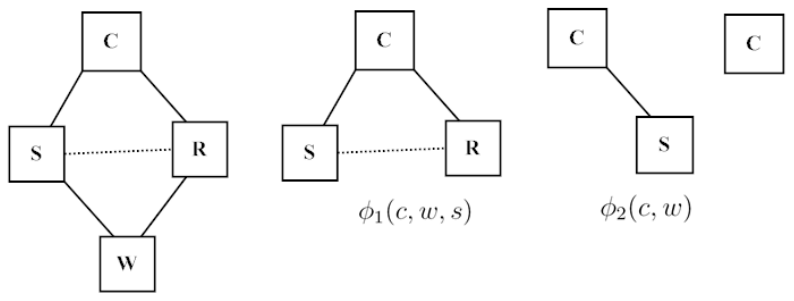


Figure 2.7: A clustered version of Figure 2.5.

The steps of the elimination algorithm applied here are exactly the same as before. But *junction* $-$ *tree* algorithms gives the opportunity to define common terms by clique potentials. After creating this tree of clusters, local message passing algorithm, explained in the next section, can be run easily by using the defined clique potentials.

The running time of these exact algorithms is exponential in the size of the largest cluster, clique (assuming all hidden nodes are discrete), which is not easy to minimize. This creates the *maximal clique problem* that find the largest clique in the graph. For many graphs which contain nodes with high fan-in, the maximal clique size is very large. Because of this problem and the evaluation is not tractable when the nodes are continuous it is necessary to use approximation.

Message-Passing

Most of the time it is needed to obtain more than a single marginal probability. So to compute these marginals, it is needed to run the elimination algorithm separately whereas there are common terms that are calculated successively. Naturally, this will be loss of time and will create computation load. There is a need to reuse these terms efficiently.

As Pearl [33] suggested, these common terms can be viewed as ”messages” attached to edges in the graph. Thus, rather than viewing inference as an elimination process, based on a global ordering, inference can be viewed in terms of local message computations and passing these messages to neighbors. A simple illustration is given in Figure 2.8.

This is a generalization of the well-known forwards-backwards algorithm for HMMs (chains) [35]. In a theoretical way, this algorithm is only feasible for graphs that are acyclic, i.e. that have no loops. The loopy structure of a graph would cause “double counting” the messages. But empirical results show that message-passing

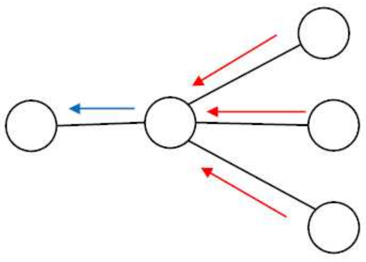


Figure 2.8: Message-passing.

algorithms produce acceptable results for graphs with loops. This is discussed in 2.3.4.2 section.

Belief Propagation

Belief propagation (BP), also known as the sum-product algorithm, is an another algorithm for computing marginals on a graphical model that is based on the concept explained above. BP is an iterative algorithm that is commonly used in pairwise Markov random fields, Bayesian networks, and factor graphs.

Considering the Bayesian framework, a general introduction about the application in tracking problems is given in section 2.1, for graphs whose prior distribution is defined by a Markov chain there are efficient recursive algorithms for exactly computing the single-node conditional marginal distributions $p(x_s|y)$. For any tree-structured graphical model, the prior distribution $p(x)$ can be factorized in as

$$p(x) = \prod_{(s,t)\in E} \frac{p(x_s, x_t)}{p(x_s)p(x_t)} \prod_{s\in V} p(x_s) \tag{2.40}$$

where $p(x_s)$ and $p(x_s, x_t)$ are marginal distributions. This equation gives an opportunity to factor $p(x)$ using pairwise clique potentials which are simple functions of the local marginal distributions at neighboring nodes. As discussed previously, despite the fact that this equation is only satisfied for tree-structured graphs, some empirical works show that it can be satisfied for graphs with cycles also.

For any $s \in V$ and any $t \in N(s)$, let $y_{s\setminus t}$ be the set of all observation nodes in the tree rooted at node s, excluding those in the subtree rooted at node t, as illustrated in Figure 2.9. The marginal distribution $p(x_s|y)$ can be decomposed

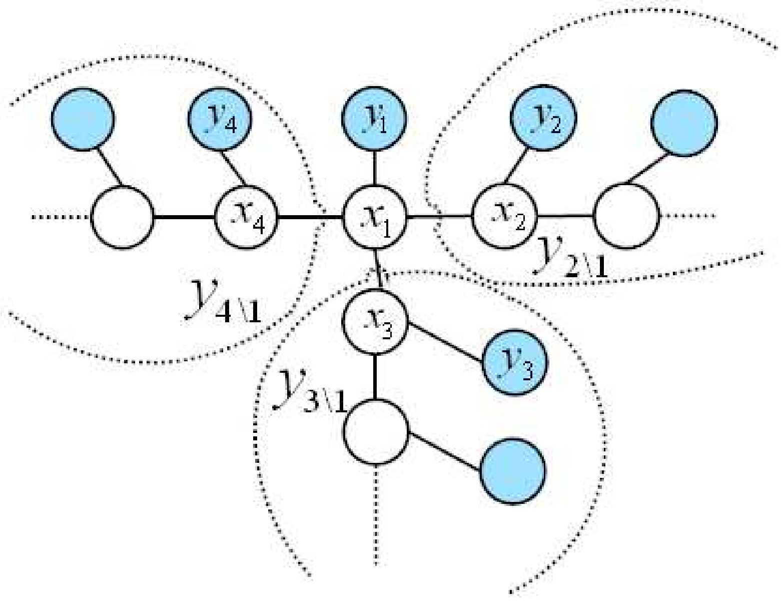


Figure 2.9: A tree-structured graph.

using Bayes' rule and the Markov properties implied by graph G as

$$p(x_s|y) = \frac{p(y|x_s)p(x_s)}{p(y)} = \alpha\, p(x_s)p(y_s|x_s) \prod_{t\in N(s)} p(y_{t\backslash s}|x_s) \tag{2.41}$$

where α denotes the normalization constant which is independent of x. Note that the normalization constant α is always chosen so that the function it multiplies integrates to unity. Thus, the numerical value of α may change from equation to equation.

From this decomposition, it can be seen that to calculate $p(x_s|y)$, the conditional likelihood $p(y_{t\backslash s}|x_s)$ is sufficient for the subtree rooted at node t. Using the conditional independencies implied by graph G, $p(y_{t\backslash s}|x_s)$ can be defined as

$$\begin{aligned} p(y_{t\backslash s}|x_s) = \frac{p(x_s|y_{t\backslash s})p(y_{t\backslash s})}{p(x_s)} &= \alpha \int_{x_t} \frac{p(x_s, x_t|y_{t\backslash s})}{p(x_s)} dx_t \\ &= \alpha \int_{x_t} \frac{p(x_s, x_t)p(y_{t\backslash s}|x_s, x_t)}{p(x_s)} dx_t \\ &= \alpha \int_{x_t} \frac{p(x_s, x_t)p(y_{t\backslash s}|x_t)}{p(x_s)} dx_t \\ = \alpha \int_{x_t} \left(\frac{p(x_s, x_t)}{p(x_s)p(x_t)}\right) p(x_t)p(y_t|x_t) &\prod_{u\in N(t)\backslash s} p(y_{u\backslash t}|x_t) dx_t \end{aligned} \tag{2.42}$$

Notice the similarity between equations 2.42 and 2.40. The prior model in equation 2.42 is in terms of the potential functions found in equation 2.40.

Factorization can continue like this. The general form of equation 2.41 is

$$p(x_u|y) = \alpha \int_{x_{V\backslash u}} \prod_{(s,t)\in E} \frac{p(x_s,x_t)}{p(x_s)p(x_t)} \prod_{s\in V} p(x_s)p(y_s|x_s)dx_{V\backslash u} \tag{2.43}$$

Using the two different representation of joint distribution in equations 2.40 and 2.26, $p(x_u|y)$ can also be defined as

$$p(x_u|y) = \alpha \int_{x_{V\backslash u}} \prod_{(s,t)\in E} \psi_{s,t}(x_s,x_t) \prod_{s\in V} p(y_s|x_s)dx_{V\backslash u} \tag{2.44}$$

These two different representations give an opportunity to write $p(x_s|y)$ as the following

$$p(x_s|y) = \alpha p(y_s|x_s) \prod_{t\in N(s)} m_{ts}(x_s) \tag{2.45}$$

$$m_{ts} = \alpha \int_{x_t} \psi_{s,t}(x_s,x_t)p(y_t|x_t) \prod_{u\in N(t)\backslash s} m_{ut}(x_t)dx_t \tag{2.46}$$

This motivates that the desired conditional marginal distribution $p(x_s|y)$ and sufficient statistic $m_{ts}(x_s)$ may be expressed in terms of the local relationships between neighboring nodes. Most of the algorithms use this property and solve these equations using local computations.

The belief propagation (BP) algorithm begins here. It defines the sufficient statistic $m_{ts}(x_s)$ as a *message* that node t will send to node s, while $(s,t) \in E$. This message includes all of the information about x_s which results from x_t and all of x_t's neighbors except x_s. If these messages are calculated, the marginal distribution $p(x_s|y)$, which is defined as *beliefs*, can be easily found from equation 2.45. Usually equations 2.46 and 2.45 are defined as *message update* and *belief update* equations respectively.

Because of this propagative structure of the algorithm, it depends on iterations. There are typically two kinds of belief propagation algorithms: Serial BP, Parallel BP. Serial BP is the algorithm in which each message and each belief is calculated serially. So there will be a scheduling to efficiently calculate beliefs. In parallel BP equation 2.46 is iteratively applied at each node in parallel, generating a sequence of messages $\{m_{ts}^n(x_s)\}$ which converge to $m_{ts}(x_s)$ as $n \to \infty$:

$$m_{ts}^n(xs) = \alpha \int_{x_t} \psi_{s,t}(x_s,x_t)p(y_t|x_t) \prod_{u\in N(t)\backslash s} m_{ut}^{n-1}(x_t)dx_t \tag{2.47}$$

When the number of iterations reach to the *diameter*, D, of the graph, the number of edges in the longest path, the messages will converge to a fixed point, so $m_{ts}^D(xs) =$

$m_{ts}(xs)$. This allows $p(x_s|y)$ to be exactly calculated for all $s \in V$. Even the steps before convergence, can provide useful information. This iteration based structure will affect belief update as

$$p(x_s|y_s^n) = \alpha p(y_s|x_s) \prod_{t \in N(s)} m_{ts}^n(x_s) \tag{2.48}$$

As a conclusion, the marginals $p(x_s|y)$ can be calculated by the BP update equations 2.46 and 2.45 for a tree-structured Markov random field. In practice, the message update integral 2.46 can be intractable for many distributions $p(x)$. There exist two general classes of tractable distributions: discrete variables and Gaussian variables. For discrete variables, the integral becomes sum operation and for Gaussian variables the integral becomes recursion operation on mean and covariance parameters of Gaussian distribution. The Gaussian version of the algorithm is explained in the next section. When the distributions are non-Gaussian, the algorithm called $non-parametric\ belief\ propagation$ [40] is used which is a non-parametric version of the BP algorithm similar to the non-Gaussian versions of Kalman filters, particle filters, etc.

Gaussian Belief Propagation

Gaussian belief propagation is the application of belief propagation on the graphs defined as Gaussian distributions. Because this thesis is based on Gaussian belief propagation, a detailed explanation is given here.

For Gauss Markov random fields, the prior distribution $p(x)$ is uniquely specified by either the full covariance matrix Σ or the inverse covariance matrix $J = \Sigma^{-1}$. However, because of the sparse structure — (s,t) element of matrix J, $(J_{s,t})$, will be zero if there is not any edge between node s and t, $(s,t) \notin E$ — J provides the more natural and efficient parameterization. Often, it is convenient to decompose J into clique potentials as in equation 2.26. Starting from the joint distribution, $p(x)$ can be decomposed into clique potentials using decomposition of J as

$$p(x) = \quad \tfrac{1}{Z}\exp\left\{-\tfrac{1}{2}x^T\Sigma^{-1}x\right\} = \tfrac{1}{Z}\prod_{s=1}^{N}\prod_{t=1}^{N}\exp\left\{-\tfrac{1}{2}x_s^T J_{s,t} x_t\right\} \tag{2.49}$$

$$= \frac{1}{Z}\prod_{(s,t)\in E} \exp \left\{-\tfrac{1}{2}\begin{bmatrix} x_s^T & x_t^T \end{bmatrix}\begin{bmatrix} J_{s(t)} & J_{s,t} \\ J_{t,s} & J_{t(s)} \end{bmatrix}\begin{bmatrix} x_s \\ x_t \end{bmatrix}\right\} = \tfrac{1}{Z}\prod_{(s,t)\in E}\psi_{s,t}(x_s,x_t)$$

where $Z = ((2\pi)^N \det \Sigma^{1/2})$ is the normalization constant.

So a clique potential can be defined as

$$\psi_{s,t}(x_s, x_t) = \exp\left\{ -\frac{1}{2} \begin{bmatrix} x_s^T & x_t^T \end{bmatrix} \begin{bmatrix} J_{s(t)} & J_{s,t} \\ J_{t,s} & J_{t(s)} \end{bmatrix} \begin{bmatrix} x_s \\ x_t \end{bmatrix} \right\} \tag{2.50}$$

where the $J_{s(t)}$ terms are chosen so that for all $s \in V$,

$$\sum_{t \in N(s)} J_{s(t)} = J_{s,s} \tag{2.51}$$

Note that a state space formalism, used in the most of the time series structures like Kalman filters, is not used here. State space models correspond to the factorization of $p(x)$ into a product of an initial distribution $p(x_0)$ and a sequence of distributions with one–step transition, $p(x_t|x_{t-1})$, which is not a convenient representation for graphs with cycles.

Focusing on the Gaussian case of BP, update equations (2.46, 2.45) will be in the form of mean and covariance update equations. However most of the time Gaussian distributions are parameterized in the *information form*. The information form parameters are defined by using mean, μ, and covariance, Σ, as

$$\vartheta = \Sigma^{-1}\mu \qquad \Lambda = \Sigma^{-1} \tag{2.52}$$

The notation, $N^{-1}(\vartheta, \Lambda)$, is used to define a Gaussian distribution with information parameters ϑ and Λ. To parameterize update equations as information parameters update equations, each of the terms in these equations should be related to a Gaussian density defined in information form, as follows

$$m_{ts}^n(x_s) = \alpha N^{-1}(\vartheta_{ts}^n, \Lambda_{ts}^n) \qquad p(x_s|y) = N^{-1}(\vartheta_s^n, \Lambda_s^n) \tag{2.53}$$

The clique potential, $\psi_{s,t}(x_s, x_t)$ was defined as a Gaussian in equation 2.50. The parameters which is undefined is the information parameters, or the mean and covariances, of the messages and beliefs in equation 2.53. To define these, the relation between x and y should be examined.

This is a well known problem in estimation theory. There is an unknown random vector x and there are observations of this vector y. It is known that x and y are jointly Gaussian with zero mean, $p(x, y) \sim N(0, \Sigma)$, –it assumed to be zero mean where a non-zero mean relations can be represented by adding an offset mean later on. According to these, the conditional distribution, $p(x|y) \sim N(\hat{x}, \hat{\Sigma})$, is also a Gaussian distribution with mean and covariances defined by *normal equations* as

$$\hat{x} \triangleq E[x|y] = \Sigma_{xy}\Sigma_y^{-1}y \tag{2.54}$$

$$\hat{\Sigma} \triangleq E[(x-\hat{x})(x-\hat{x})^T|y] = \Sigma_x - \Sigma_{xy}\Sigma_y^{-}1\Sigma_{xy}^T \tag{2.55}$$

where $\Sigma_x \triangleq E[xx^T]$, covariance of x; $\Sigma_y \triangleq E[yy^T]$, covariance of y; and $\Sigma_{xy} \triangleq E[xy^T]$, cross-covariance.

In many problems, the observations y are expressed as a linear function of x with errors because of noise

$$y = Cx + v \tag{2.56}$$

where $v \sim N(0, R)$ is a zero-mean Gaussian noise. x and v are independent. So equation 2.54 and 2.55 become

$$\hat{x} \triangleq \Sigma C^T(C\Sigma C^T + R)^{-1}y \tag{2.57}$$

$$\hat{\Sigma} \triangleq \Sigma - \Sigma C^T(C\Sigma C^T + R)^{-1}C\Sigma \tag{2.58}$$

where $\Sigma \triangleq E[xx^T]$ is the prior covariance of the unobserved variables x. Assuming Σ and R are both positive definite and hence invertible, the information form of these equations can be derived using the matrix inversion lemma.

$$(\Sigma^{-1} + C^TR^{-1}C)\hat{x} \triangleq CR^{-1}y \quad \Longrightarrow \quad \vartheta = \Sigma^{-1}\mu \tag{2.59}$$

$$\hat{\Sigma} \triangleq (\Sigma^{-1} + C^TR^{-1}C)^{-1} \quad \Longrightarrow \quad \Lambda = \Sigma^{-1} \tag{2.60}$$

Adapting this structure to every node, x_s and y_s, in the graph results in the following

$$p(x_s) = N^{-1}(0\ ,\ \Sigma_{s,s}^{-1}) \tag{2.61}$$

$$p(x_s|y_s) = N^{-1}(C_s^TR_s^{-1}y_s\ ,\ \Sigma_{s,s}^{-1} + C_s^TR_s^{-1}C_s) \tag{2.62}$$

Here, y is decomposed to $\{y_s\}_{s=1}^N$, the *local*, conditionally independent observations of the hidden variables $\{x_s\}_{s=1}^N$. So $p(y|x) = \prod_{s=1}^N p(y_s|x_s)$. C_s and R_s have become the decomposition of C and R as $C = diag(C_1, C_2, ..., C_N)$, $R = diag(R_1, R_2, ..., R_N)$, the block diagonal matrices, respectively. This makes each x_s as a subvector of x, while each $\Sigma_{s,s}$ is a block diagonal element of Σ.

Using Bayes' rule

$$p(y_s|x_s) = \alpha\frac{p(x_s|y_s)}{p(x_s)} = \alpha N^{-1}(C_s^TR_s^{-1}y_s\ ,\ C_s^TR_s^{-1}C_s) \tag{2.63}$$

Considering the belief update equation 2.45, products of Gaussian densities causes the sum operation for mean and covariances. So the parameters of $p(x_s|y_s^n) =$

$N^{-1}(\vartheta_s^n, \Lambda_s^n)$ are

$$\vartheta_s^n = C_s^T R_s^{-1} y_s + \sum_{t \in N(s)} \vartheta_{ts}^n \tag{2.64}$$

$$\Lambda_s^n = C_s^T R_s^{-1} C_s + \sum_{t \in N(s)} \Lambda_{ts}^n \tag{2.65}$$

Since the message update equation 2.46 consists of more complex terms first the product of Gaussian densities is calculated

$$\psi_{s,t}(x_s, x_t) p(y_t|x_t) \prod_{u \in N(s) \backslash s} m_{ut}^{n-1}(x_t) \propto N^{-1}(\bar{\vartheta}, \bar{\Lambda}) \tag{2.66}$$

where

$$\bar{\vartheta} = \begin{bmatrix} 0 \\ C_t^T R_t^{-1} y_t + \sum_{u \in N(t) \backslash s} \vartheta_{ut}^{n-1} \end{bmatrix} \tag{2.67}$$

$$\bar{\Lambda} = \begin{bmatrix} J_{s(t)} & J_{s,t} \\ J_{t,s} & J_{t(s)} + C_t^T R_t^{-1} C_t + \sum_{u \in N(t) \backslash s} \Lambda_{ut}^{n-1} \end{bmatrix} \tag{2.68}$$

After applying marginalization over equation 2.66, performing integration over x_t, the BP message $m_{ts}^n(x_s) = \alpha N^{-1}(\vartheta_{ts}^n, \Lambda_{ts}^n)$ is found as

$$\vartheta_{ts}^n = -J_{s,t} \left(J_{t(s)} + C_t^T R_t^{-1} C_t + \sum_{u \in N(t) \backslash s} \Lambda_{ut}^{n-1} \right)^{-1} \left(C_t^T R_t^{-1} y_t + \sum_{u \in N(t) \backslash s} \vartheta_{ut}^{n-1} \right) \tag{2.69}$$

$$\Lambda_{ts}^n = J_{s(t)} - J_{s,t} \left(J_{t(s)} + C_t^T R_t^{-1} C_t + \sum_{u \in N(t) \backslash s} \Lambda_{ut}^{n-1} \right)^{-1} J_{t,s} \tag{2.70}$$

The equations (2.64, 2.65) and (2.69, 2.70) are the parallel BP update equations. It can be easily seen that the factorization of inverse covariance matrix J into clique potentials, 2.50 is very important. It should be validated that

$$J_{t(s)} + \sum_{u \in N(t) \backslash s} J_{t(u)} = \sum_{u \in N(t)} J_{t(u)} = J_{t,t} \tag{2.71}$$

Another version of these equations that does not require a factorization is given by

$$\vartheta_{ts}^n = -J_{s,t} \left(J_{t,t} + C_t^T R_t^{-1} C_t + \sum_{u \in N(t) \backslash s} \Lambda_{ut}^{n-1} \right)^{-1} \left(C_t^T R_t^{-1} y_t + \sum_{u \in N(t) \backslash s} \vartheta_{ut}^{n-1} \right) \tag{2.72}$$

$$\Lambda_{ts}^n = -J_{s,t} \left(J_{t,t} + C_t^T R_t^{-1} C_t + \sum_{u \in N(t) \backslash s} \Lambda_{ut}^{n-1} \right)^{-1} J_{t,s} \tag{2.73}$$

$$\vartheta_s^n = C_s^T R_s^{-1} y_s + \sum_{t \in N(s)} \vartheta_{ts}^n \tag{2.74}$$

$$\Lambda_s^n = J_{s,s} + C_s^T R_s^{-1} C_s + \sum_{t \in N(s)} \Lambda_{ts}^n \tag{2.75}$$

2.3.4.2 Approximate Inference

Approximate inference is needed not only because of maximal clique sizes, but also it is needed in many cases where the variables are continuous and the corresponding integrals needed for implementing Bayes' rule cannot be performed in closed form.

