PolyPublie
Polytechnique Montréal

Polytechnique Montréal
Université d'ingénierie

Titre: Title:	Estimation of the head's rigid motion in videoconference sequences
Auteur: Author:	Kegong Niu
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Niu, K. (2003). Estimation of the head's rigid motion in videoconference sequences [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7318/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7318/
Directeurs de recherche: Advisors:	Paul Cohen
Programme: Program:	Non spécifié

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

ESTIMATION OF THE HEAD'S RIGID MOTION IN VIDEOCONFERENCE SEQUENCES

KEGONG NIU
DÉPARTMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
Septembre 2003

© Kegong NIU, 2003.

National Library of Canada

Acquisitions and Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada

Acquisisitons et services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-90873-9
Our file Notre référence
ISBN: 0-612-90873-9

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ESTIMATION OF THE HEAD'S RIGID MOTION

IN VIDEOCONFERENCE SEQUENCES

présenté par: NIU Kegong

en vue de l'obtention du diplôme de: Maîstrise ès sciences appliquées

a été dûment accepté par le jury d'exmen constitué de:

Mme. CHÉRIET Farida , Ph.D., président

M. COHEN Paul, Ph.D., membre et directeur de recherche

M. LAGANIÈRE Robert, Ph.D., membre

To my parents

Acknowledgements

Thanks to Dr. Paul Cohen, my thesis advisor. He helped me tremendously in my thesis.

Thanks to Etienne Boutin for many beneficial conversions and good research ideas. His help in French translations is greatly appreciated.

Thanks to Nicolas Plouznikoff for his works in the survey of videoconference systems.

Résumé

Une nouvelle approche pour la compression d'une séquence d'images consiste à baser le codage sur un modèle a priori de la scène. Cette approche permet entre autre d'obtenir un très faible débit de transmission dans le cas de vidéoconférences « tête et épaules ». Ce type de codage implique l'extraction en temps réel des paramètres de contrôle du visage. Ces paramètres sont directement obtenus à partir d'une séquence vidéo sur le site de codage et sont transmis afin de reconstruire une réplique virtuelle de la scène sur le site de décodage. Ce travail décrit une implantation partielle d'un système à faible débit pour la vidéoconférence, incluant la détection de la tête, la localisation d'éléments faciaux, la sélection de caractéristiques et leur suivi d'une image à la suivante, l'initialisation du modèle de visage et l'estimation des mouvements rigides de la tête. L'analyse des expressions faciales n'est cependant pas traitée.

À partir de la première image de la séquence, la région faciale est segmentée de l'arrière plan en utilisant la couleur. Des méthodes d'égalisation et de lissage d'histogrammes sont utilisées afin de réduire les effets de la luminance sur l'intervalle de couleur faciale. La régularisation de la densité et quelques autres techniques de traitement d'images sont utilisées afin d'éliminer les zones parasites de faible taille. Une autre méthode basée sur un modèle elliptique du visage permet d'éliminer les zones parasites de grande taille. Les positions des éléments faciaux sont trouvées à l'aide des méthodes de sélection basées sur des connaissances a priori concernant la région faciale, telles que le fait que les éléments faciaux ont des luminances plus basses que le reste du visage, ou encore les proportions invariantes de ces éléments.

Afin d'augmenter la ressemblance entre le modèle virtuel et la scène originale, le modèle est d'abord adapté globalement. Par la suite, la texture du visage de la scène originale est appliquée sur le modèle.

À l'aide d'un suivi de points entre deux images successives, trois méthodes récursives d'estimation des paramètres du mouvement 3D en temps réel sont comparées : la Prédiction par Moindres Carrés (PLS), la méthode PLS-Kalman et la Structure à partir du Mouvement (SFM). Les méthodes PLS et PLS-Kalman utilisent l'information de profondeur provenant du modèle virtuel. La technique de prédiction et correction utilisée par PLS et PLS-Kalman fait l'estimation dans la première image de la séquence à l'intérieur d'une boucle fermée, ce qui permet d'éviter l'instabilité et la divergence dues à l'erreur cumulative rencontrée lorsque les paramètres sont simplement estimés d'une image à l'autre en boucle ouverte. La méthode SFM utilise le Filtre Kalman Étendu pour calculer le mouvement, la structure et la distance focale pour n'importe quelle séquence d'images, sans aucune information sur la profondeur. Après l'estimation de mouvement, les paramètres obtenus permettent de contrôler le modèle virtuel.

Les méthodes introduites pour la détection de têtes et la localisation des éléments faciaux ont été testées sur la base de données AR, où un succès de 94.98% a été obtenu. Les trois méthodes pour l'estimation des mouvements rigides permettent d'estimer le mouvement 3-D de la tête d'une façon précise. Comparativement aux deux autres méthodes, PLS-Kalman a obtenu de meilleurs résultats et est plus robuste au bruit introduit lors du suivi des points. Le système entier peut opérer à une vitesse de 4 ou 5 images par seconde lorsqu'une recherche entière est utilisée comme stratégie et ce sur un ordinateur PC P-2, 300 MHz.

Abstract

Model-based video coding is a newly emerging compression technique for image sequences. In the applications of model-based coding, the coding of head-and-shoulder video sequences at very low bit rates for videoconference has been a difficult problem for many years. It may involve the real-time extraction of facial control parameters from live video at the encoder site and the reconstruction of a dynamic facsimile of the subject's face at a remote decoder. In this work we propose a system for low-bandwidth videoconference and describe solutions for the modules of face detection, facial feature localization, feature selection for tracking, feature tracking, face model initialization and face rigid motion estimation. Facial expression analysis is not covered in this work but constitutes on additional component of a complete low-bit-rate videoconference system.

In the first frame of the video sequence, the face region is segmented from the background by using color. Histogram equalization and histogram fitting method are used to reduce the luminance effects on the face color ranges in the YCrCb color space. Density regularization and some other image analysis techniques are used to eliminate erroneous skin-color regions of small areas. An elliptic shape correction method can eliminate *skin-color regions* of large areas. The positions of facial features are found by using knowledge-based selection methods in the face region. A Luminance verification method, based upon the assumption that the facial features have lower luminance values

than that of other parts of the face, is used to enhance the facial feature candidate regions. The measurements used in the knowledge-based selection method are selected by using the knowledge of the detected face and ideal face proportions.

In order to update the generic face mode to a specific face model, the generic face model is globally adapted first. Then the detected face texture is mapped onto the face model. Here we propose a method that separates the face region into 18 small regions in order to map 3-D nodes of the face model onto 2-D image pixels of the face texture.

Good features for tracking are selected at the face region by using a feature selection criterion, which is optimal for tracking. Most of the time, the features for tracking are selected at the corners of eyes, eyebrows and mouth. Block matching and tracking and monitoring methods are used to track the selected features in the following frames.

To compute in real time the head 3D motion parameters, three recursive motion estimation methods, Predictive Least-square (PLS), PLS-Kalman and Structure from motion (SFM), using the tracked feature positions between two neighbor frames, have been analyzed. The PLS and PLS-Kalman methods use the depth information from the adapted face model. The prediction and correction technique used in PLS and PLS-Kalman locks the estimation to the first frame of the video sequence through a closed loop, and thus avoids instability and drift due to error accumulation that appears if frame to frame parameters are simply accumulated in an open loop. The SFM method uses an

the Extended Kalman filter to recover the motion, pointwize structure and focal length for arbitrarily image sequences without any depth information. After motion estimation, the motion of the face model can be controlled by the motion parameters.

The methods introduced in the face detection and facial feature localization components have been tested using the AR face database, with global success rate 96.21% in face detection and 94.98% in facial feature localization respectively. All three methods for the rigid motion estimation estimate efficiently the 3-D motion of the face. Compared with the other two methods, the PLS-Kalman method shows has the best results and is robust to tracking noise. The speed of the whole system is about 4-5 frames per second when using a full-search strategy on a P-2 300 MHz PC computer.

Condensé

Introduction

La représentation d'un modèle virtuel humain du type « tête et épaules » pour la vidéoconférence à faible débit est un problème difficile que l'on tente de résoudre depuis plusieurs années (Essa et Pentland, 1997; Zhang et Cohen, 2000). Ce problème requiert l'extraction en temps réel – à partir d'une séquence vidéo – de paramètres caractérisant les mouvements et les expressions du visage afin de les reproduire dans un modèle dynamique. Dans ce travail, nous proposons une nouvelle méthode, utilisable en vidéoconférence à faible débit, et décrivons des solutions pour les modules de détection du visage, la localisation des éléments faciaux, la localisation d'éléments pour le suivi d'éléments, l'initialisation du modèle du visage et l'estimation du mouvement rigide de la tête. Le système utilise des séquences d'images monoculaires en supposant que la tête et les épaules sont visibles sur les images. L'analyse des expressions faciales n'est pas traitée dans ce travail.

Détection du visage

L'information couleur a été introduite il y a quelques années pour résoudre le problème de détection du visage et son introduction semble prometteuse. Chai et Ngan (1999) suggèrent d'utiliser le système de couleurs YCCb pour extraire le visage à l'aide de la segmentation des couleurs. Dans ce système, la valeur Y représente la luminance tandis que la chrominance est représentée à l'aide de Cb et Cr. L'auteur a d'ailleurs remarqué que la région de couleur de la peau pouvait être identifiée par la présence d'un certain intervalle de chrominance (déterminée par la distribution de Cr et Cb dans l'espace de couleurs YCrCb). Les intervalles de chrominance optimaux ont été trouvés: le Cr entre 133 et 173 et le Cb entre 77 et 127 correspondent ainsi à la couleur de la peau du visage.

D'après nos expérimentations, l'intervalle de chrominance est lié à la valeur de la luminance Y. Afin d'améliorer la méthode introduite par Chai et Ngan (1999) qui est sensible à la variation de la luminance, nous utilisons des intervalles de couleur de la peau obtenus sur plusieurs visages d'individus sélectionnés aléatoirement. Les visages sont ainsi segmentés puis une méthode de correction de la couleur est utilisée. L'idée de la méthode de correction de la couleur consiste à corriger les histogrammes de couleur des images sélectionnées. Ainsi, l'analyse des histogrammes pour chaque type d'image de visage peut être évitée avant la segmentation des visages.

Après la segmentation des couleurs, quelques techniques de traitement d'images peuvent être utilisées pour éliminer les petites régions de bruit dans l'arrière-plan. Cependant, si l'arrière-plan de ces images contient de larges régions ayant la couleur de la peau, il devient impossible d'éliminer ce bruit par un simple filtre. Dans notre système, une correction utilisant un modèle de géométrie elliptique peut éliminer ces mauvaises régions. Le modèle elliptique de correction consiste à approximer la forme ovale du visage par une ellipse. Ainsi la détection du visage se réduit directement à la détection d'un objet elliptique.

La base de données de visage AR (A.M. Martinez et R. Benavente, « The AR face database », CVC Tech. Report #24, 1998) est utilisée pour tester le nouvel algorithme de détection du visage. Cette base de données contient 133 images du visage. En utilisant l'algorithme proposé, les régions du visage sont segmentées avec succès dans 127 images (96%). Les régions du visage ne sont pas segmentées complètement sur 6 images à cause de la présence de moustaches ou de barbes dans le visage ou de cheveux de couleur peau autour du visage.

Localiser les éléments du visage

Pour adapter le modèle virtuel du visage d'après les éléments de visage recueillis sur la scène, la détermination de la position de ces éléments est nécessaire. En utilisant les informations obtenues sur la détection du visage, certains éléments tels que les yeux et la bouche peuvent être localisés.

Dans notre algorithme, la localisation de ces éléments est basée sur l'hypothèse suivante : on suppose que certaines régions sont d'une couleur différente ou plus foncée comparativement à d'autres régions du visage. Toutes les régions candidates sont ainsi obtenues à l'aide d'une méthode de vérification de la luminance. Afin de choisir les régions candidates qui correspondent vraiment à des éléments d'intérêt du visage, la méthode de sélection basée sur des connaissances est utilisée (Reinders et Al, 1995). Selon cette méthode, toutes les mesures aléatoires des régions candidates pour les éléments d'intérêts sont modélisées selon une distribution normale. La moyenne et l'écart-type de la loi normale sont déterminés par les proportions idéales du visage. Les mesures que nous utilisons dans la sélection sont utilisées afin de trouver les régions les plus semblables qui correspondent à des éléments d'intérêt. Ces mesures sont simples et faciles à calculer. Plusieurs mesures utilisées dans la sélection peuvent réduire les erreurs de localisation. Ensuite, les percentiles du «z-score » sont utilisés pour représenter les valeurs des mesures particulières. Toutes les sélections des éléments du visage sont basées sur ces rangs percentiles.

Pour obtenir un haut taux de succès pour la sélection de la bouche, nous avons développé une méthode de localisation utilisant les coins de la bouche. Il est évident que les coins de la bouche ont plus de textures que dans les autres régions du visage. Il y a de fortes probabilités de trouver des éléments de hautes textures vis-à-vis les coins de la bouche en utilisant une méthode de sélection appropriée.

Tous les coins candidats de la bouche sont obtenus en utilisant un critère de sélection adéquat d'éléments qui sera introduit plus loin. Les vrais coins de la bouche sont choisis à partir de la connaissance des coins candidats.

Nous utilisons la base de données AR pour tester cet algorithme de localisation des éléments du visage. Un nombre de 125 images (95%) de visage ont été détectées avec succès tandis que 9 images ont été mal détectées.

Initialisation du modèle du visage

Le modèle virtuel du visage que nous utilisons a été développé par Keith Waters au laboratoire de recherche Compaq Cambridge. L'initialisation d'un modèle à partir d'un modèle de visage existant utilise des techniques comme la superposition, l'adaptation ou la transformation (Zhang et Cohen, 2000). Après avoir trouvé un ensemble d'éléments clés à l'aide de l'extraction de certains éléments, un modèle virtuel peut être déformé de façon à minimiser les différences globales entre les éléments localisés sur l'usager et ceux correspondant sur le modèle tridimensionnel.

Le problème d'adaptation est décomposé en une adaptation globale et une adaptation locale. L'adaptation globale consiste à étirer, positionner et orienter le modèle. L'adaptation locale consiste à raffiner les imprécisions entre le modèle et les contours du visage pour adapter les expressions. Beaucoup de travaux sont encore nécessaires afin d'effectuer ces adaptations locales sur le modèle. Notre projet est surtout concentré sur le problème de l'adaptation globale.

Le but de l'adaptation globale est d'adapter le modèle virtuel à un modèle spécial afin que celui-ci soit dans les mêmes proportions (largeur, hauteur), positions et que la position des éléments localisés puisse être utilisée pour adapter le modèle.

Puisqu'il est très difficile d'adapter un modèle avec beaucoup de précision en utilisant des polygones et de synthétiser tous les éléments tels que les rides, les taches, etc., une technique basée sur l'adaptation de textures est introduite. Cette technique consiste à calculer les coordonnées où les textures seront extraites du visage de l'usager. Ceci est utilisé pour déterminer des points de contrôle entre l'usager et le modèle virtuel, et permet ensuite d'appliquer la texture du visage. Tous les polygones 3-D générés pourront être associés à des fragments 2-D de la texture du visage de l'usager grâce à ces points de contrôle.

Il est généralement difficile de définir objectivement les mesures appropriées pour maximiser la performance et la qualité de l'adaptation. Le modèle du visage initialisé selon la méthode proposée permet cependant d'obtenir de bons résultats visuels.

La sélection d'éléments et le suivi

Les algorithmes existants pour l'estimation des mouvements rigides incluent l'estimation par le suivi d'éléments d'intérêts, l'analyse par synthèse et les techniques basées sur la vue. Avec l'objectif temps réel de notre projet, une estimation à partir du suivi d'éléments semble être le meilleur choix. Dans notre projet, nous n'utilisons pas les éléments préalablement localisés car ces derniers ne sont pas nécessairement bons pour le suivi. Nous utilisons plutôt une sélection d'éléments proposée par Shi et Tomasi (1994) pour choisir de bons éléments pour le suivi. Ce critère est d'ailleurs utilisé lors de la détection des coins de la bouche.

Pour choisir une méthode appropriée pour le suivi d'éléments d'intérêt, les résultats expérimentaux d'une méthode traditionnelle de correspondance par régions sont comparés avec le suivi et la méthode de surveillance proposée par Shi et Tomasi (1994). Bien que le suivi et la méthode de surveillance permettent de juger de la qualité du suivi en comparant les éléments sur la première image et celle courante, cela nécessite beaucoup de calculs. D'ailleurs, il est assez difficile de définir correctement un seuil pour cette surveillance. Pour ces raisons, nous avons choisi la méthode du suivi par correspondance de régions.

Estimation des mouvements rigides de la tête

Dans un système de vidéoconférence basé sur un modèle, l'estimation du mouvement rigide de la tête est essentiel afin d'adapter le modèle du visage à la même position que le visage réel dans la séquence. Pour calculer en temps réel les paramètres du mouvement 3-D, trois méthodes récursives d'estimation utilisant le suivi d'éléments d'intérêt entre deux images consécutives ont été analysées. Ces méthodes sont : la prédiction par moindres carrés (PLS), PLS-Kalman et la structure d'après le mouvement (SFM).

Smolic et al. (1999) propose deux méthodes récursives pour estimer, en temps réel, les paramètres du mouvement à long terme en utilisant l'information de la profondeur. Un vecteur plus simple avec seulement cinq paramètres du mouvement est utilisé dans le EKF et dans l'algorithme de prédiction par moindres carrés. Ce vecteur contient la position dans l'image de tous les éléments du suivi dans une nouvelle image. Les deux algorithmes utilisant la prédiction par moindres carrés et le EKF sont appelés PLS et PLS-Kalman respectivement. La technique de prédiction et correction utilisée dans le PLS et le PLS-Kalman retient l'estimation de la première image de la séquence vidéo et celle-ci est utilisée à l'intérieur d'une boucle fermée. Ainsi, l'instabilité et les erreurs cumulatives sont évitées. Ce qui n'est pas le cas lorsque les paramètres sont simplement estimés d'une image à l'autre dans une boucle ouverte.

Le EKF est utilisé pour fusionner les informations recueillies sur les mesures et estimer la structure 3-D, la position de la tête et la distance focale de la caméra (Azarbayejani et Pentland, 1997; Jebara et Pentland, 1997). Un vecteur d'état avec les paramètres du mouvement, la distance focale et l'information de la profondeur pour tous les éléments du suivi est utilisé dans une implémentation standard du EKF. Ce qui est semblable aux algorithmes PLS et PLS-EKF. Cette méthodologie est appelée la structure d'après le mouvement (SFM).

Les trois algorithmes précédents sont testés sur des séquences d'images de visages synthétiques et réels. L'algorithme PLS-Kalman démontre les meilleurs résultats. Les trajectoires des paramètres obtenus par l'algorithme PLS-Kalman sont beaucoup plus lisses et précises que celles obtenues par les deux autres algorithmes. De plus, il y a une meilleure robustesse au bruit. L'algorithme PLS-Kalman a donc été retenu pour estimer les mouvements dans notre système de vidéoconférence.

Conclusion

Dans ce mémoire, nous proposons une méthode d'estimation du mouvement un système de vidéoconférence en temps réel. Rigide dans une séquence d'images « tête et épaules ». La méthode inclut une détection robuste du visage, la localisation des éléments du visage, la sélection et le suivi des élément d'intérêts, l'initialisation du modèle et l'estimation du mouvement. Trois méthodes courantes d'estimation du mouvement rigide, PLS, PLS-Kalman et SFM sont comparées lors des expérimentations. La vitesse du système entier est aux alentours de 5 images par seconde sur un P-2 PC. La majorité du temps de calcul est utilisée pour le suivi des éléments, l'estimation du mouvement et l'adaptation du modèle. Le temps réel pourrait être atteint assez facilement sur un ordinateur de nouvelle génération plus puissant.

Les principales contributions de ce mémoire sont les suivantes :

- La proposition de la méthode de l'histogramme d'égalisation et de l'histogramme de raccord pour réduire les effets de la luminance sur les images dans l'espace de couleur YCrCb.
- La proposition d'utiliser la méthode de correction par le modèle elliptique en vue d'éliminer les grandes régions de couleur peau sur la carte de densité.
- La proposition de la méthode de vérification de la luminance pour améliorer les régions candidates pour les éléments du visage.
- La proposition d'une méthode robuste de mesure basée sur des connaissances pour la sélection et la localisation des éléments du visage.
- La proposition d'une méthode d'approximation utilisant une distribution normale pour choisir les mesures, ce qui permet d'estimer les probabilités des régions candidates pour être des éléments d'intérêt du visage.
- La proposition d'une méthode d'enregistrement pour créer la correspondance entre la texture 2-D du visage et les polygones 3-D du modèle virtuel.
- La proposition d'une méthode d'intégration du système en temps réel.

Dans le système, plusieurs aspects doivent encore être améliorés. Premièrement, la présente méthodologie n'inclut pas l'estimation des mouvements non rigides, i.e. l'extraction des mouvements faciaux pour les reproduire sur le modèle virtuel. Deuxièmement, des détections plus stables et plus robustes du visage peuvent être obtenues si nous combinons les informations des arêtes et des contours provenant des images traitées. Troisièmement, seulement une adaptation globale est effectuée sur le modèle virtuel à l'étape d'initialisation. Les adaptations locales sur le modèle et l'utilisation de l'information de la profondeur pourraient améliorer la ressemblance entre l'usager et le modèle virtuel. Finalement, un algorithme robuste pour le suivi a besoin d'être développé pour éviter les erreurs dues à la rotation ainsi que les erreurs cumulatives.

Table of contents

List of Tables

List of figures

List of symbols and abbreviations

2-D: two dimensions.

3-D: three dimensions.

EKF: Extended Kalman Filter.

MAD: minimum mean absolute difference.

MPC: maximum matching pixel count.

MSE: minimum mean square error.

PLS: Predictive Least Square algorithm.

SFM: Structure from Motion algorithm.

CHAPTER I

INTRODUCTION

Model-based video coding is a newly emerging image sequence compression technique. In contrast to conventional image compression schemes that exploit pixel to pixel correlation, model-based coding takes a more global view of objects and their specific 3-D representations, and relies on a priori knowledge about the scene. The basic assumption in this coding technique is that the expected scenes are known and they are constrained to a few world objects, like, for example, a speaker's head in a videoconference context (Zhang and Cohen, 2000; Lee and Magnenat-Thalmann, 1999; Feng et al, 2000; Moghaddam and Pentland, 1997).

Model-based coding of head-and-shoulder video sequences at very low bit rates for videoconference has been a difficult problem for many years (Essa and Pentland, 1997; Zhang and Cohen, 2000). It may involve the real-time extraction of facial control parameters from live video at the encoder site and the reconstruction of a dynamic facsimile of the subject' face at a remote decoder. It then adapts the generic face model to the actual facial expressions in the image and then tracks their evolution and changes throughout the sequence. In this way, the motion of a face in an image sequence can be compactly represented by motion parameters, which are then used to control the face model at the decoder.

The block diagram of such a model-based coding scheme is illustrated in Figure 1.1. As shown in Figure 1.1, a generic face model is present at both the encoder and decoder sides. This generic face model is updated during the initial frame of a sequence in order to adapt it to the actual features of the face in the scene. The adaptation is based upon the information extracted from the first image by using image analysis techniques. The model, therefore, becomes a specific model when it is adapted to the face in the image.

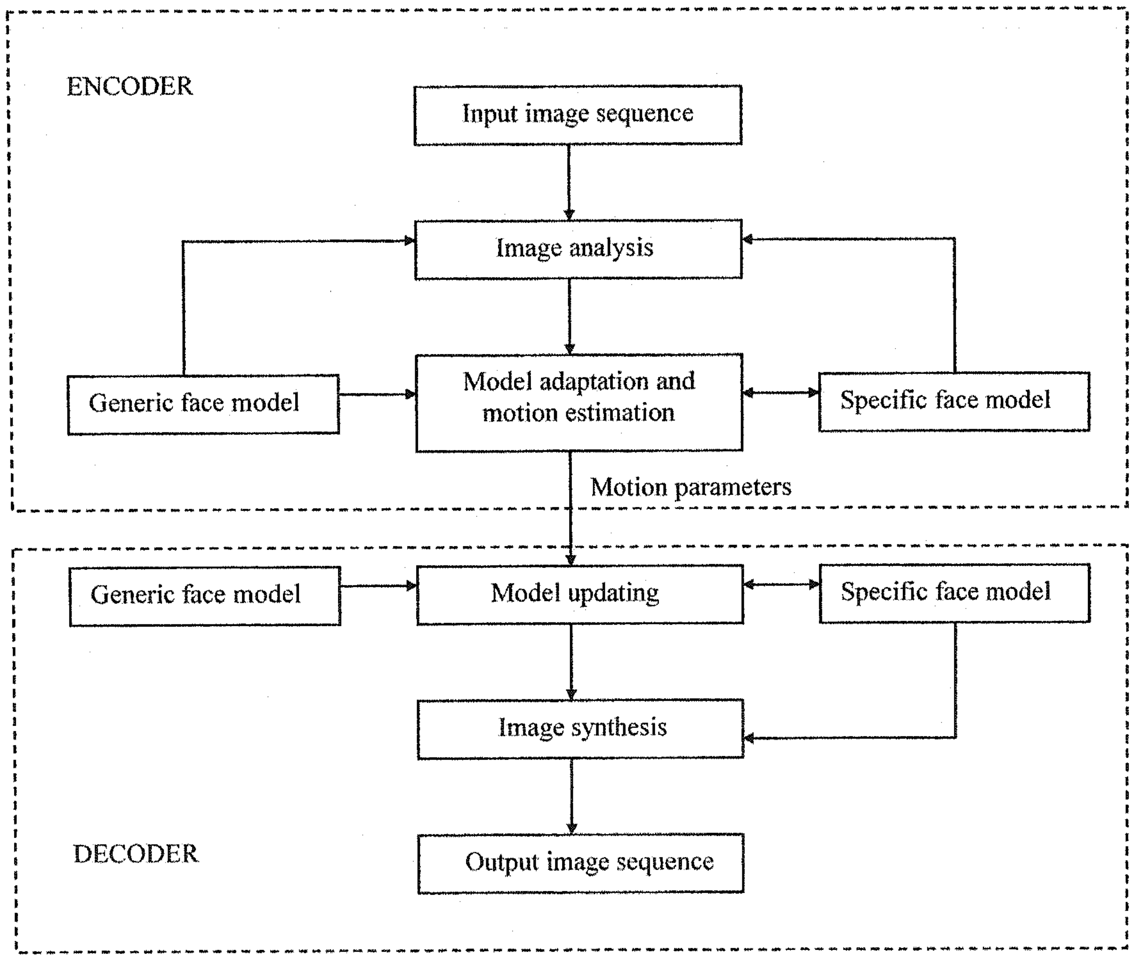


Figure 1.1 Model-based videoconference system

As the videoconference conversion proceeds, the face is tracked in the images, so that both the global motion parameters of the face as well as the local deformations corresponding to facial expressions are mimicked. The estimated motion and expression parameters are transmitted to the decoder side. At the decoder side the model is continuously adapted and animated with the incoming updates. In addition to the motion and expression data, additional data for texture and illumination changes may also be needed. The rendition of the specific face model after the model updates is implemented in the image synthesis.

1.1 The current methods of model-based coding for videoconference

Most of the work reported in the literature addresses only specific parts of the problem: face detection is now well understood and can be done in real-time (Frigola et al, 1999; Crowley and Schwerdt, 1999; Rowley et al, 1998; Satoh et al, 1999). Rigid-motion estimation and face tracking have already been studied by some authors and a few papers are available on facial action parameters (non-rigid motion estimation) (Malciu and Preteux, 2000; Essa and Pentland, 1997). As we mentioned each of those systems was developed independently and a full integrated working system has, to our knowledge, yet to be developed. This might prove difficult due to the tight coupling and interaction needed between the different parts of a real-time system.

Here we will introduce the current methods of coding for videoconference.

1.1.1 Face Detection

There are two basic classes of face detection methods, *attribute-based* (or *feature-base*d) and *template-based* methods.

The attribute-based methods take the image and search for known attributes like particular skin-color pixels, the elliptic shape of the head, or very specific facial features. This kind of methods is usually fast. As we mention later, facial feature extraction can also be treated as a particular case of feature-based face detection methods.

The template-based methods work by example. We give an example or a set of examples and try to find objects in the image similar to the example(s). Usually, template-based methods are robust but they also demand more processing power than attribute based methods. The problem is that if there are geometrical distortions, like varying pose or facial expressions, we will need to do some geometrical normalization.

We will now give you an overview of the most widely used methods for both of these two classes.

1.1.1.1 Feature-based methods

The available feature-based methods are presented in the diagram below:

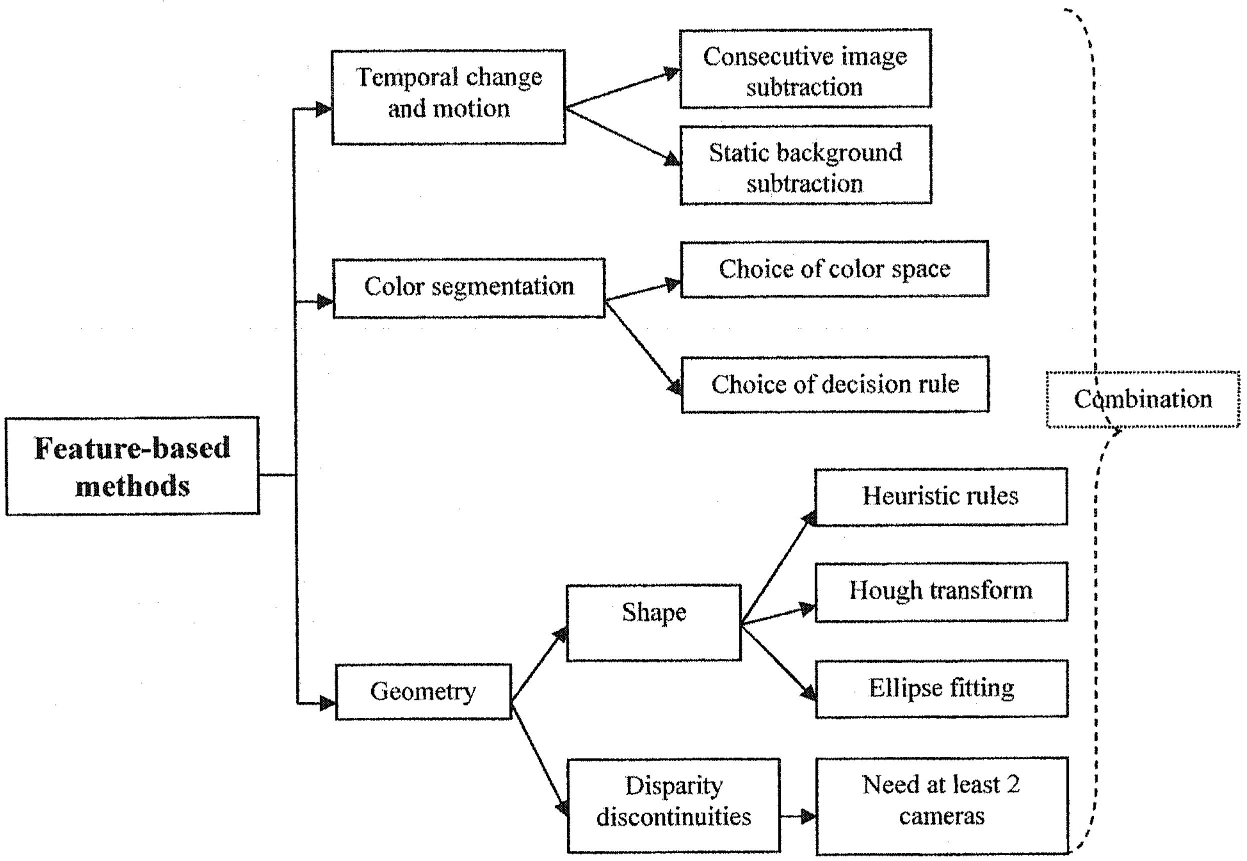


Figure 1.2 Methods of feature-based face detection

Temporal change and motion

The easiest way is to find faces in images with a controlled background: a plain background or a known static background. You'll only need the color of the plain background or an image of the background without the user. This case is trivial because a simple subtraction between the current image and the known one will remove the background and find the face boundaries. However, there are rather unrealistic assumptions involved, as you can see.

User's motion is another hint we can use to detect faces because a face is almost always moving in reality. To find the moving area we can simply subtract two consecutive images. The problems with this method are that there can be other moving objects in the background. What's more, subtracting two consecutive images produces large lateral band in the case of quick movements. The solution for the latter problem is to use differences of consecutive gradient images, then segments using the gradient orientation variation or the module, as proposed by Frigola *et al* in 1999.

Motion is a very powerful and inexpensive cue to use. The only drawback is that there can be a still person in the scene but this is very unlikely considering that a person blinks from time to time (Crowley and Schwerdt, 1999). To improve detection results, motion is usually combined with other methods, as we will discuss later.

Color

We will have access to color images so we might as well use color information, the typical skin color, to find the biggest skin-color region or to find face segments. Using color is very popular judging by the number of papers published using this method

(Rowley et al, 1998; Satoh et al, 1999; Wu et al, 1999; Chai and Ngan, 1999; Schwerdt and Crowley, 2000). A lot of interesting work has been done in this field, especially surveys addressing the choice of the color space (Terrillon et al, 2000) or the choice of the decision rules (Crowley and Schwerdt, 1999; Zarit et al, 1999).

Studies have shown that the distribution of skin colors across a given color space, especially the HSV and YCrCb color spaces, is very narrow and could be modeled by a Gaussian distribution. Researchers have also found that in those spaces the argument representing color intensity could be ignored because the possible skin color of a human being (dark skin, white skin...) is mainly characterized by this argument. The problem is to have a distribution model that is representative enough and therefore we must choose the "training" set very carefully.

Many decision rules have been employed to segment skin colored regions. Some are very fast like the ones using look-up tables and linear discriminants or thresholds. The main problem of these methods is that they cannot give any confidence of being skin for each pixel, a pixel is skin or isn't skin. This is why probabilistic decision rules like Bayes decision rule have been introduced. They can be more robust and compute the probability of a particular pixel being skin.

The computational cost of this method is very low and color cues are invariant to rotation and robust to (partial) occlusion. Even the problem of facial hair can be treated as an occlusion or can be handled by combining difference sources of information. The disadvantages is that it is not guaranteed to work with all kind of skin colors, and is not very robust under varying lighting conditions even if a linear function is used to correct for those lighting conditions.

Geometry

The idea is to use a priori knowledge of the geometry of a general human face to look for specific shape (Brunelli and Poggio, 1993). For example, the ellipse fitting method approximates the human head by an ellipse and tries to find the best ellipse match using the edges in the image but is computationally intensive. Methods robust to noise and partial occlusion, like the Hough transform (Schubert, 2000), are available to detect specific shapes but they require a lot a processing power. Other techniques include validating the shape of the regions detected by other methods: we can check the aspect ratio of the bounding box of the detected region for example.

Disparity discontinuities can also be used to detect a face. This method can segment a scene and create a layered representation of it (Liu et al, 2000). It is merely a technique allowing to group pixels together. The main drawback is that we need to match feature points from images coming from two cameras in order to know the disparity. This is computationally intensive and our setup will most likely not allow it.

Putting it all together

The idea is to use a mixture of all the methods above because combining several good approaches (using multiple sources of information) is usually more robust and normally yields even better results. The price to pay is a slightly higher computational cost but, since nearly all feature-based methods are fast, a small increase is not a problem. Good results have been obtained by combining color and motion information because color analysis can complement the restrictions of motion analysis.

A lot of ways have been proposed to perform this combination. Most of them are based upon heuristic rules like taking the overlap of the regions detected by different methods. Others have a more solid theoretical basis and use a statistical framework: the covariance of the measurements is used to take the best possible decision (Kim and Kim, 2000; Graf et al, 2000).