There are some popular approximate inference methods like Monte Carlo methods, Loopy belief propagation, etc. The simplest Sampling (Monte Carlo) methods is importance sampling, where random samples x is drawn from $P(X)$, the (unconditional) distribution on the hidden variables, and then the samples are weighted by their likelihood, $P(y|x)$, where y is the observation. A more efficient approach in high dimensions is called Markov Chain Monte Carlo (MCMC), and includes as special cases as Gibbs sampling[21]. Another approach, Loopy belief propagation is based on applying Pearl's algorithm[33], message-passing, to the original graph, even if it has loops. This algorithm was inspired by the outstanding empirical success on loopy undirected graphs such as turbo codes. Since then, further empirical works [32, 33, 51] has shown that the algorithm works well in other contexts, such as low-level vision problems.

2.4 Facial Feature Tracking

In section 2.1 the general tracking is explained and the existing methods are examined. In this section facial feature tracking, the interested subtopic of general tracking, is explained with a detailed literature overview.

Facial feature tracking is the tracking of some facial components in videos. The facial components of interest depend on the context/environment in which the tracking is performed and the end use for which the tracking information is being sought. The typical facial components of interest are eyes, eye brows, and mouth.

Facial feature tracking is of paramount importance to applications of intelligent technologies, such as human-computer interaction, face recognition, dynamic facial expression analysis, 3D face reconstruction, etc. Other than these, in applications such as MPEG-4 coding based video conferencing applications, face-based biometric person authentication systems, even in virtual reality and games there is a need of facial feature tracking system. There are also some applications that use facial feature tracking to support behavioral psychology researches.

In the scope of this thesis, human-computer interaction, facial expression analysis, fatigue detection are chosen as the target application areas for the designed tracking system. Generally, as mentioned in section 2.1, a facial expression analysis system consists of three components: feature detection, feature tracking, and expression recognition. Feature detection involves detecting the distinguishable points that can define the movement of facial components. This may involve, detection of eyes, eye brows, mouth or feature points of these components. The tracking part was explained above. The last part is the expression recognition part which outputs results such as happy, sad, surprised, etc. according to tracking results of these feature points. In an expression analysis system, the content or the aim of feature detection part and expression recognition part can vary depending on the aim of the analysis system. For example, in an expression analysis system that focus on the eye states, the feature detection part only detects eye feature points and the recognition part outputs are defined as eye states. So, the techniques used for facial feature tracking can differ in so many ways. Generally, these can be categorized in the way they represent facial features; like 2D/3D points, 2D/3D shapes, according to features they use; such as edges, color, etc and in the way the mathematical approaches they are based on.

Tracking facial features is a very challenging problem under practical conditions due to the complexity of environments with unknown and variable illumination conditions, and uncertainties of poses and appearance of faces.

There are various types of methods in the literature. It is possible to classify these methods into two groups: model-based methods and model-free methods. Model based methods simply model the shape of the facial features. Most of the models are 3-D models and model shape parameters are updated in each frame. In model-free methods, there are no shape constraints. Basically these methods are based on motion estimations. The positions of facial features in subsequent frames is found by doing a local search inside a suitable sized window for the position which correlates best with the texture around the feature in the reference frame, no trained prior knowledge is required.

2.4.1 Model-Based Methods

In [15, 2, 16, 4, 12, 17, 22] AAM/ASM is used to track facial features. A 3-D face model is constructed using wire-frame models called *Candide*. First, a training set is prepared that consists of different face shapes under different poses. The more shapes from different poses are placed in the training set, the more efficiency will be gathered under different poses. Then, this training set is transformed to

PCA space and represented as PCA coefficients. When a new frame has come, the previous shape model is deformed by finding the best matches between the current shape and the training set shapes in PCA space using a distance measure. So, training set is used as the prior information for shape model deformation. An online learning system that learns the model from previous frame is proposed in [17]. The model parameters are recovered by using an observation likelihood. This likelihood is based on a statistical texture model in which the tracking and the learning processes are running in tandem. There is also a method [54] which combines the 2-D and 3-D AAMs.

There have been some variations, modifications made to this method in time. A multi-hierarchical structure of ASM is proposed in [44]. Based on the active shape model, a two-level hierarchy in feature points is proposed to characterize global shape of human faces, and local structural details of facial components. In [24] there is a combination of AAM with Gabor wavelet networks in which there are a linear combination of 2D Gabor functions whose parameters (position, scale and orientation) and weights are optimally determined to preserve the maximum image information for a chosen number of wavelets. A multi-model probabilistic facial shape model which utilizes a mixture of probabilistic PCAs (MPPCA) to represent the global nonlinearity of shape variations from different view points is proposed in [43]. Probabilistic PCAs are a combination of several local PCAs associated with probability densities.

In addition, there are some methods [49] which use a graph matching technique to track facial features. A labeled graph matching is done in a graph consist of nodes, corresponds to facial feature points, and links between the nodes. The labels are chosen as the templates composed of 17x17 gray levels around the node.

There are also some methods which can also be viewed as model-based methods: deformable models, active contours, snakes [12]. The basic idea is that an energy function that relates a parameterized model such as snakes, deformable models, etc. to an image is formed. This energy function is minimized using any standard optimization technique, and the resulting parameter set is obtained.

2.4.2 Model-Free Methods

On the other hand, there are optical flow based model-free facial feature trackers [13, 41, 11, 10]. In [41], the method assumes that intensity values of any given region (feature window size) do not change but merely shift from one position to another. By minimizing a cost function, the translation of this region can be found. A variation of this method that applies same matching for different image resolutions in a hierarchical manner is proposed in [11, 10].

Methods based on template matching which use the intensity pattern over the feature region in the previous frame as the template and search for a region in the present frame with the closest pattern are proposed in [50, 28, 53, 42, 7]. In [53] this feature region is assumed in '*star*' pattern to reduce the computation load. 'Sum of squared intensity differences (SSD)' is used as the matching measure in [42, 7].

Furthermore, probability density function (PDF) based matching techniques are proposed in [12, 11]. One of the PDF based matching techniques is Continuously Adaptively Mean Shift Algorithm (CamShift) which is based on an adaptation of the Mean Shift algorithm. In CamShift, an image is converted to a probability image via a histogram model of the color being detected, e.g., flesh color in the case of face tracking. Then, the mean of the color distribution is found by iterating in the direction of maximum increase in probability distribution [11]. Since the color distribution changes over time the distribution is updated over time. Here, the mean of the distribution is assumed as the center of the object and the distribution determine the size of the object and updating the distribution over time performs continuous tracking.

Gabor features are used for tracking in so many methods [23, 34, 52, 18, 19]. Gabor wavelet networks (GWN) mentioned above is used in [18, 19]. The GWN represents the face as a linear combination of 2D Gabor wavelet functions and it can be repositioned to match a target face image which can be used for tracking of facial features. The disparity of a point from one frame to the next is estimated in terms of phase differences of single Gabor jets [52].

There are methods in which the motion of facial features are modeled by time-series statistical models–as they are the tools used in this thesis, two time-series statistical methods are explained in section 2.2 and 2.3. In [23, 34, 37] Kalman filters are used to track facial features. In [23, 34] facial features are represented by 2-D points and the motion of these points are fitted to Kalman filter model. The observation of these points are gathered by using Gabor wavelets. In [37] *extended kalman filter* is used for gaze direction tracking. Generalized Hough transform is used as the measurements. There is not much work done for facial feature tracking using graphical models. There is one paper [9] and its variations [39, 38, 25] which uses a directed graphical model for tracking of contours authors fitted to model facial feature motion. The graphical model is based on non-Gaussian distributions. In [56] graphical models is used for cue integration for head tracking. The integration of features like, color, shape, intensity change are based on a graphical model that uses non-parametric belief propagation.

Also there is a model-free method in which the typical affine motion model is modified to include "yaw" and "pitch" motion to the model [5]. This makes the method accurately work not only for rigid motions but also for non-rigid motions which can be viewed as a disadvantage for model-free methods.

Between these two categories, model-based and model-free, there is a method which is a combination of the tracker in [7] and model-based method.

Besides all the methods described above, there are methods where the feature detection and tracking parts are together [36, 14]. In these methods usually feature detection is made for every frame while the detection procedure is updated by the previous frames.

2.5 Discussion

In this section a comparison will be made on the methods explained in the previous sections. According to the examined recent works up to now in the field of facial feature tracking, there are lots of working methods. But they have advantages/disadvantages when compared with each others.

In most of the existing methods occlusion treatment is a problem which is not handled directly. There are two kinds of occlusions: external occlusion and self-occlusion. External occlusion is the occlusion that is caused by an external object; finger, hand, etc. In contrast, self-occlusion is because of the head movement most of the time (for example out of plane rotation) . In both cases the data are uncertain. As a natural human behavior people move their heads or occlude their faces with their hands or fingers. Apart from occlusion, head movement, on its own, is a case that should be handled. Most of the model-free methods do not consider the occlusion or head movement cases. Limited number of methods have a solution to self-occlusion but in a limited head pose variations. On the other hand, model-based methods are more robust to head pose variations, but in the limits of training sets. The more number of training images with pose variations, the more success for pose variations.

Another advantage/disadvantage area is the training phases. Model-free methods do not have a training phase. So they are simpler and are easier to use. But model-based methods have a training phase which is an extra step for the system. Because of the mathematical approaches of model-based methods, they have computationally more load compared with the model-free methods.

In model-based methods, the spatial connection of the facial features are used as the model of the methods. But in model-free methods these connections are not considered, thus most of the time drifts, physically impossible tracking results occur.

According to these, a method which considers the spatial connections between facial features as well as the temporal relation for facial features, that take into account occlusion, is needed. A method that fits this description is proposed in the next chapter.

CHAPTER 3

GRAPHICAL MODEL BASED FACIAL FEATURE POINT TRACKING

In this chapter, a facial feature point tracker that can cope with the problems recent works suffer is proposed and the graphical model framework that the tracker is based on is explained in detail.

As discussed in the section 2.5, the recent methods have advantages/disadvantages over each other. But most of the methods have some level of difficulty in handling cases involving occlusion or head movement. Face occlusion by hands, fingers or head movements results from natural human actions and these cases should be carefully considered. In addition, these methods do not use the spatial relations between facial features. Here a facial feature tracker that uses not only temporal information about feature point movements, but also information about the spatial relationships between such points is proposed. The proposed method is also robust to external occlusions and head movements.

In section 3.1, the proposed statistical model is described in detail. The nature of the data used in this framework is explained in section 3.2. The algorithm to make inference based on the proposed model is explained in section 3.3. Finally, in section 3.4 the approaches developed to cope with real-world scenario problems such as occlusion and head movements are discussed.

3.1 Proposed Model

The proposed model is a graphical model that consists of three sub-models: a temporal model, an observation model, and a spatial model. The relations are represented as undirected edges. A representative version of the model that can be used only for two facial features is illustrated in Figure 3.1.

Here nodes represent the 2-D facial feature points. Each hidden variable (x_s) are vectors, each with four elements: x-coordinates, y-coordinates, velocity at

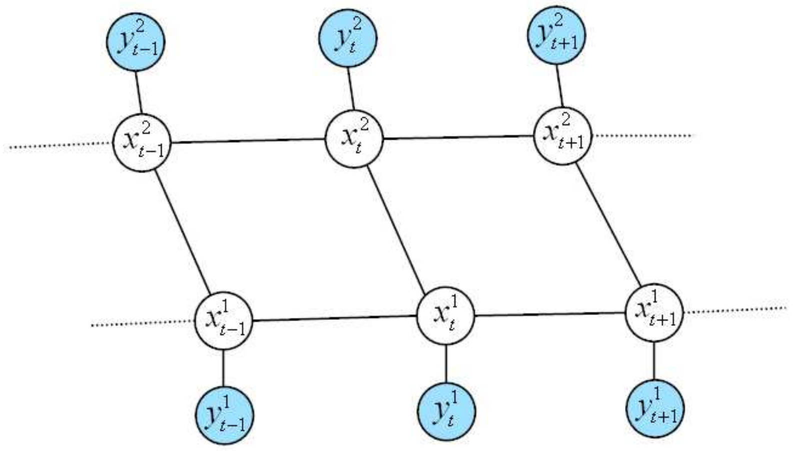


Figure 3.1: The graphical model built for tracking two facial features.

x-axis and velocity at y-axis of the points. The observed nodes (y_s) are vectors, each with two elements; x-coordinates and y-coordinates of observation data. So in this notation, (x_t^1) means hidden variable of the first feature point at time t and (y_{t+1}^2) is the observed variable of the second feature point at time $t+1$. Considering real-world coordinates, actually the hidden variables are 3-D points that move in the real-world coordinate system. But because of some problems, discussed later in this chapter, the feature points are assumed as 2-D points that move on the camera plane.

Since the head and facial components (mouth, eyes, etc.) can not move separately, the movement of feature points are dependent on two things: head movement and individual facial component movement (opening of mouth, eyes, etc.). Because facial component movement is in the scope of this thesis, the considered cases can be divided into two groups: the case where there is only facial component movement and the case where there are both head movement and facial component movement. This also creates two tracking problems: tracking only facial feature movements or tracking both head movement and facial feature movement. For this reason, the nodes in the model can represent two positions: the global position of facial features (when tracking both head and facial features) or the local position of facial features relative to head pose (when tracking facial features). The proper selection of this representation depends on different cases and scenarios. Which one of these representations is selected in this thesis is explained in the later sections when different cases are handled.

The relations between nodes are represented by clique potentials, as described in subsection 2.3.3. In this thesis, these potentials are selected as Gaussian distributions. Assuming the point coordinates and velocities are independent of each other the prior covariance matrix of hidden variables are in the following form:

$$\Sigma_{x_0} = \begin{bmatrix} {\sigma_{x_0}}_x^2 & 0 & 0 & 0 \\ 0 & {\sigma_{x_0}}_y^2 & 0 & 0 \\ 0 & 0 & {\sigma_{x_0}}_u^2 & 0 \\ 0 & 0 & 0 & {\sigma_{x_0}}_v^2 \end{bmatrix} \tag{3.1}$$

In the next subsections, construction of clique potentials in the sub-models will be explained in detail.

3.1.1 Temporal Model

The temporal model captures the temporal relations between facial feature points, hidden variables (x_s). So this model only considers the time-series relations between nodes, as illustrated in Figure 3.2-a. Since this model is about time-series, most statistical methods for tracking uses a model of this form. This temporal model can be constructed as linear or non-linear. The model used in the Kalman filter algorithm can be an example for linear temporal models. For non-linear models the model in extended Kalman filters can be an example. In this thesis, the temporal relations are modeled using linear dynamics. As in models assumed in Kalman filtering, this relation is based on

$$x_{t+1} = A \cdot x_t + w \tag{3.2}$$

where w represents the noise in temporal transitions. This transition is formed by the matrix A. Typically in a linear dynamical model used for tracking, it is assumed that the movement of the objects involved is characterized with constant velocities for each time step. Therefore, usually matrix A is selected as

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

which simply makes the position in the next time step as the summation of previous position and the constant velocity for that time step.

The temporal transition noise has a Gaussian distribution as $p(w) \sim N(0, Q)$ where Q denotes the covariance matrix. Assuming the noise in point coordinates and

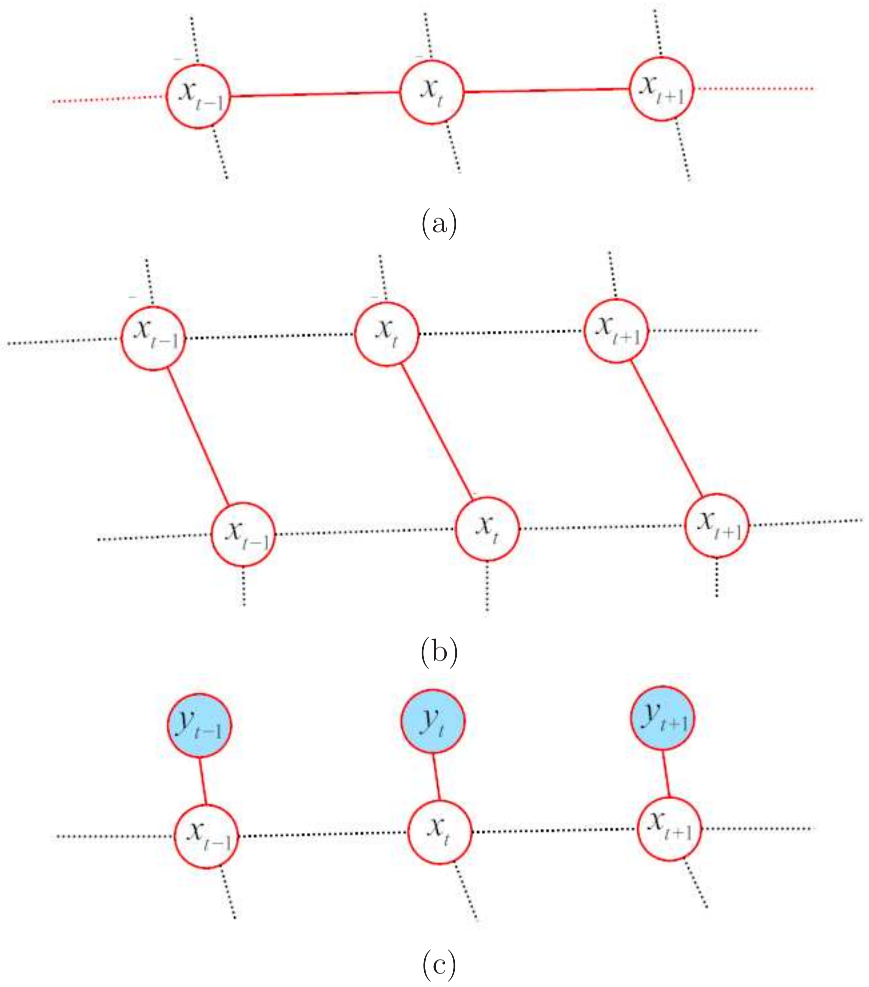


Figure 3.2: The sub-models that form the whole graphical model in Figure 3.1: (a)temporal model, (b) spatial model, (c) observation model.

velocities are independent of each other Q is formed as follows

$$Q = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_u^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix} \tag{3.4}$$

The parameters of this matrix are dependent on the scenario and they are selected according to the video input. In section 4.1 an example for these parameters is given according to the experimental data.

Using the model shown in equation 3.2, the clique potential that represents the temporal relation, in the form of the equation 2.50, becomes as

$$\psi_{t+1,t}(x_{t+1}, x_t) = N(A \cdot x_t, Q) = \alpha \exp\left\{(x_{t+1} - A \cdot x_t)^T Q^{-1}(x_{t+1} - A \cdot x_t)\right\}$$

$$= \alpha \exp\left\{ \begin{bmatrix} x_{t+1}^T & x_t^T \end{bmatrix} \begin{bmatrix} Q^{-1} & -Q^{-1}A \\ -A^TQ^{-1} & A^TQ^{-1}A \end{bmatrix} \begin{bmatrix} x_{t+1} \\ x_t \end{bmatrix} \right\} \tag{3.5}$$

3.1.2 Spatial Model

This part is the most important part of the model in the sense that it contains one of the main contributions of this thesis. This part models the spatial relations between facial features, illustrated in Figure 3.2-b. Here a model is introduced which simply insures the expected spatial distances between feature points. This is represented in a Gaussian distribution form as

$$\psi_{1,2}(x_1, x_2) = \alpha \exp\left\{ \begin{bmatrix} x_1 - (x_2 - \triangle x) \end{bmatrix}^T \Sigma^{-1} \begin{bmatrix} x_1 - (x_2 - \triangle x) \end{bmatrix} \right\} \tag{3.6}$$

where $\triangle x$ represents the difference vector and Σ represents the covariance matrix of the uncertainty. Since the hidden variables, (x_1,x_2), are four-element vectors, the difference vector, $\triangle x$, is also a four-element vector $\triangle x = [\ \triangle x_x \quad \triangle x_y \quad \triangle x_u \quad \triangle x_v\ ]^T$ where $\triangle x_x$ and $\triangle x_y$ represent the spatial differences between facial features on the x-axis and y-axis, respectively and $\triangle x_u$ and $\triangle x_v$ represent the spatial differences between velocity components on x-axis and y-axis, respectively. As mentioned in section 3.1, the proposed method tracks the locations of the projection of the feature points on the camera plane. So in this case the spatial differences are simply the 2-D spatial distances on the camera plane.