1.1.1.2 Templates-based methods

The available template-based methods are presented in the diagram below:

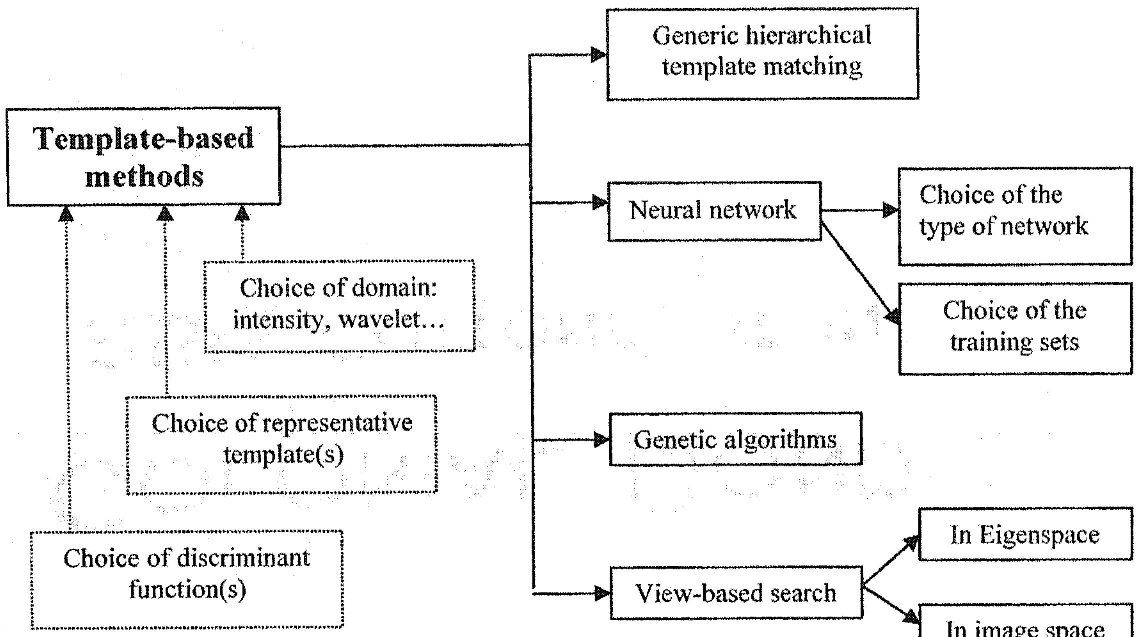

Template-based methods
Choice of domain: intensity, wavelet…
Choice of representative template(s)
Choice of discriminant function(s)
Generic hierarchical template matching
Neural network
Choice of the type of network
Choice of the training sets
Genetic algorithms
View-based search
In Eigenspace
In image space

Figure 1.3 Methods of template-based face detection

Generic template matching

Standard template matching in the image domain is the most popular method used to detect faces or features but it requires a lot of processing power. What's more, a representative template is very difficult to choose because the face appearance changes drastically across all the possible 3-D poses. The choice of a discriminant function has also to be addressed: some authors propose to use a linear discriminant to perform a first search and a quadratic one after that to refine the search (Weber and Hernàndez, 1999).

Neural networks and genetic algorithms

The most reliable methods for face detection, so far, have been connectionist methods (based upon neural networks). Neural networks can facilitate the search of faces in unconstrained scenes by implementing internally a template or a set of templates describing the training set. The choice of a representative template is then addressed automatically during the learning phase. The problem is that they are usually only used to detect frontal or nearly frontal faces (Rowley et al, 1998; Sung and Poggio, 1998) and these systems are very computationally demanding, and are therefore not very interesting in applications where real-time or near real-time performance is required.

Genetic algorithms on the other hand are well suited to perform searching and optimization but are also computationally demanding (Lin and Wu, 1999) and do not achieve real-time performance.

View-based search

To describe the varying appearances of a face across different 3-D poses we need multiple templates. This is where view-based search can be useful. These techniques can be compared to multiple observers watching a scene and we're looking for the one best describing this scene: we're looking for the template that explains in best what is in the image. What is done usually is a decomposition of face images into a weighted sum of basis images (eigenfaces) using the K-L transform: any face can be approximated by a weighted sum of eigenfaces. View based-search in eigenspace shows very promising results, in fact the results are superior to the standard template matching technique (Moghaddam and Pentland, 1997) and also confirm that a view based eigenspace is superior to a single parametric eigenspace.

An other very interesting advantage of this technique is that it can recover the 3-D pose of the object at the same time: once you know which template can best describe your scene you have the 3-D pose information associated with this template readily available. The problem is that it is not very precise and the recovered 3-D pose will only be a rough approximation: ideally we would need an infinite number of template to quantize the 3-D pose space but we can only afford to have a small number of them. The other main drawback is that it does not run in real-time: it is very computationally intensive and the more you quantize your 3-D-pose space the more intense it gets.

1.1.2 Facial Feature Localization

1.1.2.1 Facial features

A facial feature is an attribute in some way describing the face. In this context we will mostly present two kinds of facial features: specific points and contours. As we will mention latter, the location of specific facial features will most likely be used a lot in the model-based coding approach. This is why we dedicated a specific section to it.

A facial feature point is a specific point in the face, such as the corners of the mouth. The position and motion of the feature points can well describe the shape, global motion and expression of the face. For example, the motion of the corner points of the mouth describe facial expressions, coordinates of feature points on the forehead can describe the head shape and movement of inner eye corners can describe global motion. On the other hand, a contour feature is the borderline between facial areas, for example, the borderline between lips and skin, the edges of the eyelids and the border of the eyebrows. Contour features can often be interpolated from a set of feature points. Given an image and the location of the face, facial feature extraction is consequently the process of extracting coordinates of feature points or parameters describing contours. From the extracted facial features, face model parameters can be calculated, for transmission over a channel.

1.1.2.2 Facial feature localization as a specific case of face detection

As we mentioned earlier, feature detection and extraction can be considered a specific case of face detection and thus similar techniques can be used. The following diagram presents the available methods (which will be discussed) or refers to the face detection diagrams if the methods are too similar:

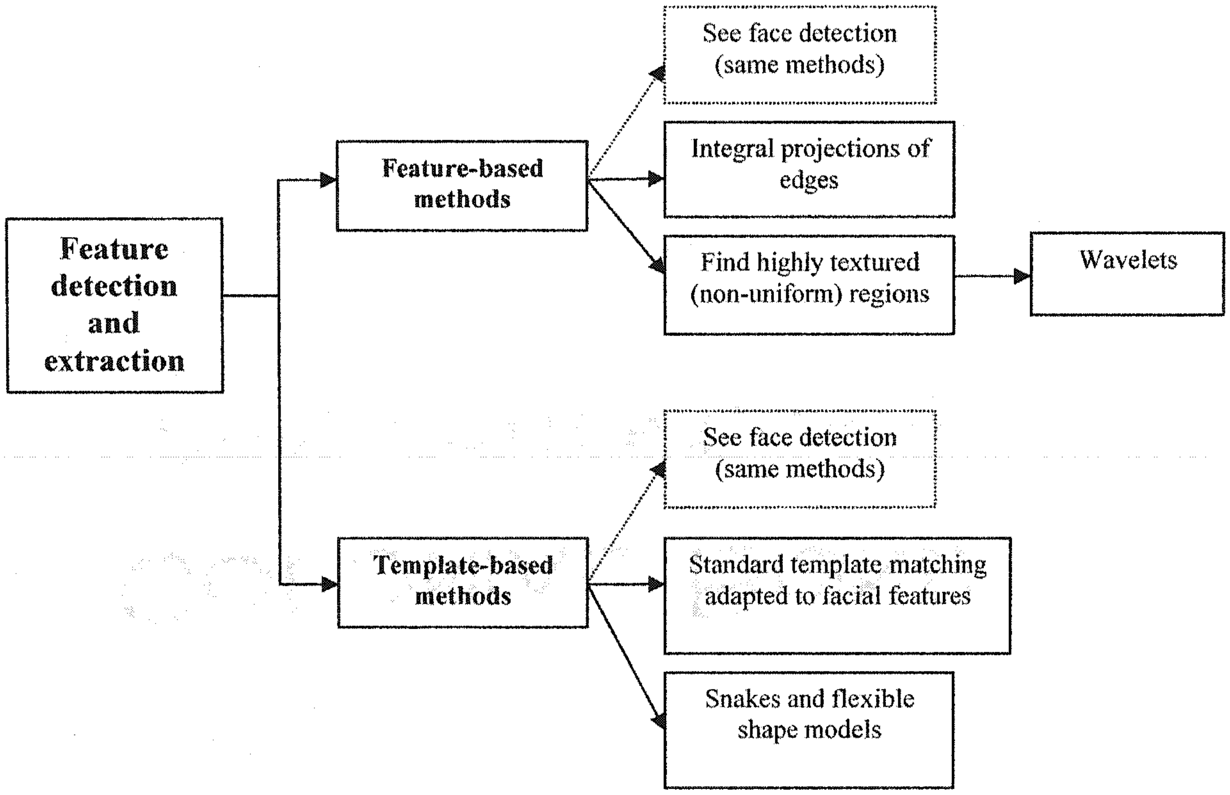

Feature detection and extraction
Feature-based methods
See face detection (same methods)
Integral projections of edges
Find highly textured (non-uniform) regions
Wavelets
Template-based methods
See face detection (same methods)
Standard template matching adapted to facial features
Snakes and flexible shape models

Figure 1.4 Methods of facial feature localization

Feature-based methods

Finding highly textured regions can help locating specific feature in the face. As mentioned by Kumar and Poggio (2000), most areas of the face with high frequency content (strong variations and a lot of edges) are localized around the eyes, the mouth and the nose. The Haar wavelet response in these regions is very strong and we might use a trained classifier to perform the detection and extraction. The proposed method is nearly real-time and can robustly capture various features. However, we will need to train a classifier and only the degrees of openness of the mouth and the smile are analyzed.

Integral projections of an edge image (Brunelli and Poggio, 1993) along (vertical and horizontal axis for example) can also be used to detect features. Locating peaks in the histograms created this way may help finding eyes for example.

Template-based methods

Specific templates are often used to detect particular features like the eyes or the mouth. For example, Tian *et al* (2000) use two different eye templates, one for open eyes and another for closed eyes, to be able to handle more situations. Using template matching to locate feature is still computationally intensive but the search is usually restricted to a much smaller region (compared to face detection) because we approximately known the region in which the feature should be.

Snakes or flexible shape models (Lanitis et al, 1997), using dynamic programming for calculation of the optimal deformation, are popular, useful and low computational complexity tools for facial feature extraction. They can be used to track specific contours and thus help in facial motion estimation. These techniques are robust to light changes, individual appearance and can handle small occlusions but are not very intelligent though, and can deform to fit to all kinds of objects, not at all looking like the object sought for. The major drawback is that the technique is heavily dependent on the pre-processing of the image, such as the edge detection and the location of the search area in which to extract the features (*i.e.*, the location of the area containing the face). Another problem if the deformable models are too complex is that they usually cannot handle rotations of more than a few degrees of the head around the vertical axis.

1.1.3 Image based view synthesis

In our study we reviewed two classes of possible view synthesis approaches: image-based view synthesis and model-based view synthesis. We will introduce the first one right away and the second one in the next section. We will try to focus on their advantages and drawbacks rather than explaining the methods in detail.

1.1.3.1 Introduction

Image-based view synthesis, as its name says, works on images only and tries to obtain a new, synthetic, but physically valid view from two given real views. Using this method there is no need for explicit 3-D reconstruction: we do not need to explain what we see in the scene. This means that we can handle scenes where body parts other that the head of the user are present, like a hand hiding a portion of the face for example. Because everything is done in the image domain, the new view produced should be more realistic than the one obtained with the model-based method.

However, this method will use more bandwidth than the model-based approach because whole images are transmitted: we're transmitting a pixel-based representation of an image sequence and even with the most efficient compression it will most likely overload the transmission channels. What's more, this kind of method can only generate a limited number of views because extrapolation is practically inexistent: the best technique we saw was capable of generating views within a 60 degrees cone of visual angle (Avidan and Shashua, 1998). Finally, all image-based methods require matching (most of the time, dense matching to work effectively) and our setup will most likely not allow it.

We classified the methods used into two main branches, as you can see in Figure 1.5, the ones requiring only scarce matching and the ones requiring dense matching.

1.1.3.2 Methods relying on scarce matching

Most of the techniques we saw relied on image interpolation. Image interpolation is quite simple but can only generate a series of new views on a straight line starting at one known viewpoint and ending at the other. This is why this kind of technique cannot be used practically.

Another interesting technique is the one using projective invariant (Havaldar et al, 1997). Under a perspective projection the ratio of points on a line (also called cross-ratio) is preserved and this technique uses this property to generate the new requested view. It works usually well with polyhedral objects because vertices can be detected reliably but curved surfaces will require dense matching.

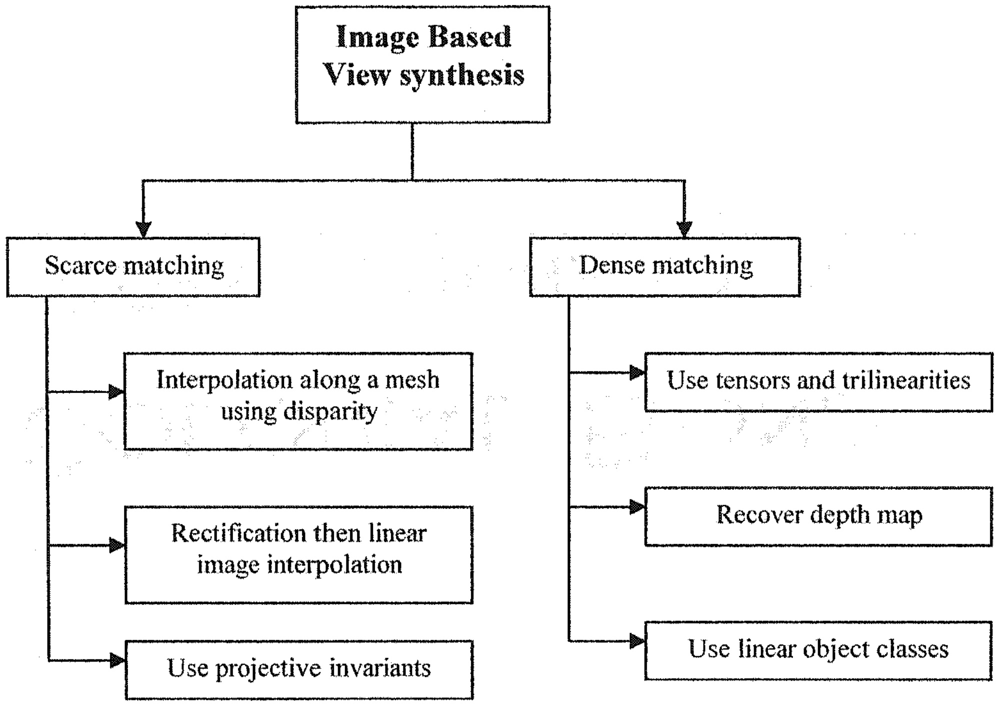


Image Based View synthesis
Scarce matching
Dense matching
Interpolation along a mesh using disparity
Rectification then linear image interpolation
Use projective invariants
Use tensors and trilinearities
Recover depth map
Use linear object classes

Figure 1.5 Methods of image-based view synthesis

1.1.3.3 Methods relying on dense matching

The problem with dense matching is that it is somehow very difficult to obtain in a natural scene. For each frame, several iterations are needed to obtain a dense matching and this makes real-time performance unattainable. On the other hand, once you have dense matching, you can most likely recover the depth map (Sengupta et al, 2000) and 3-D structure of the scene so view synthesis is rather easy.

The best results we encountered were obtained by using tensors and trilinearities combined with dense matching to perform view synthesis (Avidan and Shashua, 1998): three views are linked by a tensor representing the constraints between the three views. This technique can generate images from viewpoints within a 60-degree cone of visual angle. This is a lot better than image interpolation but it might still be not enough in our case.

Linear object classes were also used to perform view synthesis (Vetter and Poggio, 1997): if an object's 3-D shape can be represented as a linear combination of prototypical objects then its rotated view is a combination of the rotated views of the prototypes. But in this technique the 3-D pose has to be known and it is roughly equivalent to projecting the image into a prototypical space. The major problem is that is can only be applied to still object.

We think that the image-based approach is not very promising, in part because it requires transmitting each frame but also because it does not use any a priori knowledge that we have about the very specific scenes we're going to encounter. This is why we did more research on the next approach.

1.1.4 Model Based Coding for Videoconference

1.1.4.1 Introduction

Model-based view synthesis relies on high-level information extraction and attempts to explain what we're seeing in the images using a model. The advantages of this approach are obvious. It uses less bandwidth than an image-based approach because only facial animation parameters are transmitted: we're now working with a high level representation of the image not a pixel-based one. What's more, view synthesis is trivial once an approximate 3-D representation of the scene is built and can be produced by using well-known 3-D graphics methods. However, this approach is only able to handle what can be described by the model we choose, for example a hand in an image where the system expects only a face won't be correctly interpreted. We must therefore choose the model or set of models very carefully. This approach is also less realistic because of the conversion from the image to the parameters describing this image then to an image again. Also, some questions are still open: how to handle changes in the appearance (texture) of the model after the initialization? For example, what happens if a user decides to take off his glasses right in the middle of a conversation?

As we will see, the most difficult problem is how to extract the animation parameters from the frame sequence at the transmitter? Although we can set up a 3-D model, the description for this model has to be obtained from two-dimensional image plane information. Here lies the basic difficulty.

We can make the distinction between two kinds of model-based coding systems. The first one equipped with a general model like a 3-D wireframe model and able to adapt to any object is called *object-oriented coding*. The other, given a more specific model like a model of the human face and able to use the knowledge specific to this kind of object is named *semantic coding*. If we go further, we can now separate *semantic coding* into two classes:

- *Knowledge-based coding* when we know in advance which specific model to use.
- The other is when each object, during the image analysis, is recognized as a member of a class, and the specific model corresponding to that class is selected automatically.

In this work, we will only address the case of *knowledge-based coding*. We know that the object of attention is a human face.

1.1.4.2 Three major steps

We divided model-based view synthesis into three major steps:

- At the beginning, model initialization is necessary to adapt the 3-D model to reflect what is in the image.
- Rigid motion estimation (translation and rotations of the head, 6 parameters in total) is required to separate rigid motion from non-rigid motion.
- Finally non-rigid motion estimation, which is the extraction of the facial movements in order to reproduce them on the 3-D model at the other end of the system. We will not introduce it in our project at this time.

1.1.4.3 Model Initialization

This step is very important because it is where the model, specific to the current user, is created. The initialization module should provide an accurate model of the user in a neutral position but also a set of textures that will be used afterwards to render a more realistic representation of him. Initialization can be performed either by creating a new model or by adapting an existing model as shown in the Figure 1.6.

Creation of a 3-D model of the scene

Building a model for a user without using specific hardware (*i.e.* a CyberWare scanner) can be compared to recovering the 3-D structure of an unknown scene. This is very difficult and as been a major subject of research in computer vision for at least the last 25 years. Most of the methods proposed can only work under specific or controlled conditions and are therefore mostly useless to us. However, we will mention some of them briefly to demonstrate our point.

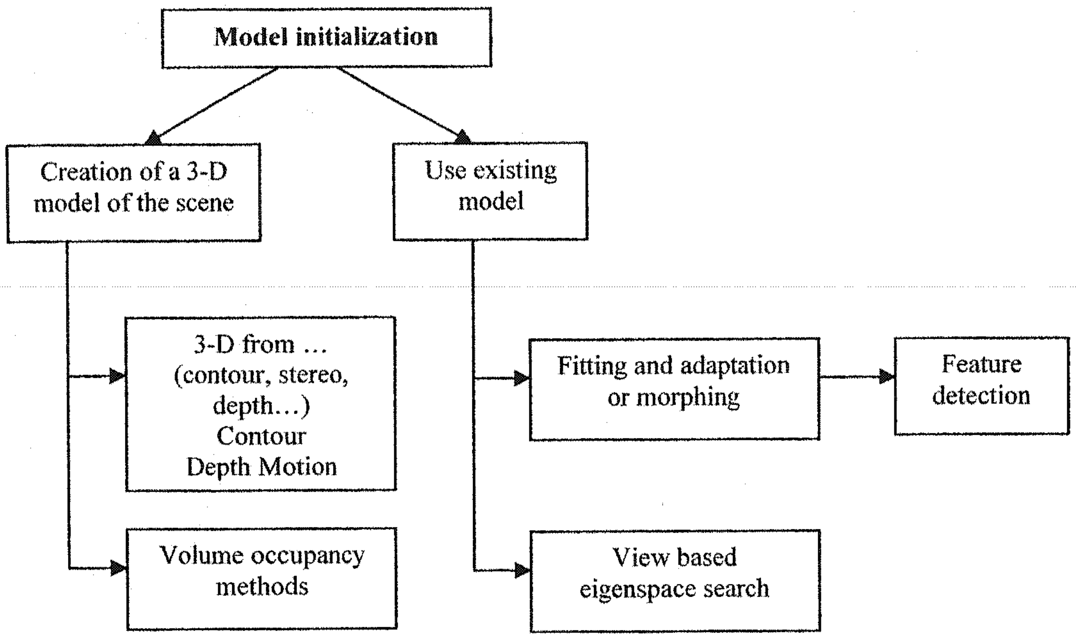

Model initialization
Creation of a 3-D model of the scene
Use existing model
3-D from ... (contour, stereo, depth...) Contour Depth Motion
Volume occupancy methods
Fitting and adaptation or morphing
Feature detection
View based eigenspace search

Figure 1.6 Methods of model initialization

The 3-D from contour technique (Zheng, 1994) works from a continuous sequence of images taken when an object rotates. The problem is that the rotation must be controlled and the user would have to remain static during all the initialization phase, which can take a long time. Therefore, it cannot be applied to our project.

3-D from depth (Sengupta et al, 2000) after the face segmentation uses multiple images to calculate the depth map of the face on the base-frame, then modify the model. The problem is that it requires at least 2 cameras and needs dense matching.

The 3-D from motion technique (Azarbayejani and Pentland, 1997) needs to track feature points in an image sequence (rigid motion only) to estimate simultaneously the focal length, the structure and the motion parameters. Unfortunately, it will only work reliably with simple objects not objects as detailed as the human face.

Aside from the "3-D from X" techniques, another family of techniques can recover the structure of an object: volume occupancy methods (Moezzi et al, 1996; Moezzi et al, 1997; Eisert et al, 2000). The idea is to divide the environment into a set of cubic volume elements (voxels). By finding which voxels are occupied we can infer the 3-D structure of the observed object. The problem is that it usually requires six or more calibrated cameras (sometimes up to 16 cameras are used) disposed strategically all around the observed object and it isn't real-time.

Using an exiting model

The other option for initialization is to use an existing model, a universal one, and adapt it to the current user according to the information we can get from the available images.

Fitting and adaptation or morphing (Zhang and Cohen, 2000) are techniques commonly used to perform such tasks. After finding a set of key feature points using feature extraction methods a generic model can be deformed in a way that minimizes the global differences between the located features points and the corresponding points on the 3-D model. Lee and Magnenat-Thalmann (1999) use 3-D feature points as a set of control points for deformation. Then the deformation of a surface can be seen as an interpolation of the displacements of the control points. Other methods rely on energy minimization techniques like the one proposed by Feng *et al* (2000), where a spring based face model is used. The nodes are connected by springs making the nodes move on the image, seen as an energy field, until they arrives at equilibrium.

A robust method used to initiate the fitting could be view-based search (Essa and Pentland, 1997; Moghaddam and Pentland, 1997). This would give us a rough approximation of the 3-D pose of the head and facilitate the model fitting.

It is important to mention that the problem if we use only one camera view is that no depth information can be used and the information about feature point is only 2-D. The polygonal mesh would fit the source image in the x-y plane and we would assume that the errors in the z coordinates of the polygon mesh (in practice we can assume that every z coordinate will be erroneous) will not be noticeable for small rotations of the head.

1.1.4.4 Rigid motion estimation

Rigid motion estimation is roughly equivalent to 3-D head tracking. We would like to recover as accurately as possible the 3-D pose of the head.

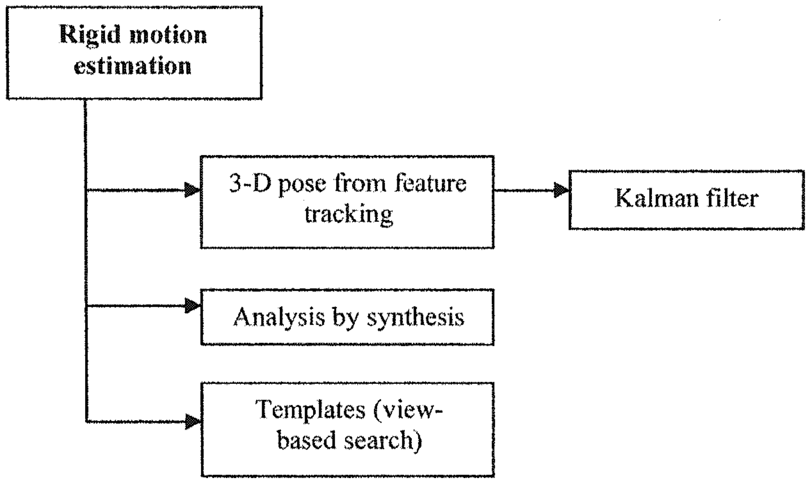


Figure 1.7 Methods of rigid motion estimation

3-D pose from feature tracking

Recovering the rigid motion of the head from feature tracking is the most widely used method. One technique is to find key feature points like the mouth and the eyes: those features form a plane that can approximate the pose of the head. Then, the pose is refined using other detected feature points. Another standard technique (Malciu and Preteux, 2000) relies on a 3-D/2-D matching between 2-D image features estimated throughout the image sequence and 3-D object features of a generic head model. If there is enough features available we can estimate the 3-D pose of the head. Matsumoto and Zelinsky (1999) use a stereo vision to recover the 3-D position of key feature points like the corner of the eyes, nose and mouth tracked using templates matching techniques. Finally, the latest systems in development use simulating annealing and statistical pattern matching to find the location, rotation and approximate shape of the face.

Finding the 3-D pose of a head from a set of feature points is not particularly hard and those kinds of systems are relatively immune to partial occlusion, facial deformations and noise. Most of the time, it is the feature detection and tracking that requires all the processing power.

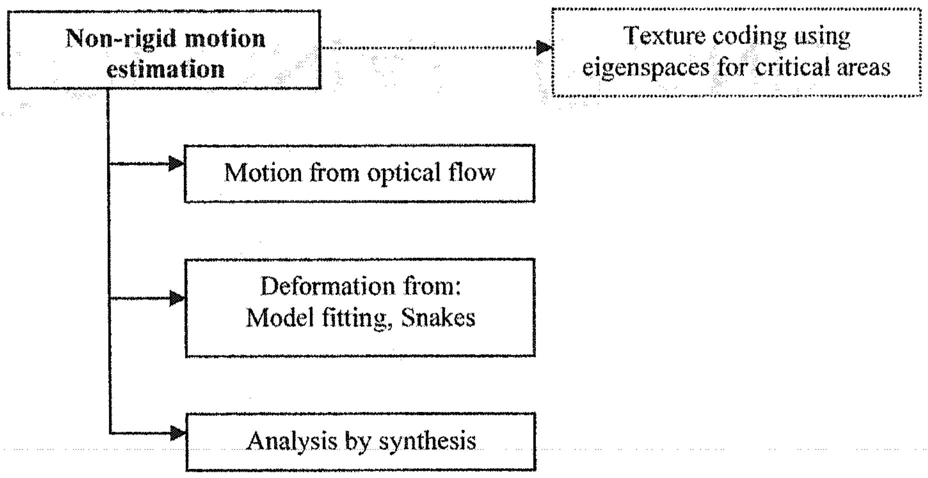
Other methods

As we mentioned earlier, a view-based search can be used to recover the rough 3-D pose of the head. See the previous descriptions for more details.

Analysis-by-synthesis methods will be described in details in the next section because it is mainly used for non-rigid motion estimation.

1.1.4.5 Non-rigid motion estimation

We need be able to distinguish what motion in the image is produced by rigid motion and what is produced by non-rigid motion. We will then be able to isolate the facial movements creating the expressions on a person's face. The following diagram presents the most commonly used methods for non-rigid motion estimation:



Non-rigid motion estimation
Texture coding using eigenspaces for critical areas
Motion from optical flow
Deformation from: Model fitting, Snakes
Analysis by synthesis

Figure 1.8 Methods of non-rigid motion estimation

Motion from optical flow

The majority of the work facial expression analysis was done through optical flow estimation. The general method is to estimate facial motion using optical flow and a Kalman filter for the control framework (Essa and Pentland, 1997). Motion is then coupled to a model (sort of finite element representation for the face) to compute the muscle control variables.

Eisert and Girod (1998) propose that the motion be estimated using with the combination of optical flow constraint and 3-D motion models. They used the whole face image to estimate motion, analyze facial expressions and finally produce a set of facial action parameters compliant with MPEG-4 SNHC standard.

From the experiments reported in the literature, we think that real-time performance can be reached or will certainly be in the next few years. However, as mentioned by Donato in 1999 that flow is a noisy measure. So many flow-based expression analysis systems employ regularization procedures such as smoothing and quantizing to estimate the principal direction of motion within an image region and this suppresses high order spatial information that might be important to recreate subtle expressions.

Deformation from model fitting or snakes

The analysis phase of the face can also be done by using deformable contours, or snakes, to track the movement of facial features. These are contours that can be set up to lock on to moving image features. They are given certain pseudo-physical attributes that enable them to interact with images as if they were force fields. Snakes are se up to track prominent facial features such as lips outline and eyebrows.

They were used to track the lips and the corner points of the eyes and thus to analyze the facial expression by Zhang (1998). Facial expressions parameters can directly be estimated from the snake state variables (Terzopoulos and Waters, 1993) and then used to deform the facial model. As we mentioned earlier, because the snakes do not incorporate prior knowledge about expected shapes, this approach maybe confused by other structures present in the image or large occlusions.

Analysis by synthesis

The basic idea is to guide the model adaptation by the measurable differences between the synthesized model reconstruction and the real camera input sequence (Koch, 1993). A model-based coder can be implemented as an *analysis-by-synthesis* coder: this means that the encoder will contain a feedback loop but also a copy of the decoding process. The coder can therefore use the synthesized image and compare it to the original image. By minimizing the residual image, the encoder can optimize the transmitted parameters. This also permits the encoder to transmit a residual image, to improve image quality at the receiving site.

This is a very promising method but the drawback is that the feedback loop is like recursive estimation: usually it will take time to converge to a good solution and therefore the real-time performance might be degraded.

Coding using eigenspaces

Some areas of the face are more critical than others in a visual communication system. In particular the eyes and the mouth of a person transmit a lot of visual information and it may not be appropriate to use facial action parameters to describe what is happening in those areas. What's more, some 3-D head model are lacking the necessary detail in those regions: they do not have any teeth, tongue, or cannot handle closed eyes.

To correct to this situation we could simply transmit the whole texture of these areas each frame. This requires a lot of bandwidth so we could also use eigenspaces (eigeneyes and eugenmouth) to compress this transmission. This way we would only have to transmit a set of coefficient describing the feature state for each frame. This method is very efficient and has already been used by Donato *et al* (1999), Lanitis *et al* (1997) and Okada *et al* (2000).

1.1.4.6 The choice of models

The choice of the different models used is quite important because everything is somehow related to them. The first model to choose is the geometrical head model type: do we want a mesh based model where each vertex is a control point or do we want a free-form parametric model that can be controlled by a few set of parameters? Most of the models used in earlier works are mesh-based whereas most of the models we encountered in the papers treating non-rigid motion estimation were free-form parametric ones. We also may want to choose a model where the polygonal resolution increases in the critical areas such as mouth and eyes.

The second important model is the facial expression extraction model: what is our high level representation of a facial expression? Do we want to represent facial expressions as a surface deformation of the model (*i.e.* moving vertices on the surface of the 3-D head model) or do we want a model closer to an actual physical human being? The first is based upon the FACS system developed by P. Ekman and W. V. Friesen in the seventies which was the basis used to develop the MPEG-4 SNHC standard. The latter model is more realistic but also more complex and the computational price needed to deform the model is high. The motion specification problem is addressed by introducing an underlying muscle model and the polygon mesh nodes are distorted to produce expressions using the muscle model which is itself parametrically controlled. The overall model is hierarchical with a few parametric muscle models controlling many polygon mesh nodes in their zone of influence. Figure 1.9 illustrates the possible choices:

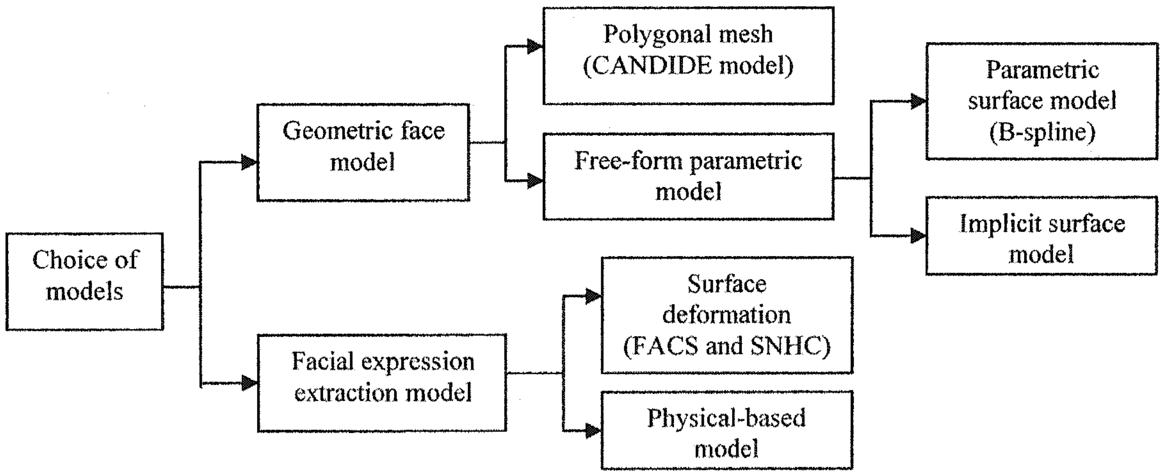

Choice of models
Geometric face model
Facial expression extraction model
Polygonal mesh (CANDIDE model)
Free-form parametric model
Parametric surface model (B-spline)
Implicit surface model
Surface deformation (FACS and SNHC)
Physical-based model

Figure 1.9 Choice of face models

1.2 Fundamental issues

The assumption in the low-bandwidth coding of head-and-shoulder image sequences is that the scenes are constrained to a few objects, in our case, only a speaker's head, for which a priori models can be developed. In addition, certain facial features are always visible, in other words, head rotations and the tilts that impede the visibility of eyes and mouth are precluded. Similarly the face is not occluded by other objects such as hands.

The main tasks of the low-bandwidth coding of head-and-shoulder image sequences consist of:

- Face detection and facial feature localization in order to supply initial information for the remaining tasks.
- Adaptation of the 3-D generic model to the actual face in the image, *i.e.* scaling and posing of the generic model and local adaptations to reflect the individual characters.
- Tracking of motion parameters as well as expression parameters.

The first task is crucial for a model-based coding system for videoconference because one cannot process things that one cannot detect and all other tasks depend upon it. Although face detection is quite simple for the human visual system, it has proven to be a difficult task for machine-vision. Human faces have the same basic structure but at the same time they are very different from each other. Even the appearance of a single face can change drastically over time, because of all the possible facial expressions, the lighting, and the pose, etc.

Once the face is detected in the image, it is not difficult to adapt the size of the face model. General low-bandwidth coding of head-and-shoulder image sequences requires the mapping of a face texture onto the wire grid of the face. A difficulty exists when one wishes to map all the 3-D nodes of the face model onto the 2-D face texture for texture mapping. A method needs to be developed to register 3-D nodes onto 2-D image pixels under the constraint of maintaining the spatial distribution of the face texture invariant.

Among the different 2-D tracking methods, block-based method is popularly used in different applications. In our case, only tracking the facial features found in the former stage are not enough for the motion estimation. Most of the time more than six features are needed for the purpose of noise reduction. More features for tracking must be selected on the face region using an appropriate feature selection criterion. In addition, the block matching criteria, searching strategy, motion model and the computational framework must be selected carefully to save computation and reduce accumulated errors.

Finally, our main goal is to achieve real-time performance, and to be able to have fluid image sequences. This constraint could simply be defined in terms of number of frames per second (fps) displayed at the receiving side: we think that 15 fps is a realistic target. Anything higher than this will likely go unnoticed by the human eye.

1.3 The proposed system

We propose a real-time model-based coding system for videoconference based upon reliable face detection and facial feature localization methods, feature tracking, a novel registration method of mapping between 3-D nodes and 2-D image pixels. For companion purposes, we use three rigid motion estimation methods. The system processes the monocular image sequences with the assumption that a face and shoulder are shown in the images. Figure 1.10 depicts the whole system and transactions among the various components.

In the face detection stage of the system, we first segment the skin-color region from the background by using the color segmentation method based upon the results of Chai and Ngan (Chai and Ngan, 1999). Histogram equalization and fitting are proposed here to reduce luminance effects on the face color. Image processing techniques and elliptic shape identification are used to eliminate noise-induced regions of the background.

Compared to other regions of the face, regions corresponding to facial features are of different color and lower luminance values. Candidate facial feature regions are segmented based upon this assumption. Facial features are then selected by using a knowledge-based selection mechanism among these candidate facial feature regions. The selection proceeds by computing a score for each candidate region, which in turn can be interpreted as the likelihood that the region corresponds to a facial feature.

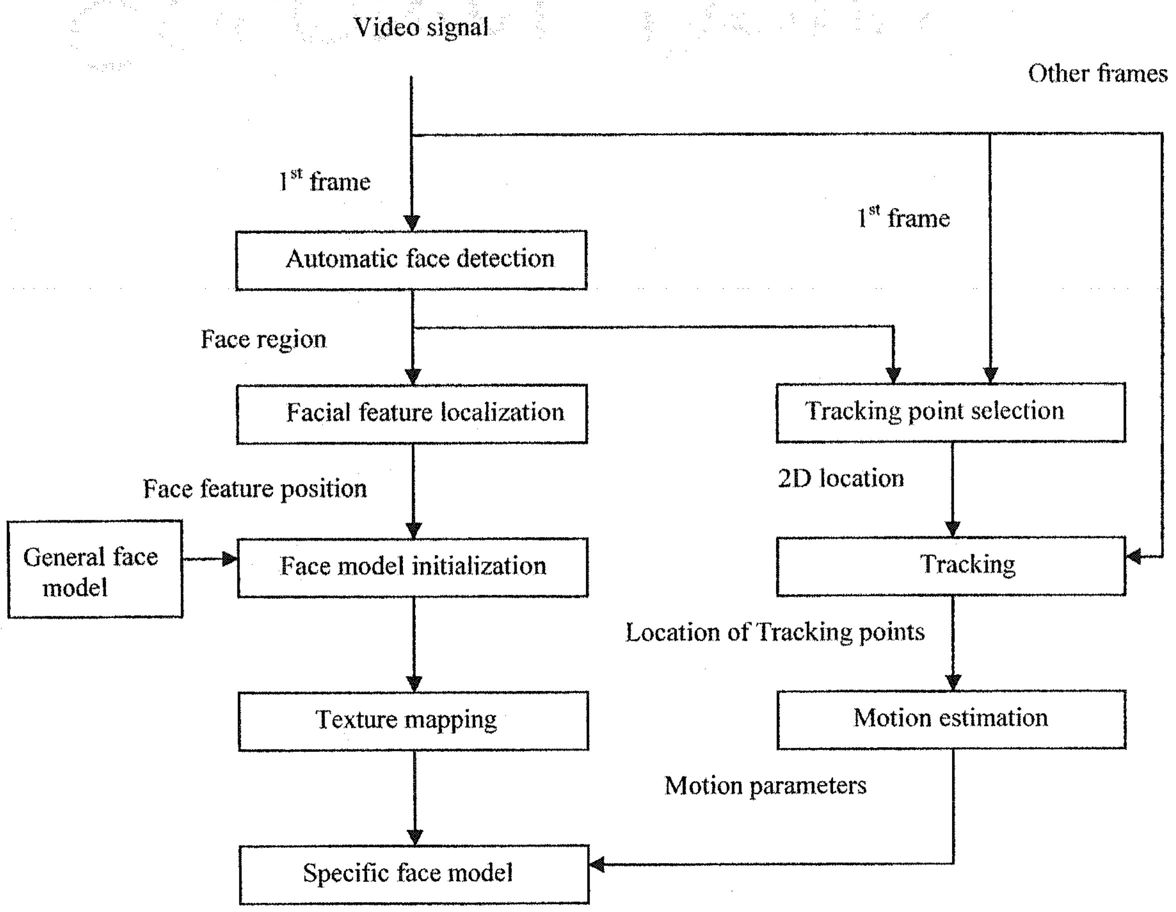


Figure 1.10 The proposed model-based coding system for videoconference

In the model initialization stage, the face model, developed by Keith Waters at Compaq Cambridge Research Laboratory (Parke and Waters, 1996) is globally adapted, based upon the knowledge of former stages. The face texture is then mapped onto the face model. We propose a novel method to register the 3-D nodes of the face model with the 2-D image pixels on the face texture for texture mapping.

In order to obtain good tracking results, features for tracking can be selected based upon some measure of texturedness or cornerness. We use a feature selection criterion, which is optimal for tracking. In order to select a appropriate tracking method for our system, both tracking and monitoring method and full-search block matching method are tested for searching the new positions of the selected features in the following frames.

Using the feature positions of two neighbor frames, three recursive methods: Predictive Least-square (PLS), PLS-Kalman and Structure from motion (SFM) of real-time estimation of 3-D motion parameters are compared at the rigid motion estimation stage. The first two methods use depth information from the face model and a prediction-and-correction technique that prevents the error accumulation. The SFM method can recursively estimate the motion parameters and focal length without using any depth information. Kalman filtering is used in the last two methods. The estimated motion parameters are used to control the face model.

1.4 Structure of the thesis

Chapter 2 describes a face detection algorithm that uses face color and some image processing techniques. The characteristics of human face color and the improvement by using color histogram equalization and fitting to correct the color of input image are introduced first. Then, an elliptic shape identification algorithm is introduced in detail.

Finally, we describe how the above algorithms and image processing techniques are used to perform the face detection.

Chapter 3 describes the proposed facial feature localization algorithm. A luminance verification method, which can provide information about facial features by using the assumption that the facial features have lower luminance values than that of other parts of face region, is proposed. After the principle of knowledge-based selection method is introduced, the measurements for facial feature selection are proposed in detail. Z-tables approximating normal distribution are used to represent the probabilities of a candidate region being a facial feature.

Chapter 4 describes the proposed model initialization method. The face model is introduced as well as the method of global adaptation. We propose a registration method to register the 3-D nodes of the face model onto 2-D image pixels for the purpose of face texture mapping.

Chapter 5 describes a feature selection method for tracking and two tracking methods.

Chapter 6 describes the three face rigid motion estimation methods used in the system. First, it introduces the motion model and computational frame of PLS and PLS-Kalman algorithms. In these two algorithms, the translational motion in the Z direction is assumed to be zero, so that only five motion parameters can be estimated. We then propose a method to obtain the depth values of selected tracking points from the face model. Finally, the SFM algorithm is introduced allowing the estimation of all the six motion parameters. Implementation details and results are presented at end of Chapter 6. Advantages and disadvantages of the different motion estimation algorithms are also discussed.

Chapter 7 concludes the thesis with a summary of the work, its contributions and the direction of future research and improvements.

CHAPTER II

FACE DETECTION

Face detection is crucial for a model-based coding system for videoconference because one cannot process things that one cannot detect and all other tasks depend upon it. All the information obtained will be used in the later stages, such as facial feature localization, tracking, motion estimation, model initialization and facial expression analysis.

To be able to perform face detection, the following basic assumptions about the head-and-should images are made:

- There exists only one object in the foreground consisting of face and shoulders of a human.
- There are no interfering skin-color regions with large areas around the face.
- Certain facial features are always visible. The face is not occluded by other objects such as hands.
- The head rotation (around the z-axis) is less than 45 degree.
- The human face has an approximate vertical symmetry.

The use of color information has been introduced to the face detection problem in recent years, and it has gained increasing attention since then. Some recent publications (Chai and Ngan, 1999; Garcia and Tziritas, 1999; Lin and Wu, 1999; Crowley and Schwerdt, 1999; Zarit et al, 1999; Kim and Kim, 2000; Terrillon et al, 2000; Herpers et al, 1999) have reported this study.

Color information is typically used for face region detection rather than edge-based segmentation. Basically, Various statistical color models are used to classify the image pixels into skin or non-skin pixels. Differences among existing skin detection systems occur primarily in the following areas: color space used, training method used, and techniques used for sorting and identifying colors corresponding to skin.

Chai and Ngan (1999) analysis several color spaces, such as RGB, HSV, YCrCb which are popularly used in the face color segmentation. They suggest that the YCrCb color, in which the Y value represents the luminance (brightness) component while the Cr and Cb value represent the chrominance components, is the appropriate one among these color spaces. One reason is that it can use chrominance information effectively for modeling human skin color. Another reason is that it can be typically used in video coding. The author also found that a skin color region can be identified by the presence of a certain set of chrominance (*i.e.* Cr and Cb values narrowly and consistently distributed in the YCrCb color space). The location of these chrominance values has been found: the Cr between 133 and 173,the Cb between 77 and 127 correspond to the face color. The experiments show a good result of the reference face map. The face detector can detect wide range of skin color, including the people of European, Asian, and African decent.

In order to cope with strong lighting variations, Garcia and Tziritas (1999) shows that in the YCrCb color space, the distribution turns out to be different for the extreme Y values corresponding to dark (around 50) and light (around 240) lighting conditions. Therefore, the author suggested more accurate color subspace related to face color using Y value.

2.1 Introduction of proposed face detection algorithm

Figure 2.1 illustrates the various modules of the proposed face detection algorithm. Details will be introduced in Section 2.2.

2.1.1 Improved color segmentation method

In order to improve the method introduced by Chai and Ngan (1999) with lighting variation limitation, we use skin-color ranges obtained from a randomly selected face image to segment the face region after performing proposed color correction algorithm. The idea of color correction method is to correct the color histograms of all face images to fit to the color histograms of the image we selected. Therefore, we can avoid to analysis the histogram for each type of face images before the face color segmentation.

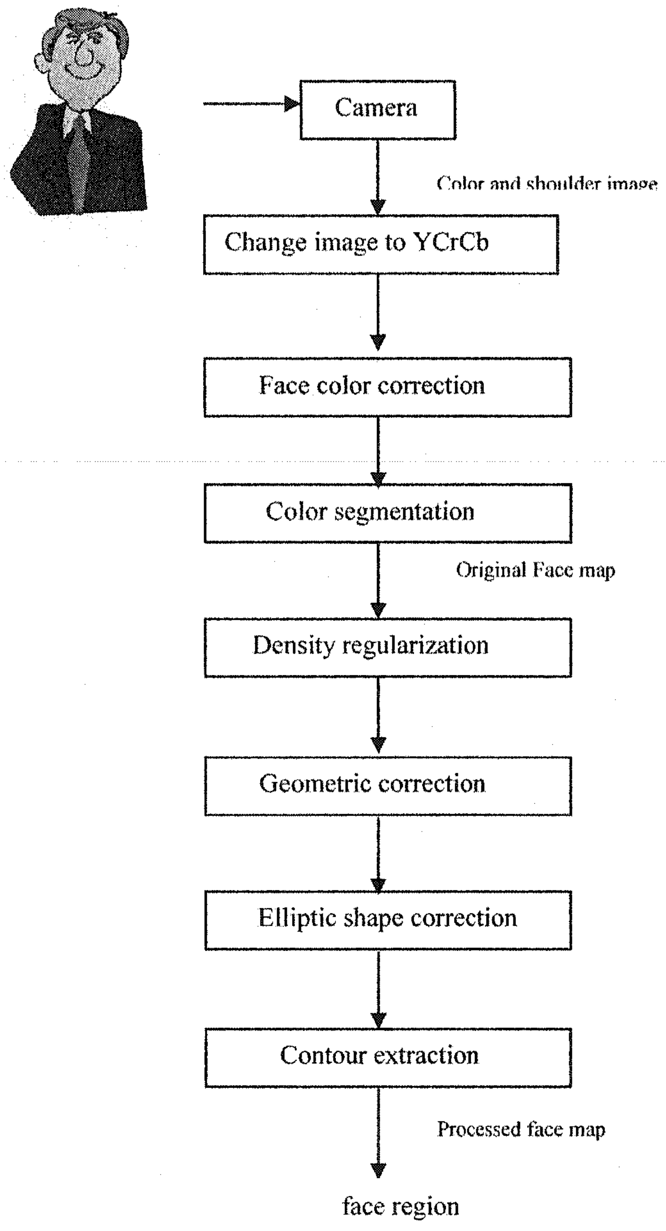


Figure 2.1 Modules in the proposed face detection algorithm

2.1.2 Detecting face efficiently

In almost all references, a uniform background is assumed at of the color-based detection process. While some papers assume a complex background that background is affected by sparse noise, so that simple image processing techniques can eliminate it. However, if images contain large skin-color regions in the background, it becomes impossible to eliminate such regions by simple filtering. In our system, elliptic shape correction after color segmentation can eliminate these erroneous regions.

Sobottka and Pitas (1997) suggest that the shape correction be performed in the image directly. However, because of the large number of pixels in the image, the calculation of gravity centers and moments becomes computationally demanding. Because the purpose of shape correction is to eliminate the large skin-color regions in the background, we can do the same thing in the interpolated image. In our project, we perform the shape correction in the density map. The size of density map in our project is 40×30. Comparing with the image size 320×240, the amount of computation is largely reduced.

The elliptic shape correction can provide several important face parameters such as the location of the face center, the face size and angle. All these parameters can, then, be used in the facial feature localization stage.

2.2 Proposed face detection algorithm

2.2.1 Face maps

Face maps are binary values that represent face information. Value one in the face map indicates whether the pixel corresponding to that value is a skin-color pixel. A zero value indicates that the corresponding pixel to that value is not a skin-color pixel.

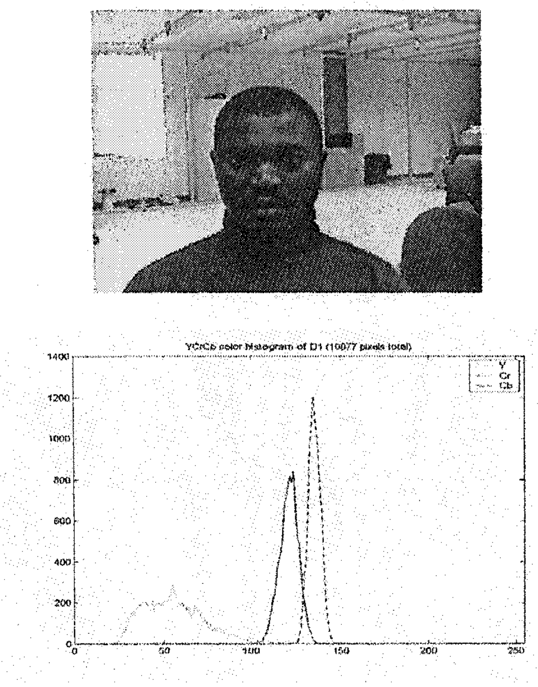
Three maps are used in our system. The first one, called the original face map, results from color segmentation and contains facial feature information that we can use later. The second one, called processed face map, results from the original face map by using the image processing methods introduced below. The processed face map contains only the information of face region. No facial feature information is contained. The third one, called density map, is an interpolated face map used for skin-color noise reduction.

2.2.2 Color segmentation in YCrCb space

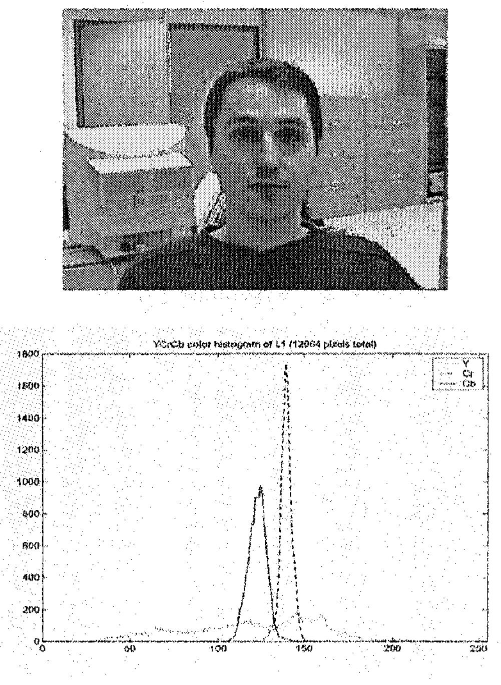
In order to analysis the face color histogram in the YCrCb space, we grabbed fifteen face images in our lab. The face images include different skin color people. We separate these images into three classes: dark-skinned people, light-skinned people and Asiatic people. Then we manually select the face region in each image and calculate the face color histograms of each image. The average histogram for all the images is calculated by normalizing the pixel number of selected face region in each image. Figure 2.2 shows three examples of face images in different classes and their color histograms. Figure 2.3 shows the average histogram of all people.

These results head to several conclusions: First, the ethnic classes do not affect the skin color distribution in Cr and Cb components in the same environment with small illumination changes. But we do not know if large illumination changes affect the Cr and Cb distributions or not. Second, the histograms of Cr and Cb components for all three classes clearly showed that the chrominance values are indeed narrowly distributed and the distributions are consistent across different classes. Third, the non-zero ranges of Cr and Cb in the average histogram are [123, 161] and [86, 149] respectively. Fourth, classes of the light-skinned and Asiatic people have similar distribution of Y component, with a range from 50 to 200. But the dark-skinned people class has a different distribution, which range from 25 to 150.

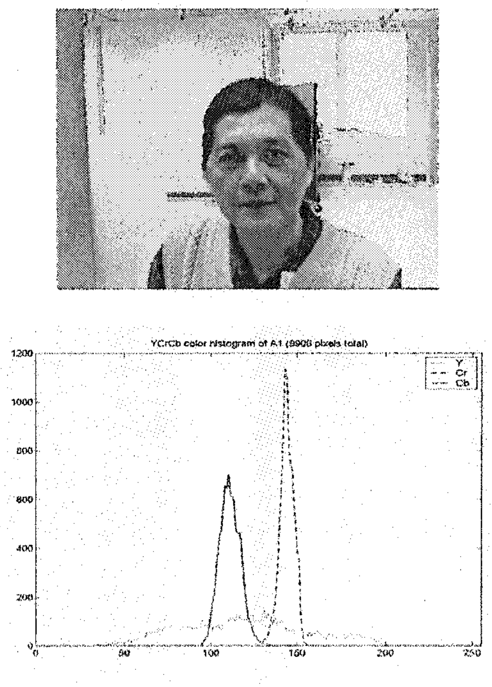
The use of the face color subspace in YCrCb space described by Garcia and Tziritas (1999) to segment the face region failed for all available face. We therefore decided to use the simple color ranges of Cr and Cb for color segmentation.



(a)



(b)



(c)

Figure 2.2 Face images and their color histogram in YCrCb color space

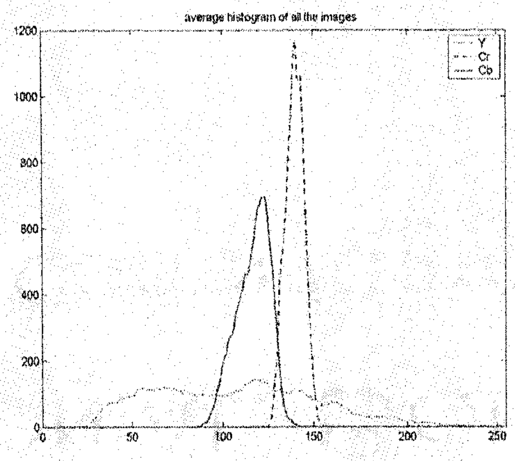


Figure 2.3 Color histogram in YCrCb color space for all face images

The above color histograms help select the appropriate ranges of Cr and Cb for face color segmentation. It is preferable to select narrow color ranges in both Cr and Cb components and the numbers of face pixels included in the selected ranges are not sensitive to the border changing. The first condition can greatly reduce skin-color regions in the background and is able to generate non skin-color holes in the face regions, which can lately be used during the facial feature localization stage. The second condition can guarantee that most of the skin-color pixels are selected by the color segmentation.

The ranges we selected are [128,152] for Cr, [93,138] for Cb for the grabbed images using the average face color histogram and two selection criteria introduced above. Comparing with the results of Chai and Ngan (1999), [133,173] for Cr, [77,127] for Cb, both the ranges of Cr and Cb have large differences.

Figure 2.4 shows several face images from different sources.

Figure 2.5 shows the results after the color segmentation by using the face color ranges indicated by Chai and Ngan (1999). The light and black points in the images represent the skin-color pixels and non skin-color pixels in the face images respectively. We can see only in image (a) and (b) the face regions can be segmented successfully. Image (c) has large holes in the face region. Others fail in the segmentation totally.

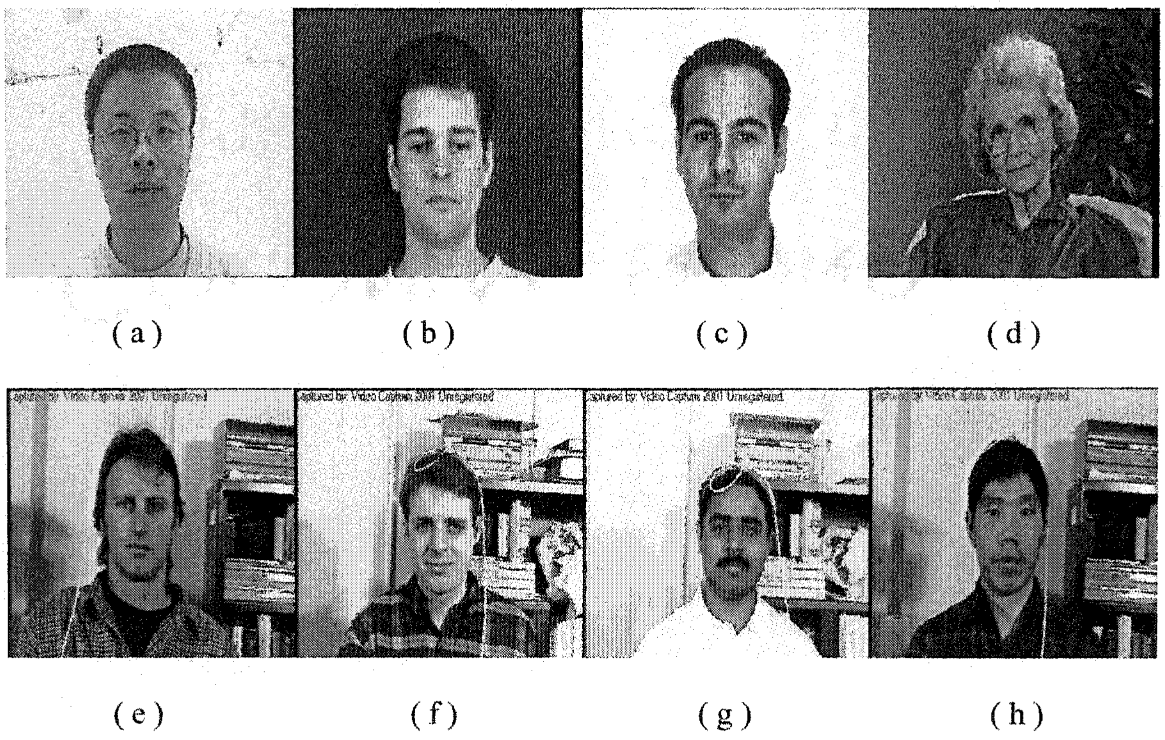


Figure 2.4. Face images from different sources

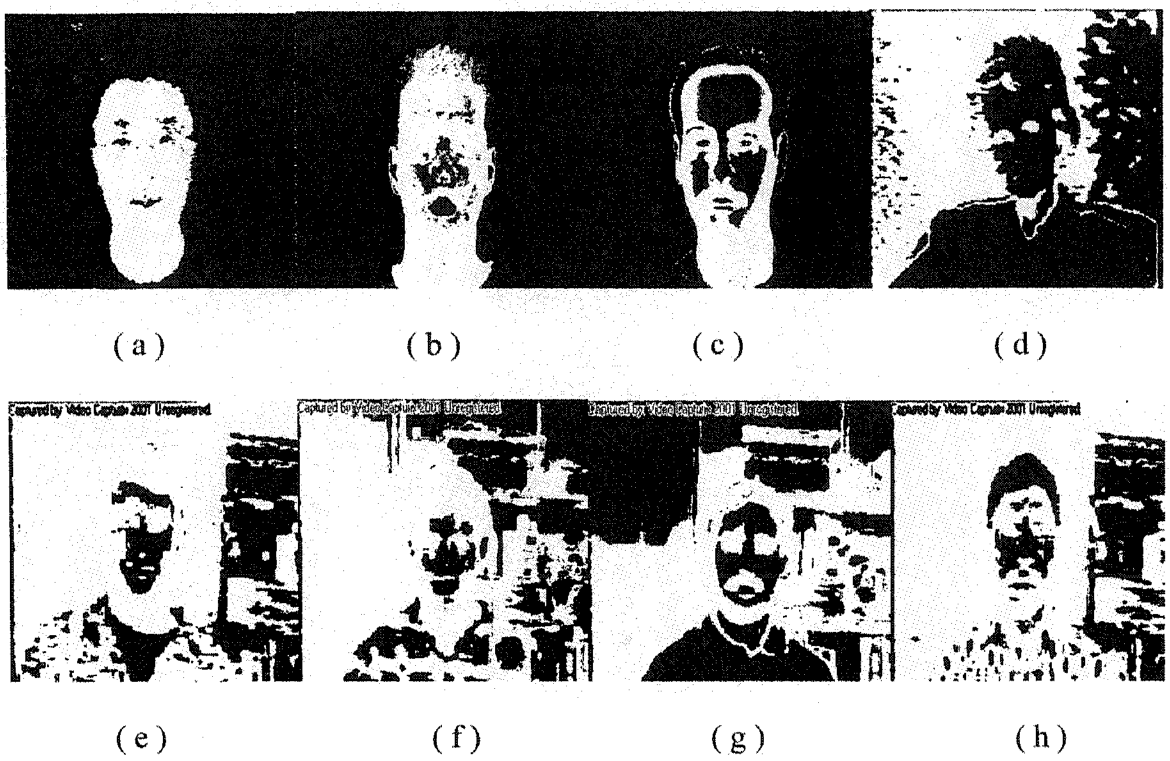


Figure 2.5 Color segmentation using the face color ranges

declared by Chai and Ngan (1999)

We can conclude that even though the chrominance values are narrowly distributed in the YCrCb color space, the positions and sizes of these ranges are related to the luminance value Y, *i.e.* the positions and the sizes of the color ranges change with the luminance. So if the image-grabbing environment changes, the ranges of face color also change. In this situation, the best way of using color segmentation for certain type of face images is to analysis their color histograms before color segmentation.

2.2.3 Face color correction

As introduced in Section 2.3 later, we use the AR face database to test our face detection algorithm. In order to have accurate results for face detection, we analyze the color histogram again for the AR face database. Good results can be obtained by using the color range [134, 207] for Cr and [66, 123] for Cb.

Figure 2.6 illustrates the results obtained using above color ranges. We can see that the face region was successfully segmented only in the first three face images. One way to solve this problem is to analysis the histogram for each type of face images before the face color segmentation. However, this is not convenient in practice and a color correction method is needed so that general face color ranges can be used for different types of face images.

Histogram equalization method is sometimes used to correct the illuminance. The idea of color correction is based upon the concept of histogram equalization. We assume that we can correct the color histograms of all face images to fit to the ideal color histograms after histogram equalization.

Two possible ways to do this are considered:

- By implementing the color correction in the RGB space and then performing the color segmentation using predefined color ranges.

- By implementing the color correction in the YCrCb space and then performing the color segmentation using predefined color ranges.

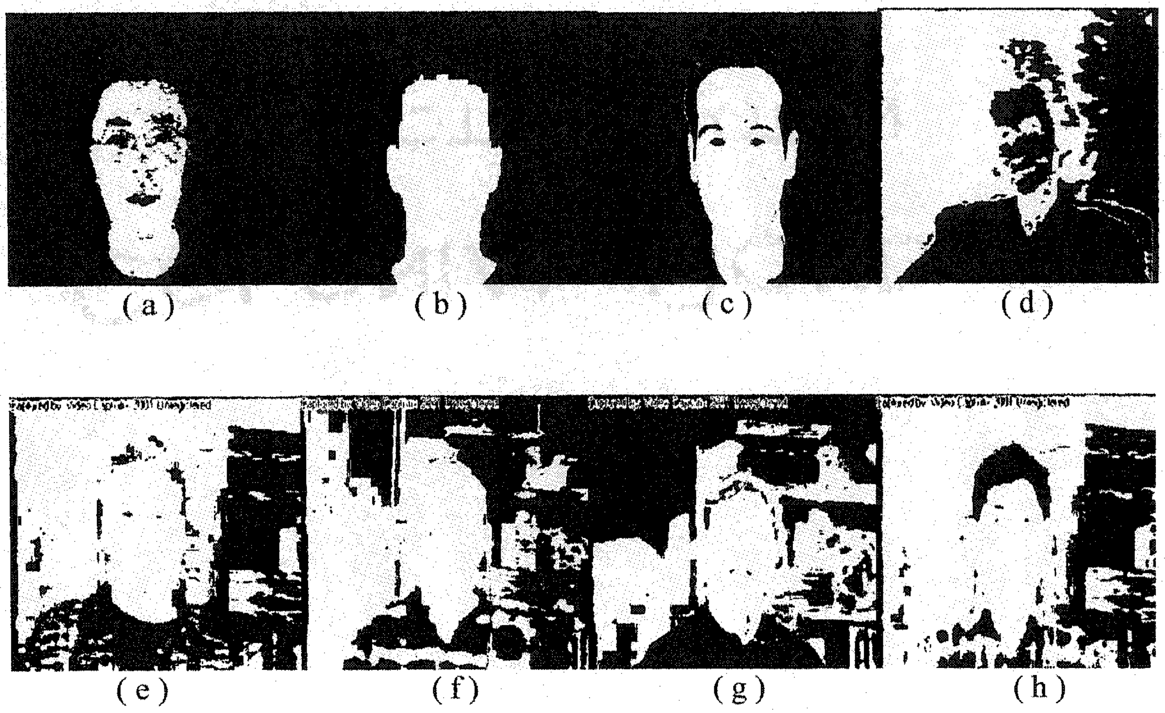


Figure 2.6 Color segmentation using the face color ranges from AR face database

2.2.3.1 Histogram equalization

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image, which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail is compressed into the dark end of the histogram. If we could `stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Histogram equalization involves finding a grey scale transformation function that creates an output image with a nearly uniform histogram. Assuming that grey values are continuous between 0 and 1, we must find a transformation T that maps grey values r in the input image F to grey values $s = T(r)$ in the transformed image $\hat{F}$. Details of histogram equalization are introduced by Castleman, 1996. Figure 2.7 is an example of histogram equalization.

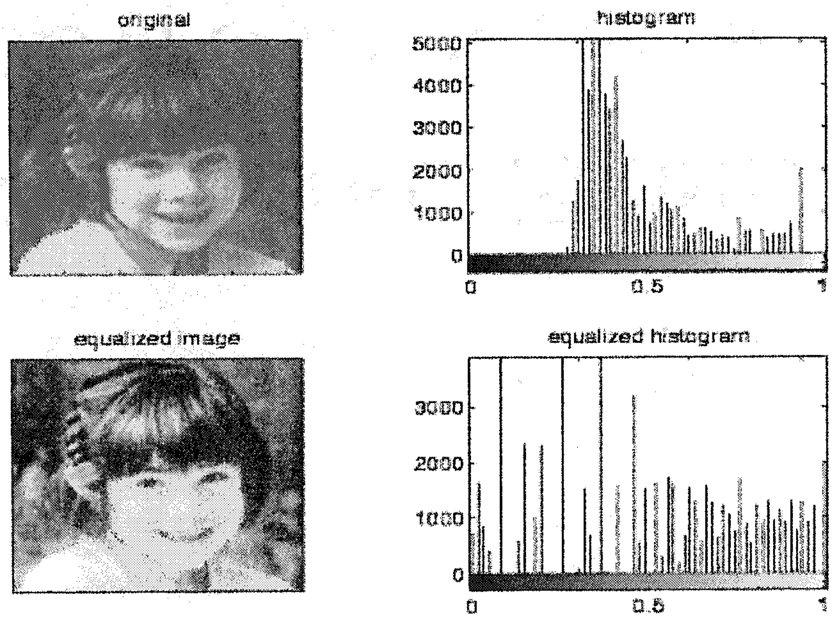


Figure 2.7 The original image and its histogram, and the equalized versions

Both images are quantized to 64 grey levels

2.2.3.2 Histogram fitting

Let $H(i)$ be the histogram function of an image, and let $G(i)$ be the desired histogram or wish to map to via a transfer function $f_{H\to G}(i)$. First, we compute a transfer function for both $H(i)$ and $G(i)$ that will map the histogram to a uniform distribution histogram, $U(i)$. The functions are $f_{H\to U}(i)$ and $f_{G\to U}(i)$ respectively. Following the histogram equalization equations introduced above:

$$f_{H\to U}(i)=\frac{\sum_{j=0}^{i}H(j)}{\sum_{j=0}^{n-1}H(j)} \qquad (2-1)$$

$$f_{G\to U}(i)=\frac{\sum_{j=0}^{i}G(j)}{\sum_{j=0}^{n-1}G(j)} \qquad (2-2)$$

where n is the number of discrete intensity levels, for 8-bit images, n=256.

To find the mapping function, $f_{H\to G}(i)$, we invert the function $f_{G\to U}(i)$ to obtain $f_{U\to G}(i)$. Since the domain and the range of the functions of this form are identical, the inverse mapping is trivial and is found by cycling through all values of the function. However, due to the discrete nature of these functions, inverting can yield a function, which is undefined for certain values. Thus, we use linear interpolation and assume smoothness to fill undefined points of the inverse function according to the values of well-defined points in the function. Thus, we generate a fully defined mapping $f_{U\to G}(i)$, which transforms a uniform histogram distribution to the distribution found in histogram $G(i)$. The mapping $f_{H\to G}(i)$ can then be defined as in the following equation

$$f_{H\to G}(i) = f_{U\to G}(f_{H\to U}(i)) \qquad (2-3)$$

2.2.3.3 Target histogram selection

We use the image shown in Figure 2.4 (c) to obtain the target (ideal) histogram. Several reasons justify this choice: (1) the image constitutes a typical sample of the AR data base on which we want to test the face detection method, (2) since a simple image cannot capture the ideal target histogram, it will allow us to test the ability of color correction.

We select the R, G and B as well as Y, Cr, Cb histograms of this image as target histograms. Good segmentation results can be obtained from Figure 2.4 (c) using the color range [139,202] for Cr and [66, 123] for Cb in YCrCb color space.