Think about that the global positions of the facial feature points are tracked. Since the head moves, these 2-D spatial distances ($\triangle x_x$ and $\triangle x_y$) on the camera plane will change. An updating procedure is needed. The solution to this case is explained in detail in subsection 3.4.2. Consider the case when the local positions of the facial features relative to head pose are tracked. In this case the 2-D spatial distances will not change (positions are relative to head pose) but, since the velocities of these points are independent of each other constraining the velocities may not be a sensible idea. So these components ($\triangle x_u$ and $\triangle x_v$) can be set as zero.

The covariance matrix of this clique potential is selected as below

$$\Sigma = \begin{bmatrix} \sigma_x'^2 & 0 & 0 & 0 \\ 0 & \sigma_y'^2 & 0 & 0 \\ 0 & 0 & \sigma_u'^2 & 0 \\ 0 & 0 & 0 & \sigma_v'^2 \end{bmatrix} \tag{3.7}$$

Similar to the parameters in matrix Q, these parameters are also dependent on the scenario. An example is given in section 4.1.

But to make everything clear about this sub-model, an application of the model for a simple case is given here. Consider two eye corner points are spatially connected and head is straight. The spatial distance between these points (on the camera plane) is 10 pixels on x-axis and 0 pixel on y-axis (head is straight, they are on the same line on y-axis). Assuming that the velocities are not spatially constrained and the error on x-axis and y-axis are ± 2 pixels, the spatial difference vector and the inverse error covariance matrix will be as follows

$$\triangle x = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \Sigma^{-1} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

Although the velocities are not spatially constrained, the spatial distances for velocities are shown as zero. But this doesn't constraint the velocities. The only thing that spatial constraints depend on is the corresponding parameter of the inverse covariance matrix. By setting the corresponding parameters for velocities in the inverse covariance matrix as zero, the velocities are set as spatially unconstrained.

3.1.3 Observation Model

As illustrated in Figure 3.2-c, this part is about the relations between the hidden variables and the corresponding noisy observations. The relation is modeled in a linear form as

$$y_t = C \cdot x_t + v \tag{3.9}$$

where v represents the measurement noise. The matrix C is the observation matrix which forms the analytical relation between hidden variables and observations. In this thesis, observations are composed of the 2-D facial feature points (the extraction of this information is explained in section 3.2 in detail). Since the observation vector is the 2-D coordinates, the matrix C is simply constructed in the way that relates the 2-D coordinate values in the hidden variables and the observations as

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.10}$$

The measurement noise has a zero mean Gaussian distribution with covariance matrix, R, namely $p(v) \sim N(0, R)$. Similar to the transition noise it is assumed that noise on the point coordinates are independent of each other. So the matrix R is formed as

$$R = \begin{bmatrix} \sigma_x''^2 & 0 \\ 0 & \sigma_y''^2 \end{bmatrix} \tag{3.11}$$

In section 4.1 an example for these parameters is given.

According to this formation, the clique potential of the observation model is

$$\psi_t(x_t, y_t) = N(C \cdot x_t, R) = p(y_t|x_t) \tag{3.12}$$

3.2 Data Pre-processing

In subsection 3.1.3, the observation model is explained but how the observed data is obtained is not mentioned. In this section the observation process of the model will be explained. This process can be considered as a pre-process that construct the observations for the model at each time step.

As illustrated in Figure 3.3, for any given image from a video sequence the interested feature point is searched in a search region by comparing with a template patch that is selected as a reference for the corresponding feature. The comparison is based on the Gabor filter outputs of both the template image region and the given image region. Thus, the search region is the region for filter convolution. The Gabor filter functions are selected as in [6]. Then, as in [23], 24 Gabor filter kernels consisting of 6 different orientations (0°, 30°, 60°, 90°, 120°, 150°) and 4 different wavelengths (4, 8, 12, 16 *pixels*) are constructed for the convolution. Convolution with these kernels produce 24 real and 24 imaginary coefficients for each pixel in the regions for both template and video sequence images. The magnitude values and phase values of the complex outputs of the filtering are compared using a similarity metric. Assuming n coefficients for each image, the similarity metric is as follows

$$Similarity = \frac{\sum_n m_n m'_n cos(\phi_n - \phi'_n)}{\sqrt{\sum_n m_n^2 \sum_n m_n'^2}} \tag{3.13}$$

where m and m' are the magnitude outputs for the template image and the given image respectively and ϕ and ϕ' are the phase outputs for the template image and the given image respectively. This metric is chosen to evaluate the "phase" similarity as well as the "magnitude" similarity. It computes a similarity between -1.0 and 1.0. This similarity metric produces these similarity values for every point in the convolution region. The location of the point with the highest similarity value is considered as the best match for the interested feature point in the given video frame. So this can be used as the observation data for the corresponding feature point.

Since this pre-processing is done in each time step for each feature point, to start such a process there is a need of initial point coordinates. In a general behavior analysis system, as explained in section 2.4, the feature detection part produces these initial coordinates to be tracked. Since feature detection is not in the scope of this thesis, these initial coordinates are produced by marking the first frames in each video sequence. Thus, the template image for comparison is selected as the first frame of the sequence. By selecting template as the first frame, comparison is made between the first frame, in which the positions are assumed as true, and the next frames of interest. In [23] consecutive comparison is proposed but the experiments show that this kind of comparison causes drifts. In consecutive comparison, because the previous filtering results are assumed as the reference for comparison, when a drift occurs in the previous frame, it begins to accumulate for the following frames. To avoid this kind of drifts the comparison is made between the first and subsequent frames. This procedure makes the filtering output of the first frame a template Gabor patch that is used as a reference for comparison.

Here Gabor filters are used as the observer of the graphical model. Gabor wavelet based feature representation is related to the psychophysical basis of human vision [46], and achieves robust performance for expression recognition and feature extraction under illumination and appearance variations. Gabor wavelets can completely represent both the time and frequency domains. This property of 2D Gabor image representations are important because of their increasing role in many computer vision applications and also in modeling biological vision, since recent neurophysiological evidence from the visual cortex of mammalian brains suggests that the filter response profiles of the main class of linearly-responding cortical neurons (called simple cells) are best modeled as a family of self-similar 2D Gabor wavelets.

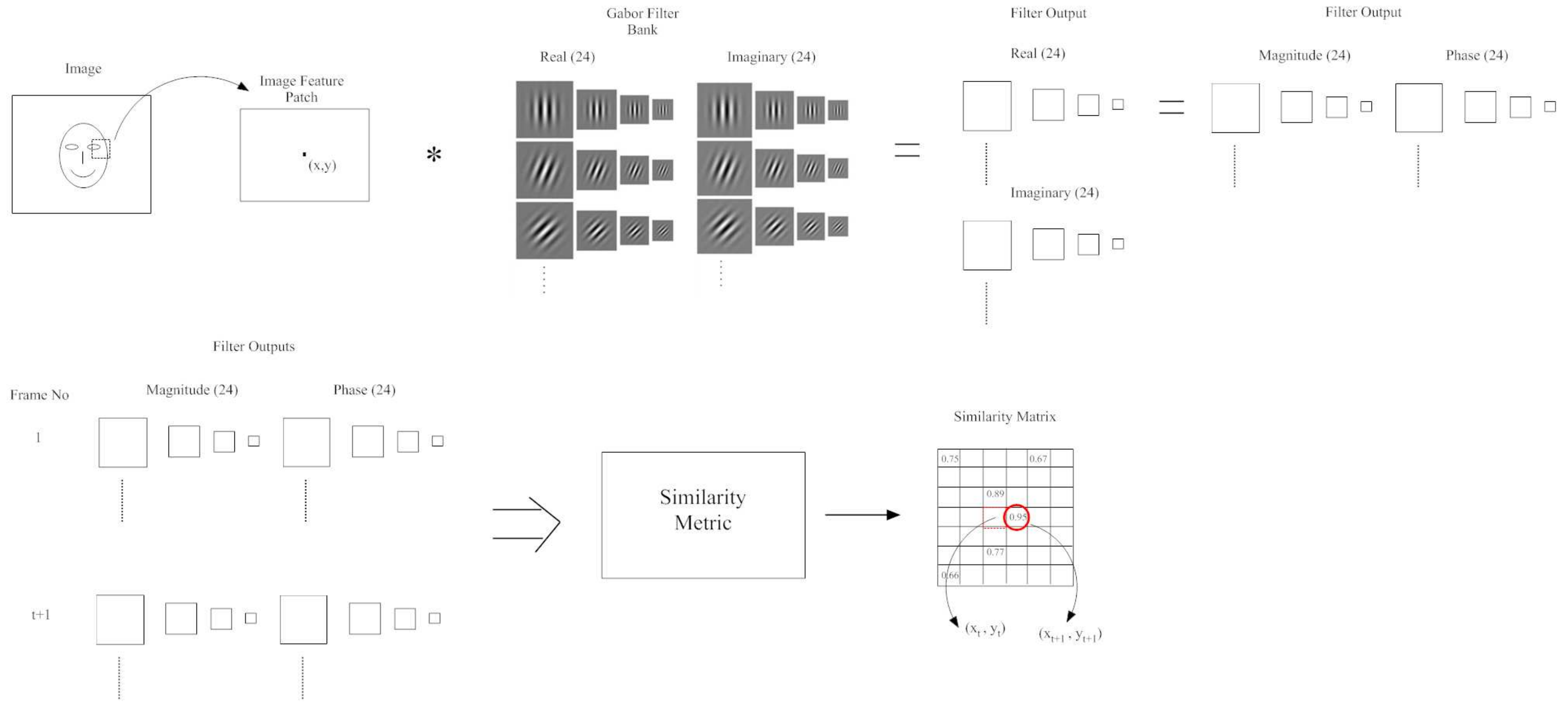


Figure 3.3: The flow diagram of the pre-processing stage.

While applying this framework to videos containing facial gestures, it has been observed that, 'eye-blinking' and 'eye-closure' cases produce a data association problem on the eyelid feature points. In both of these cases, because the eyelid is closed, the upper lid feature point and lower lid feature point begin to intersect and get mixed-up. Because the eye region is a small region and the edge structure on the eye region is changing too fast in the case of eyelid closure, the observation stage finds the same locations for both of the feature points. This causes a data association problem. As a solution to this case, the search regions of these feature points are limited in a way to avoid this misselection in the observation stage. Consider four eye feature points: inner corner, outer corner, upper lid, and lower lid, as illustrated in Figure 3.4. The line that passes through both outer and inner eye corner feature points is selected to limit the search regions of the upper lid and lower lid feature points. Because the same case occurs while the mouth is opened or closed, such a limitation can be needed for the opening of mouth. But during the trials it has been observed that, because the mouth region is a bigger area compared with the eye regions, the edge structure of search regions for the mouth feature points does not change dramatically when the mouth is opened or closed. So this does not affect the observation stage.

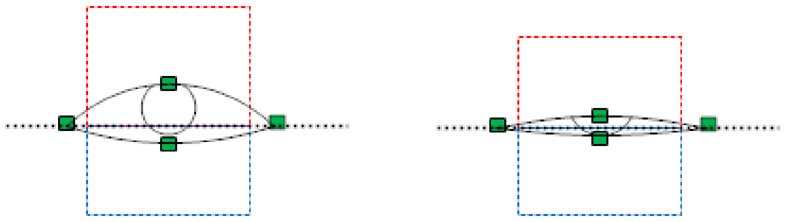


Figure 3.4: Search region limits for the eyelid feature points in the case of eyelid closure.

The output value of the similarity metric also gives a quantitative information about how similar these points are. This information can be used to detect frames in which there is an occlusion in the region of interest of a feature point. The occlusion detection part, another contribution of this thesis, is explained in subsection 3.4.1 in detail.

3.3 Inference Algorithm

As explained in subsection 2.3.4, inference algorithms are used to estimate the values of hidden nodes. In many computer vision and image processing applications, these algorithms are used to find the conditional density function $p(x_s|y)$, where x_s is a hidden variable that represents the location of a feature point and y is the set of all observed data. Since in this thesis the relations between nodes are selected as Gaussian distributions, explained in section 3.1, the inference algorithm is chosen as '*Gaussian Belief Propagation*' algorithm. Because of the parametric nature of Gaussian distributions, this inference algorithm simply turns into a mean and covariance update procedure, explained in the related part of subsection 2.3.4.1.

To make the proposed tracker usable in real-time applications, the chosen inference algorithm is constructed in a way to use only the current and the previous data. Thus, the inference algorithm used here becomes a filtering algorithm. This construction modifies the update equations, (equations 2.73, 2.72, 2.75, 2.74), in a way that they allow one-sided message passing in time. Consider the model; to make an inference on a node only the information from previous temporal neighboring node, current spatial neighboring node and current observed neighboring node is incorporated. The illustration of this one-sided structure is given in Figure 3.5.

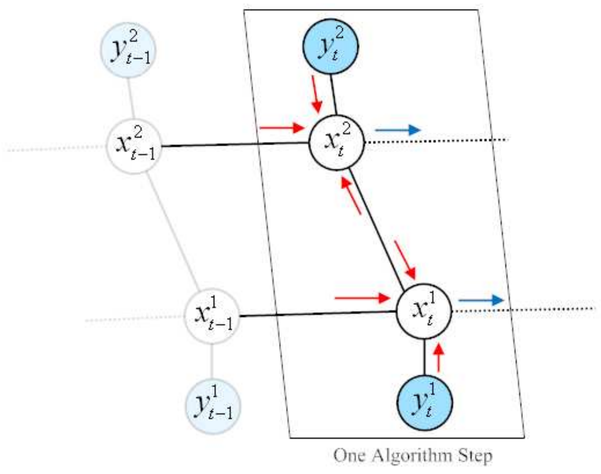


Figure 3.5: The one-sided message passing diagram of one algorithm step for the minor graphical model in figure 3.1

To illustrate the process of the update equations, here , one step of the algorithm is shown as an example. Consider two eye corner points as features of interest that are spatially connected and there are spatial constraints on x and y axes. Assume that their initial coordinates are $x^1_0 = (10, 10)$ and $x^2_0 = (20, 10)$ respectively and their prior covariance matrices are as follows:

$$\Sigma_0^1 = \Sigma_0^2 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \tag{3.14}$$

Consider that the observation in the next time step for these features are $y_1^1 = (13, 13)$ and $y_1^2 = (18, 11)$. Assuming the position errors on both x and y axes (σ_x^2, σ_y^2) are ± 4 pixels and the velocity errors on both x and y axes (σ_u^2, σ_v^2) are ± 2 pixels in the temporal model; the position errors on both x and y axes $(\sigma_x''^2, \sigma_y''^2)$ are ± 2 pixels in the observation model and the position errors on x and y axes $(\sigma_x'^2, \sigma_y'^2)$ are ± 2 pixels in the spatial model, the covariance matrices in these sub-models (equations 3.4, 3.7, 3.11) are selected as follows:

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \qquad R = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \qquad \Sigma^{-1} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

According to the update equations 2.72 and 2.73, the temporal mean and covariance messages of the x_1^1 node are

$$\vartheta_{x_0^1,x_1^1} = -J_{x_1^1,x_0^1}\left(J_{x_0^1,x_0^1} + \Lambda_{x_0^1}\right)^{-1}\left(\vartheta_{x_0^1}\right) \Rightarrow \mu_{x_0^1,x_1^1} = \begin{bmatrix} -3.6120 \\ 0.2032 \\ -2.6546 \\ -0.9143 \end{bmatrix} \tag{3.16}$$

$$\Lambda_{x_0^1,x_1^1} = -J_{x_1^1,x_0^1}\left(J_{x_0^1,x_0^1} + \Lambda_{x_0^1}\right)^{-1} J_{x_0^1,x_1^1} \Rightarrow \Sigma_{x_0^1,x_1^1} = \begin{bmatrix} -18.56 & 0 & 0.64 & 0 \\ 0 & -18.56 & 0 & 0.64 \\ 0.64 & 0 & -4.8 & 0 \\ 0 & 0.64 & 0 & -4.8 \end{bmatrix}$$

The spatial mean and covariance messages of the x_1^1 node are

$$\vartheta_{x_1^2,x_1^1} = -J_{x_1^1,x_1^2}\left(J_{x_1^2,x_1^2} + C^T R^{-1} C + \Lambda_{x_0^2,x_1^2}\right)^{-1}\left(C^T R^{-1} y_1^2 + \vartheta_{x_0^2,x_1^2}\right) \tag{3.17}$$

$$\Rightarrow \mu_{x_1^2,x_1^1} = \begin{bmatrix} 2.9711 \\ 3.6323 \\ 1.0587 \\ 0.6440 \end{bmatrix}$$

$$\Lambda_{x_1^2,x_1^1} = -J_{x_1^1,x_1^2}\left(J_{x_1^2,x_1^2} + C^T R^{-1} C + \Lambda_{x_0^2,x_1^2}\right)^{-1} J_{x_1^2,x_1^1}$$

$$\Rightarrow \Sigma_{x_0^1,x_1^1} = \begin{bmatrix} -0.1233 & 0 & 0 & 0 \\ 0 & -0.1233 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The parameters $\Lambda_{x_0^2,x_1^2}$ and $\vartheta_{x_0^2,x_1^2}$ can be evaluated as in the equation 3.16. Using the messages evaluated above, the mean and covariance belief of the x_1^1 node can updated using the equations 2.74, 2.75 as follows:

$$\vartheta_{x_1^1} = \underbrace{C^T R^{-1} y_1^1}_{\begin{bmatrix} 11.9802 \\ 8.4671 \\ 2.6561 \\ 1.5013 \end{bmatrix}} + \vartheta_{x_0^1,x_1^1} + \vartheta_{x_1^2,x_1^1} \Rightarrow \mu_{x_1^1} = \begin{bmatrix} 11.3393 \\ 12.3026 \\ 1.0602 \\ 1.2310 \end{bmatrix} \tag{3.18}$$

$$\Lambda_{x_1^1} = J_{x_1^1,x_1^1} + C^T R^{-1} C + \Lambda_{x_0^1,x_1^1} + J_{x_1^2,x_1^1}$$

$$\Rightarrow \Sigma_{x_1^1} = \begin{bmatrix} 2.6052 & 0 & 0.4619 & 0 \\ 0 & 2.6052 & 0 & 0.4619 \\ 0.4619 & 0 & 24.6493 & 0 \\ 0 & 0.4619 & 0 & 24.6493 \end{bmatrix}$$

As in the same manner the mean and covariance of the x_1^2 can be estimated. The results will be as follows:

$$\mu_{x_1^2} = \begin{bmatrix} 19.6883 \\ 11.6422 \\ 0.7674 \\ 1.1139 \end{bmatrix} \quad \Sigma_{x_1^2} = \begin{bmatrix} 2.6052 & 0 & 0.4619 & 0 \\ 0 & 2.6052 & 0 & 0.4619 \\ 0.4619 & 0 & 24.6493 & 0 \\ 0 & 0.4619 & 0 & 24.6493 \end{bmatrix} \tag{3.19}$$

As it can be seen from the estimated feature locations; although the observations seems to make the features get closer to each other (because of they are uncertain), the spatial model provides the spatial constraints and keep the expected spatial distance between the features. It can be observed from the covariance matrices that the uncertainties on the positions on x and y axes are decreased. This shows that the reliability of these estimations are increased.

3.4 Feature Point Tracking under Real-World Conditions

As a natural human actions people move their head, hands or fingers all the time. To build a facial feature tracker system that is robust to real-world conditions, these cases should be handled properly. Unfortunately, most of the recent methods, reviewed in section 2.4, do not consider these cases.

The proposed method here is designed in a way to deal with these problems. This is one of the main contributions of this thesis. These problems can be dealt in two different cases: external occlusion (moving hands, fingers, etc. in front of the face) and 3D head movements. The proposed solutions for these cases are explained in the next subsections respectively.

3.4.1 External Occlusion

The word 'occlusion' is a word that can be defined in so many ways. It contains different meanings. Looking from the general perspective of computer vision and image processing applications, occlusion case can be divided into two categories: external occlusion and self-occlusion. Self-occlusion can be defined as the case when the object of interest occlude itself, most of the time this occurs because of the movements of the object. Contrarily, external occlusion is the occlusion that is caused by external effects. In the context of this thesis, self-occlusion can be defined as the occlusion when there is out-of-plane motion of the head. For example; when the head rotates left or right, on the observed image the feature points will begin to occlude each other. This case should be handled in the scope of head movements. This is done in subsection 3.4.2. In this subsection the external occlusion case, which can be defined as the occlusion of the face by hands, fingers, and varied other objects, is explained.

When occlusion occurs, the information coming from the observation stage become useless for the occluded feature point. Consider an eye corner point that is occluded. In this case the observation stage, that is looking for an eye corner point in the search (convolution) region similar to the template of this feature point, can not find a similar point because of the occlusion. This problem will make it select an arbitrary point in the region that is not similar to eye feature but that has a edge structure alike the template eye corner. For this reason the observation stage gives a wrong location that it thinks as an eye corner point. To deal with this situation the information coming from the observation stage should be omitted. This lack of information will not cause the tracker's performance because the informations coming from the temporal and spatial relations are enough to continue the tracking.

To apply the procedure explained above, to prevent drifts caused by misleading of the observation stage, the occlusion should be detected. As mentioned in section 3.2, Gabor feature-based observation stage provides quantitative information about the similarities in the search region. It has been observed that when an occlusion occurs on a feature point, the similarity values begin to drop, as illustrated in Figure 3.6. This property can be used to detect the occlusion times in the video frames. According to the procedure mentioned above, the overall treatment for external occlusion is given in Algorithm 3.1.