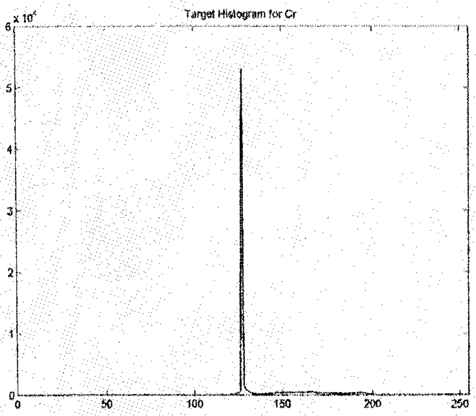
2.2.3.4 Color correction in YCrCb space

We use the face image in Figure 2.4 (e) as an example to describe the color correction process. Figure 2.8 (a) and (b) illustrate the target histograms obtained from the face image in Figure 2.4 (c). Figure 2.8 (c) and (d) are the Cr and Cb color histograms of the input image. Figure 2.8 (e) and (f) are the color histograms after equalization. Figure 2.8 (g) and (h) are the color histograms after histogram fitting.

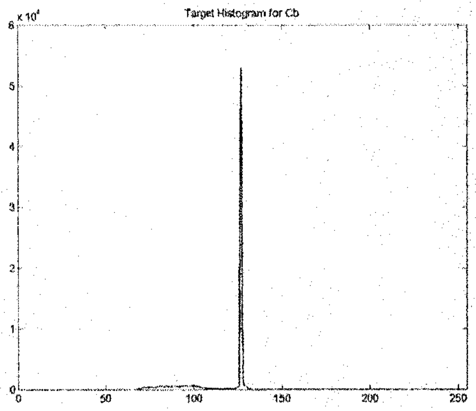
Figure 2.9 shows the face color segmentation results after color correction in the YCrCb color space. We can see how segmentation results are improved in Figure 2.9 (d), (e), (f), (g) and (h). Face regions are also segmented in Figure 2.9 (a), (b) and (c). The color correction method introduced here improves significantly the color segmentation quality by using general face color ranges in the YCrCb color space.

2.2.3.5 Color correction in RGB space

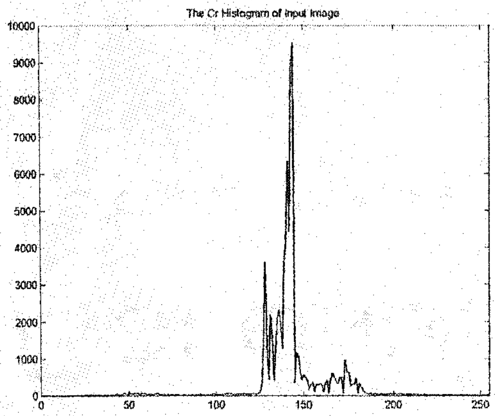
Figure 2.10 shows the color segmentation results after color correction in the RGB color space. In cases (a) and (c) only the segmentation of the face regions from the backgrounds was successful. This is because the color ranges used for color segmentation are in the YCrCb space and fitting the input color histograms into target histograms in the RGB color space cannot bring a useful correction in the YCrCb color space. The conclusion, therefore, is that the segmentation quality in the RGB space cannot be improved in our case.



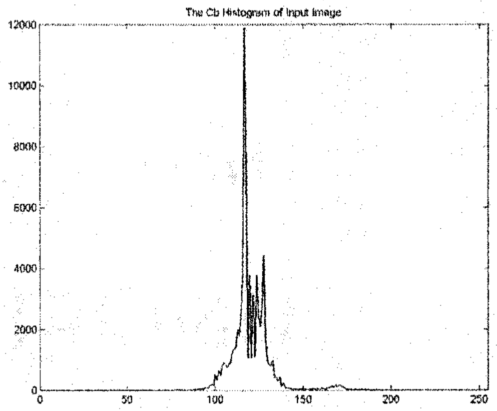
(a) The target histogram of Cr



(b) The target histogram of Cb

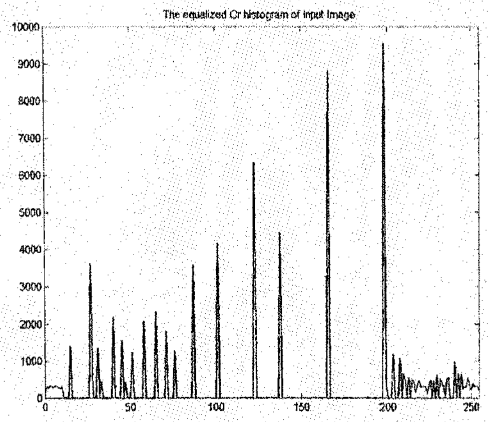


(c) The input histogram of Cr

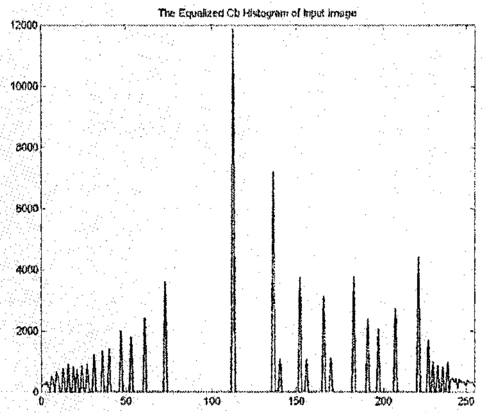


(d) The input histogram of Cb

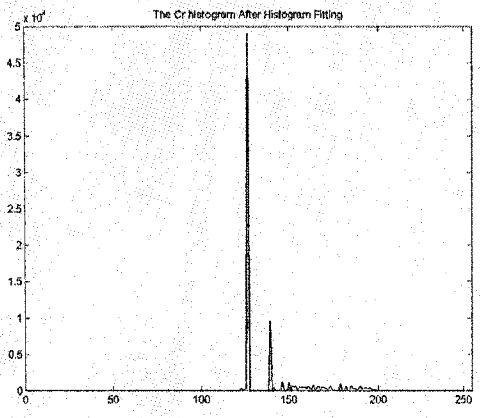
Figure 2.8 The color correction for a face image (continue)



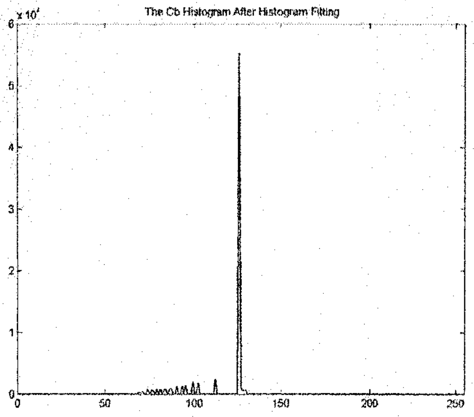
(e) The equalized histogram of Cr



(f) The equalized histogram of Cb

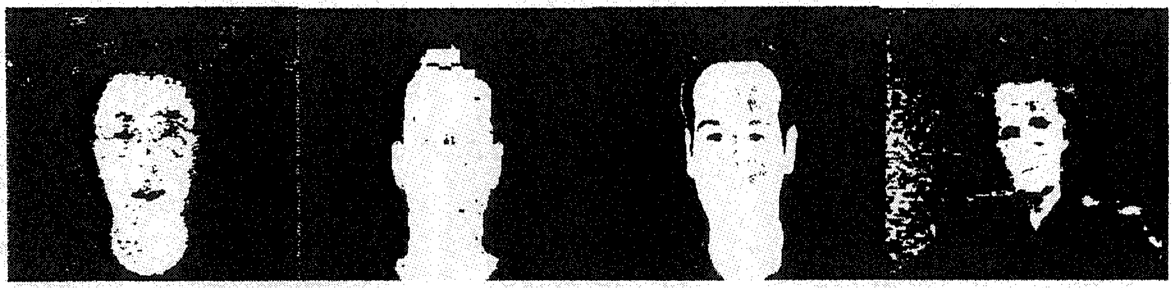


(g) The histogram of Cr after fitting

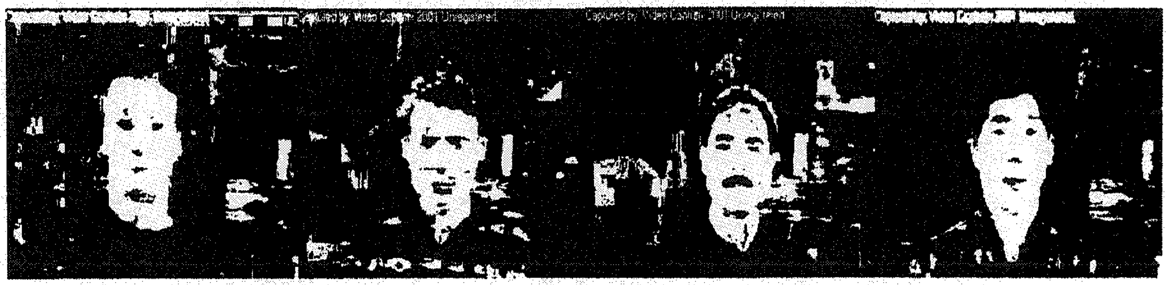


(h) The histogram of Cb after fitting

Figure 2.8 The color correction for a face image



(a) (b) (c) (d)

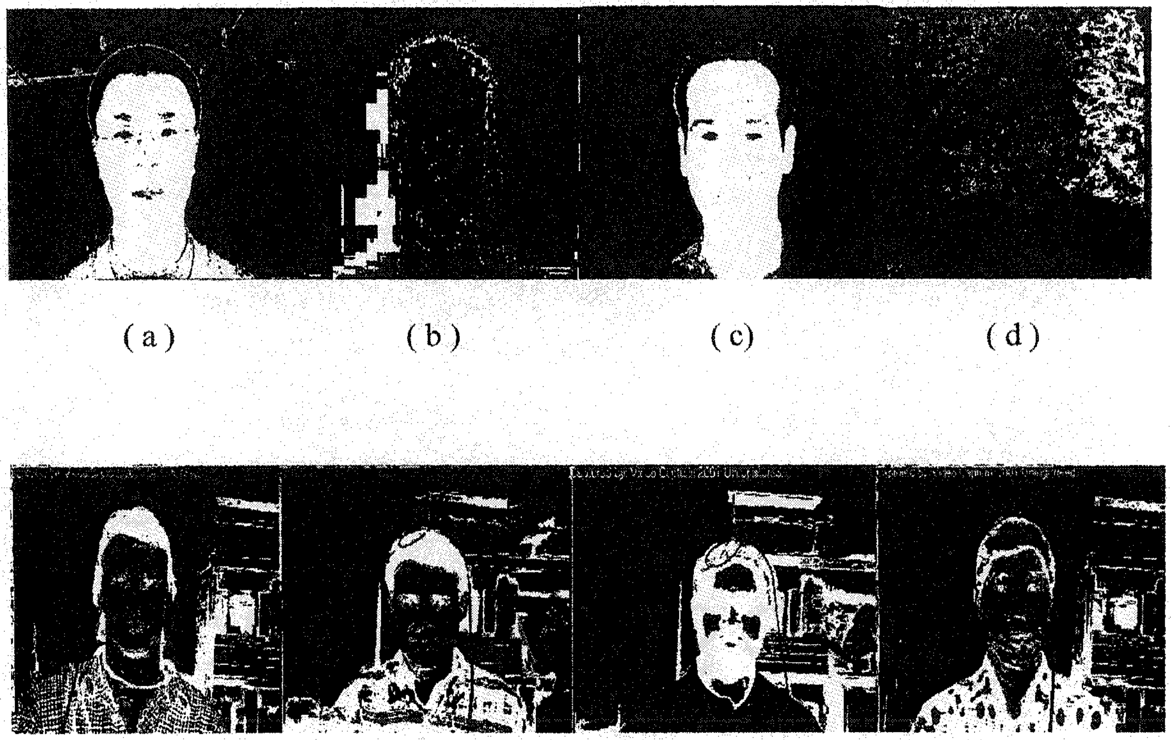


(e) (f) (g) (h)

Figure 2.9 Face color segmentation after color correction in the YCrCb space

2.2.4 Density regularisation

This stage considers that the original face map produced by previous stage contains the skin-color regions in the background and the non skin-color regions within the face. The purpose of density regularisation is to eliminate these regions in both the background and the face area.


(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2.10 Face color segmentation after color correction in the RGB color space

We first separate the image into 8×8 arrays.Then caculate how many face color pixels in each array and use the number of face color pixels in that array to represent the density value of that array. As shown in figure 2.11, some pixels in the 8×8 matrix are skin-color (black squares), others are non skin-color (white squares).

We do not write the density values onto the density map directly. The density values larger than a threshold will be set to 1 in the density map.The density values less than that threshold will be set to 0 in the density map. After that in density map, 1 represent a more likely skin-color region. 0 represent the more likely non skin-color region. How to define the threshold is closely related to the image size, the array size and image statistical characters. The values in the density map represent if the 8×8 arrays in the related positions in the face image are likely belong to face region.

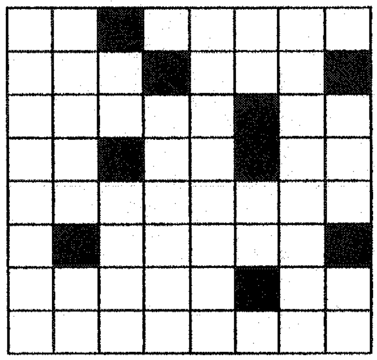


Figure 2.11 Density calculation for 8*8 array (density=9)

(black square express a skin-color pixel, white square express a non skin-color pixel)

2.2.5 Geometric correction

Procedure of geometric correction is introduced in the following:

Step 1: Discard all points at the edge of the density map. From the assumption, the skin-color pixels at the edges of the face image are impossible to be in the face region.

Step 2: Perform the dilation and erosion on the density map. A point in the density map with value 1 will not be changed if it is surrounded by more than three other points with the same value. A point with value 0 will be set to 1 if it is surrounded by more than five points with value 1.

Step 3: Commence the horizontal scan on the density map. We search for any short continous run of points that are value one. Any group of points with less than 8 horizontially connected points with the value one will be set to zero.Then we do the similar scan in the vertical direction. Any group of points with less than 8 vertically connected points with the value one will be set to zero.

2.2.6 Face elliptic shape correction

2.2.6.1 Moments

In [Reinders et al, 1995], the concept of moments for shape analysis is introduced. Several properties derived from moments are found to be useful to shape analysis. The set of moments of a bounded function $f(x,y)$ of two variables is defined by

$$M_{jk} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^j y^k f(x,y)dxdy \qquad (2-4)$$

where j and k take on all nonnegative integer values. As j and k take on all nonnegative integer values, they generate an infinite set of moments. Furthermore, this set is sufficient to specify the function $f(x,y)$ completely.

For shape-descriptive purposes, suppose $f(x,y)$ takes on the value 1 inside the object and 0 elsewhere. This silhouette function reflects only the shape of the object and ignores internal gray-level details. Every unique shape corresponds to a unique silhouette and, furthermore, to a set of moments.

The parameter j+k is called the order of the moment. There is only one zero-order moment,

$$M_{00} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)dxdy \qquad (2-5)$$

and it is clearly the area of the object. There are two first-order moments and correspondingly moments of higher orders.

The coordinates of the center of gravity of an object are

$$x_0 = \frac{M_{10}}{M_{00}} \qquad y_0 = \frac{M_{01}}{M_{00}} \qquad (2-6)$$

The central moments are computed using the center of gravity as the origin:

$$\rho_{jk} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_0)^j (y-y_0)^k f(x,y)dxdy \qquad (2-7)$$

The central moments are position invariant.

2.2.6.2 Face shape

The oval shape of a face can be approximated by an ellipse. Therefore, the detection of the facial region in the image can be performed by detecting an object of elliptic shape. We model the face-like region by an ellipse using moment-based features (Sobottka and Pitas, 1997). Let us denote the face-like area by C and best-fit ellipse by E. An ellipse is defined by its center (x_0, y_0), its orientation θ and the length a and b of its semi-major and semi-minor axes, as shown in figure 2.12.

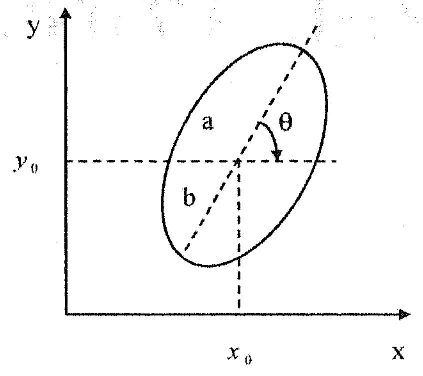


Figure 2.12 Parameters of an ellipse

The center of the ellipse is estimated by the center of gravity of the region C. The orientation of the ellipse is computed by determining the angle between the axis of the least moment of inertia and the horizontal axis of the coordination system, *i.e.*,

$$\theta = \frac{1}{2}\arctan\left(\frac{2\rho_{1,1}}{\rho_{2,0} - \rho_{0,2}}\right) \qquad (2-8)$$

where $\rho_{i,j}$ denotes the (i, j) central moment of the region C. The length of the semi-major a and the length of the semi-minor axis b can be calculated by evaluating the least and the greatest moment of the inertia, respectively. The least and the greatest moment of the inertia of an ellipse with orientation θ, $I_{\min}, I_{\max}$, are given by

$$I_{\min} = \sum_{(x,y)\in C}[(y-y_0)\cos\theta-(x-x_0)\sin\theta]^2$$
$$I_{\min} = \sum_{(x,y)\in C}[(y-y_0)\sin\theta-(x-x_0)\cos\theta]^2 \qquad (2-9)$$

where x, y denote the horizontal and vertical coordinate of a pixel. Accordingly, a and b are given by

$$a=(\frac{1}{\pi})^{\frac{1}{4}}[\frac{(I_{\max})^3}{I_{\min}}]^{1/8} \qquad b=(\frac{1}{\pi})^{\frac{1}{4}}[\frac{(I_{\min})^3}{I_{\max}}]^{1/8} \qquad (2-10)$$

respectively.

2.2.6.3 Face shape correction

To perform the shape correction, we have to discard the image background. We assume that we have already separated the face from the background. We use 0 to represent a pixel in the background, and 1 to represent a pixel in the face region.

To find the ellipse that models the given region best, we iteratively maximize the measure

$$\Delta M = \sum_{(x,y)\in E\cap C} 1 - \sum_{(x,y)\in E\cap C^c} 1 \qquad (2-11)$$

where C^c denotes the complement of the region of the region C (*i.e.* the background). The maximization of equation above corresponds to the maximization of the number of correctly modeled pixels (i.e. $(x,y) \in E \cap C$) and the minimization of the number of incorrectly modeled pixels (i.e. $(x,y) \in E \cap C^c$). In order to find the a maximum of ΔM the following recursive procedure is proposed:

Step 1: Calculate the parameters of the initial ellipse E_0 and the measure ΔM_0 for the initial region C_0.

Step 2: Find the new region $C_i = C_{i-1} \cap E_{i-1}$.

Step 3: Calculate the parameters of the ellipse E_i that fits C_i and the measure ΔM_i.

Step 4: If $\Delta M_i > \Delta M_{i-1}$ go to the step 2. Otherwise, the best ellipse is E_{i-1} that has already been found.

2.2.7 Contour extraction

The procedure of contour extraction is introduced in the following:

Step 1: Convert the density map to the face map. Because each point in the density map represent a 8×8 array in the image or face map, all the values except the edge points in the related array in the face map will be set to one if the value in the density is one. Similarly, all the values except the edge points in the related array in the face map will be set to zero if the value in the density is zero.

Step 2: In order to reduce the quantization effects in the face map, all the values in the face map will not be changed if the related points in the density map are edge points. So the face map we produced can keep all the contour information of a face.

Step 3: The resulted face maps are possible to have holes or noises in the face region because of the non skin-color facial features. So we need to fill these holes. The method of hole filling is simple. We assume the face shape we obtained is convex. So do the horizontal and vertical scan, the pixels with value zero that locate between pixels with value one are set to one.

Step 4: The parameters of obtained ellipse in the density map are changed to the image coordinate system. We use a, b, x_0, y_0 and θ to represent the semi-major, semi-minor, x coordinate and y coordinate of the center of the ellipse and the angle of the ellipse respectively in the image. We use a', b', x_0', y_0' and θ' to represent the semi-major, semi-minor, x coordinate and y coordinate of the center of the ellipse and the angle of the ellipse respectively in the density map. We use W_b and H_b to represent the block width and block height when we generate density map. Here $W_b = H_b$. The equations to change these parameters of ellipse in density map to the parameters in image coordinate system are:

$$\begin{aligned}
&a = a' \times W_b / \cos(\theta) \\
&b = b' \times W_b / \cos(PI/2 - \theta) \\
&x_0 = (x_0' + 1) \times W_b - 1 \qquad\qquad (2-12) \\
&y_0 = (y_0' + 1) \times H_b - 1 \\
&\theta = \theta'
\end{aligned}$$

In our system, we use a, b, x_0, y_0 and θ to represent the face length, face width, x coordinate and y coordinate of the face center and the face angle respectively. The equations are:

$$\begin{aligned}
&W_F = 2*b \\
&H_F = 2*a \\
&\theta_F = \theta \qquad\qquad (2-13) \\
&F_{x0} = x_0 \\
&F_{y0} = y_0
\end{aligned}$$

where W_F , H_F, θ_F, F_{x0}, F_{y0} are face width, face length, face angle, x coordinate and y coordinate of the face center respectively.

All the pixels ouside the ellipse will be considered as the non-face pixels. The face map obtained here is processed face map.

2.3 Results

Three examples of face images are shown in Figure 2.13 (a). All the images are taken in normal conditions and no special light sources are used. Figure 2.13 (b) shows the results after color segmentation. Figure 2.13 (c) shows the density map after density regularization. The density map after geometric correction is shown in figure 2.13 (d). The results of elliptic shape correction are shown in Figure 2.13 (e). The results of contour extraction are shown in figure 2.13 (f).

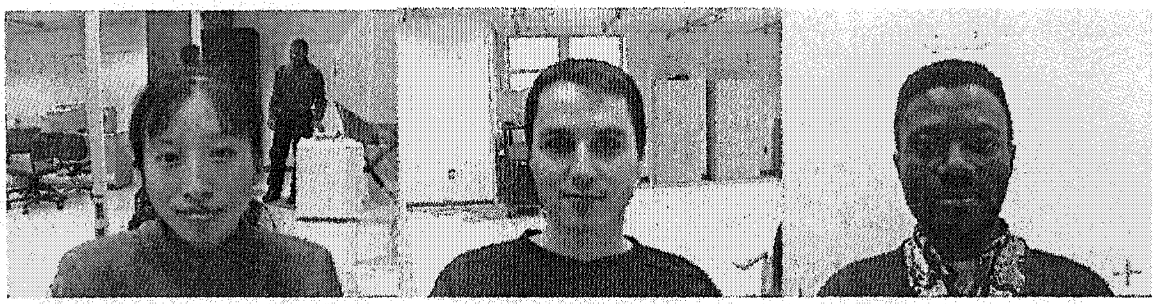
The additional two examples of face detection and facial feature localization are shown in appendix I and appendix II.

We use the AR face database (A.M. Martinez and R. Benavente, ``The AR face database", CVC Tech. Report #24, 1998) to test the proposed face detection algorithm.

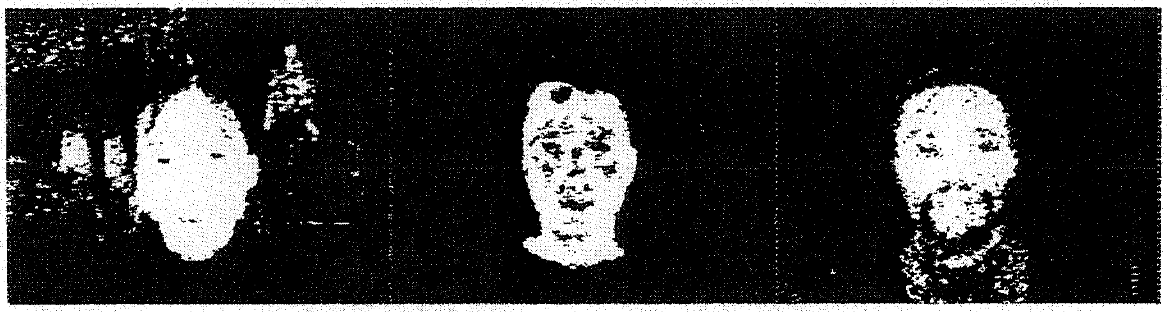
The AR face database was created by Aleix Martinez and Robert Benavente in the Computer Vision Center (CVC) of the Universitat Autonoma de Barcelona. It contains over 4,000 color images corresponding to 126 people's faces (70 men and 56 women). Images include frontal views of faces with different facial expressions, illumination conditions, and occlusions (sunglasses and scarf).

In the AR face database, each person has more than ten face images at different times, with different expression and different clothes. In our experiment, we used only the first image of each person. Therefore, there are 133 face images totally, with the exception of one poor face image. These images and their IDs are shown in Appendix III.

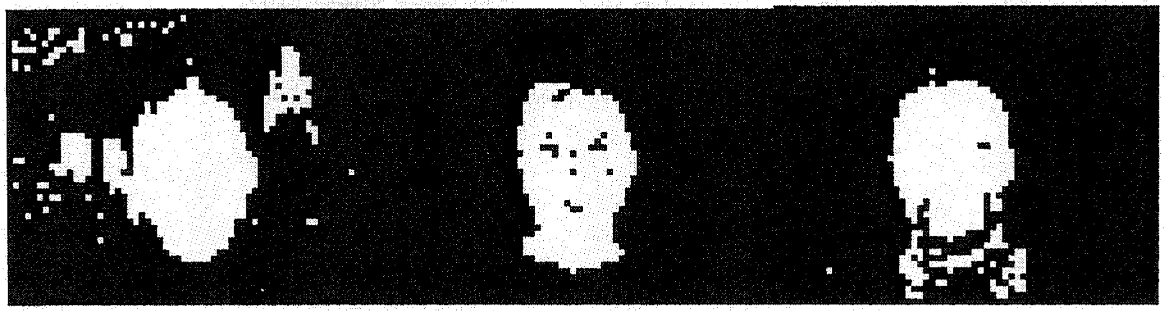
As introduced in Section 2.2.3.3, we use color range Cr of [139,202] and Cb of [66,123] for face color segmentation. By using the proposed algorithm, face regions are segmented successfully in 127 face images. The success rate is 96.21%. Face regions are not segmented completely in 6 images because of the moustaches and beards on the face regions or skin-color hair around the face.



(a)

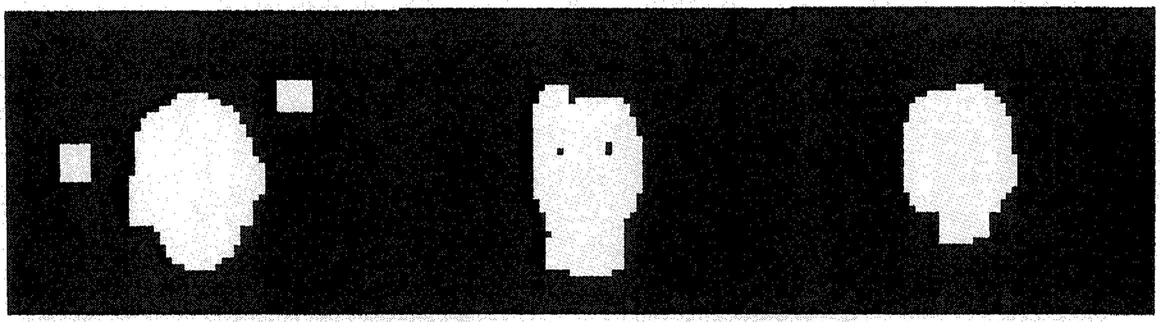


(b)

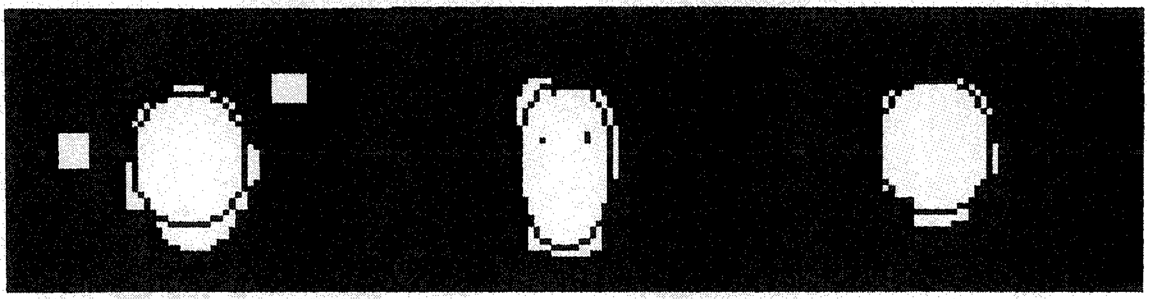


(c)

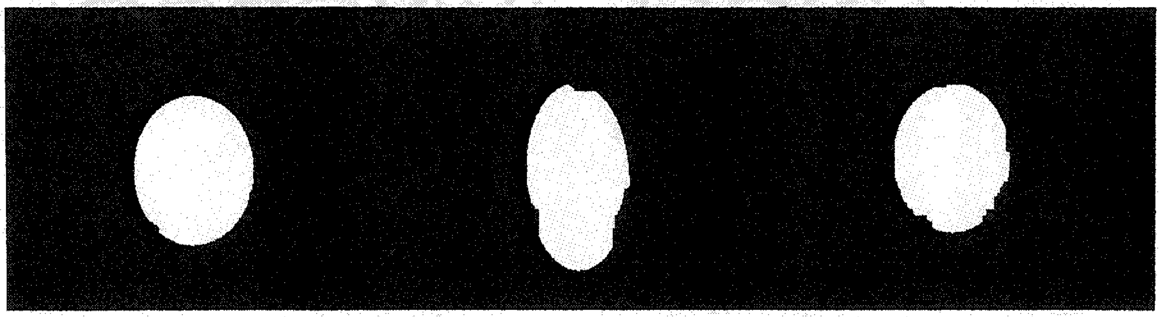
Figure 2.13 Results of face detection (continue)



(d)



(e)



(f)

Figure 2.13 Results of face detection

(a) face image (b) results after color segmentation (c) results after density regularization (d) results after geometric correction (e) results after ellipse shape correction (f) results after contour extraction

CHAPTER III

FACIAL FEATURE LOCALIZATION

In order to adapt a generic face model to the actual features of the face in the scene, the positions of facial features are needed. Based upon the knowledge obtained from the face detection component, we can localize the facial features, such as eyes and mouth, in the face regions. In our project, we just need to know the locations of the centers of the eyes and mouth. Details such as sizes and states of the eyes and mouth are not required.

3.1 Introduction of proposed facial feature localization algorithm

Figure 3.1 illustrates the various modules of the proposed facial feature localization algorithm. Details will be introduced in Section 3.2.

3.1.1 Using color efficiently

Almost all the work reported in the literature related to color-based face detection using color only concern the separation of the face region from the background. No further work has been done to use color information in the facial feature localization. Although some papers (Wu et al, 1999;Chai and Ngan, 1999; Garcia and Tziritas, 1999; Terrillon et al, 1999; Schwerdt and Crowley, 2000) introduce some facial feature localization algorithms, they do not use the color information directly after facial feature localization.

In our algorithm, facial feature localization is based upon the assumption that some parts of facial features have different colors or lower luminance values compared to face regions.

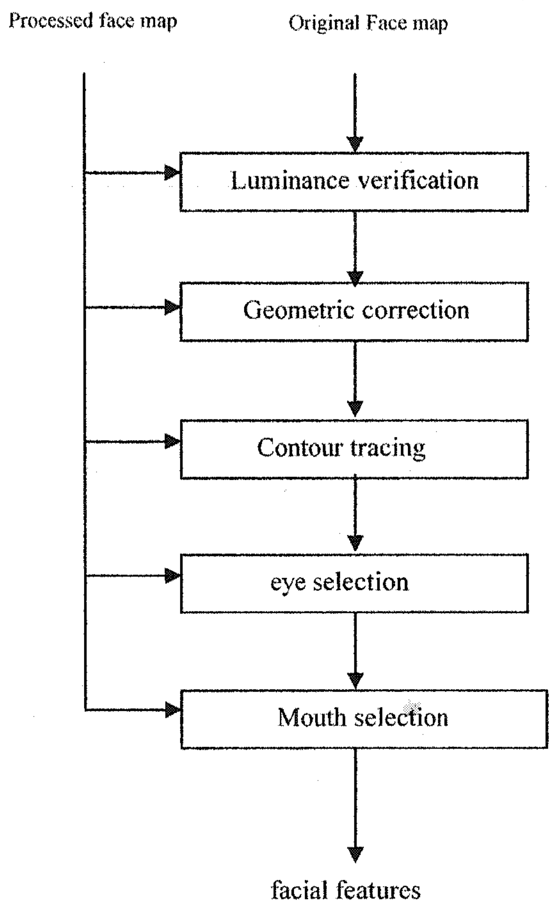


Figure 3.1 Modules in the proposed facial feature localization algorithm

3.1.2 Simple knowledge-based selection mechanism

In order to select candidate regions corresponding to real facial features, the knowledge-based selection mechanism (Reinders et al, 1995) is used. In the knowledge-based selection method, all measurements of candidate regions of facial feature are approximated as normal distribution. The mean and the standard deviation of normal distribution are determined by the ideal face proportions. The measurements we used in the selection can be interpreted as the likelihood that the region corresponds to the facial features. The measurements are simple and easy to calculate. Multiple measurements used in the selection can reduce the localization errors. Then, the percentile ranks from the z-score are used to represent the particular measurement values. All the facial feature selections are based upon these percentile ranks.

3.2 Proposed facial feature localization algorithm

3.2.1 Luminance verification

Compared to other areas of the face, regions corresponding to facial features have different colors. Therefore, after face color segmentation, some regions of facial features appear as non skin-color holes in the original face map, as shown in Figures 3.2 (b), (e), (h). In Figure 3.2(b), the eyes, mouth and even eyebrows are clearly distinct from the face region. In Figure 3.2(e), however, we cannot extract eyes directly from the original face map.

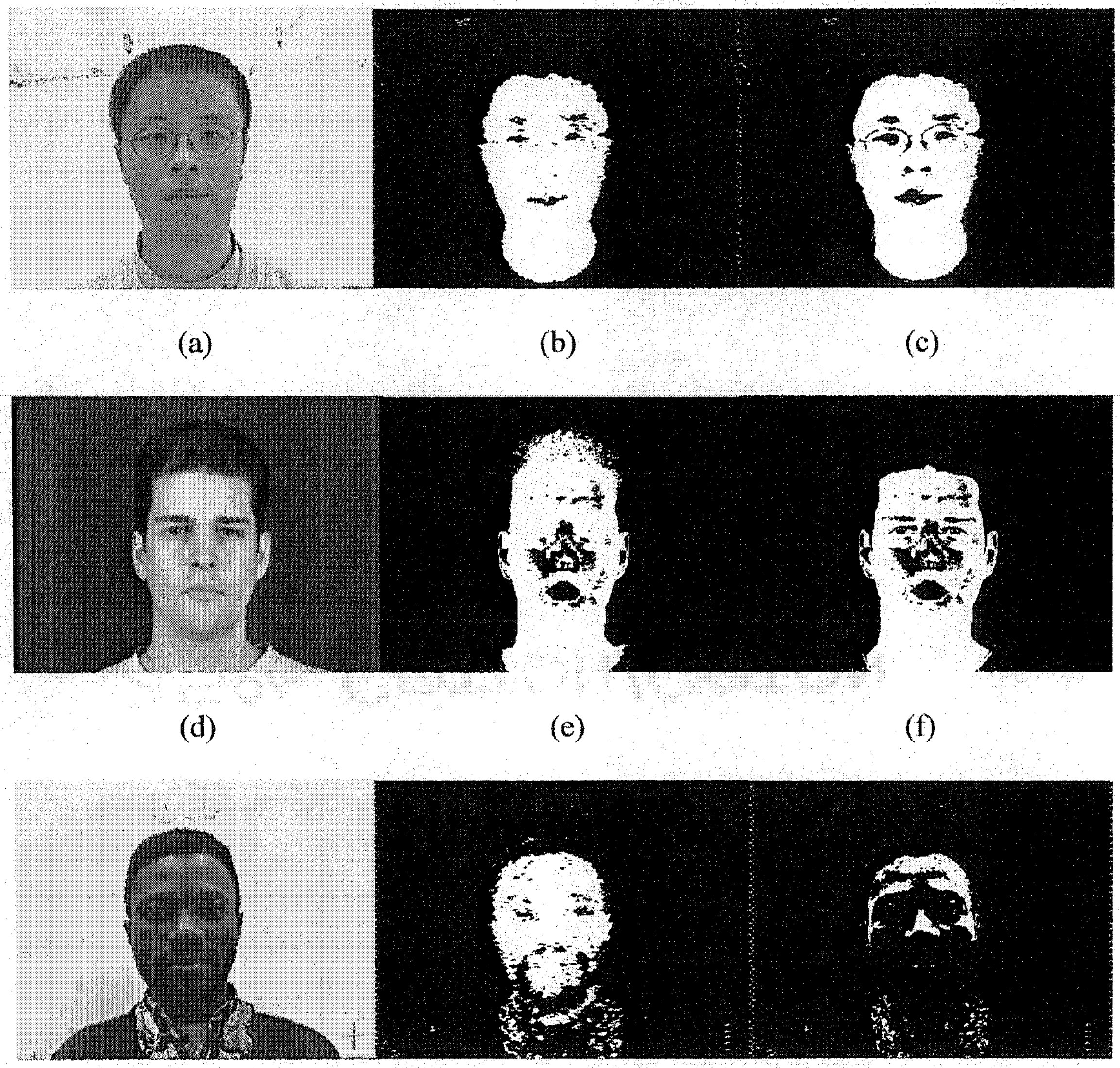


Figure 3.2 Face images and face map. (a), (d), and (g): Face images. (b), (e) and (h): Face map after face color segmentation, no luminance verification used. (c), (f) and (i) Face map after face color segmentation, with luminance threshold Y=80.