Algorithm 3.1: Overall procedure in the case of occlusion

```
Evaluate the similarity metric;
Find the point with the highest similarity;
if $this\, similarity\, value > threshold$ then
    take the point as data;
else
    close the data term;
end
```

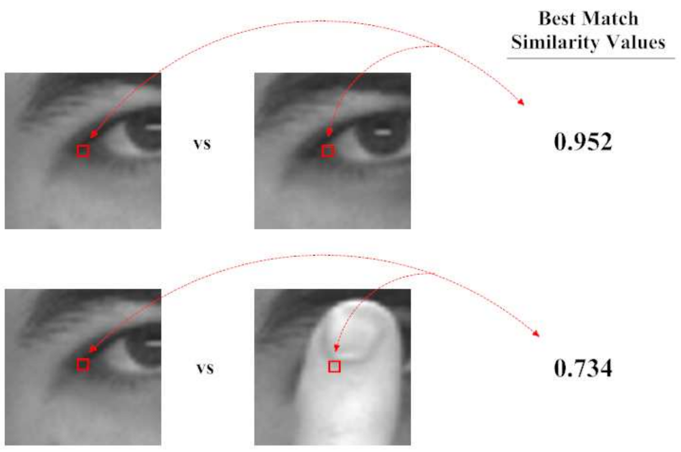


Figure 3.6: The best match similarity value outputs of the observation stage in case of occlusion.

3.4.2 3-D Head Movements

In the context of facial analysis, head movement is often divided in two categories: in-plane head motion and out-of-plane head motion. Defining these in terms of six degree of freedoms; in-plane head motion includes the translation in x-axis, translation in y-axis and the rotation on z-axis (rolling). On the other hand, out-of-plane head motion includes the translation in z-axis, rotation on x-axis (pitching) and y-axis (yawing).

To deal with these kinds of 3-D movements of the head, ideally (and unlike the development up to this point) a framework in which the feature points are represented as 3-D points is needed. In real-world coordinate system, facial feature points are 3-D points. But, since the observations are 2-D videos taken from a monocular camera, the framework up to this point is designed on the camera plane and feature points are represented as 2-D points. This produces a problem of building a 3-D model using 2-D data.

Considering this problem, there is a need of back projection that will transform the 2-D observation data (y_t) to 3-D coordinate system (x_t). Going from 2-D to 3-D is a hard problem which does not have an exact solution because of the ill-posed structure of the problem. Considering the literature, there are two main approaches to this problem. One approach is stereo vision. The other approach is using the motion (Chapter 8 of [45]). In stereo vision applications, one needs to take the advantage of stereo cameras. Since in this thesis monocular cameras are used, this approach is out of the scope. There are many proposed methods in the literature to obtain the 3-D information from motion. Some experimental analysis on 3D reconstruction given the kind of data of interest is conducted in this thesis, and the ill-posed nature of this problem has been observed. As a result, it has been decided not to pursue that path.

To suppress this deficiency of the 2-D data for 3-D movements, using the head pose can be a solution. If this information is obtained somehow and penetrated into the model, then the 2-D model can act according to the pose of the head and it can continue tracking in the case of 3-D head movements. For this reason the instant head pose is needed. This can be obtained by using a head pose tracker that works in parallel with the proposed method or for off-line applications head tracking can be done beforehand. Another way can be to use the existing system and extend it in a way to take the advantage of reliably tracked points. There are some facial feature points, like eye corners, that their movements only depend on head movement. Thus these can be defined as more reliably tracked points and they can be used to get the head pose information. These two methods are explained in the next sections.

Tracking Based On Given Head Pose

To cope with 3D head movements, one approach can be to use the head pose information that is given by a head pose tracker- for instance; a vision-based head pose tracker. This head pose information consists of three-dimensional translation (T_x, T_y, T_z) and orientation information $(\theta_x, \theta_y, \theta_z)$ with respect to the origin. This information can be inserted into the model in two ways. One way is to transform back the observed data of the model, explained in section 3.2, using the head pose information. With this transformation, head movement effects will be removed from the observation data, then it can be used as if there was no head movement. Another way is to update the spatial distances of the spatial model, defined in subsection 3.1.2, according to the head pose change. In the next paragraphs these two methods will be explained.

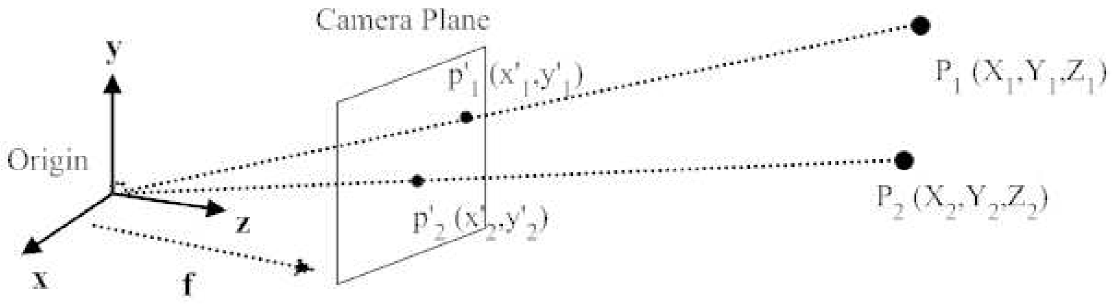


Figure 3.7: The illustration of data back transformation.

Consider a 3-D coordinate system, as illustrated in Figure 3.7, in which there is a 3-D point $\mathbf{P} = [X, Y, Z]^T$. Here $P_1 = [X_1, Y_1, Z_1]^T$ is any arbitrary 3-D feature point on the face before head pose change. Point $p'_1 = [x'_1, y'_1]^T$ is the projection of point P_1 onto the camera plane. The coordinates of the point p'_1 can be found as follows using the pin-hole camera model's projection equations:

$$\mathbf{p} = \frac{f}{Z}\mathbf{P} \qquad \begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \tag{3.20}$$

where f is the focal length of the camera. It is assumed that when the head pose is changed point P_1 will become the point $P_2 = [X_2, Y_2, Z_2]^T$. The projection of P_2 onto the camera plane will be $p'_2 = [x'_2, y'_2]^T$ which is the point that the observation stage will produce for the corresponding feature point. The coordinates of p'_2 can also be found from P_2 using equation 3.20.

Since P_2 is assumed as the new location of P_1 after the head pose change, P_1 can be written in terms of P_2 and the head pose parameters, (T_x, T_y, T_z) and $(\theta_x, \theta_y, \theta_z)$, as

$$P_1 = R \cdot P_2 + T \tag{3.21}$$

where R is the rotation matrix defined as follows:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\theta_x) & -\sin(-\theta_x) \\ 0 & \sin(-\theta_x) & \cos(-\theta_x) \end{bmatrix} \cdot \begin{bmatrix} \cos(-\theta_y) & 0 & \sin(-\theta_y) \\ 0 & 1 & 0 \\ -\sin(-\theta_y) & 0 & \cos(-\theta_y) \end{bmatrix} \cdot \begin{bmatrix} \cos(-\theta_z) & -\sin(-\theta_z) & 0 \\ \sin(-\theta_z) & \cos(-\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

and $T = [-T_x, -T_y, -T_z]^T$ is the translation vector. The head pose parameters taken from the tracker are the translation and rotation parameters when P_1 (the point when the is on previous position) turns into P_2 (the point when the is on current position). Thus, to define the transformation from P_2 to P_1 these parameters are negated.

By writing the P_2 in equation 3.21 in terms of p'_2, since the coordinates of p'_2 are known from observation stage, the coordinates of P_1 can be found. Notice that the z-axis parameter of P_2 is unknown. But the z-axis parameter of P_1 can be found from the first head pose parameters obtained by the tracker. By using the z-axis parameter of P_1, the z-axis parameter of P_2 can be easily found using equation 3.21. According to this procedure, the observation data is transformed back and tracking can continue without the effects of head movement.

On the other hand, the approach where the spatial distances in the spatial model are updated is different from the previous approach. Consider Figure 3.8. As in the previous approach, the points P_1 and P_2 are two 3-D feature points on the face and they are spatially connected. Their projections on the camera plane are p'_1 and p'_2. When the head pose is changed, it is assumed that P_1 will become P_3 and P_2 will become P_4. Since the transformation parameters are known from the head pose tracker, the coordinates of the new points, P_3 and P_4, can be found using equation 3.21. Notice that, as in the previous approach, the z-axis parameters of P_1 and P_2 are not known. But these parameters can also be found from the first head pose parameters obtained by the tracker. According to equation 3.20, P_3 and

P_4 can be projected onto the camera plane as p'_3 and p'_4. This projection provides the exact coordinates of the new points on the image. Using these coordinates the spatial distances on x-axis and y-axis in the model can be easily updated.

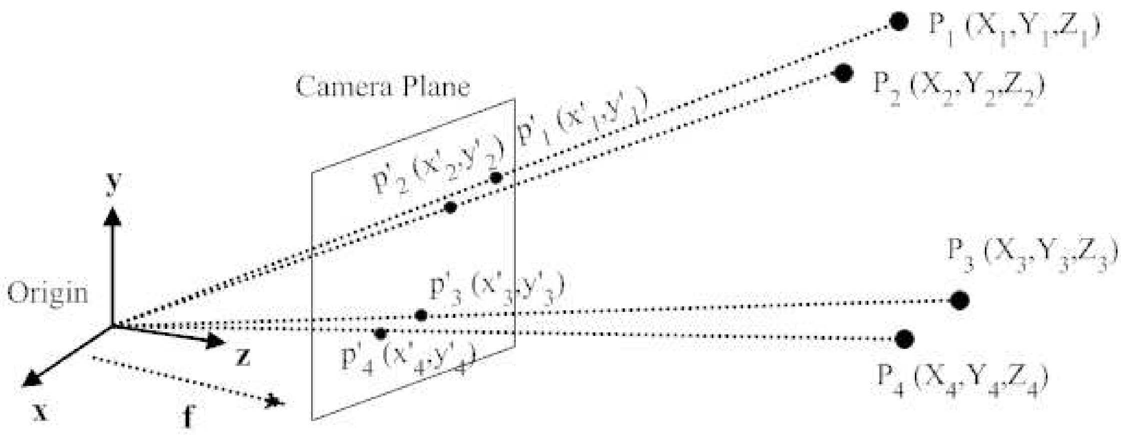


Figure 3.8: The illustration of updating spatial distance procedure.

Tracking Using Reliable Points

Other than the methods explained above, head pose information can be obtained by using feature points whose movement only depends on head movement. Because the movement of these feature points only depends on head movement, these points can give information about the movement of the head. To get reliable information from these points, one requirement is that these points should be reliably tracked. One example is inner eye corner points. The inner eye corner points can be easily detected by the observation stage, due to their edge structure, and their movement only depends on head movement. So these points can be used to obtain the pose information and update spatial distances in the spatial model (subsection 3.1.2), such as the distances between inner and outer corner points.

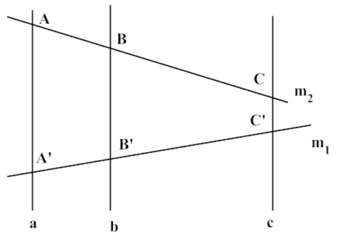


Figure 3.9: The Affine-ratio rule.

Updating is based on using the Affine-ratio rule [3]. Consider Figure 3.9. Affine-ratio is defined assuming there are three parallel lines (a, b, c) and there are two arbitrary lines (m_1, m_2) that pass through these lines. The rule satisfies that the ratio between distances $(|AB|, |BC|$ and $|A'B'|, |B'C'|)$ is always constant $(\frac{|AB|}{|BC|} = \frac{|A'B'|}{|B'C'|})$, i.e. it is affine invariant. To apply this ratio rule to the proposed model, it is assumed that the projection of the 3-D feature points on the camera plane is orthographic projection. Ideally the assumption should be perspective projection and in this case the cross-ratio invariance should be used. But to use such an invariance, there is a need of three reliable points that are on the same line (Chapter 10 of [45]). Since there are not any three reliable points on face that are on the same line, the affine-ratio invariance is selected for updating.

Consider A,B,C points as the true point locations (like the ones that are initially marked) and there is a spatial connection between point B and point C. Then assume that, A',B',C' points are the new point locations after the head pose is changed. If A',B' points are assumed to be reliably tracked points then the spatial distance between B' and C' in the model can be updated using the ratio rule. For different head poses, the application of this procedure on eye corner feature points is illustrated in Figure 3.10.

In Figure 3.10 it is assumed that there is a spatial connection between eye corner points (P_1 and P_2), (P_3 and P_4). Consider that the points P_1, P_2, P_3, P_4 are true point locations that are initially marked and points P_2 and P_3 are the points that can be reliably tracked. The spatial distances between eye corner points can be easily updated for the new position using the Affine-ratio as follows:

$$\frac{\triangle x_1}{\triangle x_2} = \frac{\triangle x'_1}{\triangle x'_2} = \frac{\triangle y'_2}{\triangle y'_1} = \frac{\triangle x''_1}{\triangle x''_2} = \frac{\triangle x'''_1}{\triangle x'''_2} = \frac{\triangle x''''_1}{\triangle x''''_2} \tag{3.23}$$

It can be concluded that the ratio rule can be used to update the spatial distances (between spatially connected feature points) in the model. Notice that to apply this rule, there is a need of at least two reliably tracked points. Usually the inner eye corner points can be selected as the reliable points. But in the case of the head rotations around x-axis and y-axis these points can be occluded by other feature points because of the head movement. This can be a limitation for the tracker (make it work up to a point where feature points begin to occlude each other).

Discussion

Considering the approaches proposed to handle the 3-D movements of the head, since the head pose is an additional and a very useful information, tracking facial features while the head pose is known will give better results. In a scenario

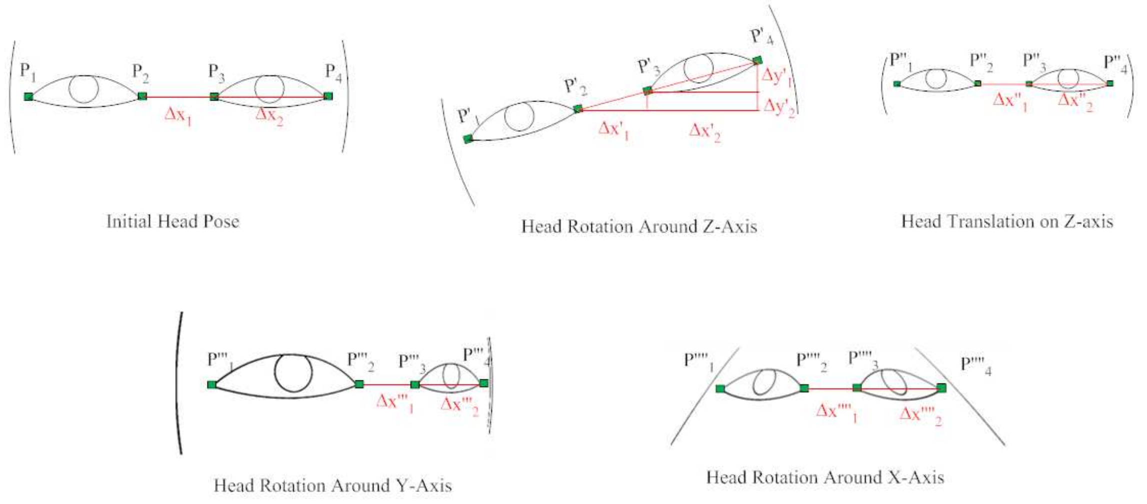


Figure 3.10: The application of Affine-ratio rule on eye corner feature points for different head poses.

where the head pose information can not be obtained, although there is the limitation above, the proposed method can still work and it can continue facial feature tracking using the reliable points.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter the performance of the proposed method explained in the previous chapter is demonstrated on data recorded in various conditions. The setup and the parameter selection procedure is explained in section 4.1. In section 4.2, data that consist only of head movements is used to test the framework's performance in the case of simple movement. Since the main motivation of this thesis is facial expression recognition systems, the proposed method is tested with videos that include facial gestures; the results of which are presented in section 4.3. Finally Section 4.4 contains the results of the proposed method applied to real-world data recorded in a vehicle environment.

4.1 Setup & Parameter Selection

The data used in experiments are grayscale videos with 640x480 resolution. Videos are recorded with a rate of 30 frames per second without compression. As explained in subsections 3.1.1, 3.1.2, and 3.1.3, the parameters of the corresponding noise covariance matrices should be selected based on the experimental data. But here for simplicity these parameters are selected identically in all scenarios. In these identically selected covariance matrices; to decrease the number of parameters and to make simple assumptions, the position (and the velocity) errors on x and y axes are selected identically for all features. The selected transition model noise (Q) and the observation model noise (R) are as follows:

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \qquad R = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \tag{4.1}$$

In the transition model it is assumed that the position errors on both x and y axes (σ_x^2, σ_y^2) are ± 4 pixels and the velocity errors on both x and y axes (σ_u^2, σ_v^2) are ± 2 pixels. For observation model the position errors on both x and y axes ($\sigma_x''^2, \sigma_y''^2$) are assumed as ± 2 pixels.

In the spatial model, covariance matrices are formed in $information\ form$. Consider the case in which there are constraints only on x and y axis positions for two spatially connected feature points. Assuming the position errors on x and y axes $(\sigma_x'^2, \sigma_y'^2)$ are ± 2 pixels, the inverse covariance matrix (Σ^{-1}) for the corresponding spatial model is selected as follows:

$$\Sigma^{-1} = \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.2}$$

As mentioned above, the position errors $(\sigma_x'^2, \sigma_y'^2)$ are selected as equal for simplicity.

Since the covariance matrices for hidden variables are updated in each algorithm step, the prior covariance matrix, explained in section 3.1, is chosen with large error values. Assuming the uncertainties on positions and velocities on both x and y axes are ± 10 pixels and ± 5 pixels respectively, the selected prior covariance matrix is as follows:

$$\Sigma_{x_0} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \tag{4.3}$$

The proposed method is run under Matlab v6.5 on a Celeron 1.5Ghz computer. The process time for the non-optimized code is slower than the real-time processing rates. The slowest part of the algorithm is the convolution operation performed in the Gabor filter-based pre-processing procedure described in Section 3.2. The inference part for the proposed framework is much faster than the convolution part.

4.2 Head Tracking Experiments

First of all, the proposed framework is tested in simple scenarios involving only head movement. Head movement is a simpler movement than the movement of facial components such as eyes, mouth, etc. In this section, the results of the proposed method for such a case are shown. Since the main motivation of this thesis is facial feature tracking not head tracking, the purpose of showing these results is not to compete with other head tracking methods in the literature.

Four eye corner points are selected to be tracked to capture head movement. As shown in Figure 4.1, simply eye corner points are connected spatially to each other for each eye individually. These spatial connections put constraints on x and y axes. To handle the head movements Affine invariance is used and the inner eye corners are selected as reliable points.

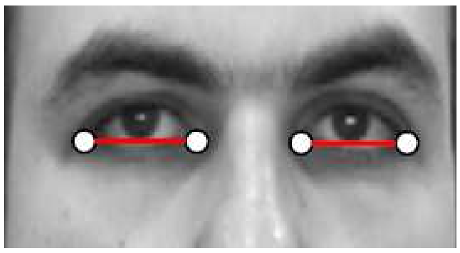


Figure 4.1: The selected feature points and the spatial connections for head movement sequences.

In Figure 4.2 the results for tracking these four points while there is a head rotation around z-axis and a translation in x-axis are shown. To display the performance of the spatial difference update procedure, the real difference vectors (in cm) between the left eye corners (measured on face) and the projection of these differences to the camera plane (in pixels) are shown in Table 4.1. Since the real differences are the real distances between the eye corner points they do not change. As it can be seen in the results, the proposed method performs well in the case of head movements and the spatial differences are updated properly. To test the occlusion detector in the case of head movements the framework is applied to videos containing occlusions. Consider a case that includes head translation in z-axis, rotation around z-axis and occlusion of one facial feature by hand. The results for such a case are shown in Figure 4.3. In Table 4.2 the real difference vectors and the projection of the difference vectors on to the camera plane in this sequence are shown. It can be observed that even in the case of occlusion on a facial feature, the algorithm continues tracking successfully. The occlusion detector senses the occlusion on the left eye outer corner and the information coming from data of the occluded feature is closed, but the information from temporal and spatial relations is sufficient to continue tracking.

4.3 Facial Expression Experiments

One of the main motivations of this thesis is solving the problem of facial feature tracking in a facial expression analysis system. In this section, tracking re-

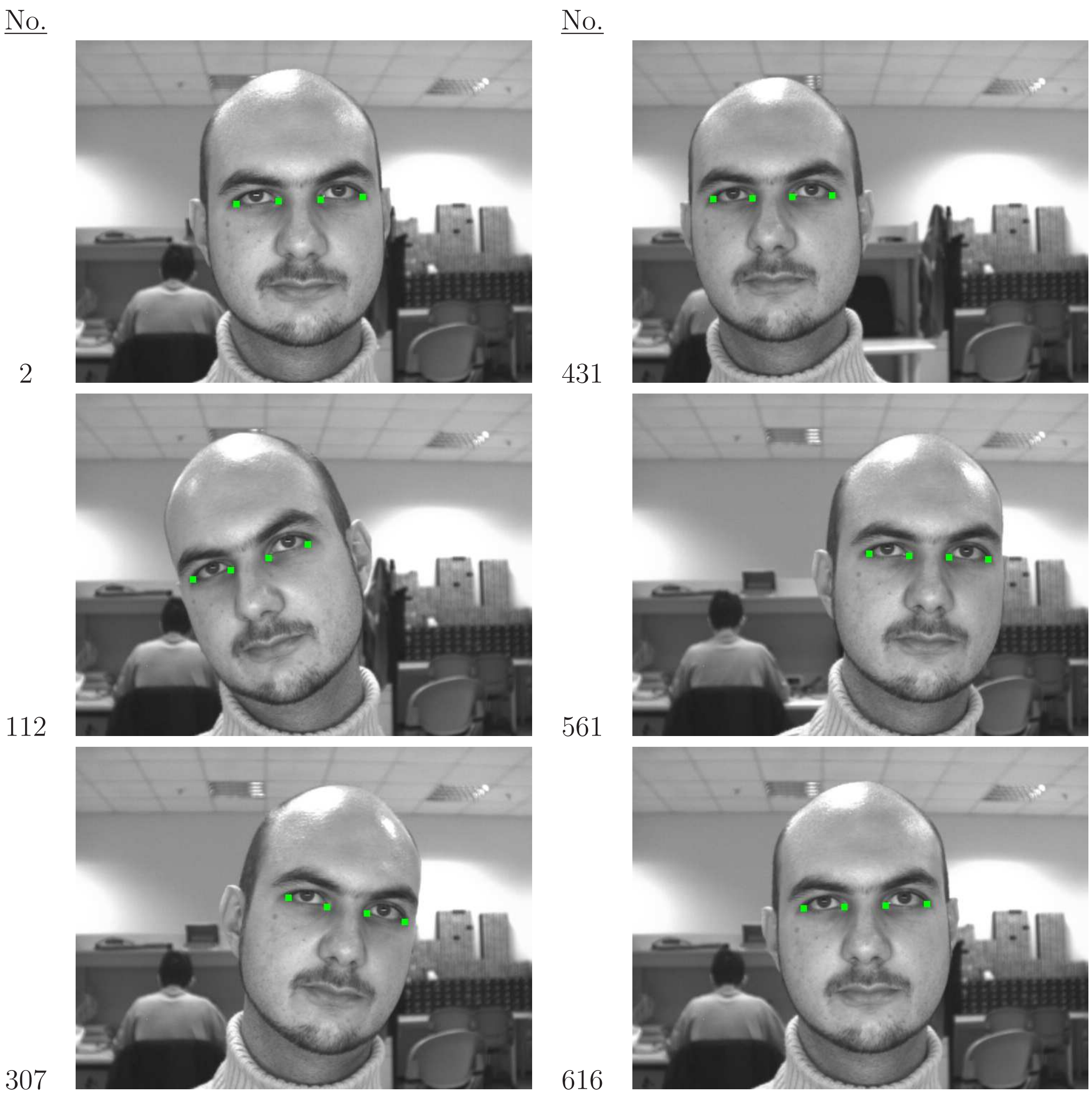


Figure 4.2: Tracking results of the proposed method for a sequence that includes head rotation around z-axis and translation in x-axis.

No.	$\triangle x$ (in pixels)				$\triangle x$ (in cm)			
	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$
1	59.00	2.00	0.00	0.00	3.80	0.4	0.00	0.00
2	58.82	4.99	0.00	0.00	3.80	0.4	0.00	0.00
112	51.50	14.01	0.00	0.00	3.80	0.4	0.00	0.00
307	53.28	12.69	0.00	0.00	3.80	0.4	0.00	0.00
431	55.74	1.19	0.00	0.00	3.80	0.4	0.00	0.00
561	55.71	2.28	0.00	0.00	3.80	0.4	0.00	0.00
616	57.45	2.48	0.00	0.00	3.80	0.4	0.00	0.00

Table 4.1: The updated difference vectors and the real spatial difference vectors between the left eye corner points for the sequence in Figure 4.2.

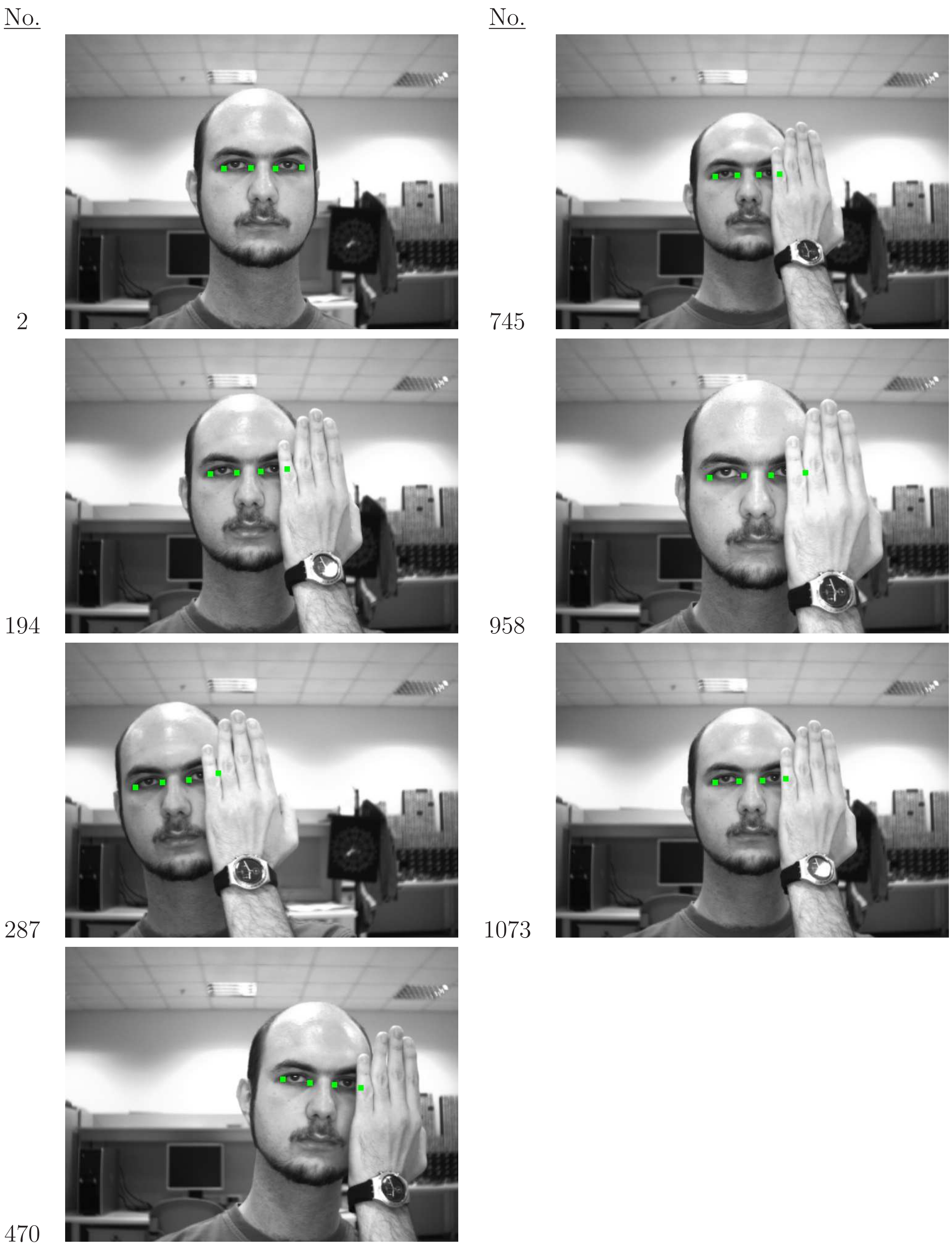


Figure 4.3: Tracking results of the proposed method for a sequence that includes head rotation around z-axis, translation in z-axis, and an occlusion of a facial feature by hand.

No.	$\triangle x$ (in pixels)				$\triangle x$ (in cm)			
	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$
1	43.00	0.00	0.00	0.00	3.80	0.4	0.00	0.00
2	42.99	1.07	0.00	0.00	3.80	0.4	0.00	0.00
194	42.41	3.53	0.00	0.00	3.80	0.4	0.00	0.00
287	43.97	10.38	0.00	0.00	3.80	0.4	0.00	0.00
470	42.85	5.17	0.00	0.00	3.80	0.4	0.00	0.00
745	33.86	1.40	0.00	0.00	3.80	0.4	0.00	0.00
958	55.81	4.67	0.00	0.00	3.80	0.4	0.00	0.00
1073	37.88	2.79	0.00	0.00	3.80	0.4	0.00	0.00

Table 4.2: The updated difference vectors and the real spatial difference vectors between the left eye corner points for the sequence in Figure 4.3.

sults based on this motivation are shown. Basically some distinct facial gestures are selected for the experiments. These include mouth opening, eye closure, smiling, and eye (wide) opening. The sequence consisting of such gestures is recorded continuously, and also includes natural eye blinking movements. 12 feature points are selected including 4 points for each eye and 4 points for the mouth. In Figure 4.4, the selected feature points and the spatial connections, that are simply selected, for such an experiment are illustrated. In Table 4.3, the spatial constraints of the corresponding connections are shown by using the projection of the initial spatial differences and the real differences. It should be noticed that the distance values are dependent on the scenario.

Connected Couples	$\triangle x$ (in pixels)				$\triangle x$ (in cm)			
	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$
1-4	28.00	0.00	0.00	0.00	1.90	0.00	0.00	0.00
3-4	50.00	0.00	0.00	0.00	3.80	0.00	0.00	0.00
7-4	28.00	0.00	0.00	0.00	1.90	0.00	0.00	0.00
2-5	24.00	0.00	0.00	0.00	1.90	0.00	0.00	0.00
4-5	0.00	2.00	0.00	0.00	0.00	0.05	0.00	0.00
6-5	50.00	0.00	0.00	0.00	3.80	0.00	0.00	0.00
8-5	24.00	0.00	0.00	0.00	1.90	0.00	0.00	0.00
9-11	42.00	0.00	0.00	0.00	2.95	0.00	0.00	0.00
10-11	0.00	4.00	0.00	0.00	0.00	0.05	0.00	0.00
12-11	42.00	0.00	0.00	0.00	2.95	0.00	0.00	0.00

Table 4.3: The selected spatial constraints of the corresponding spatial connections shown in Figure 4.4 for the experiments in section 4.3.

The results for the sequence described above are shown in Figure 4.5-b. To make a comparison, an algorithm that uses only temporal relations and that is based on Kalman filter [23] is also applied to the same sequence and the results are shown in

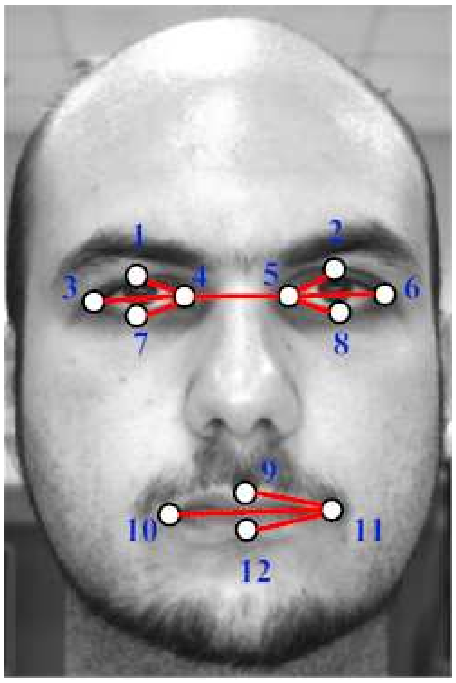
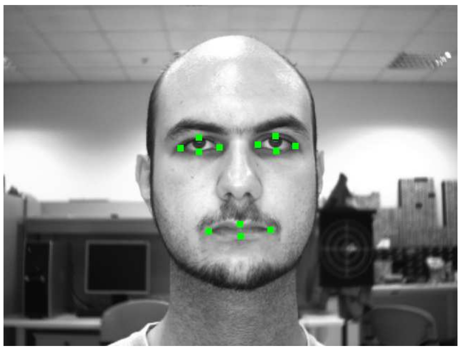
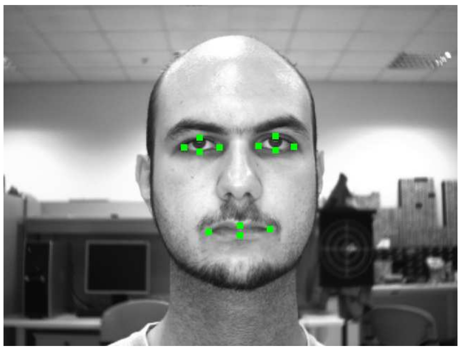


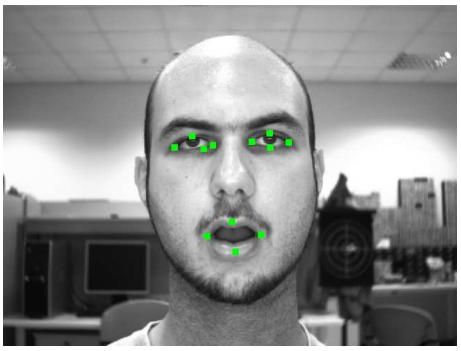
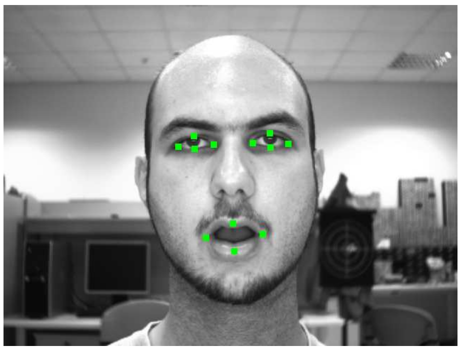
Figure 4.4: The selected feature points and the spatial connections for facial expression experiments.

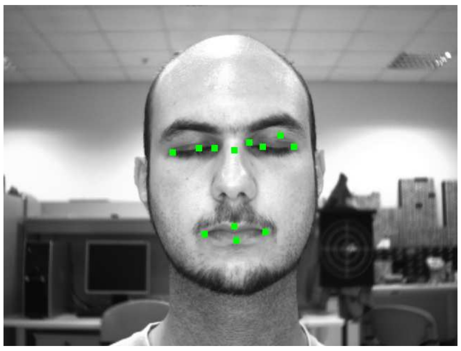
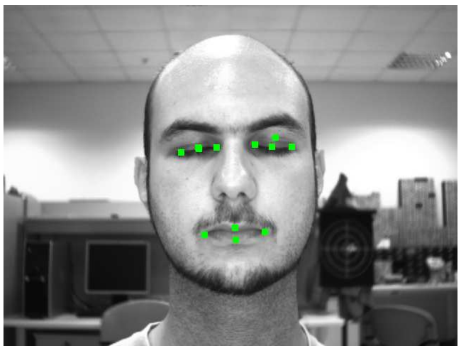
Figure 4.5-a. It can be clearly seen that the proposed method gives more reliable results, while drifts occur for the other algorithm. Most of the drifts occur in the eye region points because eye blinking and eye closure cause the observed region to change rapidly. Since the other algorithm doesn't use spatial relations, these rapid changes make it lose tracking. It should be also noticed that although the resolution of the video frames are 640x480, the size of the head region occupies a much smaller region. After all it can be concluded the proposed method can successfully track even the size of the head is small.

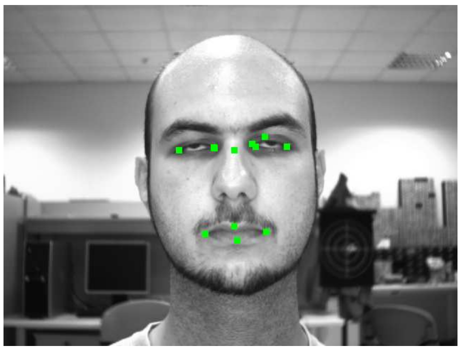
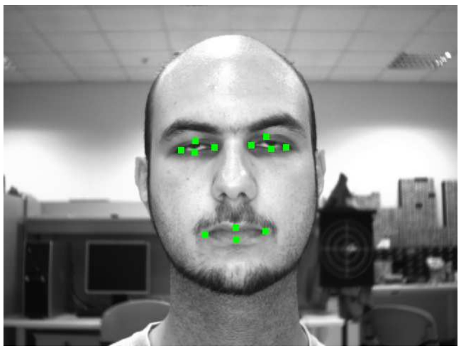
To extend these results, some variations of this sequence have been used to test the tracker under different conditions. In subsection 4.3.1, the tracker is tested in a sequence in which there is an occlusion on mouth and eye points. In many practical applications, available video resolution is not much higher than that can be provided for example by webcams. To test the practicality of the method, it has been tried with videos that have low resolution. Such results are shown in subsection 4.3.2. Another practical application problem can be the illumination changes. These tests are displayed in subsection 4.3.3. One of the main contributions of this thesis is the facial feature tracking in the case of head movements. In subsections 4.3.4 and 4.3.5 the tracker's performance when there is in-plane and out-of-plane head motion are given respectively. In these experiments it is assumed that there is not any external source that can give the head pose parameters. The head pose information is obtained by using the reliably tracked points.

No.



2



272



377



432

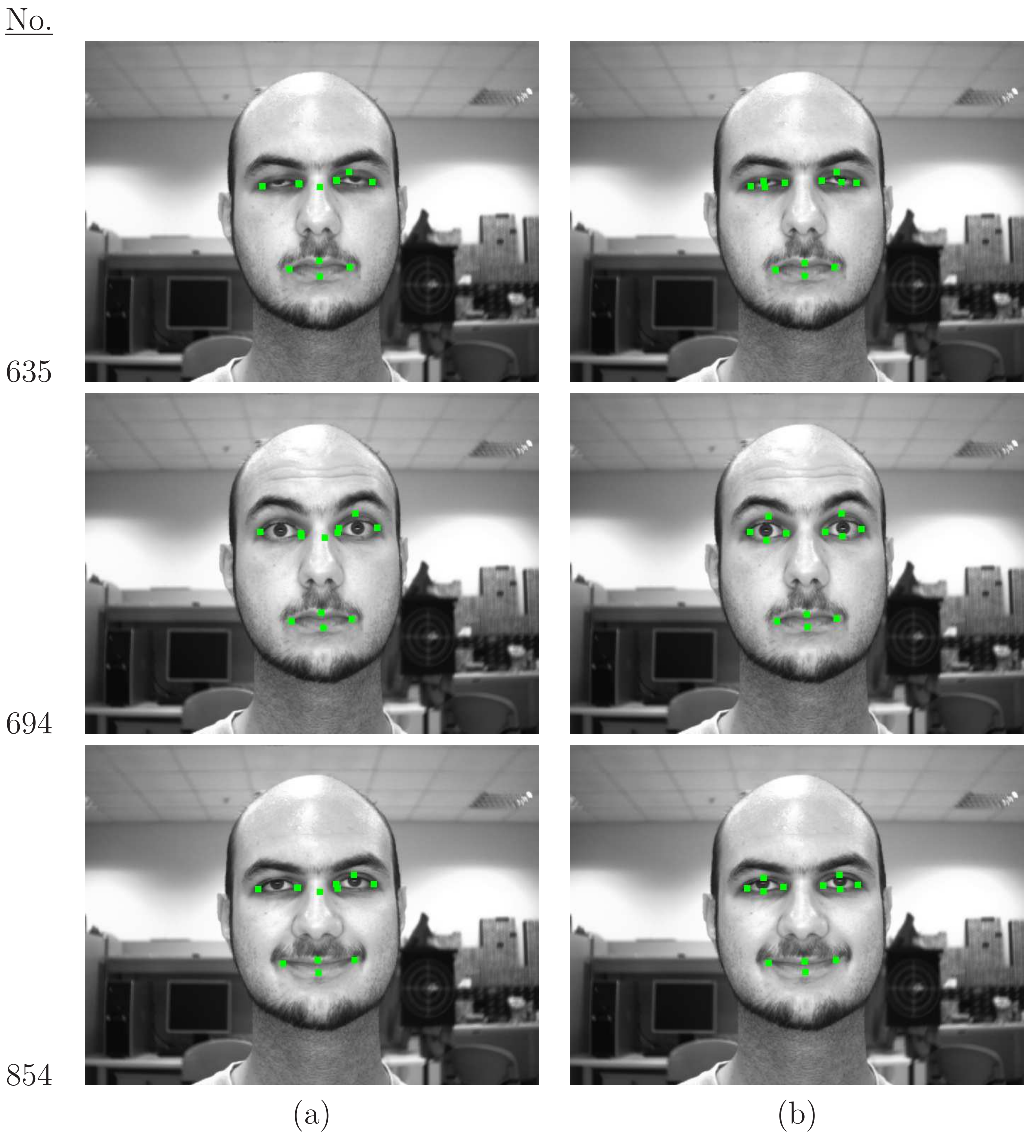


Figure 4.5: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes facial gestures such as mouth opening, eye closure, opening eyes wide, and smiling.

4.3.1 Under External Occlusion

The purpose of this experiment is to test the performance of the occlusion detection part, explained in subsection 3.4.1, in facial expression sequences. Here, a sequence that consists of the occlusion of a mouth feature point and an eye corner point as well as facial gestures like mouth opening and wide opening eyes is used. There is also natural eye blinking movements in the sequence. The proposed tracker's performance in such a sequence is illustrated in Figure 4.6-b. In Figure 4.6-a, the results of an existing technique [23] is shown. To make a proper comparison, the same occlusion detector is also used for the existing technique.