Although some regions of the facial features are classified to skin-color regions in the color segmentation, they have lower luminance than other regions of the face. By setting a luminance threshold appropriately, we can better select the candidate regions of facial features. Any pixel with a luminance value lower than this threshold is classified as a pixel of a facial feature region. We call this operation luminance verification.

Figures 3.2(c), (f) and (i) show the results by using a randomly selected threshold of 80. Now Figures 3.2(c) and (f) have more details about the facial features than Figure 3.2(b) and (e), respectively. But in Figure 3.2(i), the result becomes worse if we use the luminance threshold. This means that using luminance threshold is not appropriate in this situation. We need an automatic method of determining if luminance verification is needed, as well as a method of determining an appropriate luminance threshold.

In order to determine the luminance threshold, we need to consider two difficulties:

- Different type of skin colors may have different luminance values in the face images.
- Similar skin colors also have different luminance values in different lighting environments.

The suggested method to determine the luminance threshold is to assume that facial feature regions contain a certain percentage of the total face area. As long as the total area of the candidate facial feature regions is less than the set percentage of the total face area, we may then assume that the candidate regions need to be increased. Otherwise, we consider that candidate regions are sufficient.

In the implementation, we calculate the luminance histogram in the face region first, then use the luminance histogram to calculate the threshold that can segment a certain percentage of face area to facial feature candidate regions by using both the original face map and the processed face map. If the facial feature regions already contain more than a certain percentage of total face area, this means that we have enough information for the facial features and do not need to perform the luminance verification.

In the experiments, we see that the percentage of 1/8 leads to good results in luminance verification. Small changes of this percentage have no obvious effects on the results of luminance verification.

3.2.2 Geometric correction

This stage does not work on the processed face map, but on the original face map after the face luminance verification stage. The purpose of this stage is to eliminate the noises in the face region and simplify the facial feature selection. Candidate regions of facial features are reduced after geometric correction. The procedure of geometric correction is described below:

First, we perform the erosion and dilation on the original face map. All the pixels belonging to the candidate regions with less than 3 neighbors in the candidate regions are set to non candidate region pixels. All the pixels belonging to the non candidate region pixels with more than 5 neighbors in the candidate regions are set to candidate region pixels.

Second, we perform the horizontal and vertical scan on the original face map respectively. We search for any short continous run of pixels that belong to the candidate region. Any group of pixels that have two or fewer horizontally or verticallly connected pixels in the candidate regions will be set to non candidate region pixels.

3.2.3 Contour tracing

Here we use the Moore-neighborhood contour tracing algorithm to find all the contours for the candidate facial feature regions in the face region in the original face map. With the Moore-neighborhood algorithm, when the current pixel p is black, the Moore-neighborhood of p is examined in a clockwise manner starting with the pixel from which p was entered and advancing pixel by pixel until a new black pixel in P is encountered, as illustrated in Figure 3.3.

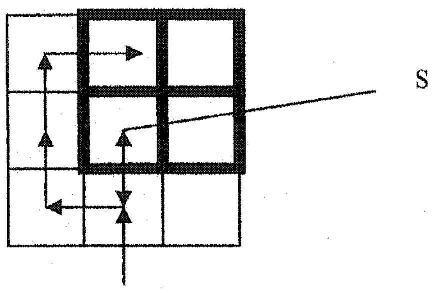


Figure 3.3 Searching for the next black contour pixel in Moore-neighborhood algorithm

The algorithm is described more precisely below:

Input: A square tessellation, **T,** containing a connected component, **P,** of black cells.

Output: A sequence, **B**(b1,b2,...,bk), of boundary pixels.

Begin: {initialization: find a pixel in P, initialize B, define M(p) to be the Moore-neighborhood of the current pixel p}

Step 1: Set **B** to be empty.

Step 2: From bottom to top and left to right scan the cells of **T** until a pixel, s, of **P** is found (until the current pixel p=s is a black pixel of **P**), insert s in **B**. Let b denote the current boundary point, *i.e.*, b=s.

```
            Step 3: Backtrack (move the current pixel to the pixel from which s was
            entered) and advance the current pixel, p, being examined to the next
            clockwise pixel in M(b).

        While p is not equal to s do

            If p is black, insert p in B, set b=p and backtrack (move the current pixel
               to the pixel from which p=b was entered);
            else advance the current pixel, p, to the next clockwise pixel in M(b).

    end while
end
```

3.2.4 Facial feature selection

3.2.4.1 Knowledge-based selection method

Knowledge-based selection aims at selecting the most probable regions corresponding to objects, which are facial features in our case, from a set of candidate regions. We use F to represent facial features. Measurements, such as level of symmetry, areas of candidate regions *etc.*, can be used in selection. We are interested in the probability that region r_i represent F, given the set of measurements $\vec{m}_j$ made on all regions:

$$P(\text{region } r_i \text{ represents } F \mid \bigcup_{j=1}^{N_r} \vec{m}_j) \qquad (3.1)$$

where N_r is the number of candidate regions and $\vec{m}_j$ is the set of measurements made on region r_i. The region, which is most supported by the evidence, given by all measurements $\vec{m}_j$, resembles facial feature F most. Thus, the region which maximizes the above equation is selected to represent facial feature F.

The next step would be to break this expression up into simpler terms, involving only probabilities conditioned on an individual set of measurements $\vec{m}_i$, $P(F \mid \vec{m}_i)$, because these are the only distributions which can be determined *a priori*. However, factorizing the above equation into such terms requires independence of the measures $\vec{m}_i$. This independence cannot be guaranteed because certain regions may belong to the same facial feature or to facial features which have measurements in common. For example, the distance between the eyes is very similar to that between the eyebrows. Therefore, we are forced to adopt a suboptimal approach: we proceed by selecting the region possessing a maximum probability that it belongs to the facial feature, based upon its individual measurements only. In our system, this simplification does not seem to have any noticeable effects.

The selection process, then, reduces to the maximization of $P(F \mid \vec{m}_i)$ over all candidate regions r_i, or:

$$\max_{i=1,\ldots,N_r} (P(F \mid \vec{m}_i)) \tag{3.2}$$

Applying Bayes rule to the above gives:

$$\max_{i=1,\ldots,N_r} [\frac{P(\vec{m}_i \mid F)P(F)}{P(\vec{m}_i)}] \tag{3.3}$$

We would like to point out that the maximization takes place over all measurement vectors, rather than over the different facial features as is the case in classical pattern recognition, where one tries to classify a region into one of the possible classes, given its measured property vector. Consequently, $P(F)$ is now constant over the maximization while $P(\vec{m}_i)$ would have been held constant in a pattern recognition problem. The optimal decision rule, under the assumption that different measurements within each measurement set are independent, becomes:

$$\max_{i=1,\ldots,N_r} [\frac{P(\vec{m}_i \mid F)}{P(\vec{m}_i)}] = \max_{i=1,\ldots,N_r} [\frac{P(\vec{m}_{i,1} \mid F)}{P(\vec{m}_{i,1})} \times \cdots \times \frac{P(\vec{m}_{i,l} \mid F)}{P(\vec{m}_{i,l})} \times \cdots \times \frac{P(\vec{m}_{i,N_M} \mid F)}{P(\vec{m}_{i,N_M})}] \tag{3.4}$$

where the set of measurements $\vec{m}_i$ is split into its N_m different measurements.

Each ratio $P(m_{i,l} \mid F) / P(m_{i,l})$ above can be considered as a score representing the degree of fit to the object class F. In other words, the selection process suggests that region whose set of measurement values $\vec{m}_i$ are closest to the expected values based upon a *priori* knowledge about the facial features has high scores.

Although there are no fixed rules for property selection, it is desirable that they have high discriminative power, whereby the ratio $P(m_{i,l} \mid F)/P(m_{i,l})$ peaks for the sought facial feature region. Furthermore, to apply Equation (3.4), these properties should be independent. In experiments the set of properties are chosen heuristically, and in the absence of further knowledge, we assume that the measured property values have uniform prior distributions. Further, their posterior distributions are assumed to be independent and Gaussian with means μ and variances σ, which we can estimate from training or ideal situations. Details are introduced in next two sections.

3.2.4.2 Normal distribution approximation

The distribution of normal distributions can be described in two parameters: mean (μ) and standard deviation (σ). The standard normal distribution is a normal distribution with a mean of 0 and a standard deviation of 1.

The standard normal distribution is sometimes called the z-distribution. A z-value always reflects the number of standard deviations above or below the mean of a particular value. For instance, if a person scored a 70 on a test with a mean of 50 and a standard deviation of 10, then he or she scored 2 standard deviations above the mean. Converting the test scores to z scores, an X of 70 would be:

$$Z = \frac{70-50}{10} = 2 \qquad (3-5)$$

Thus, a z-value of 2 means the original score is 2 standard deviations above the mean. Note that the z-distribution will only be a normal distribution if the original distribution (X) is normal.

If the z-value is known, it is relatively easy to figure out the probability of a variable obtaining a specific value. To calculate the probability, we just calculate the area below the curve and in the range of $[-\infty, -|Z|]$ and $[|Z|, \infty]$.

A Z table with 0.01 precision is shown in Table 3.1.

Table 3.1 z-table

Z value	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Probability	1	0.92	0.84	0.76	0.69	0.62	0.55	0.48	0.42	0.37
Z value	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
Probability	0.31	0.27	0.23	0.19	0.16	0.13	0.11	0.09	0.07	0.06
Z value	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9
Probability	0.05	0.04	0.03	0.02	0.02	0.01	0.01	0.01	0.01	0.00
Z value	3.0	Above 3								
Probability	0.00	0								

We can notice that when the z-value is 0, the probability value is 1. And when the z-value is larger than 3, the probability will be 0. We can use these properties when we use the normal distribution to approximate the face measurements.

3.2.4.3 Ideal face proportions

After the face detection, we have a processed face map that can tell us whether a pixel in the image belongs to the face area or not. The face height, face width, face center and face angle are also obtained in the elliptic shape correction stage. We can use these face parameters and the ideal face proportions to select the measurements, which can be used in the knowledge-based selection for facial features.

We assume the face width is four units. In the ideal face proportions, for example, the distance between two eye centers is two units, the vertical distance between eye centers and mouth center is two units. An ideal face proportion map is shown in Figure 3.4.

In practice, we cannot guarantee that the face is always horizontal and must address this problem in the implementation. A face with an angle is shown in Figure 3.5.

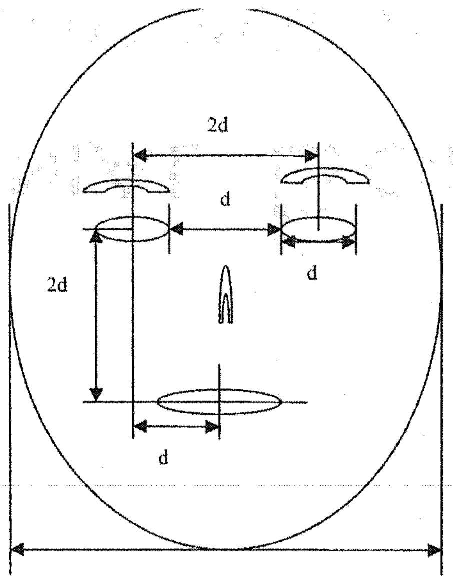


Figure 3.4 Ideal face proportions

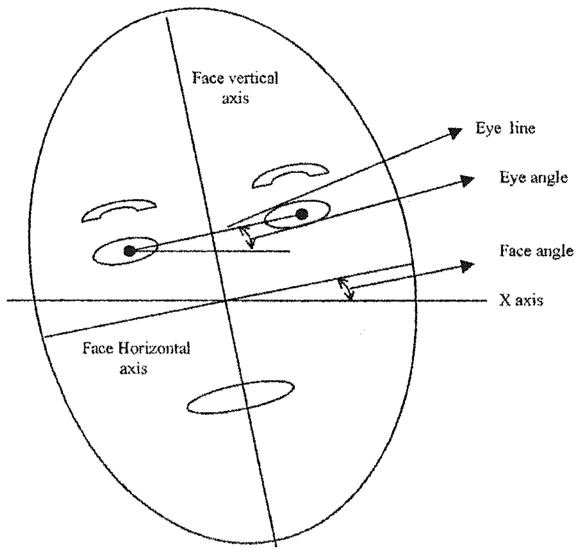


Figure 3.5 Face with an angle

3.2.4.4 Measurement selection and normal distribution approximation

3.2.4.4.1 Eye selection

In the knowledge-based method, the following measurements are used to search for the eyes in the eye candidate regions:

Measurement 1: Distance between the centroids of candidate eye pair

We call the distance between two eye centers the eye distance. Figure 3.4 shows that, in ideal face proportions, the eye distance is about half face width. Thus, if the distance between the two centroids of a candidate eye pair is close to half face width, this candidate eye pair has a large probability of being an eye pair.

We denotes D_e and W_F the distance between the centroids of a candidate eye pair and the face width, respectively. The distance between the eyes is $1/2 \times W_F$ ideally. Consequently, we define the mean of normal distribution as $\mu = W_F$. Considering that the eye width is $1/4 \times W_F$ ideally, the largest distance between the eyes is $3/4 \times W_F$, and the shortest distance between the eyes is $1/4 \times W_F$. We must set the probability of shortest distance and largest distance to be zero when we define the standard deviation σ. It is easy to see from Section 3.2.4.2 that $\sigma = 1/3 \times 1/4 \times W_F = 1/12 \times W_F$ can fit this condition according to the z value introduced above. The z-value of distance between the centroids of candidate eye pair, therefore, is:

$$Z_d = \frac{\left| D_e - \frac{1}{2} \times \mathrm{W_F} \right|}{\frac{1}{12} \times W_F} \qquad (3-6)$$

After we have Z_d, we can use the z-table to find the probability $P(eye\ pair \mid D_e)$.

Measurement 2: Eye angle

We call the line linking two eye centers the eye line. The eye angle is defined as the angle between the eye line and the x-axis. Normally, the eye angle should be equal to the face angle, as shown in Figure 3.5. Therefore, if the candidate eye pair has an eye angle that is close to the face angle, it has a large probability of being an eye pair.

We use θ_E and θ_F to represent the eye angle of a candidate eye pair and the face angle respectively. Because the candidate pair whose eye angle has a value similar to the face angle has a large probability of being the eye pair, we define the mean of normal distribution as $\mu = \theta_F$. If the difference between the eye angle of candidate eye pair and face angle is +45 degrees or −45 degrees, then there is no chance that this candidate eye pair could be a true pair of human eyes. In normal distribution, as a result, the probability of being the true eyes in this situation is zero. We select $\sigma=1/3\times45=15$ degree to be the standard deviation. The z-value of the eye angle is thus represented by:

$$Z_e=\frac{|\theta_E-\theta_F|}{15} \qquad (3-7)$$

We can use Z_e to find the probability $P(eye\ pair\,|\,\theta_E)$ from the z-table.

Measurement 3: Symmetry

We call the line passing through the middle point between the two eye centers and perpendicular to the eye line the eye vertical. In an ideal situation, the eye vertical is identical to the face vertical axis. In practice, a difference between the eye vertical and the face vertical axis exists. As well, the face angle we obtained in ellipse shape correction is not accurate, compared to the eye angle. Thus, we can use the eye angle to represent the face angle. Here, the symmetry will be represented by the similarity of areas between the left side and the right side of the eye vertical. The candidate eye pair that has high symmetry has a large probability of being an eye pair.

The symmetry of a candidate eye pair is represented by the difference between face areas at two sides of the eye vertical. Normal distribution is therefore used to approximate the difference of areas. We use N_l, N_r and N_d to represent the number of pixels on the left side and on the right side of the eye vertical, and the difference of these two areas, respectively. Thus $N_d=|N_l-N_r|$. According to the ideal face symmetry property,

N_d should be zero. We define the mean of normal distribution as μ=0. We assume that when the N_d is larger than ¼ of the face area, the symmetry of the candidate eye pair is too poor to be an eye pair. Thereby we define σ=1/3×1/4×face area=1/12×face area. The Z value of symmetry is represented by:

$$Z_s = \frac{|N_d|}{\frac{1}{12} \times face\ area} \qquad (3-8)$$

We can use Z_s to find the probability $P(eye\ pair \mid N_d)$ from the z-table.

i. **Measurement 4: Region areas**

Among all the candidate eye regions, the two eye regions have a large probability of having the largest region area in the upper half face. The larger area of a candidate eye pair, the larger probability that that candidate eye pair is the eye pair. Thus we can avoid selecting the small noise regions around the real eye regions.

When we use normal distribution to approximate the probabilities of the candidate eye pairs of being the eye pair, we should give the candidate regions with large areas a high probability and those with small areas a low probability. The difficulty here is we do not know the average eye region area. Naturally, then, we cannot use it to define the mean. In addition, the total area of the eyes is related to the luminance environment and the luminance threshold we used in luminance verification. The method we adopt here is to find the largest area $A_{\max}$ and the smallest area $A_{\min}$ of the candidate eye pair. We use A to represent a random area of a candidate eye pair. The candidate eye pair that has the largest area has the largest probability to be the eye pair. So we define $\mu = A_{\max}$. The candidate eye pair that has the smallest area has the lowest probability of being eye pair. Thus we define $\sigma = 1/3 \times (A_{\max} - A_{\min})$. The Z value of region area is represented by:

$$Z_a = \frac{|A - A_{\max}|}{\left(\frac{|A_{\max} - A_{\min}|}{3}\right)} \qquad (3-9)$$

We can use Z_a to find the probability $P(eye\ pair \mid N)$ from the z-table.

According to the knowledge-based selection method, the candidate eye pair that has

$$\begin{aligned}\max[P(eye\ pair \mid m_i)] = \max[&P(eye\ pair \mid D_e) \times P(eye\ pair \mid E_e) \\ &\times P(eye\ pair \mid N_d) \times P(eye\ pair \mid N)]\end{aligned} \qquad (3-10)$$

will be classified as the eye pair.

3.2.4.4.2 Mouth selection by luminance verification

After we have the location of the eyes, we can use them to locate the mouth. The measurements we used in the mouth localization are as follows:

Measurement 1: Distance between the centroids of candidate mouth regions and the face vertical axis

In ideal face proportions, the mouth is ventered on the face vertical axis. Therefore the closer the location of the centroid of the candidate mouth region to the face vertical axis, the larger probability that it is the actual mouth.

We use D_v to represent the distance between the centroids of candidate mouth region and the face vertical axis. The candidate mouth region that has D_v=0 has a high probability of being the mouth. Therefore, we set the mean of normal distribution at μ=0. From Figure 3.2, we see that the candidate mouth region that has D_v larger than ¼ face width has low probability of being the mouth. Consequently, we set the standard deviation σ=1/3×1/4×W_F=1/12×W_F. The Z value of D_v is represented by:

$$Z_v = \frac{|D_v|}{\frac{1}{12}\times W_F} \qquad (3-11)$$

We can use Z_v to find the probability $P(mouth \mid Z_v)$ from the z-table.

Measurement 2: Distance between the centroids of candidate mouth regions and eye line

In ideal face proportions, the distance between the mouth center and the eye line is half the face width. Therefore the closer this distance is to half face width, the larger the probability that it is the mouth.

We use D_h to represent the distance between the centroids of the candidate mouth region and the eye line. According to the ideal face proportion, the candidate mouth region that has D_h=1/2×W_F has a high probability of being the mouth. Thus we set the mean of normal distribution as μ=1/2×W_F. We assume that the candidate mouth region which has D_h larger than one face width and less than 0 has a low probability of being the mouth. Thus, we set the standard deviation σ=1/3×1/2×W_F=1/6×W_F. The Z value of D_h is represented by:

$$Z_h = \frac{\left| D_h - \frac{1}{2} \times W_F \right|}{\frac{1}{6} \times W_F} \qquad (3-12)$$

We can use Z_h to find the probability $P(mouth \mid Z_h)$ from the z-table.

i. Measurement 3: Region areas

Among all the candidate mouth regions, the mouth region has a large probability of having the largest area in the lower half face. Therefore, the larger the area of a candidate mouth region, the larger the probability that the candidate region is the mouth.

Similar to the eye selection, we use normal distribution to approximate the probabilities that the candidate mouth regions are the actual mouth regions. We should give the large area a high probability and the small area a low probability. We use $A_{\max}$ and $A_{\min}$ to represent the largest area and the smallest area respectively. A represents a random area of a candidate eye pair. The candidate mouth region that has the largest area has the largest probability of being the mouth. Thus we define $\mu = A_{\max}$. The candidate mouth region that has the smallest area has the lowest probability of being the eye pair. Thus we define $\sigma = 1/3 \times (A_{\max} - A_{\min})$. The Z value of region area is represented by:

$$Z_a = \frac{|A - A_{\max}|}{\left(\frac{|A_{\max} - A_{\min}|}{3}\right)} \qquad (3-13)$$

We can use Z_a to find the probability $P(mouth \mid N)$ from the z-table.

According to the knowledge-based selection method, the candidate eye pair that has

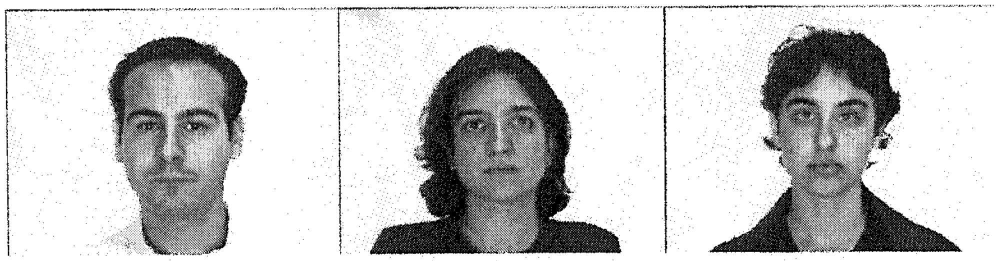
$$\max[P(eye\ pair \mid m_i)] = \max[P(mouth \mid D_v) \times P(mouth \mid E_h) \times P(mouth \mid N)] \qquad (3-14)$$

will be set as the mouth pair.

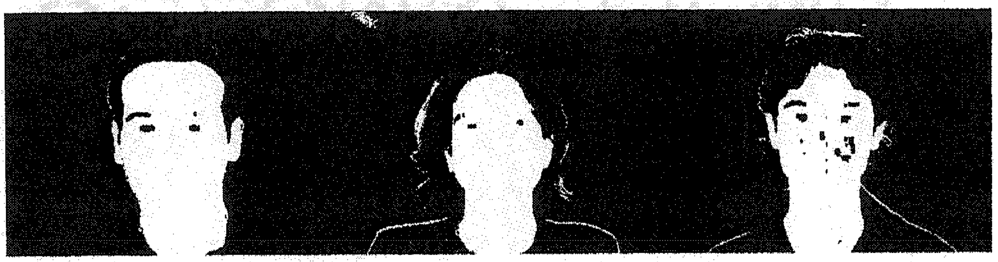
3.2.4.4.3 Mouth selection by corners

The facial localization methods introduced above can work well in the face images that were obtained in our lab and from some other sources. But we find that the methods can not be generally used in all the face images. In the tests of using the AR face database, the mouth is supposed to have non skin-color and lower luminance compared with other parts of the face. However this assumption is not always true in the AR face database. The success rate for mouth localization is low in these tests.

We can see from Figure 3.6, after color segmentation and luminance verification in the original face maps, there are no mouth candidate regions at all in (a) or (b), and only a few candidate mouth regions with small areas in (c).



(a) Face images



(b) Face maps

Figure 3.6 Failing examples for mouth localization using luminance verification

There are two reasons for the localization fails:

- Good luminance in the AR face database. The assumption of low luminance in the mouth area is not valid anymore given the good lighting condition.
- The differences between the mouth color and that of other parts of the face become small given the good lighting condition.

In order to solve this problem, we develop a mouth localization method using mouth corners. It is clear that the mouth region has more textures than other face regions do. There have high probabilities in finding high texture features at the corners of the mouth by using the feature selection method. The features we find in the lower face region are treated as the candidates of mouth corners. In order to find the real mouth corners, the knowledge-based selection method can be used among the candidates. Then we can use the center of the two corners to locate the mouth itself.

There are two advantages to using the mouth corners of the mouth to locate the mouth center:

- It is easier to find high texture features in different luminance situations.
- Compared with one mouth feature as reported in Section 3.2.4.4.2, two mouth features have more selection rules when we use knowledge-based selection method.

The high texture selection method used is reported by Shi and Tomasi (1994), and will be introduced in Chapter V.

After we obtain the high texture features in the lower half face, all the selected features are grouped to the possible pairs of candidate mouth corners. Then, a knowledge-based mouth corner selection method is used. The obtained location of eyes is also used in the mouth localization.

The measurements we selected and the way to use the normal distribution to approximate are introduced below:

Measurement 1: Distance between the middle of two candidate mouth corners and the face vertical axis

In the ideal face proportions, the center of the mouth is located at the face vertical axis. Therefore, the closer the location of the middle of the candidate mouth corner pair to the face vertical axis, the larger the probability that the candidate mouth corner pair is the actual mouth corner pair.

As reported before, we use normal distributions to approximate the probability of being the mouth. We use D_v to represent the distance between the middle of the candidate mouth corner pair and the face vertical axis. The candidate mouth corner pair that has

D_v=0 has a high probability to be mouth. Thus, we set the mean of normal distribution as μ=0. We assume the candidate mouth region that has D_v larger than ¼ of the face width has a low probability of being the mouth. Thus, we set the standard deviation $\sigma=1/3\times1/4\times W_F=1/12\times W_F$. The Z value of D_v is represented by:

$$Z_v=\frac{|D_v|}{\frac{1}{12}\times W_F} \qquad (3-15)$$

We can use Z_v to find the probability $P(mouthcorne\ rs \mid Z_v)$ from the z-table.

Measurement 2: Angle of the line linking two candidate mouth corners

In ideal face proportion, the two lines that link the two eyes and the two mouth corners are supposed to be parallel. We call the line linking the two eyes eye line, as introduced before. We call the line linking the two mouth corners mouth horizontal. Therefore, the closer the angle of the line linking the candidate mouth corner pair is to that of the eye line, the larger the probability that that candidate mouth corner pair is the actual mouth corner pair.

We use θ_E and θ_m to represent the eye angle and the mouth angle, the mouth angle being the angle between the line linking the candidate mouth corners and the x-axis. The candidate pair whose mouth angle has a value similar to the eye angle has the large probability of being the mouth corner pair. Thus, we define the mean of normal distribution as $\mu=\theta_E$. If the difference between the mouth angle and face angle is +45 degrees or –45 degrees, then there is no chance that this candidate mouth corner pair can be a real mouth corner pair. In normal distribution, the probability of being the actual mouth corners is zero in this situation. We select $\sigma=1/3\times45=15$ degree to be the standard deviation. Now the Z value of mouth angle is represented by:

$$Z_m = \frac{|\theta_m - \theta_E|}{15} \qquad (3-16)$$

We can use Z_m to find the probability $P(mouthcorners \mid E_m)$ from the z-table.

Measurement 3: Distance between two candidate mouth corners

In ideal face proportions, the distance between the two mouth corners is about half that of the eye distance. Therefore, the closer the distance between the pair of candidate mouth corners is to half of the eye distance, the larger the probability that the pair of candidate mouth corners is the actual mouth corner pair.

We use D_c to represent the distance between the two mouth corners. According to ideal face proportions, the candidate mouth corner pair that has distance $D_c=1/4\times W_F$ has a high probability of being the mouth corner pair. Consequently, we set the mean of normal distribution as $\mu=1/4\times W_F$. We assume that the candidate mouth region that has D_c larger than half of face-width and less than 0 has a low probability of being the mouth. Consequently, we set the standard deviation σ=1/3×1/4×W_F=1/12×W_F. The z-value of D_c is represented by:

$$Z_c = \frac{\left| D_c - \frac{1}{4} \times W_F \right|}{\frac{1}{12} \times W_F} \qquad (3-17)$$

We can use Z_c to find the probability $P(mouthcorners \mid Z_c)$ from the z-table.

Measurement 4: Distance between the middle of two candidate mouth corners and the eye line

In ideal face proportion, the distance between the mouth center and the eye line is about half of face width. Therefore, the closer the distance of the pair of candidate mouth corners is to the half of the face width, the larger the probability that it is the actual pair of mouth corners.

We use D_h to represent the distance between the middle of pair of candidate mouth corners and the eye line. According to ideal face proportion, the candidate mouth region that has D_h=1/2×W_F has a high probability of being the mouth. Thus, we set the mean of normal distribution as μ=1/2×W_F. We assume the candidate mouth region that has D_h larger than one face width and less than 0 has a low probability of being the mouth. Thus, we set the standard deviation $\sigma = 1/3 \times 1/2 \times W_F = 1/6 \times W_F$. The Z value of D_h is represented by:

$$Z_h = \frac{\left| D_h - \frac{1}{2} \times W_F \right|}{\frac{1}{6} \times W_F} \qquad (3-18)$$

We can use Z_h to find the probability $P(mouthcorners \mid Z_h)$ from the z-table.

Measurement 5: Symmetry of two candidate mouth corners

We call the line that passes through the middle point of the two mouth corners and perpendicular to the mouth horizontal as the mouth vertical. In an ideal situation, the mouth vertical is the same as the face vertical axis. In practice, the difference between the eye vertical and the face vertical axis exists. Here, the symmetry will be represented by the similarity of areas of the left side and of the right side of mouth vertical. The candidate mouth corner pair that has high symmetry has a large probability of being the mouth corner pair.

We use N_l, N_r and N_d to represent the number of pixels on the left side and the right side of the eye vertical, and the difference of these two areas respectively. Thus, $N_d = |N_l - N_r|$. According to the ideal face symmetry property, N_d should be zero. We define the mean of normal distribution as $\mu = 0$. We assume that when the N_d is larger than ¼ of the face area, the symmetry of the candidate eye pair is too poor to be an actual eye pair. Thus, we define σ=1/3×1/4×face area=1/12×face area. The z-value of symmetry is represented by:

$$Z_s = \frac{|N_d|}{\frac{1}{12} \times face\ area} \qquad (3-19)$$

We can use Z_s to find the probability $P(mouthcorners \mid N_d)$ from z-table.

According to the knowledge-based selection method, the candidate eye pair that has

$$\begin{aligned} \max[P(mouthcorners \mid m_i)] = \max[&P(mouthcorners \mid D_v) \times P(mouthcorners \mid E_m) \\ &\times P(mouthcorners \mid D_c) \times P(mouthcorners \mid D_h) \\ &\times P(mouthcorners \mid N_d)] \qquad (3-20) \end{aligned}$$

will be set as the mouth corner pair.

3.2.4.5 Facial feature selection procedure

The procedure for facial feature localization is introduced in the following steps:

Step 1: After color segmentation, luminance verification and noise elimination, we obtain many candidate regions for the facial features. Calculate the centroids and the areas for all the candidate regions. The measurements, which describe the probabilities of these candidate regions to be facial features, are based upon the relative locations of the centroids or the sizes of those candidate regions.

Step 2: Use face parameters obtained from face detection to find face vertical axis and face horizontal axis.

Step 3: Use the face vertical axis and the horizontal axis to classify all the candidate regions into three groups: candidate regions for left eye, candidate regions for right eye and candidate regions for mouth. The candidate regions below the face horizontal axis are classified as candidate regions for the mouth. The candidate regions above the face horizontal axis are classified as candidate regions for the eyes. The human face is assumed to be vertically symmetric, among the eye candidate regions, the regions at the left of the vertical axis are classified as candidate regions for left eyes. Similarly, the regions in the right of the face vertical axis are classified as candidate regions for right eye. Then we can use all the candidate left eye regions and candidate right regions to compose the possible candidate eye pair.

Step 4: Use the knowledge-based method to select the candidate eye pair with the largest probability to be the actual eye pair. If another candidate pair has a probability close to the largest one, we compare the y-coordinates of these two pairs and select the candidate pair with the lower location. The centroids of the selected eye pair are used to represent the locations of the eye centers. The measurements used in the knowledge-based method will be introduced ...

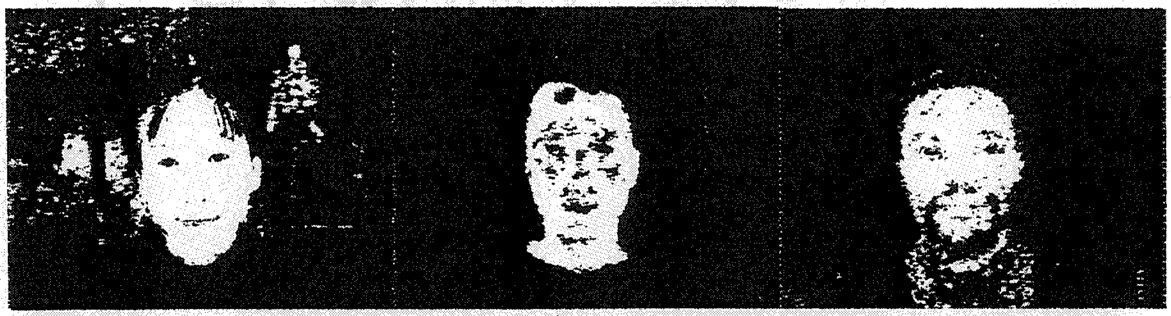
Step 5: Use the locations of the eye centers and the ideal face proportions to select the candidate mouth region with the largest probability. The mouth center will be represented by the centroid of the selected candidate region.

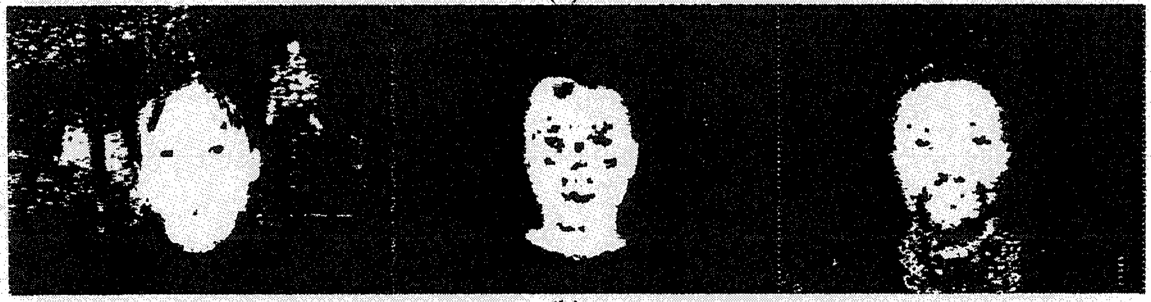
3.3 Result

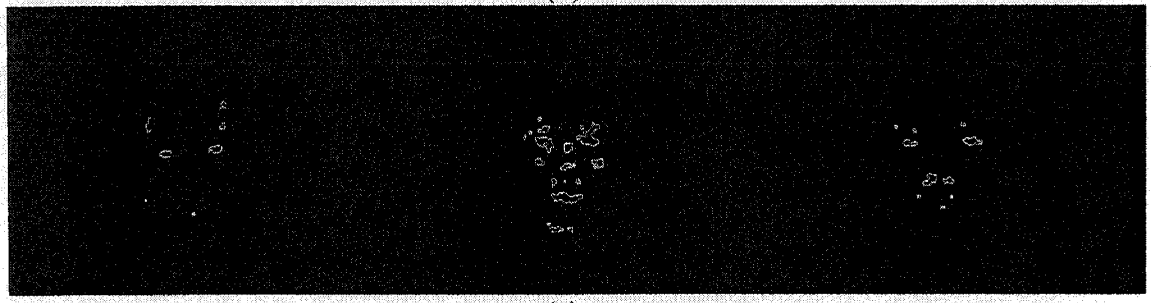
Three examples of facial feature localization using luminance verification method for both the eye selection and mouth selection are shown in Figure 3.7. Figure 3.7 (a), (b), (c) and (d) are original face maps after luminance verification, original face maps after geometric correction, contours of candidate regions and results of facial feature localization respectively. The results of these processing procedure can also be seen in Appendix I and Appendix II.

Three examples of mouth localization using corners are shown in Figure 3.8. Figure 3.8 (a) and (b) are candidates of mouth corners and results of facial feature localization respectively.

Same as last chapter, we use AR face database to test the proposed facial feature localization algorithm. Luminance verification and selecting mouth by corners are used in the test. In the face images, 31 men and 8 women have glasses, 12 men have beards, and 16 men have mustaches. It is sure that the glasses, beards, and mustaches can affect the results of the feature localization.


(a)


(b)


(c)

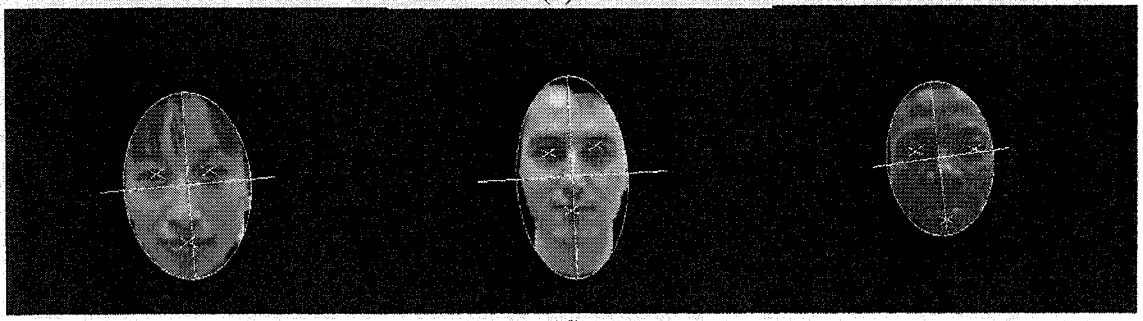
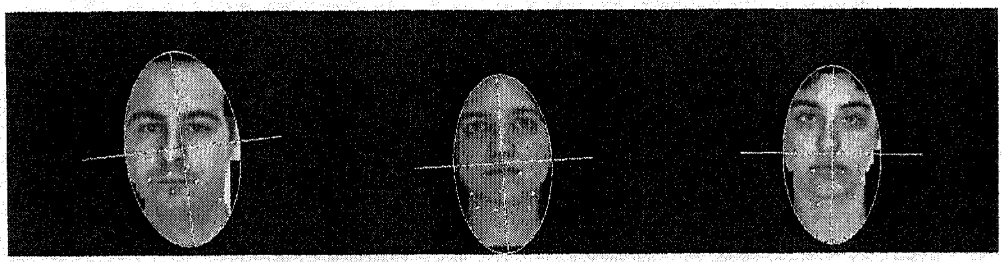
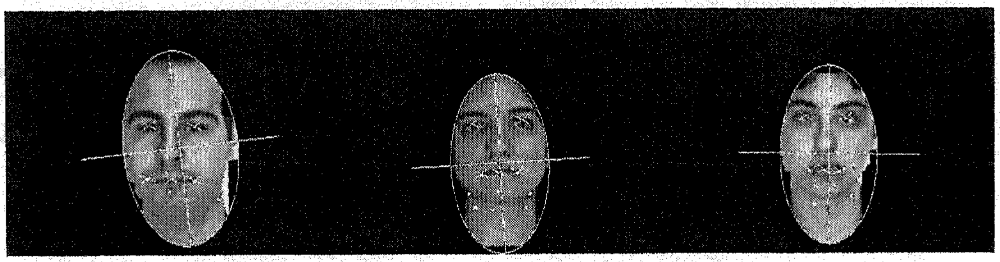

(d)

Figure 3.7 face map after contour extraction

(a) Original face map after luminance verification (b) Original face map after geometric correction (c) Contour of candidate regions (d) Results after facial feature localization



(a)



(b)

Figure 3.8 High texture feature selection and mouth localization

(a) Candidates of mouth corners. (b) Results of facial feature localization

Because the mouth localization method uses the information of eye positions, if the eyes cannot be localized, the mouth also cannot be localized. Therefore, if any one of three facial features cannot be localized from an image, we define that it is failed in localization.

In the experiment, 125 face images were successfully detected and 9 images failed. The rate of success was 94.98%. The successful results of the detected face images are shown in Appendix IV and the failed results are shown in Appendix V.

In order to know the accuracy of the facial feature localization results, we manually select the centre of each facial feature for each face image, and use it to compare with the automatically localized results.

Table 3.2 shows the average differences between the manually selected facial feature locations and the automatically localized locations. Result 1 is calculated from all the 133 face images. Result 2 is calculated from 84 people who have no glasses, no beards, and no moustaches. We can see that the results of eye location become better if the people have no glasses, no beards, and no moustaches.

Table 3.2 The average difference between manually selected feature l` ocations and automatically detected locations

	Left eye x	Left eye y	Right eye x	Right eye y	Mouth x	Mouth y
Result 1	1.86	1.13	2.14	1.38	1.37	2.22
Result 2	1.21	1.03	1.95	1.08	1.41	2.14

Several problems cause errors in the localization results:

- Glasses on face cause larger non skin-color regions around the eyes and give more difficulties to find the eye centers. Glasses on the face can also reflect lights. The high light points on the glasses show non-face color. This can affects the accuracy of the results too.
- The moustache and beard show non skin-color on the face. It is difficult to detect the face using color when someone has a heavy moustache or beard. The edges of the moustache or beard show as high texture area. Sometimes the high texture features at the mouth corners can be confused by features at the edges of the moustaches and beards.
- The hair falling on eyes can lead to errors in eye localization.
- Blonde hair has skin-color some time.
- Some people have uncommon face proportion.

From Table 3.2, we can see the results of eye locations become better if we only select the face images without glasses. The two results of mouth center in Table 3.2 show that the accuracy remains almost the same. The reason is that most of the time the center of the two mouth corners is similar to the center of the two high texture features selected by the knowledge-based selecting method.

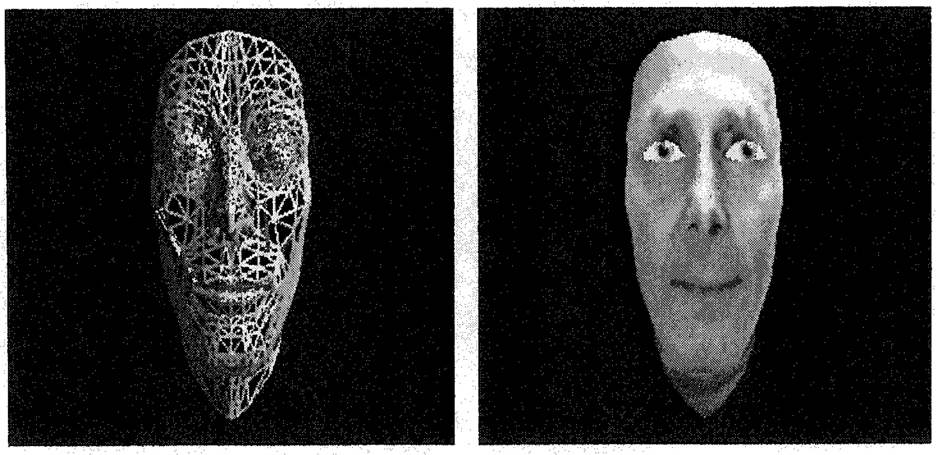
From the above results, we can see that the face detection and facial feature localization methods we introduced can detect the face and localize the facial features efficiently.

CHAPTER IV

FACE MODEL INITIALIZATION

Model-based view synthesis relies on high-level information extraction and attempts to explain what we see in images using a model. As a result, model-based coding of image sequences requires 3-D models for all objects in the scene. In our case, because we focus on head and shoulder scenes for videoconference applications, we need a face model.

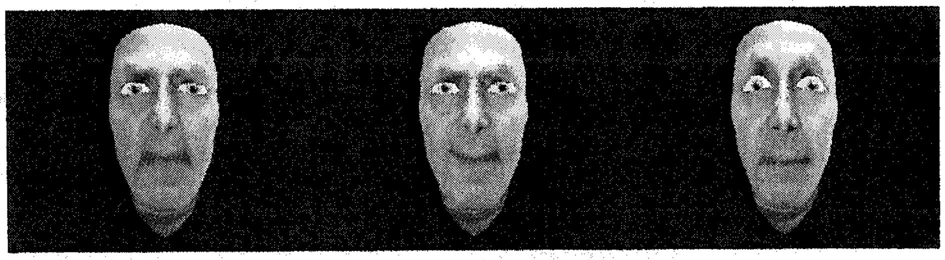
The face model we used was developed by Keith Waters at Compaq Cambridge Research Laboratory. The face wireframe model and texture mapped model are shown in figure 4.1.



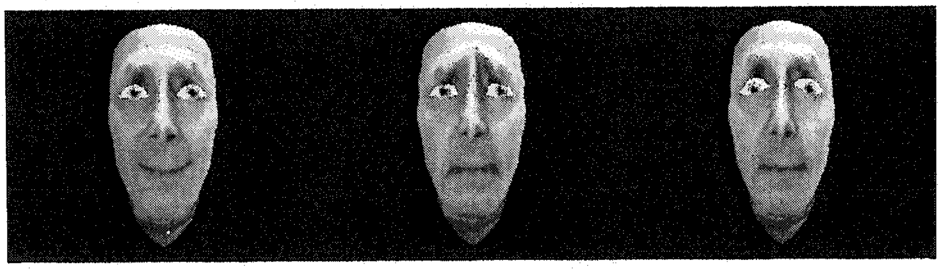
(a) Wireframe model (b) Texture mapped model

Figure 4.1 Face model

This face model is made by 256 3-D nodes, which represents the facial geometry as a collection of triangular polygons to be displayed and rendered. In addition, pre-determined indices are defined for the jaw and eyelid pointers so they can be opened and closed. The face model also has eighteen paired muscle types. These basic muscles are combined into a muscle expression macro to create a single facial expression. Figure 2 shows the six expressions of this model.



(a) Anger (b) Disgust (c) Fear



(d) Happiness (e) Sadness (f) Surprise

Figure 4.2 Face model with different expressions

Model initialization of an existing face model, uses techniques such as fitting and adaptation or morphing (Zhang and Cohen, 2000). After finding a set of key feature points using feature extraction methods, a generic model can be deformed in a way that minimizes the global differences between the located feature points and the corresponding points on the 3-D model. Lee and Magnenat-Thalmann (1999) use 3-D feature points as a set of control points for deformation. Then, the deformation of a surface can be seen as an interpolation of the displacements of the control points. Other methods rely on energy minimization techniques like the one proposed by Feng *et al* (2000), where a spring based face model is used. The nodes are connected by springs causing the nodes to move on the image, which is seen as an energy field, until they arrive at equilibrium.

The adaptation problem is decomposed into global and local adaptations. The global adaptation should account for the scale, position and orientation of the model. The local adaptation should refine the mismatch between the generic model and the measured facial contours, to correct for individual characters. There is more work that needs to be done in order to adapt the face model locally. Our project concentrates on the global adaptation problem.

It is important to mention that no depth information can be used and the information about feature points is merely 2-D if we use only one camera view. The polygonal mesh would fit the source image in the x-y plane and we would assume that the errors in the z-coordinates of the polygon mesh will not be noticeable for small rotations of the head.

Since it is impossible to adapt the face model to produce a model that corresponds accurately to the real human head using the nodes, and to synthesise all the features in the face like wrinkles, marks, *etc.*, a texture mapping technique is introduced for help. Now, with the help of hardware, the graphics workstation is capable of rendering texture mapping in real time.

4.1 Global adaptation for face model

After face detection and facial feature localization, all the parameters that could be used in face model adaptation are the face width, face height, and the centers of the mouth and the eyes. The purpose of the global adaptation is to adapt the general face model to a special face model so that the adapted face model has the same size (width, height), position, and initial pose as those of the real face. The global adaptation of the generic face model consists of 3-D rotation, translation, and scaling operations of the face. Thus, in principle, nine parameters must be estimated: three for scaling (S_x, S_y, S_z), three for translation (T_x, T_y, T_z), and three for rotation (R_x, R_y, R_z). According to the detection results in our project, T_z, R_x, and R_y cannot be estimated because we cannot obtain depth information from one camera view. We therefore assume these parameters are zero in the global adaptation.

The procedure of face model global adaptation is shown in Figure 4.3.

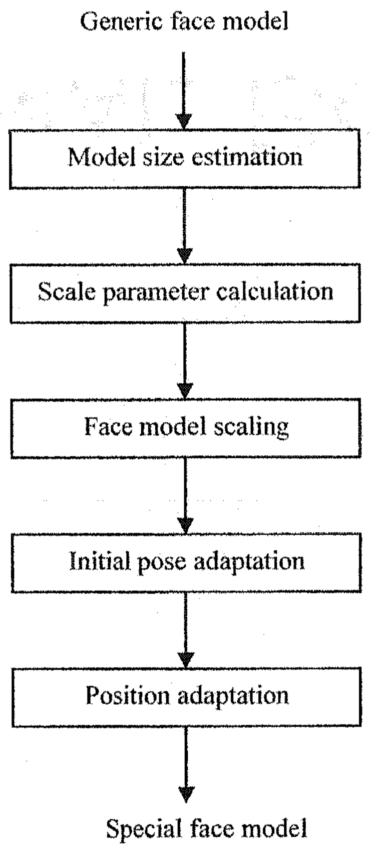


Figure 4.3 Procedure of face model global adaptation

Details of face model global adaptation are introduced in the following:

Model size estimation: In order to find the face width and face height of the face model, we search all the nodes of the face model to find the smallest or largest values in the x and y coordinate. Here we use $X_{\max\_m}$, $X_{\min\_m}$, $Y_{\max\_m}$, and $Y_{\min\_m}$ to represent the largest and smallest value in x and y coordinate of all the nodes, respectively. Thus, the face width W_M and face height H_M of the face model can be obtained from above values.

Scale parameter calculation: In our project, three rigid motion estimation methods are used to compare the results, two of which use the orthographic projection method to display the face model, one of which uses the perspective projection method. When we use orthographic projection, the face width and face height of the model should be equal to the detected face width and face height absolutely. When we display the perspective projection, the displayed size of the face model is related to the distance between the face model and the camera (image). The farther the face model is from the camera, the smaller it appears in the image. In our project, we select the distance between the face model and the camera manually to give the face model an appropriate displaying size in perspective projection.

The scale parameters can be calculated as below:

$$\begin{aligned} S_x &= W_F / W_M \\ S_y &= H_F / H_M \end{aligned} \qquad (4-1)$$

where S_x, S_y, W_F, and H_F represent the scale parameters in the x and y directions, face width, and face height from face detection, respectively. Because we have no depth information from one camera view, we use the average of scale parameters in the x and y directions to represent the scale parameter S_z in the z direction:

$$S_z = (S_x + S_y)/2 \qquad (4-2)$$

Face model scaling: The coordinates of each node on the face model can be scaled by the obtained scale parameters. The scale equations are:

$$x' = x \times S_x \quad y' = y \times S_y \quad z' = z \times S_z \qquad (4-3)$$

The x', y', and z' are the scaled coordinates of the nodes, and x, y, and z are the original coordinates of the nodes, respectively.

Because the eye models are independent parts of the face model, the radius and positions of the two eyes also need to be adapted. The radius of the eye models can be scaled by S_z. The positions of the eye models can be scaled in the same way as the 3-D nodes.

Initial pose adaptation: In videoconference, people's faces are usually horizontal in the first frame of the sequence. Although small head inclination angle R_z is not noticeable in practice, most faces are not truly horizontal. Since face angle detected in the face detection using elliptic shape correction is not as accurate as the eye angle, see Chapter III. We use the eye angle to represent the face angle here.

Position adaptation: We need to adapt the face model to the same position in the displaying window as that of the real face in the image. We simply move the face model to the same position by using the face center position obtained, provided the displaying window of the face model has the same size as the image window. If the size of the displaying window is different from the image size, we can use the relative position to adapt the position of face model. In this case, we can normalize the image coordinates to [0,1], then use the normalized position of the face center to calculate the relative position of the face center in the displaying window of the face model.

4.2 Texture mapping

The texture mapping method we developed is based upon the information we obtained at the face detection and facial feature localization. The main task of texture mapping is to calculating the mapping coordinates.

Before we calculate the mapping coordinates, three pre-processing procedures are needed. First, we need to obtain the face texture map. In OpenGL, the texture map must be a rectangular image. It is easy to cut the face texture from the image using the parameters obtained from face detection, such as the face width, face height, or center of the ellipse (face center). Second, we need to scale the face texture map. The face texture map we cut from the image cannot be used in mapping directly because the size of the face map cannot possibly satisfy the OpenGL condition: width and height of the texture map must have the form x^m, where m is a non-negative integer. Third, the face texture map needs to be adapted to the horizontal position because the faces in the images are not guaranteed to be horizontal. In our project, we first scale all the texture maps to a 128 by 128 image. Then we use the eye angle, as introduced above to rotate the face to the horizontal position.

Details of mapping coordinates calculation are introduced in the following:

Two-dimensional textures are square or rectangular images that are typically mapped to the polygons that make up a polygonal model. This means we need to do a 2-D to 3-D mapping. Using the normalized coordinates to describe the positions in the texture map is a good method for utilizing the different sizes of the texture maps. In OpenGL, we do not need to do the mapping for each pixel in the image. We only need to find some important mapping points, called control points, in the face model and the face texture, and then match all of them. By following the positions of those control points, the computer will do the remaining work of texture mapping.

In our case, each node on the face model needs to be mapped to a pixel in the face texture. The coordinates of pixels in the face texture are the normalized coordinates of the texture map. They tell computer the exact positions that the nodes should be matched onto the texture map.

The available information from the face detection includes the positions of the facial feature centers and the 2D size of the face. In the texture mapping, we can use the known face sizes to calculate the normalized coordinates for pixels in the face texture and use facial features as the control points to match with those on the face model. The positions of facial features can be manually selected on the face model at the beginning. Because all the generic models at the beginning are the same, we select the positions of those facial features only once. We must remember that these parameters must be scaled at the face model global adaptation stage with other 3-D nodes. The scaled method is the same as that of 3-D nodes on the face model.

The idea of suggested method is to match all the control points between the texture map and the face model first. After that, all 3-D nodes can map to 2-D pixels in the face texture according to their relative positions with the control points.

Centers of two eyes and mouth are used as control points. In practice, the facial features that we detected before cannot alone implement the mapping accurately. In order to keep the shapes of the facial features after texture mapping, five control points are used in the face region. Two additional control points are:

Mouth lower: Because the face model includes the neck and we do not know the size of the neck in the texture map, it is difficult to keep the shape of mouth in the texture mapping when we just use the mouth center to do the texture mapping at that region. Also, the sizes of the necks in the texture map and the face model are not the same, normally. In order to solve this problem, we select an additional point under the mouth to control the texture mapping in the mouth area, we call it mouth lower. If the point of mouth lower in the face texture can be matched with that on the face model, the mouth shape will be kept, even if the sizes of the necks are not matched. The position of the mouth lower is selected to be 1/8 of the face width lower than the mouth center and has the same y coordinate as the mouth center, as shown in Figure 4.4. We call the facial features and the mouth lower control points.

Nose center: The nose center is another necessary control point to keep the face shape.

Therefore, the whole face is separated into eighteen regions by these five control points, as shown in Figure 4.4.

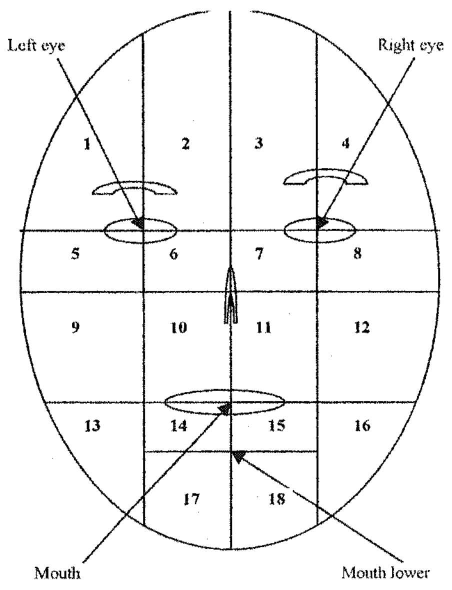


Figure 4.4 Face separation for mapping coordinate calculation

The method we suggested to perform 3-D to 2-D mapping is introduced in the following steps:

Step 1: Calculate the normalized positions of facial features in the texture map.

We use x_{leye_f}, y_{leye_f}, x_{reye_f}, y_{reye_f}, x_{mouth_f}, and y_{mouth_f} to represent the x and y coordinates of the left eye, right eye and mouth in the face texture respectively. These are the parameters we know. We use x_{nose_f}, y_{nose_f} x_{ml_f}, and y_{ml_f} to represent the x and y coordinates of the nose center and the mouth lower in the face texture.

Considering the ideal face proportions, the nose center should be located at the middle of the eyes and mouth. The position of nose center can be calculated by the following equations:

$$\begin{aligned} x_{nose\_f} &= \frac{x_{leye\_f}}{4} + \frac{x_{reye\_f}}{4} + \frac{x_{mouth\_f}}{2} \\ y_{nose\_f} &= \frac{y_{leye\_f}}{4} + \frac{y_{reye\_f}}{4} + \frac{y_{mouth\_f}}{2} \end{aligned} \qquad (4-4)$$

The mouth lower can be obtained by:

$$\begin{aligned} x_{ml\_f} &= x_{m\_f} \\ y_{ml\_f} &= y_{ml\_f} + \frac{W_F}{8} \end{aligned} \qquad (4-5)$$

Here, W_F is the detected face width.

We use the following equations to calculate the normalized coordinates for these control points:

$$x_{leye\_f\_n} = \frac{x_{leye\_f}}{W_F} \qquad y_{leye\_f\_n} = \frac{y_{leye\_f}}{H_F} \qquad (4-6)$$

where $x_{leye\_f\_n}$ and $y_{leye\_f\_n}$ are the normalized coordinates of the left eye.

We use $x_{reye\_f\_n}$, $y_{reye\_f\_n}$, $x_{mouth\_f\_n}$, $y_{mouth\_f\_n}$, $x_{nose\_f\_n}$, $y_{nose\_f\_n}$, $x_{ml\_f\_n}$, and $y_{ml\_f\_n}$ to represent other normalized coordinates of the control points. They can be calculated in the same way.

On the face model, x_{leye_m}, y_{leye_m}, x_{reye_m}, y_{reye_m}, x_{mouth_m}, y_{mouth_m}, x_{nose_m}, y_{nose_m}, x_{ml_m}, and y_{ml_m} are used to represent the x and y coordinates of those control points accordingly.

Step 2: Mapping control points between face texture and face model.

All the control points on the face model must map onto those in the texture map. We call the coordinate of a pixel in the face map a mapping coordinate of the node to which the node on the face model mapped. As a result, all the mapping coordinates of those control points on the face model are equal to the normalized coordinates of the control points in the texture map accordingly. We use tx_{leye_m}, ty_{leye_m}, tx_{reye_m}, ty_{reye_m}, tx_{mouth_m}, ty_{mouth_m}, tx_{nose_m}, ty_{nose_m}, tx_{ml_m} and ty_{ml_m} to represent the mapping coordinates of the control points in the x and y direction on the face model. Thus:

$$\begin{array}{ll} tx_{leye\_m} = x_{leye\_f\_n} & ty_{leye\_m} = y_{leye\_f\_n} \\ tx_{reye\_m} = x_{reye\_f\_n} & ty_{reye\_m} = y_{reye\_f\_n} \\ tx_{mouth\_m} = x_{mouth\_f\_n} & ty_{mouth\_m} = y_{mouth\_f\_n} \\ tx_{nose\_m} = x_{nose\_f\_n} & ty_{nose\_m} = y_{nose\_f\_n} \\ tx_{ml\_m} = x_{ml\_f\_n} & ty_{ml\_m} = y_{ml\_f\_n} \end{array} \qquad (4-7)$$

Step 3: Calculate mapping coordinates for all the nodes on the face model.

From Figure 4.4, we see that the face is separated into three large regions: the left eye region, the right eye region and the mouth region. According to the relative positions of control points, we further separate the three large regions into smaller regions. The total number of smaller regions on the face is 18. The left eye area includes small areas 1, 2, 5, and 6. The right eye area includes small areas 3, 4, 7, and 8. From Figure 4.3, we can see that there are more regions below the mouth. More regions below the mouth can keep the mouth shape, even if the face has different size of neck. For each node on the face model, we first find the region to which it belongs, then use the relative distance with the control points we selected before to calculate the mapping coordinates.

In the face texture map, $D_{len\_x\_f}$, $D_{len\_y\_f}$, $D_{ren\_x\_f}$, $D_{ren\_y\_f}$, $D_{mn\_x\_f}$, $D_{mn\_y\_f}$, $D_{mlow\_x\_f}$, and $D_{mlow\_y\_f}$ are used to represent the distances between the left eye center and the nose center, the right eye center and the mouth center, the mouth center and the nose center and the mouth center and the mouth lower, in both the x and y directions respectively. For the face model, $D_{len\_x\_m}$, $D_{len\_y\_m}$, $D_{ren\_x\_m}$, $D_{ren\_y\_m}$, $D_{mn\_x\_m}$, $D_{mn\_y\_m}$, $D_{mlow\_x\_m}$, $D_{mlow\_y\_m}$, $D_{mn\_x\_m}$, and $D_{mlow\_x\_m}$ are used to represent the distances between the respective control points as above. $D_{mn\_x\_f}$, $D_{mlow\_x\_f}$, $D_{mn\_x\_m}$, and $D_{mlow\_x\_m}$ are zero at this time. Figure 4.5 shows the distances between the control points in both the text map and face model.

For each node on the face model, we can know the region to which it belongs by using the control points that we selected. We assume that the x and y coordinates of a node on the face model are x and y, and the mapping coordinates of this node are t_x and t_y. The largest and smallest value in x and y coordinates of all the nodes are $X_{\max\_m}$, $X_{\min\_m}$, $Y_{\max\_m}$, and $Y_{\min\_m}$, respectively. Take note here, that the y-axis of the coordinate system of the face model is from face bottom to face top, the x-axis is from left to right. The method to calculate the mapping coordinates for arbitrary nodes on the face model in small area 1 is described in Equation (4-8).

$$t_x = tx_{leye\_m} + tx_{leye\_m} \times \frac{x - x_{leye\_m}}{x_{leye\_m} - X_{\min\_m}}$$
$$t_y = ty_{leye\_m} + (1 - ty_{leye\_m}) \times \frac{y - y_{leye\_m}}{Y_{\max\_m} - y_{leye\_m}} \qquad (4 - 8)$$

The equations to calculate the mapping coordinates in other regions are similar to Equation (4-8).

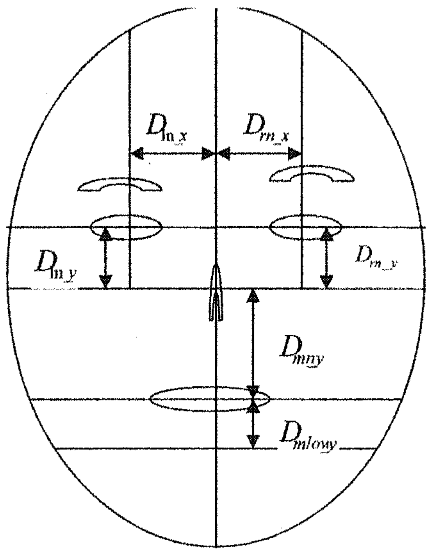
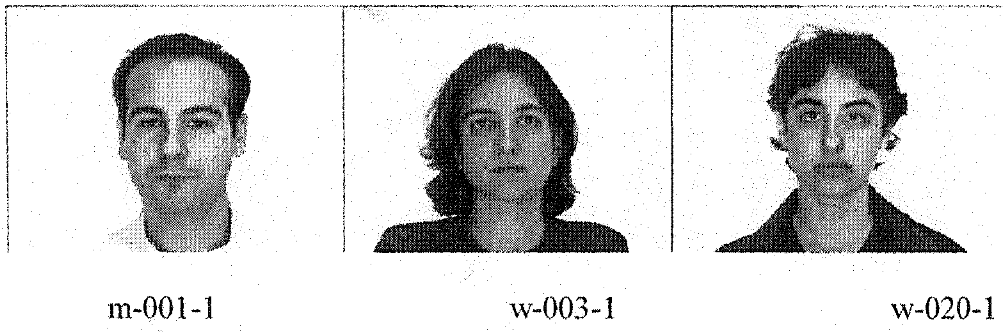


Figure 4.5 The distances between the control points

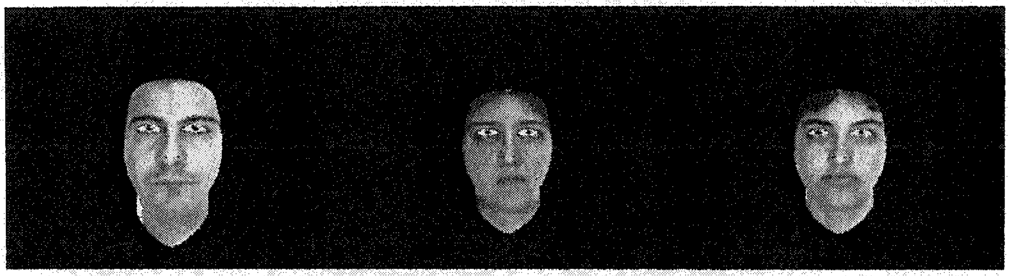
4.3 Results

Figure 4.6 shows the results of face model initialization using three face images in the AR face database. Figure 4.6 (a) shows the original face images, (b) shows the initialized face model by using orthographic projection and (c) shows the initialized face model by using orthographic projection.

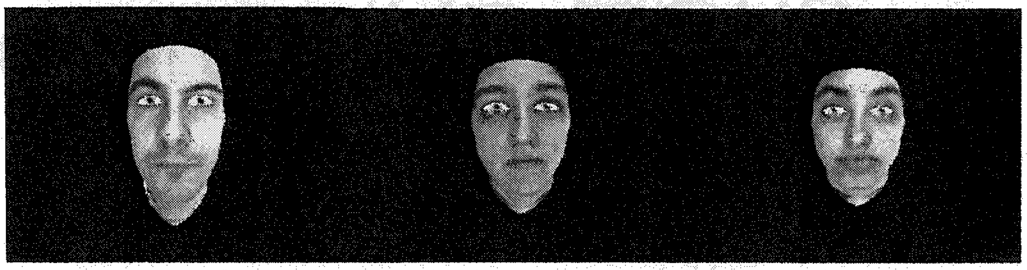
It is difficult in general to find objective measures to assess the performance and quality of the model adaptation. Part of the difficulties reside in the fact that the wireframe model itself is rather coarse. In addition, the parameters we use to adapt the face model are not accurate enough. For example, in our project, the face width and face height detected in the 2-D image cannot express the face width and face height in the 3-D world accurately. The necks of the people are also included in the face height because the neck skin has the same color as that of the face skin. Further contour detection methods can be used to separate the neck from the face.



(a) Face images



(b) Using orthographic projection



(c) Using perspective projection

Figure 4.6 Generic model after initialization

CHAPTER V

GOOD FEATURE SELECTION AND FEATURE TRACKING

After the face detection and model initialization stages, we obtain a modified face model with a size and position similar to those of the real face in the first frame of a sequence. Rigid motion estimation is, then, used to adapt the face model to the same pose as the real face in the subsequent frames. Rigid motion estimation aims at recovering as accurately as possible the 3-D pose of the head.

Existent algorithms for rigid motion estimation include estimation by feature tracking, analysis by synthesis and view-based techniques. Given the real time objective of our project, estimation from feature tracking appears to be a good choice. It can avoid the face structure analysis and iterative adaptation in the estimation period. The motion parameters can be calculated using the tracked positions of all selected features and a motion model.

The three facial features that were localized in the former stages are not sufficient for motion estimation due to tracking noise and the requirements of the motion model in terms of minimum number of features. The motion estimator will be more robust to noise if we use more features in the estimation. On the other hand, tracking more features requires more computation time. In our project, we do not use the facial features that were localized because we do not know whether or not the facial features are good features for tracking. A feature selection criterion should be used to select good features for tracking.

5.1 Feature selection

Feature can be selected based upon some measures of textureness or cornerness, such as a high standard deviation in the spatial intensity profile, the presence of zero crossings of the Laplacian of the image intensity, and corners. The feature selection criteria used in our system is proposed by Shi and Tomasi (1994). They proposed a feature selection criteria that is optimal for tracking and a feature monitoring method that can detect occlusions, disocclusions, and features that do not correspond to real points in the world. The suggested monitoring method uses an affine motion model to monitor the quality of image features during tracking.

5.1.1 Two models of image motion

As the camera moves, the patterns of image intensities change in a complex way. However, away from occluding boundaries and near surface markings, these changes can often be described as image motions:

$$I(x,y,t+\tau)=I(x-\xi(x,y,t,\tau),y-\eta(x,y,t,\tau)) \qquad (5-1)$$

Thus, a later image taken at time $t+\tau$ can be obtained by moving every point in the current image, taken at time t, by a suitable amount. The mount of motion $\vec{\sigma}=(\xi,\eta)$ is called the displacement of the point at $\vec{x}=(x,y)$.