It can be observed that in the case of data loss because of occlusion, spatial constraints enable the tracking to continue. Without the spatial relations, the positions of the points cannot be accurately estimated, as demonstrated by the results of [23] in Figure 4.6-a. When the occlusion occurs the information coming from the data term of the occluded feature is closed in both tests. The proposed method continues to track using the spatial information and temporal information, but the other algorithm fails to track because only temporal information is not enough to track. Using only temporal relationships will cause the feature points to go on with constant velocity (due to the nature of the linear state space model used). The drifts are because of this lack of information. As explained in subsection 3.4.1, the best match similarity outputs for the corresponding feature points give quantitative information about occlusion. These similarity values for four eye corner points are given in Figure 4.7. It can be clearly seen that thresholding the similarity values below 0.95 (red line) can detect the occlusion.

4.3.2 Lower Video Resolution

There are many facial feature tracking methods in the literature and most of these methods are tested in ideal cases. But the practical application of these systems can suffer from some assumptions. High video resolution is one of these assumptions. The higher the video resolution, the bigger the head region in frames which makes tracking easier. To test the proposed method in low-resolution the same sequence in Figure 4.5, this time with 320x240 and 160x120 resolutions, is used. Since the video data sizes are half or quarter of the previous sizes the wavelengths of Gabor filter in section 3.2 are selected as half or quarter of the previous wavelengths.

In Figure 4.8-a,b and 4.9-a,b the results for the method in [23] and the results for the proposed method are given respectively. It can be clearly seen that, when the video resolution is 320x240 the proposed method can continue tracking

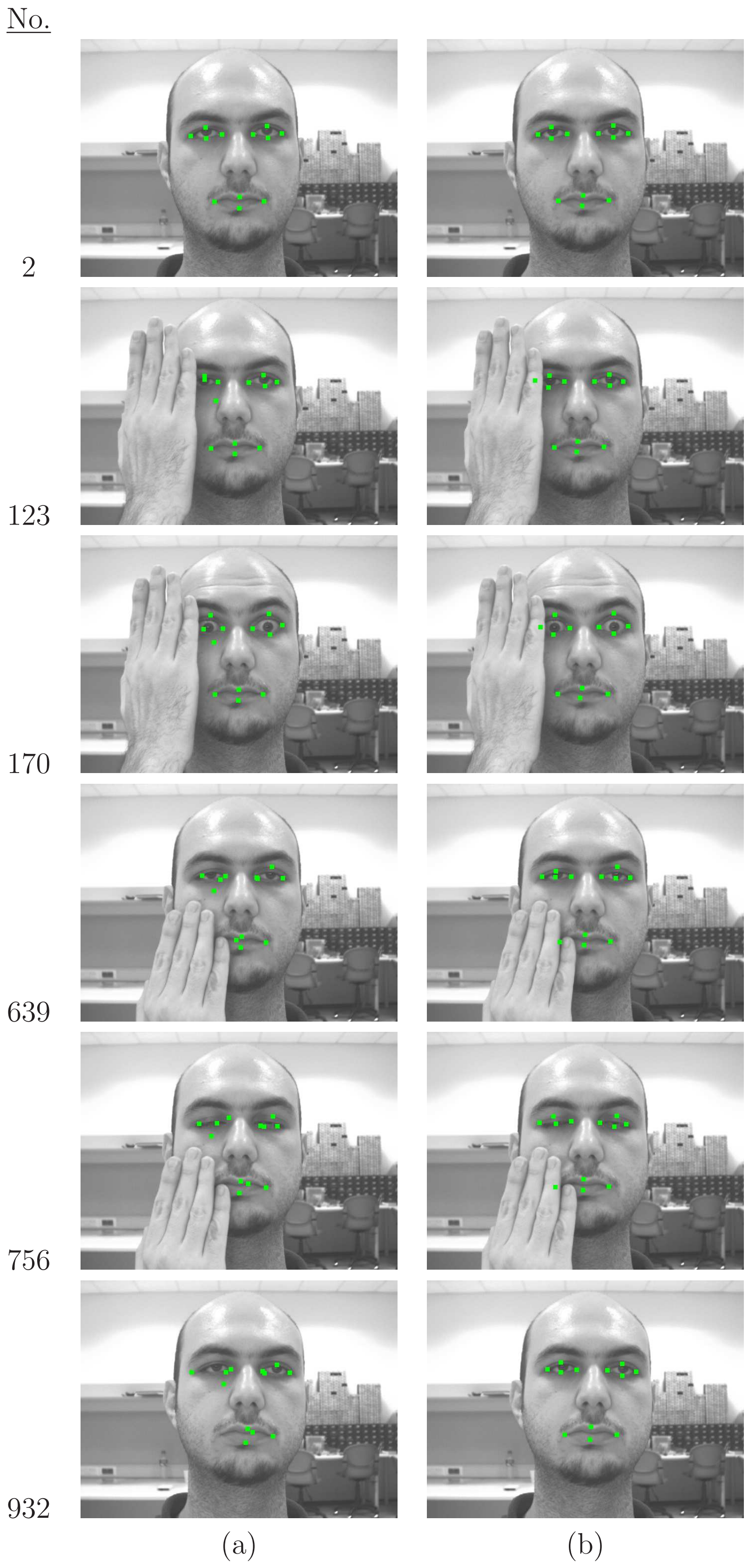


Figure 4.6: Tracking results of (a) the method in [23] (b) the proposed method for a sequence includes facial gestures and external occlusion by hand.

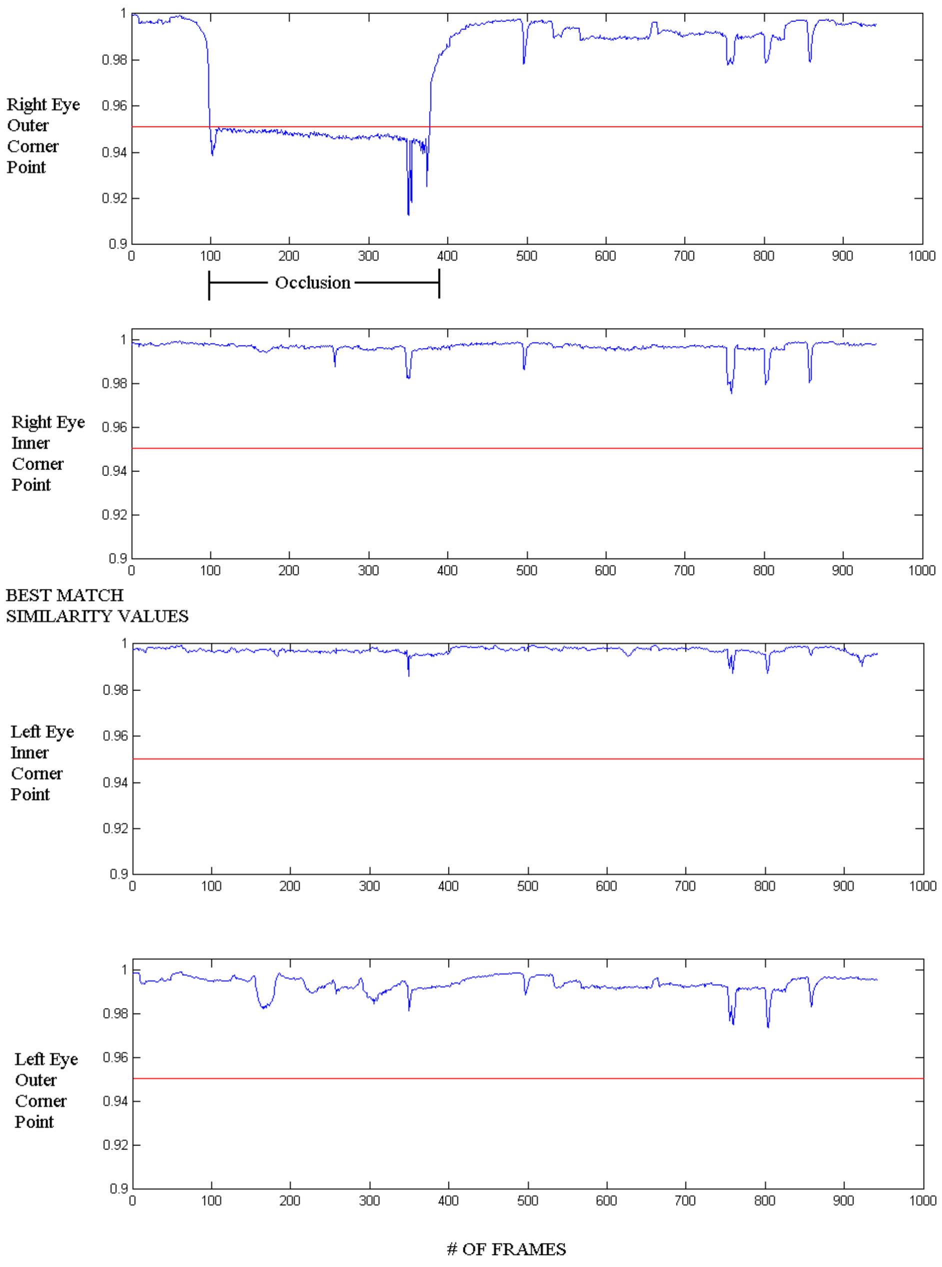


Figure 4.7: The similarity outputs for four eye corner point in the sequence shown in Figure 4.6. Red line represents the threshold value.

robustly. But when the video resolution is 160x120, the proposed method can lose tracking of some features. Here drifts occur because of the misleading information coming from the observation stage. The main reason is missing edge structure of the feature of interests because of the very low resolution data. Low resolution causes the edge structures to disappear. There are drifts in the other algorithm again because of the eye blinking and eye closure. Although the proposed method can lose tracking when the data have very low resolution (160x120) it can be concluded that, in the case of practical applications, when the video data resolutions are not so much low, the proposed method can still be a solution for the facial feature tracking problem.

4.3.3 Illumination Changes

In the previous subsection the robustness of the proposed method to limitations in video resolution has been examined. In this subsection, the issue of robustness to another important variable, illumination is considered. Illumination change is a well-known problem in practical applications. The illumination in scenes can change in time because of the nature of the real-world conditions. So this is a case that should be handled.

The results of the method in [23] and the proposed method is given in Figure 4.10-a,b respectively. Tracking results show that there can occur drifts when the illumination in the scene is very low (frame no.s: 312, 357, 415). But when the illumination is very high (frame no: 139) the proposed method tracks better than the existing technique. The reason of the drifts is again missing edge structure of the feature of interests. In the case of low illumination, Gabor filters cannot detect the feature locations due to the lack of visibility of the edge structure. This reduces the quality of the data fed into the system from the observations stage. But as compared to the results of the existing technique the proposed method gives more reliable results.

4.3.4 In-plane Head Motion

As mentioned in subsection 3.4.2, head movement is a natural human action that should be taken into account in facial feature tracking systems. In-plane head motion is the motion of head in three degrees of freedom : T_x, T_y, θ_z. In-plane head motion can be thought of more simple movements as compared with out-of-plane head motion. In this section the proposed method is applied to a sequence that consists of the combination of motions involving these three degrees of freedom. The sequence also includes a mouth opening gesture.

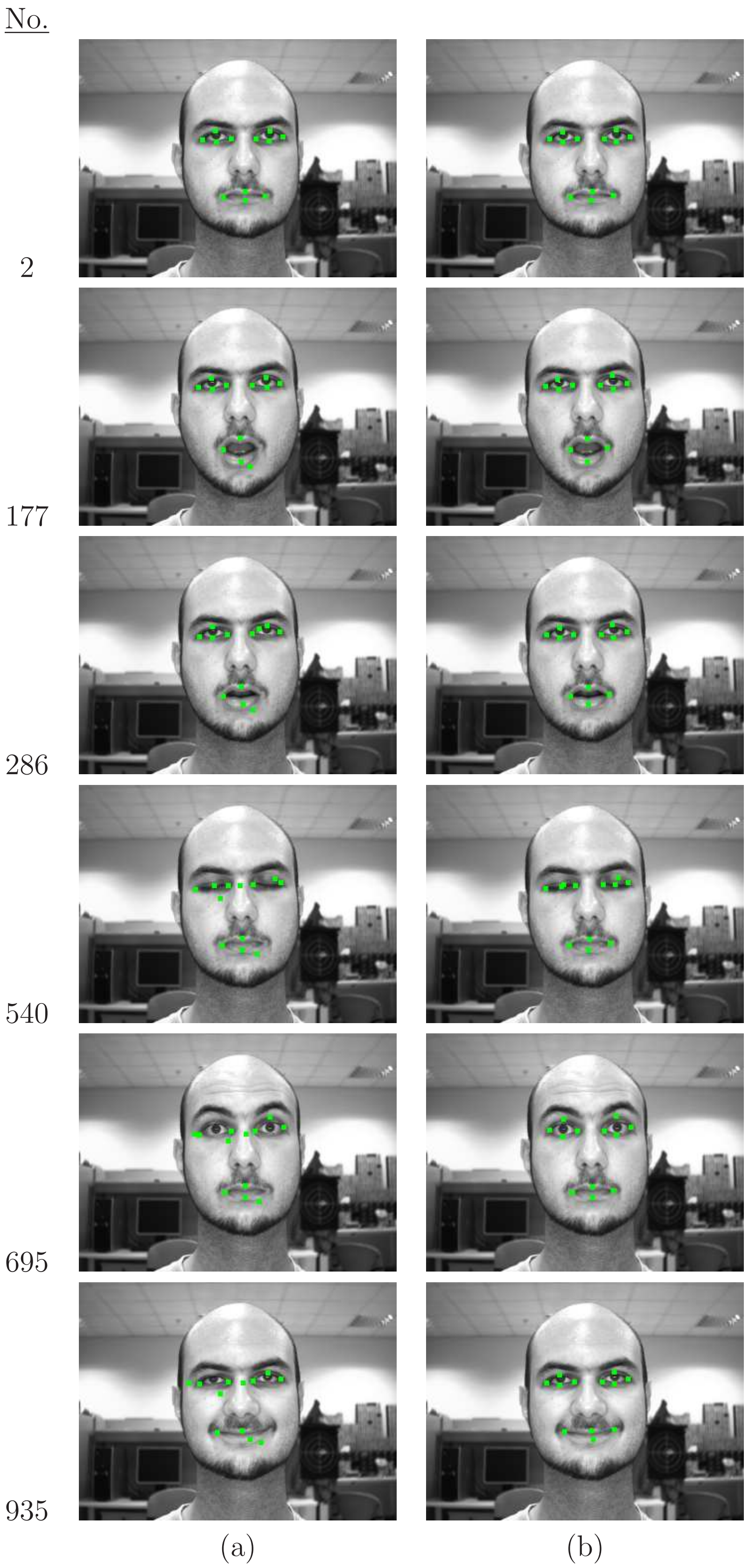


Figure 4.8: Tracking results of (a) the method in [23] (b) the proposed method for the same sequence in Figure 4.5 with 320x240 resolution.

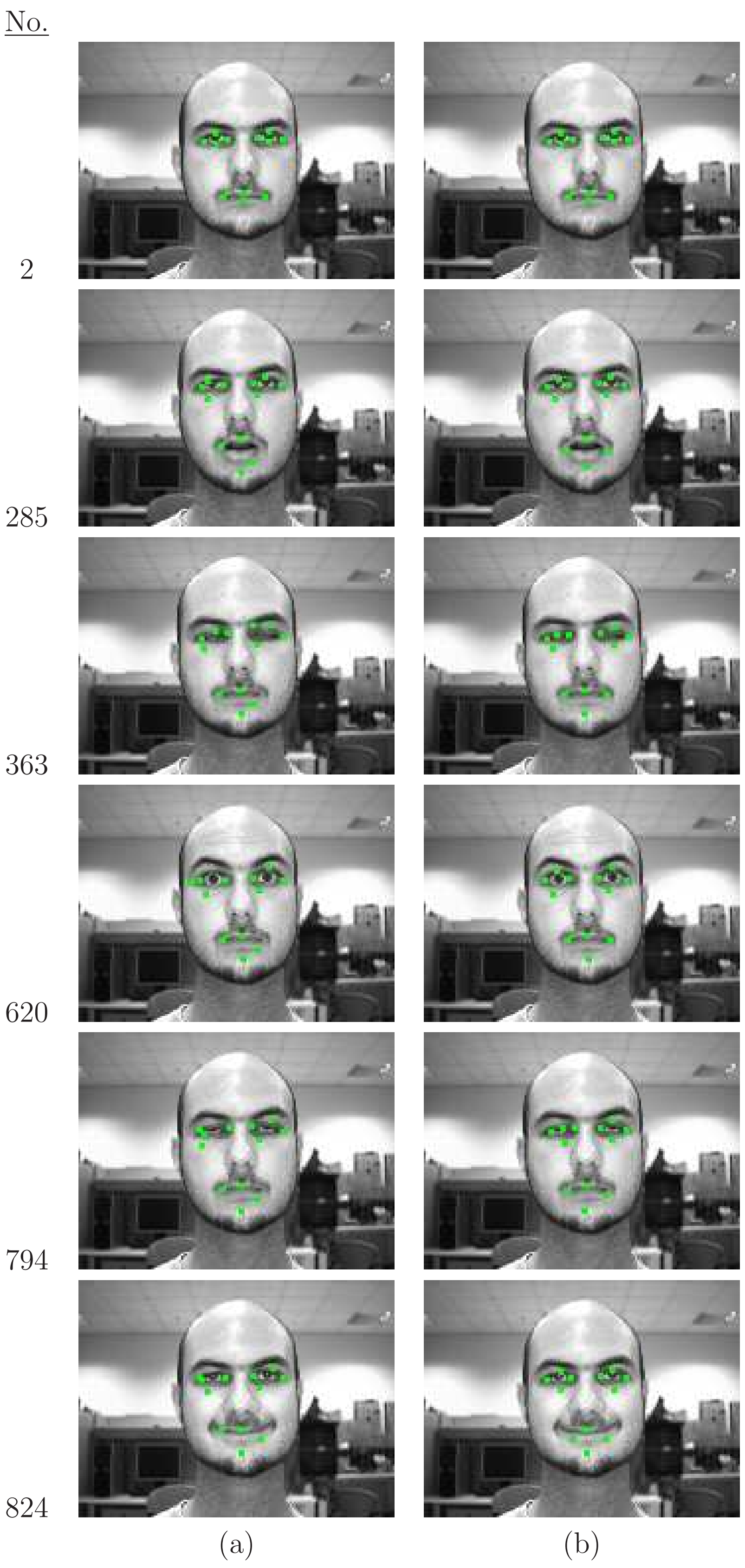


Figure 4.9: Tracking results of (a) the method in [23] (b) the proposed method for the same sequence in Figure 4.5 with 160x120 resolution.