5.1.1.1 Affine motion model

The affine motion model is shown in the following:

$$\vec{\sigma}=D\vec{x}+\vec{d} \qquad (5-2)$$

where

$$D=\begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \qquad (5-3)$$

is a deformation matrix, and $\vec{d}$ is the translation of the feature window's center. The image coordinates $\vec{x}$ are measured with respect to the window's center. Then, a point $\vec{x}$ in the first image, I, moves to point $A\vec{x}+\vec{d}$ in the second image J, where $A=\vec{1}+D$ and $\vec{1}$ is the 2*2 identity matrix:

$$J(A\vec{x}+\vec{d})=I(\vec{x}) \qquad (5-4)$$

5.1.1.2 Translation model

When the deformation matrix D is assumed to be zero, the translation model is obtained:

$$\vec{\sigma}=\vec{d} \qquad (5-5)$$

5.1.2 Computing image motion

Because of image noise and because the affine motion model is not perfect, Equation (5-4) is in general not satisfied exactly. The problem of determining the motion parameters is then that of finding the A and $\vec{d}$ that minimize the dissimilarity

$$\varepsilon = \iint_W [J(A\vec{x} + \vec{d}) - I(\vec{x})]^2 \omega(\vec{x}) d\vec{x} \qquad (5-6)$$

where W is the given feature window and $\omega(\vec{x})$ is a weighting function. In the simplest case, $\omega(\vec{x}) = 1$. Alternatively, ω could be a Gaussian-like function to emphasize the central area of the window. Under pure translation, the matrix A is constrained to be equal to the identity matrix. To minimize the residual (5-6), we differentiate it with matrix D and the displacement vector $\vec{d}$ and set the result to zero.

$$\frac{1}{2}\frac{\partial \varepsilon}{\partial D} = \int_W [J(A\vec{x} + \vec{d}) - I(\vec{x})]\vec{g}\vec{x}^T \omega d\vec{x} \qquad (5-7)$$
$$\frac{1}{2}\frac{\partial \varepsilon}{\partial \vec{d}} = \int_W [J(A\vec{x} + \vec{d}) - I(\vec{x})]\vec{g}\omega d\vec{x} \qquad (5-8)$$

where $\vec{g} = (\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y})^T$.

Assuming small motion, $A\vec{x} + d = \vec{x} + (D\vec{x} + \vec{d}) = \vec{x} + \vec{d}$. We then linearize the resulting system by the truncated Taylor expansion

$$J(A\vec{x}+\vec{d}) = J(\vec{x}) + \vec{g}^T\vec{u} \qquad (5-9)$$

where $\vec{u} = D\vec{x}+\vec{d}$. Combining (5-7) (5-8) and (5-9), we have

$$\int_W \vec{g}\vec{x}^T(\vec{g}^T\vec{u})\omega d\vec{x} = \int_W [I(\vec{x}) - J(\vec{x})]\vec{g}\vec{x}^T\omega d\vec{x} \qquad (5-10)$$

$$\int_W \vec{g}(\vec{g}^T\vec{u})\omega d\vec{x} = \int_W [I(\vec{x}) - J(\vec{x})]\vec{g}\omega d\vec{x} \qquad (5-11)$$

We can rewrite the (5-10) and (5-11) as a linear system of 6 equations and 6 unknowns. This yields the following 6*6 system:

$$T\vec{z} = \vec{a} \qquad (5-12)$$

where $\vec{z}^T = [d_{xx}, d_{yx}, d_{xy}, d_{yy}, d_x, d_y]$ collects the entries of the deformation D and displacement $\vec{d}$, the error vector

$$\vec{a} = \int_W [I(\vec{x}) - J(\vec{x})]\begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix}\omega d\vec{x} \qquad (5-13)$$

depends on the difference between the two images, and the 6*6 matrix T, which can be computed from one image, can be written as

$$T = \iint_V \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} \omega d\vec{x} \qquad (5-14)$$

where

$$U = \begin{bmatrix} x^2 g_x^2 & x^2 g_x g_y & xy g_x^2 & xy g_x g_y \\ x^2 g_x g_y & x^2 g_y^2 & xy g_x g_y & xy g_y^2 \\ xy g_x^2 & xy g_x g_y & y^2 g_x^2 & y^2 g_x g_y \\ xy g_x g_y & xy g_y^2 & y^2 g_x g_y & y^2 g_y^2 \end{bmatrix} \qquad (5-15)$$

$$V^T = \begin{bmatrix} x g_x^2 & x g_x g_y & y g_x^2 & y g_x g_y \\ x g_x g_y & x g_y^2 & y g_x g_y & y g_y^2 \end{bmatrix} \qquad (5-16)$$

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \qquad (5-17)$$

Even when affine motion is a good model, equation (5-12) is only approximately satisfied, because of the linearization of equation (5-9). However, the correct affine change can be found by using equation (5-12) iteratively in a Newton-Raphson style minimization.

During tracking, the affine deformation D of the feature window is probable to be small, since motion between adjacent frames must be small in the first place for tracking to work at all. It is then safer to set D to the zero matrix. Consequently, when the goal is to determine $\vec{d}$, the smaller system

$$Z\vec{d} = \vec{e} \qquad (5-18)$$

should be solved, where $\vec{e}$ collects the last two entries of the vector $\vec{a}$ of equation (5-12).

5.1.3 Feature selection criteria

We can track a window from frame to frame if the system in Equation (5-18) represents good measurements, and if it can be solved reliably. Consequently, the symmetric 2*2 matrix Z of the system must be both above the image noise level and well-conditioned. The noise requirement implies that both eigenvalues of Z must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. Two small eigenvalues mean a roughly constant intensity profile within a window. A large and small eigenvalue correspond to a unidirectional texture pattern. Two large eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

In practice, when the smaller eigenvalue is sufficiently large to meet the noise criteria, the matrix Z is usually also well-conditioned. In fact, the intensity variations in a window are bounded by the maximum allowable pixel value, so that the greater eigenvalue cannot be arbitrarily large. In conclusion, if the two eigenvalues of Z are λ_1 and λ_2 we accept a window if

$$\min(\lambda_1, \lambda_2) > \lambda \qquad (5-19)$$

where λ is a predefined threshold. We set λ to 1 in our project.

5.1.4 Gradient of image

The vector $\vec{g}$ used in Equation (5) and (6) is the gradient of the image. g_x and g_y are the two elements of $\vec{g}$ in x and y directions respectively. Gradient calculation is impaired by image noise. Image noise is assumed to be white noise, *i.e.*, independently and identically distributed over each pixel and with a zero-mean Gaussian distribution. One approach used to reduce the impact of image noise is to convolve the image with a 2-D Gaussian filter.

Because the derivative operator and the convolution are linear operators, we have:

$$\frac{\partial}{x}[Gauss(x,y)\otimes I(x,y)] = \frac{\partial}{x}Gauss(x,y)\otimes I(x,y)$$
$$\frac{\partial}{y}[Gauss(x,y)\otimes I(x,y)] = \frac{\partial}{y}Gauss(x,y)\otimes I(x,y) \qquad (5-20)$$

where $I(x,y)$ is the image intensity at location on (x, y) and $Gauss(x,y)$ is the 2-D Gaussian function. In practice, this process can be accomplished through the use of a two-dimensional Gaussian function or combination of a one-dimensional Gaussian function in both the x and y directions.

A convolution mask can be created to represent the operation of Equation (5-20). The values of the first derivative Gaussian mask depend upon the choice of sigma in the Gaussian filter:

$$\mathrm{Gauss}(x) = \frac{1}{\sqrt{2\pi\sigma}}e^{\frac{-x^2}{2\sigma^2}} \qquad \mathrm{Gauss}'(x) = \frac{-x}{\sqrt{2\pi\sigma^3}}e^{\frac{-x^2}{2\sigma^2}} \qquad (5-21)$$

5.1.5 Feature selection procedure

We use following steps to select the good features for face tracking:

i. **Step 1**: Face region selection.

For face tracking purposes, all the features used in rigid motion estimation must be selected in the face region. Therefore, we need to use the face position parameters that obtained during the face detection stage in order to isolate the face region from the rest of the image.

Step 2: Gradient calculation.

Use the method introduced in 5.1.4 to calculate the gradients for all pixels in the face region.

Step 3: Z matrix calculation.

Calculate all the elements g_x^2, g_xg_y and g_y^2 of matrix Z for each pixel. Then calculate the sums of g_x^2, g_xg_y, and g_y^2 for all pixels in the feature window respectively.

ii. **Step 4**: Tracking ability calculation.

The smaller one of two eigenvalues of matrix Z, which is calculated in the feature window, is used to represent the tracking ability of this pixel. We call the smaller one of two eigenvalues quality of feature here. All the pixels are sorted by using the quality of feature.

Step 5: Feature selection

According to the feature selection criteria introduced before, the larger the quality of feature, the greater the possibility that it is a good feature. Consequently, we select the good features beginning from the largest value of quality of feature to the smaller one. In order to avoid selecting good features in the close neighborhoods, we also define a minimum distance. A selected feature that is within the minimum distance of a better one will be deleted. In our project, setting minimum distance to 10 can obtain good result. The number of features selected depends upon the noise level and the motion estimation algorithm used.

Figure 5.1 shows the results of good feature selection using three face images from the AR face database: (a) shows the original face images. (b) shows the smoothed face-region images using Gaussian filter. (c) shows the gradient image in the x and y directions. The white points in (d) are the features selected using the feature selection criteria introduced above. Twelve features are selected in each image.

In the facial feature localization stage, we also use the feature selection method introduced above to detect the mouth corners. The mouth corners are assumed to have higher possibilities of being good features. In this case, we set the number of features, feature window size and minimum distance to 12, 11*11, and 14 respectively. Figure 5.2

shows the selected features in the mouth regions of the face images for facial feature localization. The original images are shown in Figure 5.1 (a).

5.2 Feature tracking

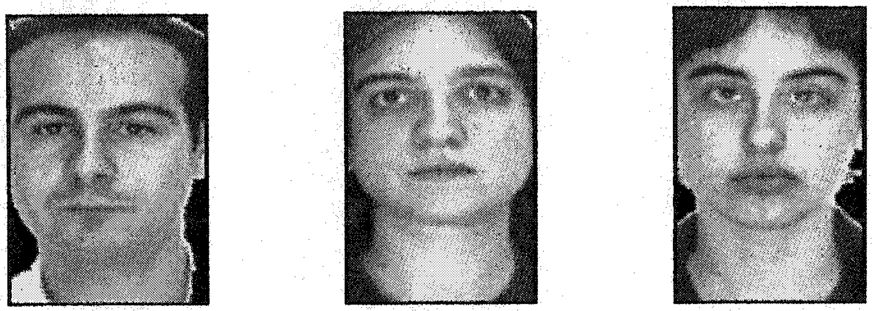
5.2.1 Tracking and monitoring method

Shi and Tomasi (1994) show that the best combination of two motion models in (5-4) and (5-5) is pure translation for tracking, because of its higher reliability and accuracy over the small inter-frame motion of the camera, and over the affine motion for comparing features between the first and the current frames in order to monitor their quality.

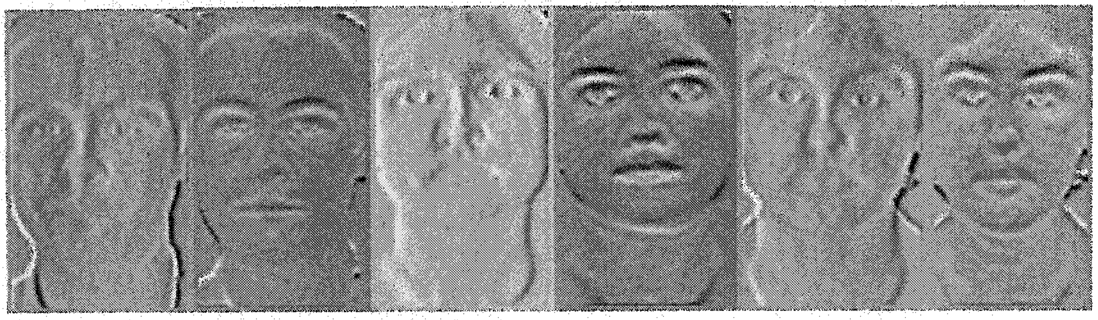
As introduced in 5.1.2, we set the affine deformation D of the feature window to zero during tracking, since the motion between adjacent frames must be small. In fact, attempting to determine deformation parameters in this situation is not only fruitless but can lead to poor displacement solutions: the deformation D and the displacement $\vec{d}$ interact through the 4×2 matrix V of equation (5-14), and any errors in D would cause errors in $\vec{d}$.

Copyright 2010

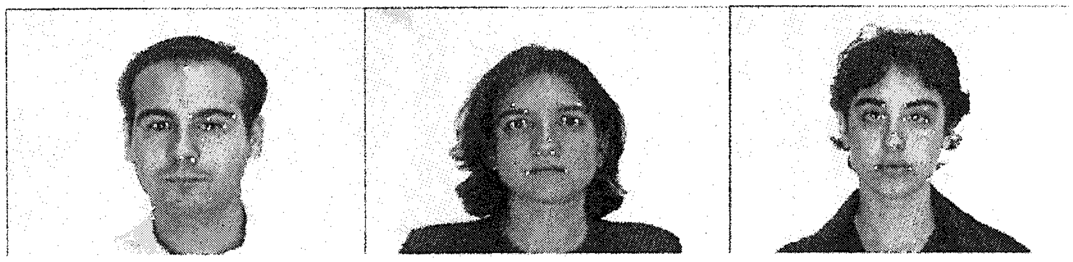
(a)



(b)



(c)



(d)

Figure 5.1 Good feature selection

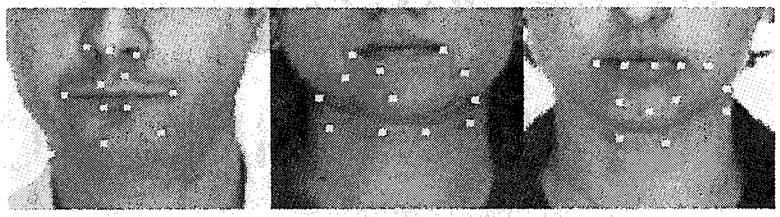


Figure 5.2 The features in the mouth region

When monitoring features for dissimilarities in their appearance between the first and the current frame, on the other hand, the full affine motion system (5-12) should be solved. In fact, motion is now too large to be described well by the pure translation model. Furthermore, in determining dissimilarity, the whole transformation between the two windows is of interest, and a precise displacement is less critical, so it is acceptable for D and $\vec{d}$ to interact to some extent through the matrix V.

In our project, in order to improve the tracking quality for motion estimation, we select more tracking points than the minimum number required for motion estimation. In the following frames, we only use points with high tracking qualities into the motion estimator by monitoring. We also need to update the input reference positions in the first frame if the tracking points are changed. Details of reference positions are introduced in Chapter VI.

5.2.2 Block matching method

Block matching is the most popular method for practical feature tracking. In block matching, the best motion vector estimate is found by a pixel-domain search procedure. The basic idea of block matching is depicted in Figure 5.3, where the displacement for a pixel (n_1, n_2) in frame k (the present frame) is determined by considering a $N_1 \times N_2$ block centered about (n_1, n_2), and searching frame k+1 (the search frame) from the location of the best matching block of the same size. The search is usually limited to a region called the search window for computational reasons.

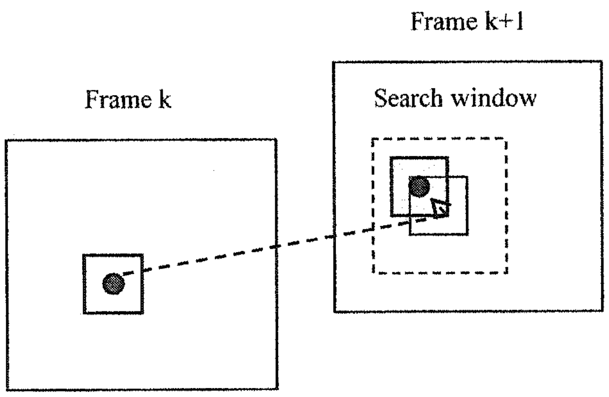


Figure 5.3 Block matching

Block Matching algorithms differ in the matching criteria, the searching strategy, and the determination of block size.

The matching of the blocks can be quantified according to various criteria including the maximum cross-correlation, the minimum mean square error (MSE), the minimum mean absolute difference (MAD), and the maximum matching pixel count (MPC) (Kenneth 1996). In our project, we select the MSE criteria in the block matching.

We evaluate the MSE, defined as

$$MSE(x,y)=\frac{1}{N_1N_2}\sum_{(n_1,n_2\in B}[s(n_1,n_2,k)-s(n_1+x,n_2+y,k+1)]^2 \qquad (5-22)$$

where B denotes an $N_1 \times N_2$ block for a set of candidate motion vectors (n_1,n_2). The estimate of the motion vector is taken to be the value (x, y) which minimizes the MSE.

Finding the best matching block requires optimizing the matching criteria over all possible candidate displacement vectors at each pixel. This can be accomplished by the so-called "full search" which evaluates the matching criteria for all values at each pixel, and is time-consuming. In order to reduce the computational load, some algorithms, such as the three-step search algorithm, evaluate the criteria function at a predetermined subset of the candidate motion vector locations only. Although the three-step search algorithm can save time, the tracking results become worse. In the experiments, the tracking results of the three-step search algorithm become unacceptable. Thus the full-search method is suggested in the project.

5.3 Results

At beginning, it seems that the tracking and monitoring method is more robust than traditional blocking matching method. Compared with blocking matching method, the tracking and monitoring method has better tracking results. At the same time, the tracking and monitoring method can know the tracking quality of each feature by monitoring. Thus, it can always select the good tracking points for the motion estimator.

In the experiments, we found some critical disadvantages in the tracking and monitoring method:

- It requires more computation. The speed of tracking is about 1 frame per second.
- It is difficult to define an appropriate threshold for monitoring. Most of the time, a low dissimilarity threshold can cause all the points to be lost during several frames, and a high dissimilarity threshold can cause the monitoring function to be completely lost. Even if we can define a good threshold for one sequence, the same situation exists if we change the sequence.
- As introduced below, it has the same problems as block matching problem when the object has rotations. Because only translation is assumed in tracking.

For the above reasons, we select the block matching method in our system. The speed of block matching method is faster than 5 frames per second and the results are acceptable. All the results of motion estimation in the following chapters are based upon the results of block matching method. In tracking, using searching region with size 11×11 and searching window with size 9×9 can obtain good results.

In order to test the tracking results, we use one tracking point on the face from one face sequence. The point used in our experiment was the corner of right eyebrow. It is shown as a white point in Figure 5.4.

We first manually select the same corner of that eyebrow in all 200 frames. Then we use the positions obtained as the true positions of that corner in the sequence to compare with the results of the tracking.

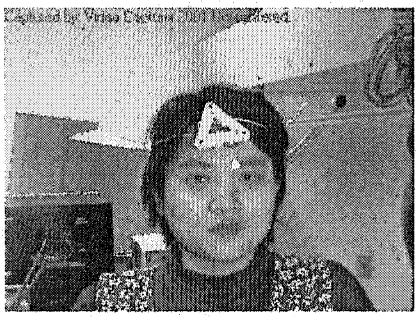
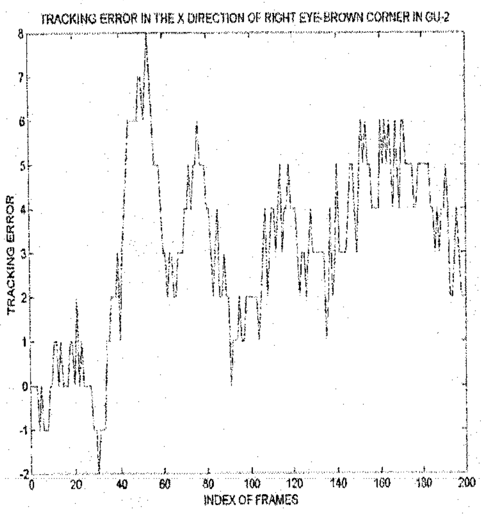


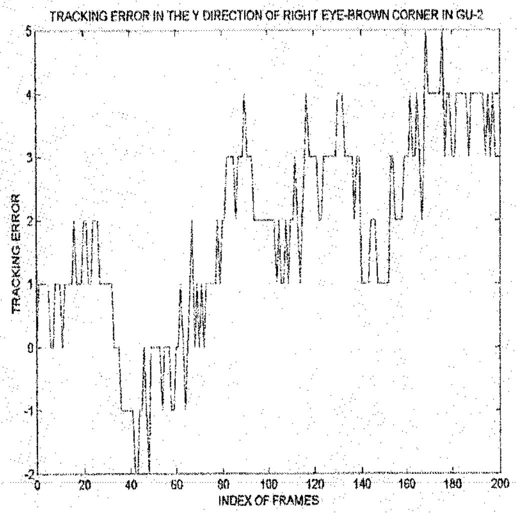
Figure 5.4 First frame of a face sequence

Our experiments show that the blocking matching method can obtain good results when the object in the image has only translation. The rotation can cause an error accumulations. In the sample sequence mentioned above, the head has both translation and rotation. Figure 5.5 shows the differences between the manually selected corner positions and tracking positions in the x and y coordinates respectively. Figure 5.6 shows the squared differences for both the x and y coordinates between the manually selected results and tracking results. From the results, we can see that errors become larger when more images are tracked.

Figure 5.7 and Figure 5.8 show the intensity differences between the matched neighbour blocks by using manually selecting and the block matching method, respectively. We can see the intensity differences of the block matching method have smaller values than those (assumed to be true values) of manually selecting method. This situation is caused by the failure of constant intensity assumption for the tracking points in different frames when the object has large rotations. Figure 5.9 further describes the limitations of the block matching method. In 5.9 (b), we can see it is impossible to track the correct positions of that point by using block matching, especially in frames 110, 120, and 130.

In order to solve the error accumulation and the intensity changing problems, a method that using two tracking templates obtained in the first and previous frame and matching with rotations in the current frame can obtain better results.





(a) Errors in x direction (b) Errors in y direction

Figure 5.5 Intensity differences in x and y direction between the manually selected results and the tracking results

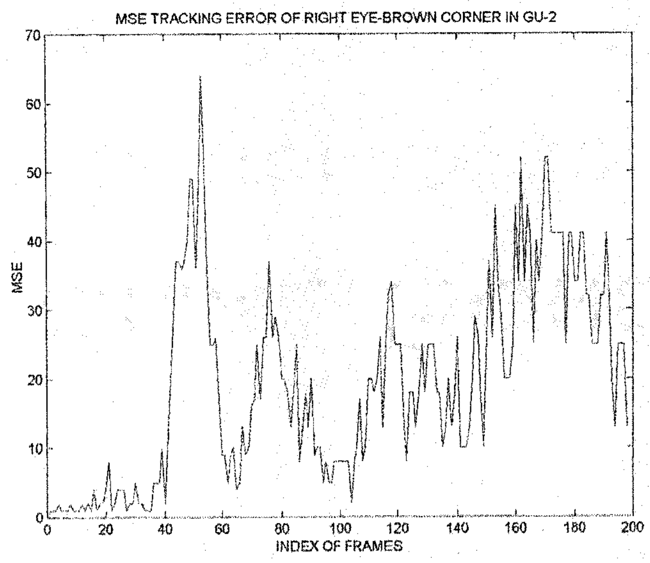


Figure 5.6 Squared intensity differences between the manually selected results and the tracking results in both x and y coordinates

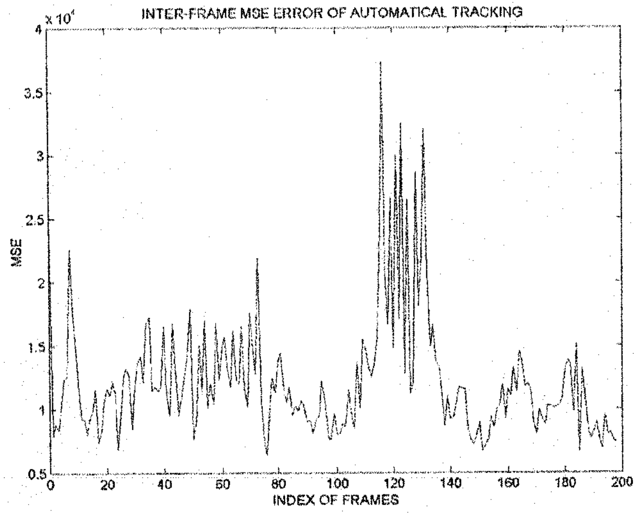


Figure 5.7 Squared intensity differences between the matched neighbor frames by using tracking method (Note that the unit of error is $\times 10^4$)

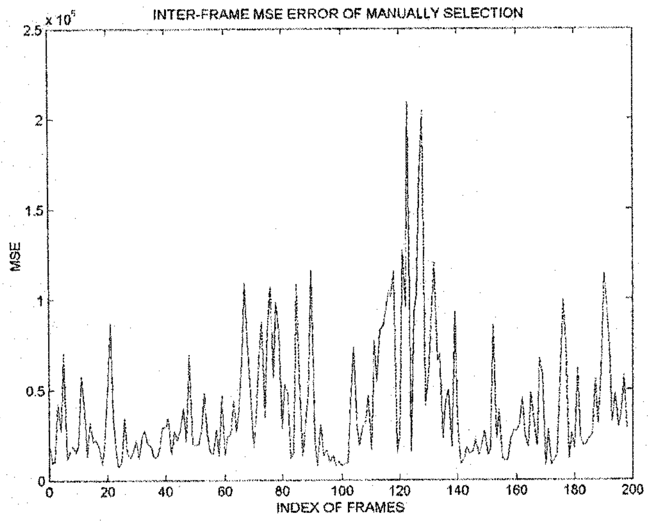
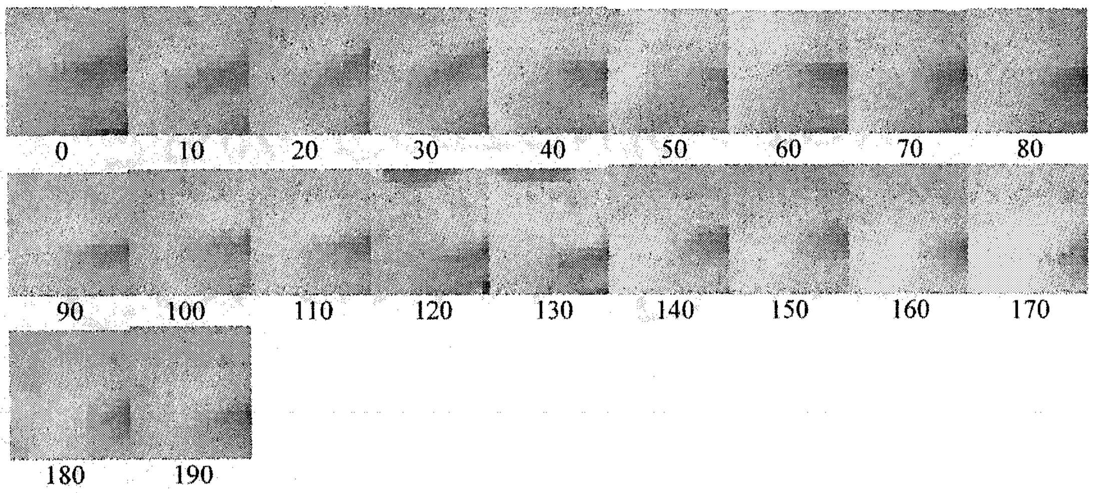
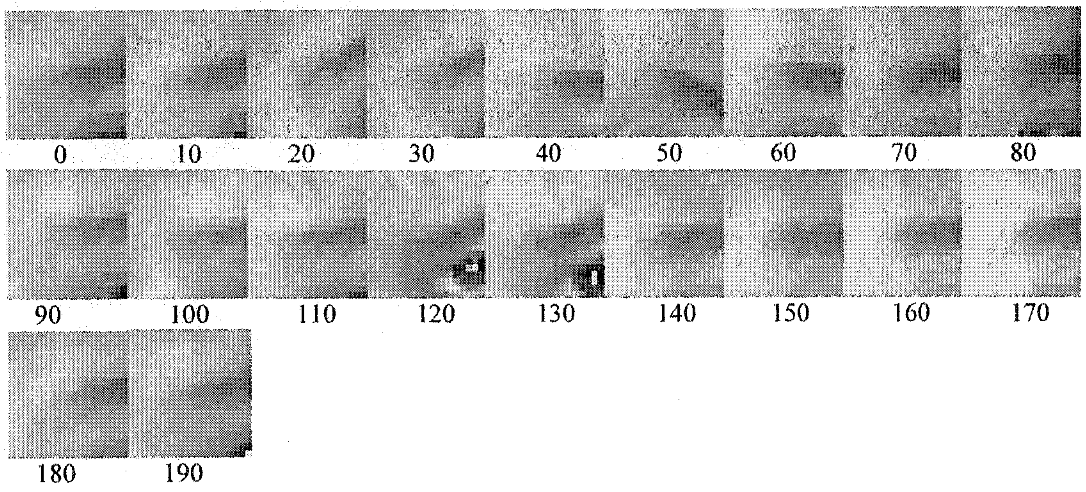


Figure 5.8 Squared intensity differences between the matched neighbor frames by using the manual selecting method (Note that the unit of error is $\times 10^5$)



(a) The matched blocks in the face sequence by using block matching method



(b) The matched blocks by using manually selecting method

Figure 5.9 The matched blocks for tracking the corner of the left eyebrow in the face sequence. The numbers under the images are the indices of the frames.

CHAPTER VI

FACE RIGID MOTION ESTIMATION

In a model-based videoconference system, in order to adapt the face model to the same pose as the real face in the image sequences, motion estimation is needed.

Motion estimation in the context of model-based image coding has been treated by many researchers (Li *et al*, 1993; Azarbayejani and Pentland, 1997; Jebara and Pentland, 1997; Smolie *et al*, 1999). Using the Extended Kalman Filter (EKF) for recursive estimation on image sequences is a primary focus in these works.

In some papers (Azarbayejani and Pentland, 1997; Jebara and Pentland, 1997), the EKF is used to provide a convenient mechanism for fusing all the information about the reliability of the measurements into an estimate of the 3-D structure, the pose, and the focal length of the camera. A state vector with the motion parameters, focal length, and depth information for all the tracking points is used in a standard EKF implementation. The measurement vector contains the image locations of all the tracked points in a new frame. This methodology is called Structure From Motion (SFM).

Smolic *et al.* (1999) propose two recursive methods for the real-time estimation of the long-term 3-D motion parameters, using the depth information from a modified face model. A simpler state vector with only five motion parameters is used in the EKF and the predictive least-square algorithm. The measurement vector is the same as that of SFM. The two algorithms proposed by Smolic *et al* are called PLS and PLS-Kalman, respectively.

In our project, we implement the three algorithms mentioned above to compare the results.

6.1 Rigid motion estimation using depth information

In this section, we will introduce a predictive least-square algorithm (PLS) and a PLS-Kalman algorithm for real-time motion estimation. We assume that the moving 3-D object in the real world is mapped onto a 2-D projection. Measurements in the form of feature points are made in the 2-D image plane. Based upon the 2-D observation, an object model, a motion model, and a camera projection model, the 3-D motion parameters are estimated with the PLS or PLS-Kalman algorithm. The scheme of the model-based coding system for videoconference using the PLS and PLS-Kalman is described in Figure 6.1.

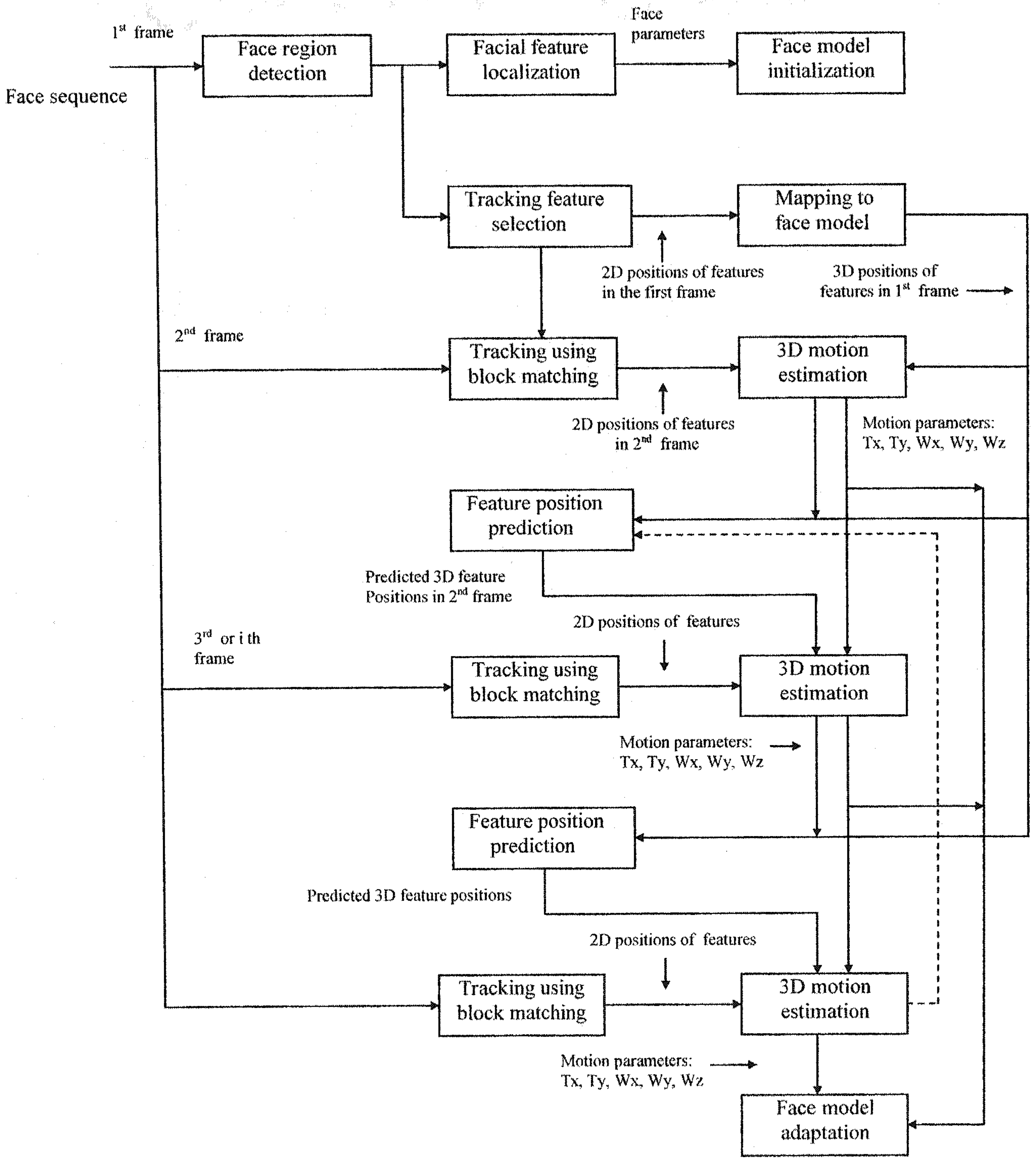

1st frame
Face sequence
Face region detection
Facial feature localization
Face parameters
Face model initialization
Tracking feature selection
Mapping to face model
2D positions of features in the first frame
3D positions of features in 1st frame
2nd frame
Tracking using block matching
3D motion estimation
2D positions of features in 2nd frame
Motion parameters: Tx, Ty, Wx, Wy, Wz
Feature position prediction
Predicted 3D feature Positions in 2nd frame
2D positions of features
3rd or i th frame
Tracking using block matching
3D motion estimation
Motion parameters: Tx, Ty, Wx, Wy, Wz
Feature position prediction
Predicted 3D feature positions
2D positions of features
Tracking using block matching
3D motion estimation
Motion parameters: Tx, Ty, Wx, Wy, Wz
Face model adaptation

Figure 6.1 Motion estimation using PLS and PLS-Kalman

6.1.1 Depth initialization

The estimation of 3-D motion parameters from monocular image sequences requires knowledge about the 3-D structure of the observed object. This information can be obtained if a generic face model is adapted to the face in the first frame. The method used is the same as that of texture mapping introduced in Chapter IV. We first separate the face region into eighteen smaller regions. Then, the selected tracking points in the image are mapped to the face model. Lastly, all the tracking points in the first image of the sequence can be matched approximately to the closest node on the face model by using the relative positions between this point and the control points on the face. The depth values of these matched nodes are used to represent the depth values of those points in the image.