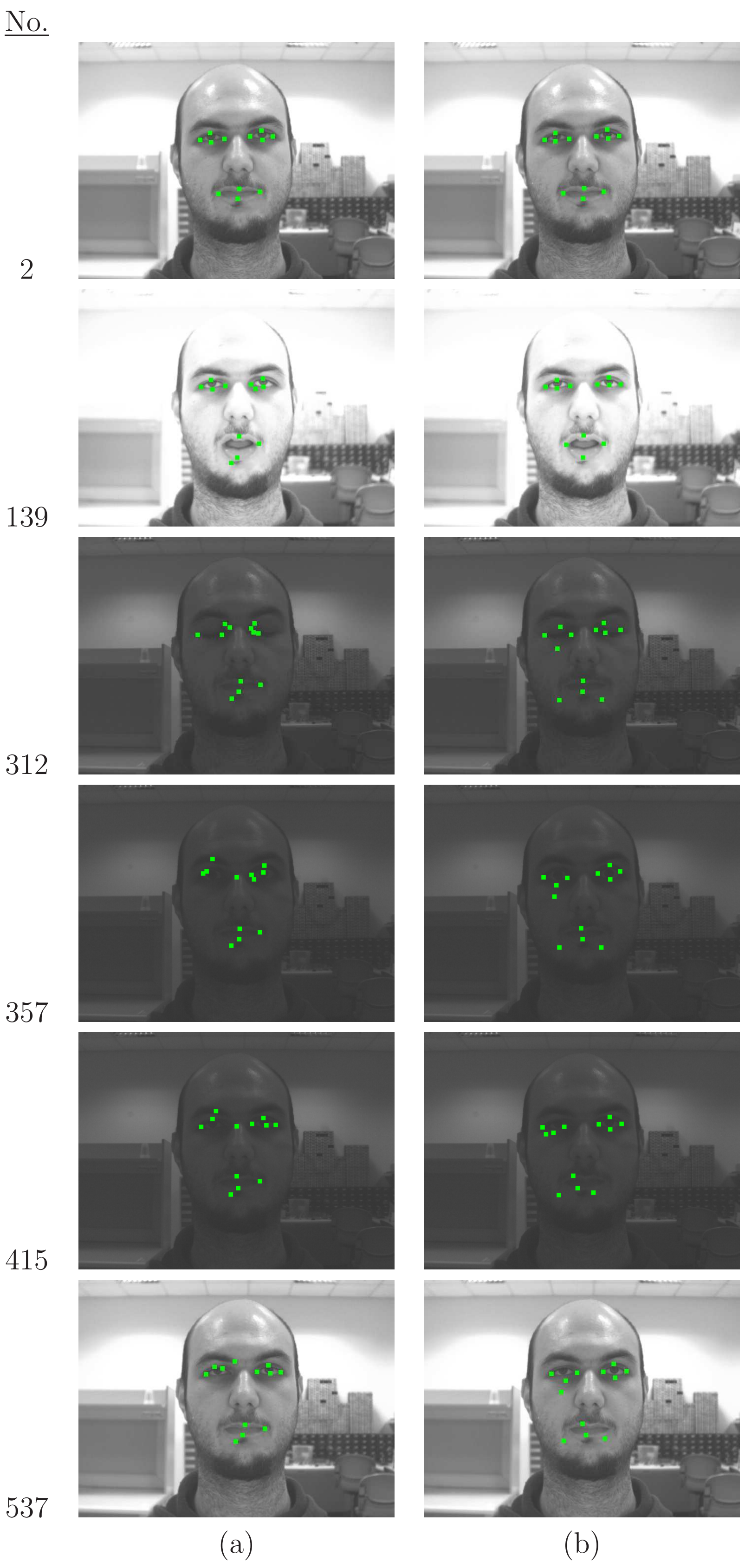
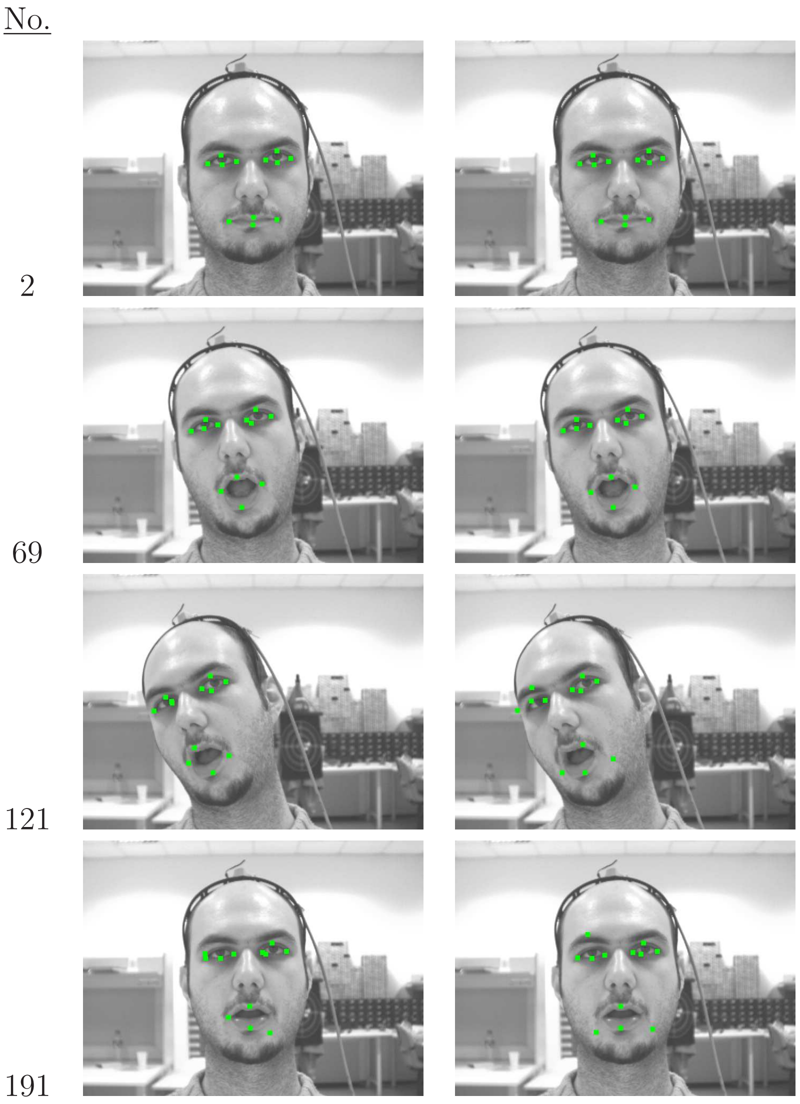


Figure 4.10: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes facial gestures and illumination changes.

As a solution to head movement cases two different approaches are given in subsection 3.4.2, one requiring and one not requiring information about the head pose parameters from an external source. Here, it is assumed that the head pose parameters can not be obtained by an external source. For this reason, the approach based on reliably tracked points is used to get the missing head pose information (by Affine invariance) and continue tracking. The left and right inner eye corner points are selected as reliably tracked points. The results of this approach for such a sequence explained above is given in Figure 4.11-b. The same sequence is tested using the method in [23] to make a comparison. These results are given in Figure 4.11-a.



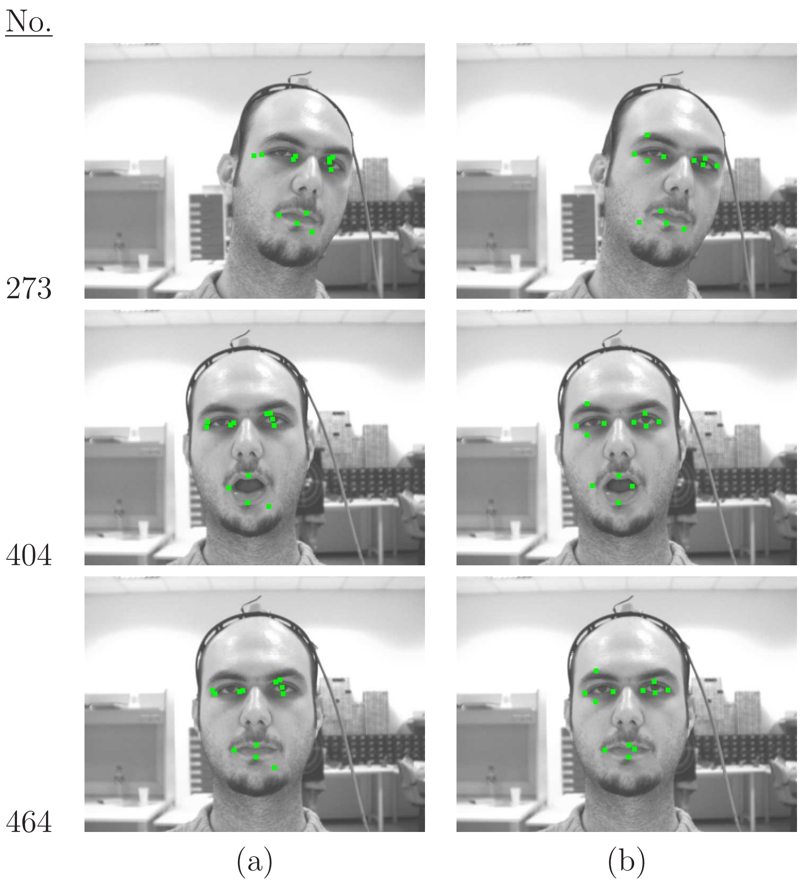


Figure 4.11: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes translation on x and y axes, rotation around z-axis and mouth opening.

The real spatial differences (in cm) and the projection of the distances on camera plane (in pixels) between the left eye corner points through the sequence in Figure 4.11 are given in Table 4.4. It can be clearly observed that the spatial distances are properly updated when the head rotates around z-axis. For example; when the head rotates left (frame no. 121) and right (frame no. 273) around z-axis, the 2-D distance on y-axis between eye corner points will be larger than the distance when the head is straight. According to the projected distances on y-axis in Table 4.4, it can be seen that this increment is properly handled.

Although the spatial distances are properly updated there occur some drifts in the results of the proposed method. The main reason of the drifts in the proposed method's results is the out-of-plane movements of the head. Here the assumption is in-plane head motion, but in practice, limiting the head to perfect in-plane motion is hard. It can be seen that in frames 121 and 273 there is a little rotation around x-axis which affects the reliable points. Thus, because of the drifts

No.	$\triangle x$ (in pixels)				$\triangle x$ (in cm)			
	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$
1	47.00	3.00	0.00	0.00	3.80	0.4	0.00	0.00
2	47.03	2.57	0.00	0.00	3.80	0.4	0.00	0.00
69	43.44	7.67	0.00	0.00	3.80	0.4	0.00	0.00
121	44.42	17.50	0.00	0.00	3.80	0.4	0.00	0.00
191	44.75	3.00	0.00	0.00	3.80	0.4	0.00	0.00
273	48.13	5.41	0.00	0.00	3.80	0.4	0.00	0.00
404	48.29	2.20	0.00	0.00	3.80	0.4	0.00	0.00
464	48.94	1.45	0.00	0.00	3.80	0.4	0.00	0.00

Table 4.4: The updated difference vectors and the real spatial difference vectors between the left eye corner points for the sequence in Figure 4.11.

in the reliable points the head pose information can not be obtained properly from these points and there occur drifts in all other points. In a case when the proper head pose information can be obtained from an external source these results will be improved.

4.3.5 Out-of-plane Head Motion

The out-of-plane motion is the motion of head in the following three degrees of freedom: T_z, θ_x, θ_y. For problems involving 2-D data, out-of-plane motion of the head can be considered as a more complex motion as compared with the in-plane motion. In this section, the proposed method is tested on three different sequences each of which includes one of these three motions and a facial gesture (mouth opening).

Figures 4.12, 4.13, and 4.14, show respectively the results for a sequence that includes rotation around y-axis (θ_y, yawing), rotation around x-axis (θ_x, pitching), and translation on z-axis as well as mouth opening. The results of these sequences for the method in [23] are shown in Figures 4.12-a, 4.13-a, and 4.14-a. Moreover, the results for the proposed method are shown in Figures 4.12-b, 4.13-b, 4.14-b.

In Table 4.5 the real spatial differences (in cm) and the projection of the distances on camera plane (in pixels) between the left eye corner points are given through the sequence in Figure 4.14. It can be observed that the spatial differences are properly updated when the head translates backwards and forwards on z-axis.

Considering all the results, it can be seen that there occurs some drifts in

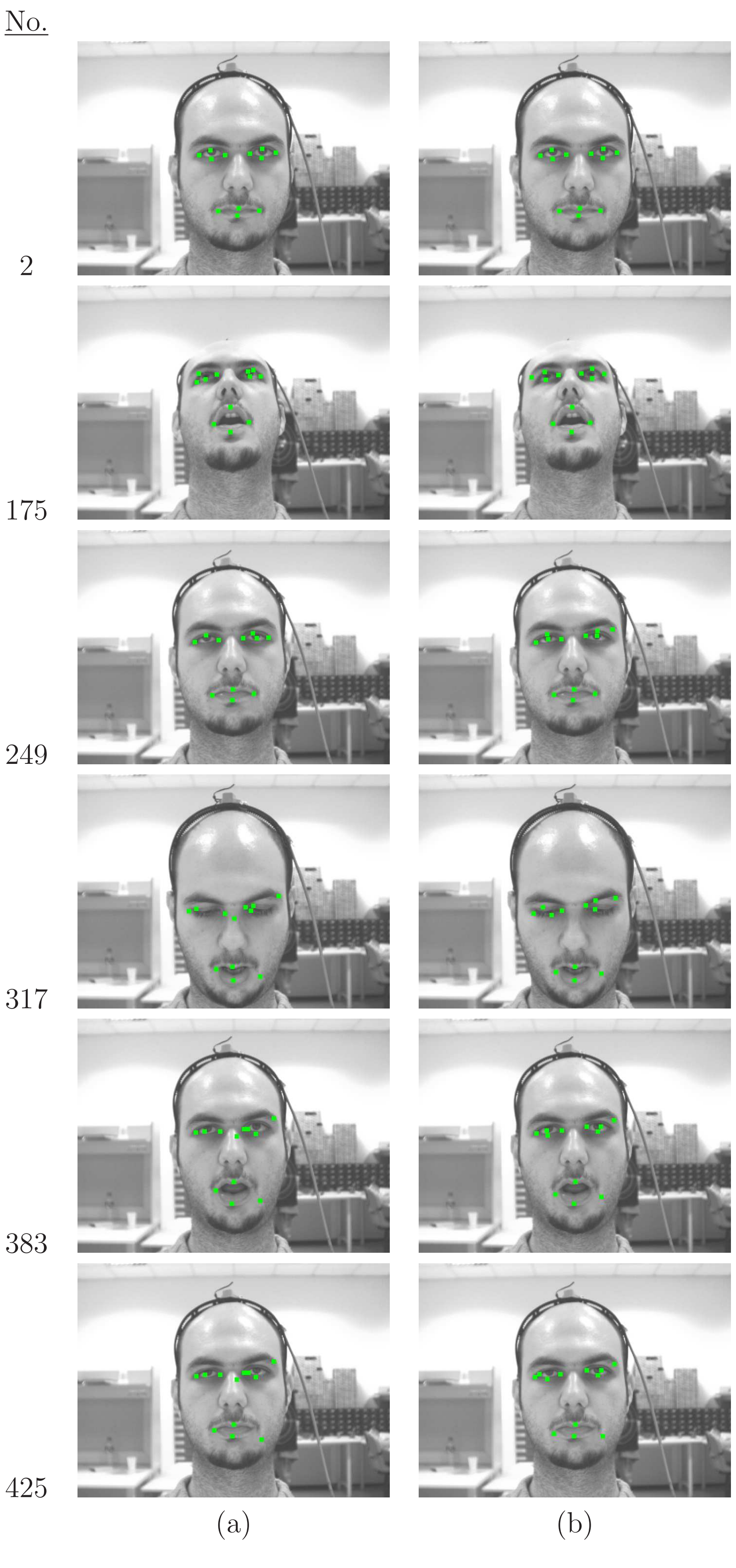
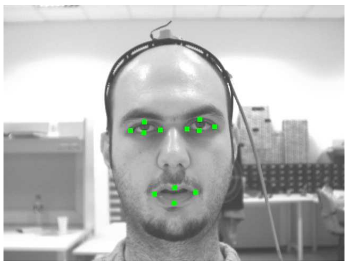
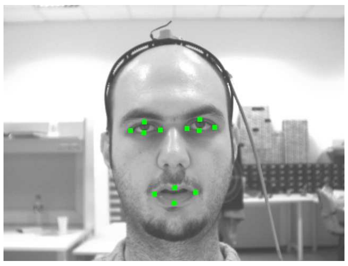
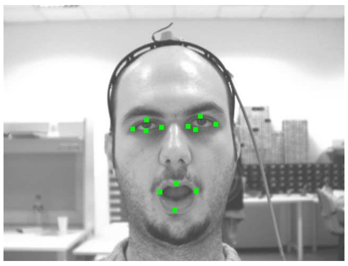
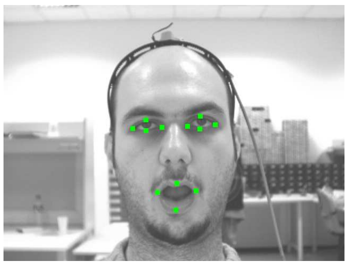


Figure 4.12: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes rotation around y-axis (θ_y) and mouth opening.

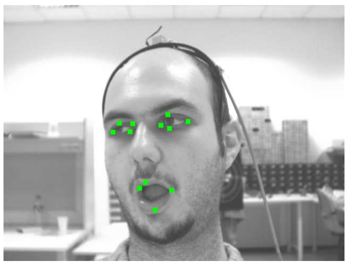
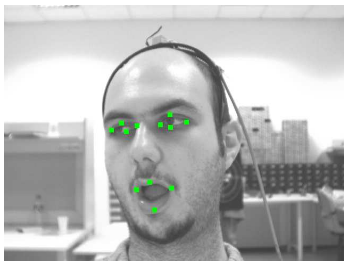
No.

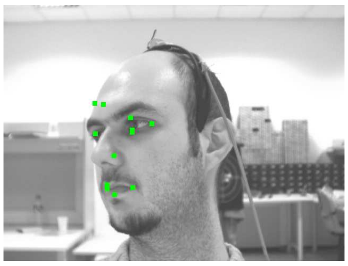
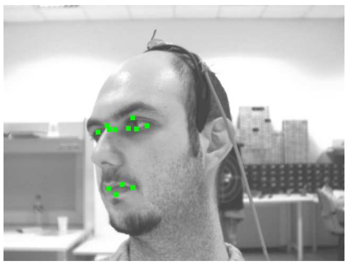
2

40

84

147

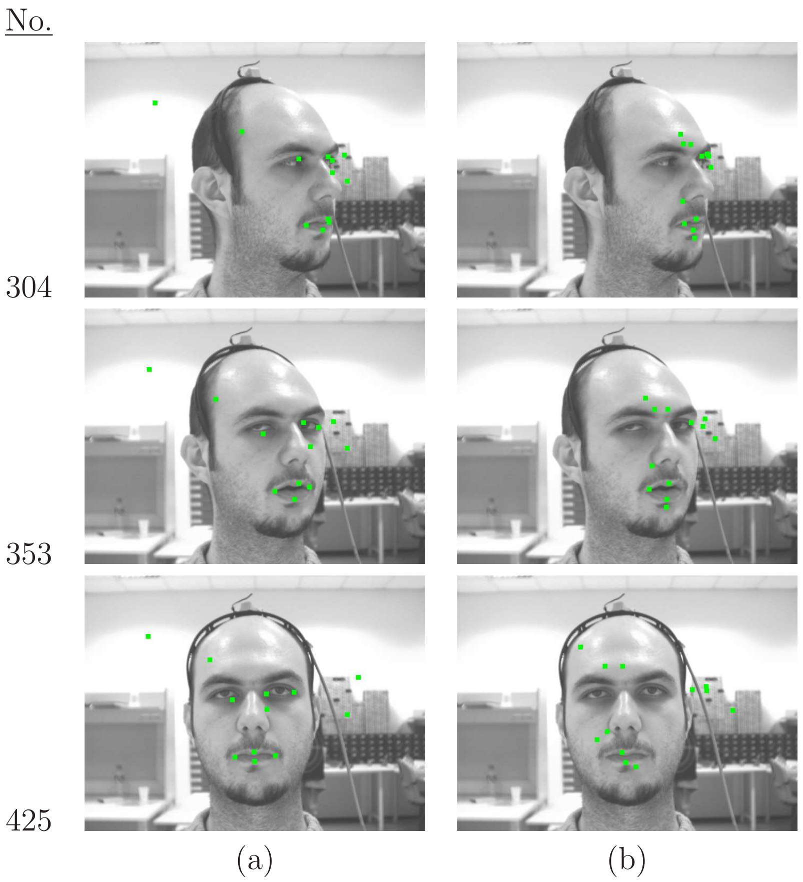


Figure 4.13: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes rotation around x-axis (θ_y) and mouth opening.

No.	$\triangle x$ (in pixels)				$\triangle x$ (in cm)			
	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$	$\triangle x_x$	$\triangle x_y$	$\triangle x_u$	$\triangle x_v$
1	49.00	4.00	0.00	0.00	3.80	0.4	0.00	0.00
2	49.26	2.73	0.00	0.00	3.80	0.4	0.00	0.00
53	47.85	2.29	0.00	0.00	3.80	0.4	0.00	0.00
128	34.18	1.62	0.00	0.00	3.80	0.4	0.00	0.00
251	47.07	3.01	0.00	0.00	3.80	0.4	0.00	0.00
289	54.07	4.10	0.00	0.00	3.80	0.4	0.00	0.00
413	43.59	0.86	0.00	0.00	3.80	0.4	0.00	0.00

Table 4.5: The updated difference vectors and the real spatial difference vectors between the left eye corner points for the sequence in Figure 4.14.

the results of the proposed method. One reason of this is again reliably tracked points (left and right eye inner corners). As explained in subsection 3.4.2 when the feature points that are accepted as reliably tracked points can not be tracked well, the proposed method can lose tracking. Because in this case the head pose information can not be obtained well using reliable points, there occur drifts in all other feature points. Another reason is the orthographic assumption of Affine invariance. Affine invariance provides accurate results up to a point of head

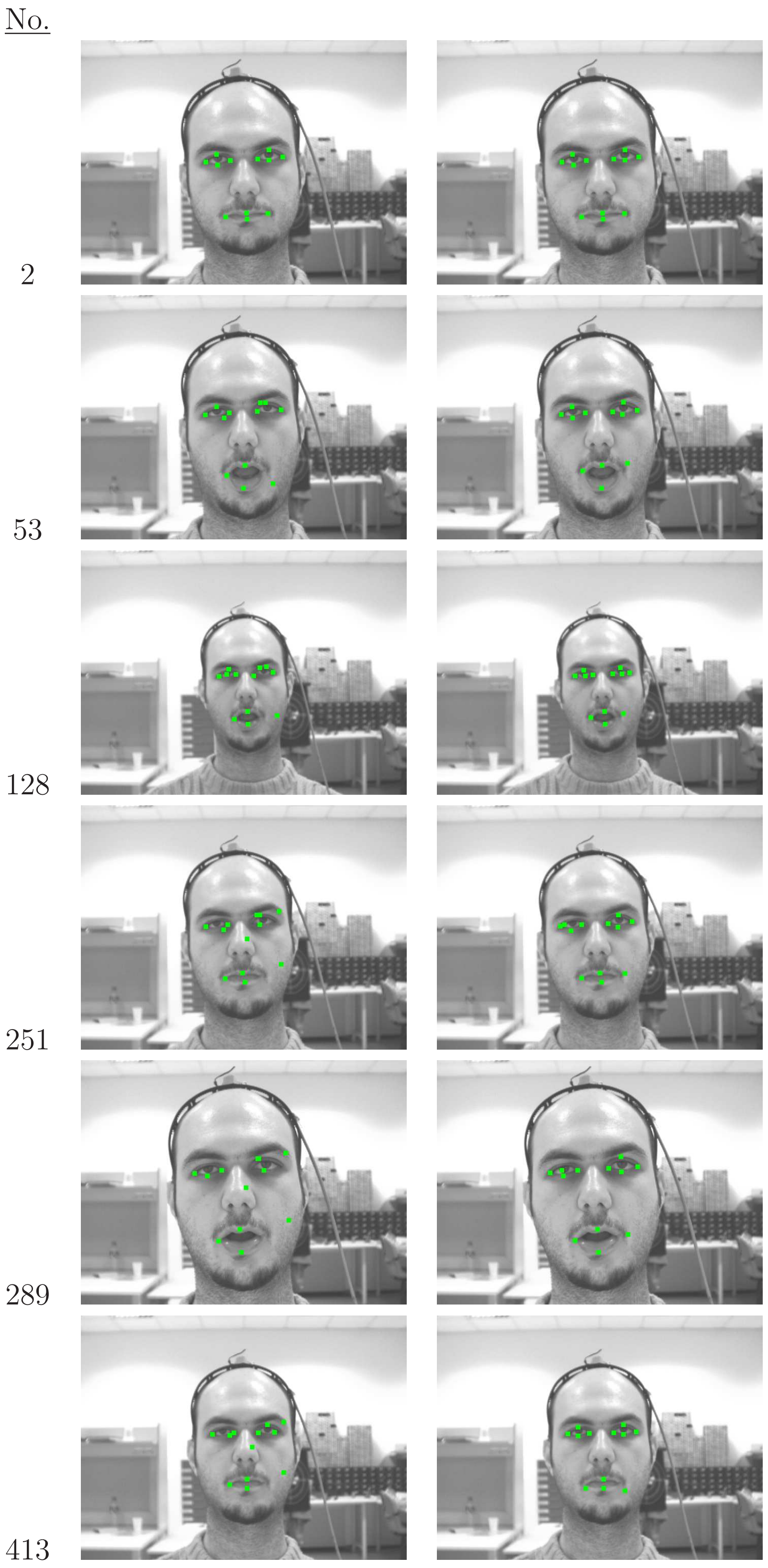


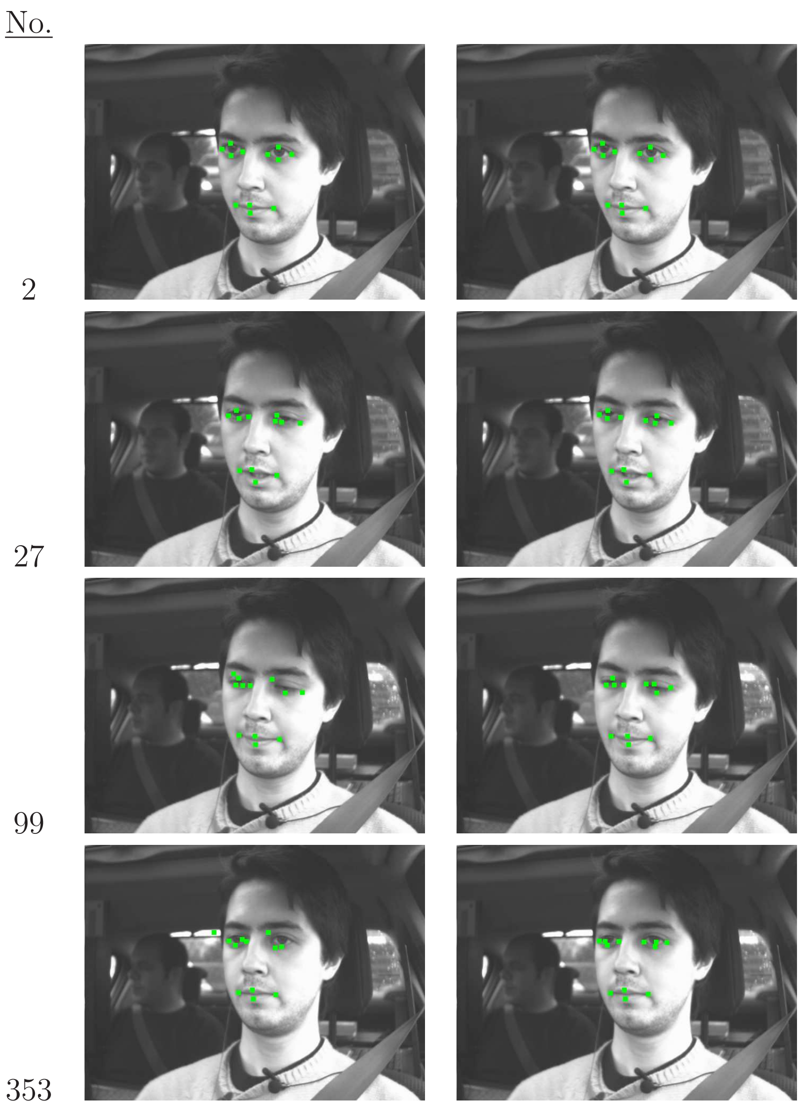
Figure 4.14: Tracking results of (a) the method in [23] (b) the proposed method for a sequence that includes translation on z-axis and mouth opening.

movements. When there is large movements of head the assumption fails. But considering the results of the method in [23], the proposed method appears to be a better tracker. If the head pose information can be obtained from an external source then these results will be much improved.

4.4 Vehicle Environment Results

In this section, in order to evaluate the performance of the proposed method in a real-world application, the framework is tested on videos recorded in vehicle environment [1]. The sequences are 640x480 resolution videos, compressed with DIVX codec, that include head movements and facial component movements. As in the experiments in subsections 4.3.4, 4.3.5 reliably tracked points are used to cope with head movements. The left and right inner eye corner feature points are selected as reliable points. In Figure 4.15-a,b the results for the method in [23] and for the proposed method are shown respectively. This test sequence is taken from the part when the driver is performing a "phone-banking" task. For this reason the sequence also includes mouth movements because of speaking.

It can be clearly seen that except for some drifts on the lower lid feature points the proposed method can successfully track while the method in which only temporal relations are used [23] fails to track and loses the positions. When there occurs occlusions because of the eye blinks the spatial connections in the proposed method are sufficient to continue tracking. But in the other method , because of this lack of information feature points go on with constant velocity (due to the nature of the linear state space model used). It should also be noticed that an external source that gives the head pose parameters is not used in these cases. Such information will improve the results. Considering the results from the vehicle environment, it is concluded that the proposed tracker has the potential be used in practical real-world problems.


No.
2
27
99
353

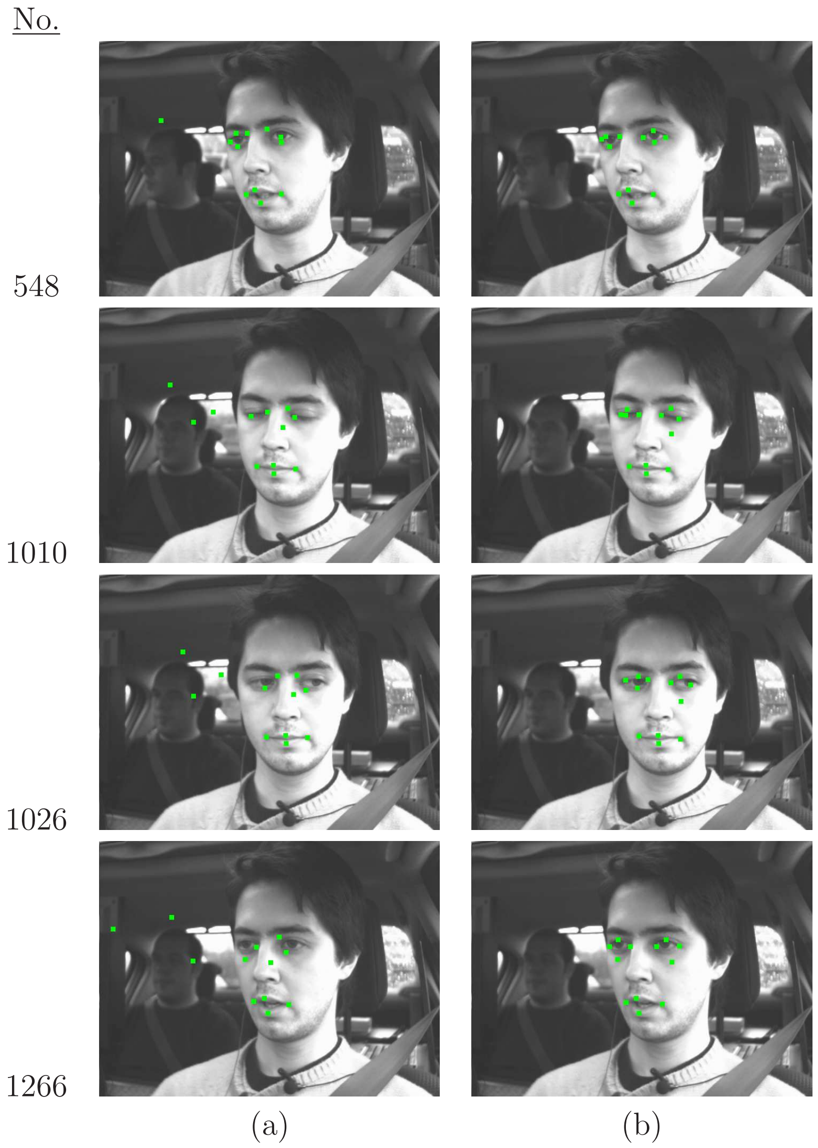


Figure 4.15: Tracking results of (a) the method in [23] (b) the proposed method for a sequence, recorded in vehicle environment [1], that includes facial gestures (speaking, mouth opening) and head movements.

CHAPTER 5

CONCLUSION

In this thesis a facial feature point tracker that can be used in applications such as human-computer interfaces, facial expression analysis systems, driver fatigue detection systems, etc. is proposed and the performance of the tracker is evaluated under various conditions.

5.1 Summary

The proposed tracker is based on a graphical model framework. The position of the facial features are tracked through video streams by incorporating statistical relations in time and the spatial relations between feature points. In most of the recent works, the spatial connections are not used, the tracking of feature points are done individually. This causes drifts in the case of arbitrary head movements. By using the spatial relations, the proposed method provides facial feature tracking in a holistic way and provides robust tracking in the head movement cases. Another advantage of the proposed method is the treatment of external occlusions. In the case of occlusion, the data in the occluded region become useless. To prevent drifts because of occlusions, a Gabor feature based occlusion detector is developed and used in the proposed method.

According to the given results it can be concluded that the proposed method provides a general framework for facial feature tracking. It is a robust tracker for facial expression sequences where there is uncertain and useless data because of external occlusion. There can occur drifts in the case of arbitrary head movements. Especially in the cases in which there is large movements and in the cases when the reliable points can not be tracked well, since the proper head pose information can not be obtained drifts occur in all points. But in a case in which the head pose information is obtained properly by an external source the proposed method can continue tracking robustly. Although there occur some drifts when the video resolution is too low, the proposed method can be thought of a robust tracker when the data resolution is low. In the conditions where the illumination level of the scene is very low the proposed method can suffer because of the misleading of observation stage. But when the illumination is very high the method can continue tracking reliably. The results in the vehicle environment show that the proposed method can be a practically usable tool for a analysis system such as driver behavior analysis, driver fatigue detection, etc.

5.2 Suggestions for Future Work

There can be extension of the proposed method in different ways. Since the algorithm works on Matlab platform, this prevents the algorithm to work in real-time rates. This can be a disadvantage for practical implementations. The algorithm can be modified to run on C++ platform and can be much faster.

One extension can be in the observation phase. When the eyes are closed or there is eye blinking, the edge structure in the eye region is totally different from the structure when the eyes are open. To handle these cases totally, two different edge structures can be represented by using two Gabor feature patches. There can be some modifications in the occlusion detection part. The threshold selection for the occlusion part is manual. This can be done automatically. Instead of thresholding according to the similarity outputs, similarity values can be used to change the uncertainty of the observation model. This can weight the information coming from observations.

As mentioned above, the proposed method provides a general framework and it is flexible. The framework can be manipulated in different ways to extend the system; for example by assigning different spatial relations, increasing the number of facial features, etc. The proposed framework can be also modified in a way to model the relations between facial features hierarchically. This can be done by connecting low-level feature points (e.g. right eye feature points, mouth feature points) together as one high-level point (e.g. center of right eye, center of mouth) and then connecting each high-level point together. This will provides a multi-resolution tracker. The representation of facial features can be in 3-D.

Another extension can be to learn these relations from training sets. In this framework the statistical relations between feature points are represented using Gaussian distributions. This is the simplest way to perform inference parametrically. The framework would allow more complex relationships that use non-parametric density models. Again a training phase can also be useful to choose the proper distributions to be assigned for relations between features.

REFERENCES

[1] H. Abut, H. Erdogan, A. Ercil, B. Curuklu, H.C. Koman, F. Tas, A.O. Argunsah, S. Cosar, B. Akan, H. Karabalkan, E. Cokelek, R. Ficici, V. Sezer, S. Danis, M. Karaca, M. Abbak. M.G. Uzunbas, K. Ertimen, C. Kalaycioglu, M. Imamoglu, C. Karabat, and M. Peyic. Data collection with 'uyanik': Too much pain; but gains are coming. In Proc. Biennial on DSP for In-Vehicle and Mobile Systems, Istanbul, Turkey, June 2007.

[2] J. Ahlberg. Using the active appearance algorithm for face and facial feature tracking. In IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, pages 68–72, 2001.

[3] Aleksandr Danilovitch Aleksandrov, Andrei Nikolaevich Kolmogorov, and Mikhail Alekseevich Lavrent'ev. Mathematics: Its Content, Methods and Meaning. M.I.T. Press, Cambridge, Mass, USA, 1963.

[4] P. Antoszczyszyn, J.M. Hannah, and P. Grant. Local motion tracking in semantic-based coding of videophone sequences. In Sixth International Conference on Image Processing and Its Applications, pages 46–50, 1997.

[5] Michael J. Black and Yaser Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In ICCV, pages 374–381, 1995.

[6] David S. Bolme. Elastic bunch graph matching. Master's thesis, Colorado State University, 2003.

[7] F. Bourel, C. Chibelushi, and A. Low. Robust facial feature tracking, 2000.

[8] R.G. Brown and P.Y.C. Hwang. Introduction to Random Signals and Applied Kalman Filtering. John Wiley & Sons, Inc., third edition, 1996.

[9] Su C and Huang L. Spatio-temporal graphical-model-based multiple facial feature tracking. EURASIP Journal on Applied Signal Processing, 13:2091–2100, 2005.

[10] Jingying Chen and B. Tiddeman. A robust facial feature tracking system. In IEEE Conference on Advanced Video and Signal Based Surveillance, pages 445–449, 2005.

[11] Jingying Chen and B. Tiddeman. Robust facial feature tracking under various illuminations. In IEEE International Conference on Image Processing, pages 2829 – 2832, 2006.

[12] T. Chen and R. Rao. Audio-visual integration in multimodal communications, 1998.

[13] Jeffrey Cohn, Adena Zlochower, Jenn-Jier James Lien, and Takeo Kanade. Feature-point tracking by optical flow discriminates subtle differences in facial expression. In Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition (FG '98), pages 396 – 401, April 1998.

[14] Antonio Colmenarez, Brendan Frey, and Thomas S. Huang. DETECTION AND TRACKING OF FACES AND FACIAL FEATURES. pages 657–661.

[15] F. Dornaika and J. Ahlberg. Efficient active appearance model for real-time head and facial feature tracking. In IEEE International Workshop on Analysis and Modeling of Faces and Gestures, pages 173 – 180, 2003.

[16] F. Dornaika and J. Ahlberg. Fast and reliable active appearance model search for 3-d face tracking. IEEE Transactions on Systems, Man and Cybernetics, Part B, 34(4):1838 – 1853, August 2004.

[17] F. Dornaika and F. Davoine. Online appearance-based face and facial feature tracking. In Proceedings of the 17th International Conference on Pattern Recognition, pages 814 – 817, 2004.

[18] Rogério Schmidt Feris and Roberto Marcondes Cesar Junior. Locating and tracking facial landmarks using gabor wavelet networks. In ICAPR '01: Proceedings of the Second International Conference on Advances in Pattern Recognition, pages 311–320, London, UK, 2001. Springer-Verlag.

[19] R.S. Feris and Jr. Cesar, R.M. Tracking facial features using gabor wavelet networks. In Proceedings XIII Brazilian Symposium on Computer Graphics and Image Processing, pages 22–27, 2000.

[20] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. pages 564–584, 1987.

[21] W. Gilks, S. Richardson, and D. Spiegelhalter. Markov Chain Monte Carlo in Practice. Chapman and Hall, 1996.

[22] T. Goto, S. Kshirsagar, and N. Magnenat-Thalmann. Automatic face cloning and animation using real-time facial feature tracking and speech acquisition. IEEE Signal Processing Magazine, 18:17–25, May 2001.

[23] Gu H and Ji Q. Information extraction from image sequences of real-world facial expressions. Mach. Vis. Appl., 16(2):105–115, 2005.

[24] C. Hu, R. Feris, and M. Turk. Real-time view-based face alignment using active wavelet networks. In IEEE International Workshop on Analysis and Modeling of Faces and Gestures, pages 215 – 221, 2003.

[25] L. Huang and C. Su. Bayesian network enhanced prediction for multiple facial feature tracking. 5(3):157–169, 2005.

[26] O. L. R. Jacobs. Introduction to Control Theory. Oxford University Press, 2nd edition, 1993.

[27] R. E. Kalman. A new approach to linear filtering and prediction problems. Transaction of the ASME-Journal of Basic Engineering, 82:35–45, March 1960.

[28] Jin-Woo Kima, Munjae Song, Ig-Jae Kim, Yong-Moo Kwon, Hyoung-Gon Kim, and Sang Chul Ahn. Automatic fdp/fap generation from an image sequence. In The 2000 IEEE International Symposium on Circuits and Systems, Proceedings. ISCAS Geneva., pages 40 – 43, 2000.

[29] Richard Lewis. Optimal Estimation with an Introduction to Stochastic Control Theory. John Wiley & Sons, Inc., 1986.

[30] P. S. Maybeck. Stochastic models, estimation and control. Volume I. 1979.

[31] Kevin P. Murphy. An introduction to graphical models. http://www.cs.ubc.ca/ murphyk/Bayes/bnintro.html, May 2001.

[32] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. pages 467–475.

[33] J. Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufman, San Mateo, 1988.

[34] Ji Q and Yang X. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. Real-Time Imaging, 8(5):357–377, 2002.

[35] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.

[36] H. Sako, M. Whitehouse, A.V.W. Smith, and A. Sutherland. Real-time facial-feature tracking based on matching techniques and its applications. pages B:320–324, 1994.

[37] A. Schubert. Detection and tracking of facial features in real time using a synergistic approach of spatio-temporal models and generalized hough-transform techniques. In FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000, page 116, Washington, DC, USA, 2000. IEEE Computer Society.

[38] C. Su, H. Zhou, and L. Huang. Multiple facial feature tracking using multi-cue based prediction model. pages 214–226, 2004.

[39] C. Su, Y. Zhuang, L. Huang, and F. Wu. A two-step approach to multiple facial feature tracking: Temporal particle filter and spatial belief propagation. pages 433–438, 2004.

[40] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. Nonparametric belief propagation. Technical Report Memo 20, MIT Artificial Intelligence Lab, MIT AI Lab, Cambridge, MA., 2002.

[41] Ying-Li Tian, Takeo Kanade, and Jeffrey Cohn. Dual-state parametric eye tracking. In Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), pages 110 – 115, March 2000.

[42] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[43] Yan Tong and Qiang Ji. Multiview facial feature tracking with a multi-modal probabilistic model. In ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, pages 307–310, Washington, DC, USA, 2006. IEEE Computer Society.

[44] Yan Tong, Yang Wang, Zhiwei Zhu, and Qiang Ji. Facial feature tracking using a multi-state hierarchical shape model under varying face pose and facial expression. In 18th International Conference on Pattern Recognition, pages 283 – 286, 2006.

[45] Emanuele Trucco and Alessandro Verri. Introductory Techniques for 3-D Computer Vision. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[46] Baran Çürüklü. Layout and function of the intracortical connections within the primary visual cortex. Licentiate thesis, December 2003.

[47] Jessica JunLin Wang and Sameer Singh. Video analysis of human dynamics - a survey. Real-Time Imaging, 9(5):321–346(26), October 2003.

[48] Liang Wang, Weiming Hu, and Tieniu Tan. Recent developments in human motion analysis. Pattern Recognition, 36(3):585–601(17), March 2003.

[49] Mei Wang, Yoshio IWAI, and Masahiko Yachida. Recognizing degree of continuous facial expression change. In ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 2, page 1188, Washington, DC, USA, 1998. IEEE Computer Society.

[50] Ru-Shang Wang and Yao Wang. Facial feature extraction and tracking in video sequences. In IEEE First Workshop on Multimedia Signal Processing, pages 233–238, 1997.

[51] Yair Weiss. Correctness of local probability propagation in graphical models with loops. Neural Comput., 12(1):1–41, 2000.

[52] Jan Wieghardt, Rolf P. Würtz, and Christoph von der Malsburg. Gabor-based feature point tracking with automatically learned constraints. In Proceedings ECCV 2002, Copenhagen, 2002.

[53] Kwok-Wai Wong, Kin-Man Lam, Wan-Chi Siu, and Kai-Ming Tse. Face segmentation and facial feature tracking for videophone applications. In Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, pages 518 – 521, 2001.

[54] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2d+3d active appearance models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 535 – 542, June 2004.

[55] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. ACM Comput. Surv., 38(4):13, 2006.

[56] X. Zhong, J. Xue, and N. Zheng. Graphical model based cue integration strategy for head tracking. page I:207, 2006.