6.1.2 Motion model

For the estimation of the global motion parameters, the person's face is regarded approximately as a rigid body. The motion of a rigid body has six degrees of freedom and can be described with six parameters

$$\vec{P}' = \vec{r} \cdot \vec{P} + \vec{t} \qquad (6-1)$$

where $\vec{P} = (x, y, z)^T$ and $\vec{P}' = (x', y', z')^T$ denote the 3D positions of a point on the rigid body before and after displacement, respectively. The translational part of the motion is represented by the vector $\vec{t} = (t_x, t_y, t_z)^T$. The matrix $\vec{r} = \vec{f}(\omega_x, \omega_y, \omega_z)$ describes the rotational part of the motion and is a nonlinear function of the rotational parameters $\omega_x, \omega_y, \omega_z$, which represent the rotation angles around the corresponding axes relative to the reference

$$\vec{r} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \qquad (6-2)$$

with

$$\left.\begin{aligned}
r_{11} &= \cos\omega_z \cos\omega_y \\
r_{12} &= \sin\omega_z \cos\omega_y \\
r_{13} &= -\sin\omega_y \\
r_{21} &= -\sin\omega_z \cos\omega_x + \cos\omega_z \sin\omega_y \sin\omega_x \\
r_{22} &= \cos\omega_z \cos\omega_x + \sin\omega_z \sin\omega_y \sin\omega_x \\
r_{23} &= \cos\omega_y \sin\omega_x \\
r_{31} &= \sin\omega_z \sin\omega_x + \cos\omega_z \sin\omega_y \cos\omega_x \\
r_{32} &= -\cos\omega_z \sin\omega_x + \sin\omega_z \sin\omega_y \cos\omega_x \\
r_{33} &= \cos\omega_y \cos\omega_x
\end{aligned}\right\} \qquad (6-3)$$

Assuming an orthographic camera projection, (6-1) can be written as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \qquad (6-4)$$

In this case, the 3D coordinate x, x', y, and y' are equal to the image coordinates, but the depth translation t_z cannot be recovered. If there is significant depth translation in the sequence, the results will be poor. The values (x', y') are measured in every frame for a set of tracking points $p_1 \dots p_m$. The tracking points $\vec{p} = (x, y, z)^T$ are measured in the first image. The depth of each tracking point z is obtained from the adapted face model as explained before.

We define a state vector, $\vec{\Xi}$, combining the directly unobservable 3-D motion parameters, which have to be estimated

$$\vec{\Xi} = (t_x, t_y, \omega_x, \omega_y, \omega_z)^T \qquad (6-5)$$

Equation (6-6) describes the measurement process, and denotes the dependencies of the measured quantities from unobservable state variables

$$\vec{\Psi}_n = \vec{C}(\Xi_n) + \vec{\varepsilon}_n \qquad (6-6)$$

The measurement vector $\vec{\Psi}_n$ consists of the measured coordinates of the tracking points (x', y'). The operation $\vec{C}(\ )$ describes the relationship between the measurement $\vec{\Psi}_n$

and the inaccessible state vector $\vec{\Xi}_n$. It is a combination of (6-4) for all tracking points. The vector $\vec{\varepsilon}_n$ denotes the measurement error at time n. The statistics of this vector are described by a covariance matrix $\vec{R}$. Since the operation $\vec{C}(\ \ )$ is nonlinear, (6-6) cannot be solved by a simple linear algorithm.

6.1.3 Predictive least-square algorithm (PLS)

The estimated parameters at time n, $\hat{\Xi}_n$, are calculated as the sum of a predicted value $\hat{\Xi}_{n-1}$, the estimated parameter vector at time $n-1$, and an update term $\Delta\hat{\Xi}_n$

$$\hat{\Xi}_n = \hat{\Xi}_{n-1} + \Delta\hat{\Xi}_n \qquad (6-7)$$

From the predicted parameters $\hat{\Xi}_{n-1}$, we calculate predicted control points $\hat{\Psi}_{n-1}$ by applying (6-4) to every control point of the reference frame

$$\hat{\Psi}_{n-1} = \vec{C}(\hat{\Xi}_{n-1}) \qquad (6-8)$$

Using the nonlinear operation $\vec{C}(\ \ )$ for prediction implies the utilization of a full nonlinear motion model of a rigid body. Thus, arbitrary global rotations can be represented in our approach. Next, we assume that the amount of rotation between the measured and the predicted control points is small, and we thus use the linearized equation from (6-4) for the estimation of the update term, which describes the motion between $\bar{\Psi}_n$ and $\hat{\Psi}_{n-1}$

$$\begin{pmatrix} x' - \hat{x} \\ y' - \hat{y} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -\hat{z} & \hat{y} \\ 0 & 1 & \hat{z} & 0 & -\hat{x} \end{pmatrix} \cdot \begin{pmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\omega_x \\ \Delta\omega_y \\ \Delta\omega_z \end{pmatrix} \qquad (6-9)$$

with certain predicted control points $\hat{p} = (\hat{x}, \hat{y}, \hat{z})^T$. This equation is obtained from (6-4) by assuming $\sin\omega \cong \omega$ and $\cos\omega \cong 1$ because of small differences between consecutive frames. The approximation at $\hat{p}$ is a linearization around the last estimated value $\hat{\Xi}_{n-1}$, enabling long-term estimation of arbitrary global rotation. Equation (6-9) can be combined for all control points into the compact form

$$\bar{\Psi}_n - \hat{\Psi}_{n-1} = D \cdot \Delta\hat{\Xi}_n \qquad (6-10)$$

with

$$D = \begin{pmatrix} 1 & 0 & 0 & -\hat{z} & \hat{y}_1 \\ 0 & 1 & \hat{z}_1 & 0 & -\hat{x}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & -\hat{z}_m & \hat{y}_m \\ 0 & 1 & \hat{z}_m & 0 & -\hat{x}_m \end{pmatrix} \qquad (6-11)$$

This can be solved with a standard least-square algorithm

$$\Delta\hat{\Xi}_n = (D^T \cdot D)^{-1} \cdot D^T \cdot (\vec{\Psi}_n - \Psi_{n-1}) \qquad (6-12)$$

This prediction and correction technique locks the estimation to the reference frame through a closed loop, and thus avoids instability and drift due to error accumulation that would appear if frame-to-frame parameters were simply accumulated in an open loop.

6.1.4 Using Extended Kalman Filtering (PLS-Kalman)

6.1.4.1 The dynamical system

The first central equation within a Kalman filter framework describes the model of the system dynamics, that is the propagation of the state vector over time

$$\vec{\Xi}_n = \vec{\Xi}_{n-1} + \vec{\omega}_n \qquad (6-13)$$

In general, this equation consists of a deterministic part and a stochastic part. The deterministic part can be derived from known dependencies between the state variables. In our formulation, it is simply an identity transform. The stochastic input to the observed system is modeled by the random vector $\vec{\omega}_n$. The statistics of this vector are described by a covariance matrix $\vec{Q}$.

The second central equation is the measurement process introduced in (6-6), which describes the dependencies of the measured quantities from the unobservable state variables as mentioned before. The statistics of the stochastic measurement error $\vec{\varepsilon}_n$ are modeled by its covariance matrix $\vec{R}$. The measured error is mainly influenced by the overlaid local motion and inaccuracies in the tracking.

Following the theory of extended Kalman filters and based upon the system dynamics in (6-13) and the measurements process (6-6), the optimum estimation $\hat{\vec{\Xi}}_n$ of the unobservable state vector $\vec{\Xi}_n$ is given at each step by

$$\hat{\vec{\Xi}}_n = \hat{\vec{\Xi}}_{n-1} + \vec{K}_n \cdot \left[\vec{\Psi}_n - \vec{C}(\hat{\vec{\Xi}}_{n-1})\right] \qquad (6-14)$$

This equation can be viewed as the sum of a predicted term and a correction term defining a recursive structure for the estimation process. The predicted term is given in our case by the estimated state vector at the previous time step. This corresponds to the deterministic part of the system. The correction term is calculated in our formulation as the difference between the current measurement vector $\vec{\Xi}_n$ and a predicted measurement vector $\vec{C}(\hat{\Xi}_{n-1})$ weighted by the current Kalman-gain matrix $\vec{K}_n$. This corresponds to the stochastic part of the system, which cannot be predicted and therefore must be updated at each time step through the following procedure:

$$\vec{\Lambda}^*_{n+1} = \vec{\Lambda}_n + \vec{Q} \qquad (6-15)$$
$$K_n = \vec{\Lambda}^*_n \cdot \vec{H}^T_n \cdot (\vec{H}_n \cdot \vec{\Lambda}^*_n \cdot \vec{H}^T_n + \vec{R})^{-1} \qquad (6-16)$$
$$\vec{\Lambda}_n = \vec{\Lambda}^*_n - \vec{K}_n \cdot \vec{H}_n \cdot \vec{\Lambda}^*_n \qquad (6-17)$$

The matrices $\vec{\Lambda}_n$ and $\vec{\Lambda}^*_n$ can be interpreted as the covariance matrices of the estimation error and the extrapolation error, respectively. They are only needed inside the calculation of the Kalman-gain-matrix. Since we know the initial value of the state vector exactly ($\vec{\Xi} = \vec{0}$), we initiate the system with $\vec{\Lambda}_n = \vec{0}$. The matrix $\vec{H}_n$ above is a linearization of the nonlinear operation in the measurement equation

$$\vec{H}_n = \frac{\partial \vec{C}(\vec{\Xi}_n)}{\partial \vec{\Xi}_n}\bigg|_{\vec{\Xi}_n = \hat{\vec{\Xi}}_{n-1}} \qquad (6-18)$$

Within an extended Kalman filter framework, the linearization is done around the predicted value of the state vector, which in our case is the last estimated $\hat{\vec{\Xi}}_{n-1}$. We call the linearization matrix of $\vec{H}_n$ the measurement Jacobian matrix.

6.1.4.2 Measurement Jacobian of PLS-Kalman

According to Equation (6-4), the nonlinear operation in the measurement process is described by:

$$\hat{\Psi}_n = \vec{C}(\Xi_n) = \begin{bmatrix} C_{x,0}(\vec{\Xi}_n) \\ C_{y,0}(\vec{\Xi}_n) \\ C_{x,1}(\vec{\Xi}_n) \\ C_{y,1}(\vec{\Xi}_n) \\ \vdots \\ C_{x,m-1}(\vec{\Xi}_n) \\ C_{y,m-1}(\vec{\Xi}_n) \end{bmatrix} = \begin{bmatrix} r_{11}x_0 + r_{12}y_0 + r_{13}z_0 + t_x \\ r_{21}x_0 + r_{22}y_0 + r_{23}z_0 + t_y \\ r_{11}x_1 + r_{12}y_1 + r_{13}z_1 + t_x \\ r_{21}x_1 + r_{22}y_1 + r_{23}z_1 + t_y \\ \vdots \\ r_{11}x_{m-1} + r_{12}y_{m-1} + r_{13}z_{m-1} + t_x \\ r_{21}x_{m-1} + r_{22}y_{m-1} + r_{23}z_{m-1} + t_y \end{bmatrix} \qquad (6-19)$$

where m is the number of the tracking points, (x_k, y_k, z_k) are the 3-D coordinates of the tracking points, k is the index of the tracking point and $0 \le k < m$. $C_{x,k}(\vec{\Xi}_n)$ and $C_{y,k}(\vec{\Xi}_n)$ are the even and odd rows of the nonlinear matrix $C(\vec{\Xi}_n)$. The measurement Jacobian matrix is calculated as folllows.

$$\begin{cases} \vec{H}_{n,x,k} = \dfrac{\partial \vec{C}_{x,k}(\vec{\Xi}_n)}{\partial \vec{\Xi}_n} = \begin{bmatrix} \dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_x} & \dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_y} & \dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_x} & \dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_y} & \dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_z} \end{bmatrix} \\ \vec{H}_{n,y,k} = \dfrac{\partial \vec{C}_{y,k}(\vec{\Xi}_n)}{\partial \vec{\Xi}_n} = \begin{bmatrix} \dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_x} & \dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_y} & \dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_x} & \dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_y} & \dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_z} \end{bmatrix} \end{cases} \qquad (6-20)$$

where $\vec{H}_{n,x,k}$ and $\vec{H}_{n,y,k}$ are the even and odd rows in $\vec{H}_n$ respectively. The elements of $\vec{H}_{n,x,k}$ and $\vec{H}_{n,y,k}$ can be calculated using equation (6-3) and (6-4) and the results are shown below:

$$\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_x} = 1$$

$$\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_y} = 0$$

$$\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_x} = 0$$

$$\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_y} = (-\cos\omega_z \sin\omega_y)x_k + (-\sin\omega_z \sin\omega_y)y_k + (-\cos\omega_y)z_k$$

$$\frac{\partial C_x(\vec{\Xi}_n)}{\partial \omega_z} = (-\sin\omega_z \cos\omega_y)x_k + (\cos\omega_z \cos\omega_y)y_k$$

$$\frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_x} = 0$$

$$\frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_y} = 1$$

$$\begin{aligned}\frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_x} = {} & (\sin\omega_z \sin\omega_x + \cos\omega_z \sin\omega_y \cos\omega_x)x_k + (-\cos\omega_z \sin\omega_x \\ & + \sin\omega_z \sin\omega_y \cos\omega_x)y_k + (\cos\omega_y \cos\omega_x)z_k\end{aligned}$$

$$\begin{aligned}\frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_y} = {} & (\cos\omega_z \cos\omega_y \sin\omega_x)x_k + (\sin\omega_z \cos\omega_y \sin\omega_x)y_k \\ & + (-\sin\omega_y \sin\omega_x)z_k\end{aligned}$$

$$\frac{\partial C_{y,k}(\bar{\Xi}_n)}{\partial \omega_z} = (-\cos\omega_z \cos\omega_x - \sin\omega_z \sin\omega_y \sin\omega_x)x_k + (-\sin\omega_z \cos\omega_x + \cos\omega_z \sin\omega_y \sin\omega_x)y_k \qquad (6-21)$$

In this algorithm, all the parameters in the state vector are the global motion parameters, which describe the motion of the current frame reference at the first frame of the sequence. Therefore, the input point coordinates (x_k, y_k, z_k) are those of the points in the first frame.

6.2 Rigid motion estimation without depth information (SFM)

Recovering scene structure and camera geometry from image sequences of rigid motion has been an important topic in computer vision and has been approached in many different ways. We call this entire class of motion-based estimation research as the structure from motion (SFM) approach. It should be clarified that SFM usually means recovery of motion and structure, and focal length as well. Azarbayejani and Pentland presented, in 1995, a formulation for recursive recovery of motion, pointwise structure and focal length from point correspondences tracked through an image sequence. In addition to adding focal length to the state vector, several representational improvements are made over earlier structure from motion formulations, yielding a stable and accurate estimation framework, which applies uniformly to both true perspective and orthographic projection.

The SFM algorithm introduced here also uses the EKF as the frame of the dynamical system. The differences with the algorithm introduced above are in the motion model, state vector selection. In the following, we review and discuss the formulation proposed by Azarbayejani and Pentland for a recursive recovery of 3-D structure, 3-D motion, and camera geometry from point correspondences over an image sequence.

The scheme of the system using SFM is described in Figure 6.2.

6.2.1 The dynamical system

The dynamical system used, as proposed by Azarbayejani and Pentland, is the same as the EKF algorithm introduced in the PLS-Kalman algorithm:

$$\vec{\Xi}_n = \vec{\Xi}_{n-1} + \vec{\omega}_n \qquad (6-22)$$

$$\vec{\Psi}_n = \vec{C}(\vec{\Xi}_n) + \vec{\varepsilon}_n \qquad (6-23)$$

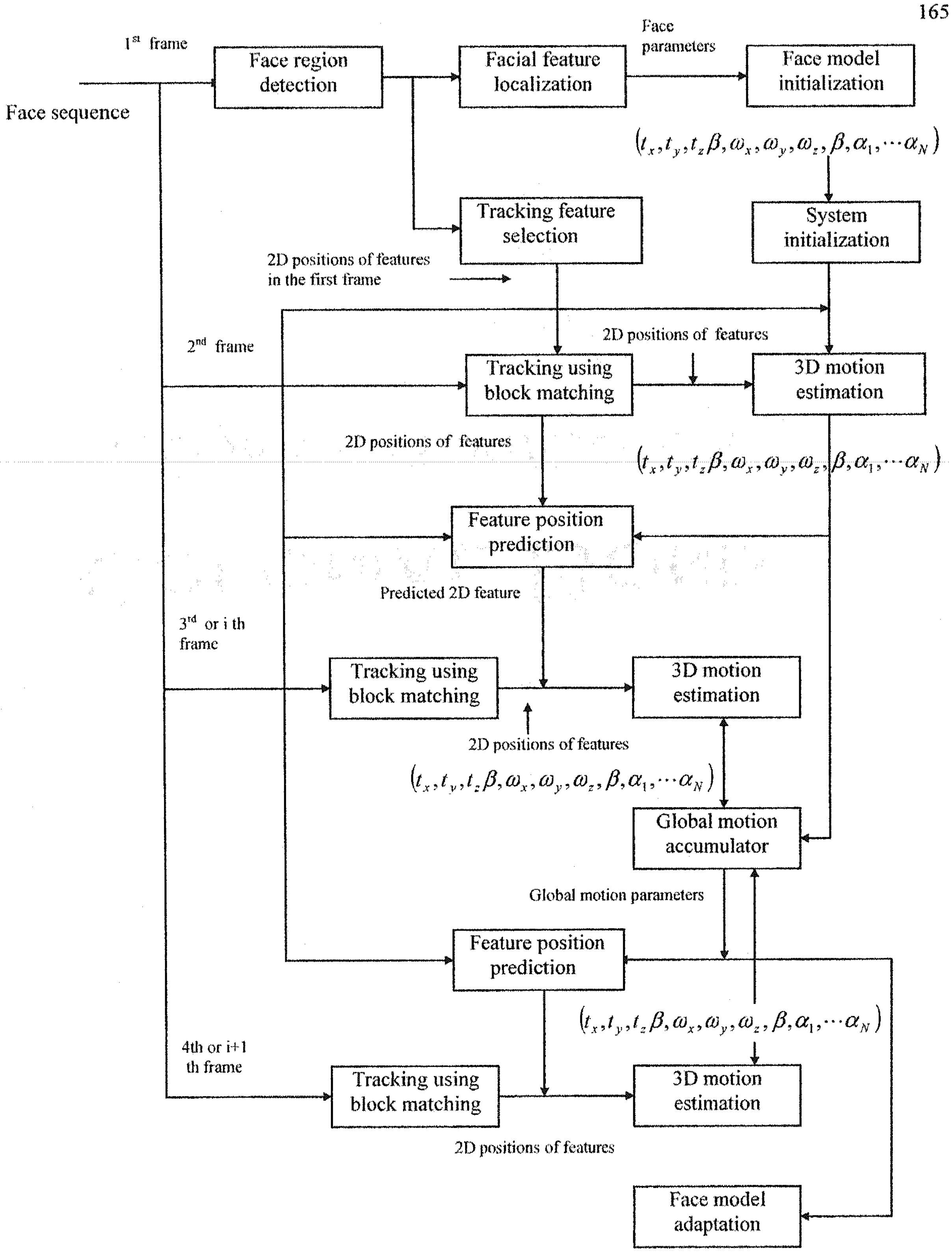

1st frame
Face sequence
Face region detection
Facial feature localization
Face parameters
Face model initialization
$(t_x, t_y, t_z\beta, \omega_x, \omega_y, \omega_z, \beta, \alpha_1, \cdots \alpha_N)$
Tracking feature selection
System initialization
2D positions of features in the first frame
2nd frame
2D positions of features
Tracking using block matching
3D motion estimation
2D positions of features
$(t_x, t_y, t_z\beta, \omega_x, \omega_y, \omega_z, \beta, \alpha_1, \cdots \alpha_N)$
Feature position prediction
Predicted 2D feature
3rd or i th frame
Tracking using block matching
3D motion estimation
2D positions of features
$(t_x, t_y, t_z\beta, \omega_x, \omega_y, \omega_z, \beta, \alpha_1, \cdots \alpha_N)$
Global motion accumulator
Global motion parameters
Feature position prediction
$(t_x, t_y, t_z\beta, \omega_x, \omega_y, \omega_z, \beta, \alpha_1, \cdots \alpha_N)$
4th or i+1 th frame
Tracking using block matching
3D motion estimation
2D positions of features
Face model adaptation

Figure 6.2 Motion estimation using SFM

Here in (6-23), the measurements are the 2-D points (in x, y coordinates) which are concatenated into an observation vector $\vec{\Psi}_n$ for each moment in time. The measurements are obtained by the internal state of the system, $\vec{\Xi}_n$ which contains the scene's 3-D structure, the relative 3-D motion between the camera as well as the scene and the camera's internal geometry. The mapping $\vec{\Xi}_n$ from $\vec{\Psi}_n$ is also nonlinear and is corrupted by noise. Here, the noise is represented as an additive Gaussian (normal) process with zero-mean and time-varying covariance $\vec{R}$.

In (6-22), the 3-D structure, 3-D motion and camera geometry do not vary wildly but are linearly dependent upon their previous values with Gaussian noise added. The noise process is additive with zero-mean and covariance $\vec{Q}$. Just as before, we use the previous value to predict the current value.

6.2.2 Internal Geometry and Projection Model

In standard perspective projection, the mapping from a 3-D coordinate onto the image plane is accomplished via the projection equation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_C \\ Y_C \end{pmatrix} \frac{f}{Z_C} \qquad (6-24)$$

However, we instead use the central projection representation as depicted in Figure 6.3. Here, the coordinate system's origin is fixed at the image plane instead of at the center of projection (COP). In addition, the focal length is parameterized by its inverse. The projection equation thus becomes Equation (6-25).

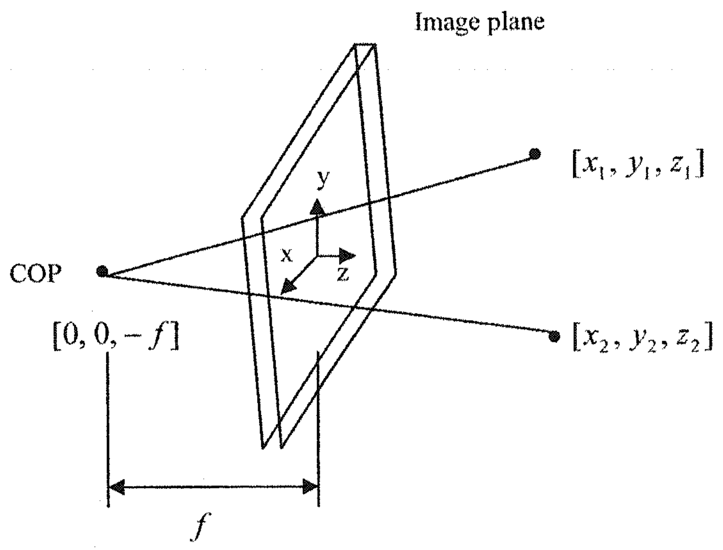


Figure 6.3 Central projection

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_C \\ Y_C \end{pmatrix} \frac{1}{1+Z_C\beta} \qquad (6-25)$$

Note how this projection decouples the camera focal length (f) from the depth of the point (Z_C). In the traditional projection Equation (6-24), if Z_C is fixed and the f is altered, the imaging geometry remains the same while the scale of the image changes. In other words, the cone of perspective rays remains fixed while the focal plane translates along the optical (Z) axis. We note that in the standard projection model, the imaging geometry (*i.e.* the perspective rays) are only altered by varying depth Z_C. Thus, f only acts as a scaling factor and the imaging geometry and the depth are encoded in Z_C.

In Equation (6-25), however, the inverse focal length β alters the imaging geometry independently of the depth value Z_C. State variable decoupling is known to be critical in Kalman filtering frameworks and is applicable here since we plan to put both camera internal geometry β and structure Z_C into the internal hidden state .

Another critical property of β, as opposed to f, is that it does not exhibit numerical ill-conditioning. It can span the wide range of perspective projection but also the special case of orthographic projection which occurs when we set the focal length $f = \infty$ and all rays project orthogonally onto the image plane. However, under orthographic projection, $\beta = 0$ maintains numerical stability in EKF frameworks. We can thus combine both perspective and orthographic projection into the same so-called central projection framework without any numerical instabilities (this is demonstrated experimentally in the next section).

6.2.3 Structure model

We assume that N feature points are to be tracked over F frames in an image sequence. In the first frame, each ``feature point" is initially in terms of an image location (x, y). Subsequent frames are then observed and the image location of the point is measured with some noisy zero-mean Gaussian error. One may think of the 3-D structure of the model as the true (X_C, Y_C, Y_C) coordinates of each of the 3-D points which then project into (x, y) 2-D coordinates. However, this obvious parameterization is not compact and stable. For one thing, it contains 3 unknowns (or 3 degrees of freedom) per point being tracked resulting in the concatenation of **3 x** N dimensions to our internal state vector **x**. Another problem with this representation is that it in no way encodes the rigidity of the object being tracked.

A more compact representation of the 3-D location is shown in Equation (6-26) below. Here, there is only one degree of freedom per point, α. The first term represents the initial image location and the second term represents the perspective ray scaled by the unknown depth α:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} x\beta \\ y\beta \\ 1 \end{pmatrix} \qquad (6-26)$$

Point-wise structure, therefore, can be represented with one parameter per point (instead of 3).

In our formulation, constraints (1+2N) outnumber degrees of freedom (6+1+N) for motion, camera, and structure at every frame when N>7. The more measurements available, the larger the gap.

6.2.4 3-D motion model

6.2.4.1 Translation model

The translational motion is represented as the 3-D location of the object reference frame relative to the current camera reference frame using the vector $t=(t_x,t_y,t_z)$. The t_x and t_y components correspond to directions parallel to the image plane, while the t_z component corresponds to the depth of the object along the optical axis. As such, the sensitivity of image plane motion to t_x and to t_y motion will be similar, while the sensitivity to t_z motion will differ, to an extent dependent upon the focal length of the imaging geometry.

For typical focal lengths, even in the case of ``wide angle" lenses, there is already much less sensitivity to t_z motion than there is to (t_x,t_x) motions. For longer focal lengths the sensitivity decreases until, in the limiting orthographic case, there is zero image plane sensitivity to t_z motion. For this reason, t_z cannot be represented explicitly in our estimation process. Instead, the product of t_z and β is estimated. The coordinate frame transformation equation

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C\beta \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ t_z\beta \end{pmatrix} + \begin{pmatrix} 1 & & \\ & 1 & \\ & & \beta \end{pmatrix} \bar{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad (6-27)$$

combined with Equation (6-25) demonstrates that only $t_z\beta$ is actually required to generate an equation for the image plane measurements (x, y) as a function of the motion, structure, and camera parameters (rotation is discussed below).

Furthermore, the sensitivity of $t_z\beta$ does not degenerate at long focal lengths as does t_z. For example, the sensitivities of the x image coordinate to both t_z and $t_z\beta$ are

$$\frac{\partial x}{\partial t_z} = \frac{-X_C\beta}{(1+Z_C\beta)^2} \qquad \frac{\partial x}{\partial (t_z\beta)} = \frac{-X_C}{(1+Z_C\beta)^2} \qquad (6-28)$$

demonstrating that $t_z\beta$ remains observable from the measurements and is therefore estimable for long focal lengths, while t_z is not (β approaches zero for long focal lengths). Thus we parameterize translation with the vector $(translation) = (t_x, t_y, t_z\beta)$.

6.2.4.2 Rotation model

The 3-D rotation is defined as the relative rotation between the object reference frame and the current camera reference frame. This is represented using a unit quaternion, from which the rotation matrix ($\vec{R}$) can be generated:

$$\begin{pmatrix} q_0^2+q_1^2+q_2^2+q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix} \qquad (6-29)$$

The four elements of the unit quaternion only have three degrees of freedom due to the normality constraint. Thus, all four cannot be estimated independently; only a nonlinear constrained minimization will work to recover the quaternion directly. Since the EKF uses a linearization at each step, the nonlinear normality constraint cannot be rigidly enforced within the EKF computational structure.

However, a 3-parameter incremental rotation representation can be used in the EKF to estimate interframe rotation at each frame. Incremental Euler angles centered around zero (or discrete-time ``rotational velocity") do not overparameterize rotation and are approximately independent and therefore can be used reliably in a system linearization. The incremental rotation quaternion is a function of these three parameters:

$$\delta_q = \left(\sqrt{1-\varepsilon}, \frac{\omega_x}{2}, \frac{\omega_y}{2}, \frac{\omega_z}{2} \right) \qquad (6-30)$$

$$\varepsilon = {(\omega_x^2 + \omega_y^2 + \omega_z^2)}/{4} \qquad (6-31)$$

This incremental rotation can be computed at each frame and then composed with an external rotation quaternion to maintain an estimate of the global rotation. The global quaternion is then used in the linearization process at the next frame. Thus, we obtain the interframe rotation $(\omega_x, \omega_y, \omega_z)$ and the global rotation (q_0, q_1, q_2, q_3), where the interframe rotation is part of the EKF state vector and the global rotation is maintained and used in the linearization at each step.

6.2.5 The issue of scale

It is well known that the shape and motion geometry in SFM problems are subject to arbitrary scaling and that the scale factor cannot be recovered. The imaging geometry β and the rotation are recoverable and are not subject to this scaling. In two-frame problems with no information about true lengths in the scene, the scale factor is usually set by fixing the length of the ``baseline" between the two cameras. This corresponds to the magnitude of the translational motion. A better approach to setting the scale is to fix a static parameter. Since we are dealing with rigid objects, all of the shape parameters $\{\alpha_i\}$ are static. Thus, fixing any one of these establishes a uniform scale for all motion and structure parameters over the entire sequence. The result is a well conditioned, stable representation. Setting scale is simple and elegant in the EKF; the initial variance on, say α_0 is set to zero, which will fix that parameter at its initial value. All other parameters then automatically scale themselves to accommodate this constraint. This behavior can be observed in the experimental results.

6.2.6 The EKF implementation

Using the representations discussed so far, our composite state vector consists of 7+N parameters: 6 for motion, 1 for camera geometry, and N for structure, where N is the number of points tracked: $\vec{\Xi} = \left(t_x, t_y, t_z \beta, \omega_x, \omega_y, \omega_z, \beta, \alpha_1, \cdots \alpha_N\right)$

The vector $\vec{\Xi}$ is the state vector used in a standard EKF implementation, where the measurement vector contains the image locations of all the tracked points in a new frame. As described earlier, an additional quaternion is required for maintaining a description of the global rotation external to the EKF.

The dynamics model in the EKF can be chosen trivially as an identity transform plus noise, unless additional prior information on dynamics is available. The measurement equation is simply obtained by combining Equations (6-25), (6-26), and (6-27). The final implementation of the EKF is straightforward with the only additional computation being the quaternion maintenance.

6.2.7 Measurement Jacobian of SFM

We assume that the coordinates of the k^{th} tracking point in the reference frame are (u_k, v_k), the 3-D coordinates of that point are (X_k, Y_k, Z_k), the coordinates of that point in the current frame are $(u_k^{'}, v_k^{'})$, and the 3-D coordinates of that point in the current frame are $(X_{C,k}, Y_{C,k}, Z_{C,k})$. The nonlinear matrix $C(\vec{\Xi}_n)$ is obtained by using Equations (6-25), (6-26), and (6-27).

$$\begin{cases} X_k = u_k(1+\alpha\beta) \\ Y_k = v_k(1++\alpha\beta) \\ Z_k = \alpha \end{cases} \qquad (6-32)$$

$$\begin{pmatrix} X_{C,k} \\ Y_{C,k} \\ Z_{C,k}\beta \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ t_z\beta \end{pmatrix} + \begin{pmatrix} 1 & & \\ & 1 & \\ & & \beta \end{pmatrix} \vec{R} \begin{pmatrix} X_k \\ Y_k \\ Z_k \end{pmatrix} \qquad (6-33)$$

$$\begin{cases} u_k^{'} = \dfrac{X_{C,k}}{1+Z_{C,k}\beta} \\ u_k^{'} = \dfrac{Y_{C,k}}{1+Z_{C,k}\beta} \end{cases} \qquad (6-34)$$

A relation between $(u_k^{'}, v_k^{'})$ and (u_k, v_k) is set up using the above equations. We can use these three equations to calculate the measurement Jacobian matrix $\vec{H}_n$.

The elements of the measurement Jacobian matrix are shown in the following:

$$\begin{cases} \vec{H}_{n,x,k} = \dfrac{\partial \vec{C}_{x,k}(\vec{\Xi}_n)}{\partial \vec{\Xi}_n} = \begin{bmatrix} \dfrac{\partial C_{x,k}}{\partial t_x} & \dfrac{\partial C_{x,k}}{\partial t_y} & \dfrac{\partial C_{x,k}}{\partial \beta t_z} & \dfrac{\partial C_{x,k}}{\partial \omega_x} & \dfrac{\partial C_{x,k}}{\partial \omega_y} & \dfrac{\partial C_{x,k}}{\partial \omega_z} & \dfrac{\partial C_{x,k}}{\partial \beta} & \dfrac{\partial C_{x,k}}{\partial \alpha_0} \cdots \dfrac{\partial C_{x,k}}{\partial \alpha_{m-1}} \end{bmatrix} \\ \vec{H}_{n,y,k} = \dfrac{\partial \vec{C}_{y,k}(\vec{\Xi}_n)}{\partial \vec{\Xi}_n} = \begin{bmatrix} \dfrac{\partial C_{y,k}}{\partial t_x} & \dfrac{\partial C_{y,k}}{\partial t_y} & \dfrac{\partial C_{y,k}}{\partial \beta t_z} & \dfrac{\partial C_{y,k}}{\partial \omega_x} & \dfrac{\partial C_{y,k}}{\partial \omega_y} & \dfrac{\partial C_{y,k}}{\partial \omega_z} & \dfrac{\partial C_{y,k}}{\partial \beta} & \dfrac{\partial C_{y,k}}{\partial \alpha_0} \cdots \dfrac{\partial C_{y,k}}{\partial \alpha_{m-1}} \end{bmatrix} \end{cases} \quad (6-35)$$

where $\vec{H}_{n,x,k}$ and $\vec{H}_{n,y,k}$ are the even and odd rows in $\vec{H}_n$ respectively. The elements of $\vec{H}_{n,x,k}$ and $\vec{H}_{n,y,k}$ can be calculated using Equations (6-32), (6-33), and (6-34).

We assume:

$$\begin{array}{lll} c_0 = 2(q_0 q_1 + q_2 q_3) & c_1 = 2(q_0 q_1 - q_2 q_3) & c_2 = 2(q_0 q_2 + q_1 q_3) \\ c_3 = 2(q_0 q_2 - q_1 q_3) & c_4 = 2(q_0 q_3 + q_1 q_2) & c_4 = 2(q_0 q_3 - q_1 q_2) \\ k_0 = q_0^2 + q_1^2 - q_2^2 - q_3^2 & k_0 = q_0^2 - q_1^2 + q_2^2 - q_3^2 & k_0 = q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{array} \quad (6-36)$$

Therefore, the rotation matrix $\vec{R}$ becomes

$$\vec{R} = \begin{bmatrix} k_2 & c_0 & c_3 \\ c_1 & k_1 & c_4 \\ c_2 & c_5 & k_0 \end{bmatrix} \quad (6-37)$$

The partial derivatives with respect to translation are as follows:

$$
\begin{aligned}
&\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_x}=\frac{1}{1+Z_{C,k}\beta} &\qquad& \frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_x}=0\\
&\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_y}=0 && \frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_y}=\frac{1}{1+Z_{C,k}\beta}\\
&\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_z\beta}=\frac{-X_{C,k}}{(1+Z_{C,k}\beta)^2} && \frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_z\beta}=\frac{-Y_{C,k}}{(1+Z_{C,k}\beta)^2}
\end{aligned}
\qquad (6-38)
$$

The partial derivatives with respect to rotation are as follows:

$$
\begin{cases}
\dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_j}=\dfrac{\dfrac{\partial X_{C,k}}{\partial \omega_j}}{1+Z_{C,k}\beta}+\dfrac{\partial Z_{C,k}\beta}{\partial \omega_j}\cdot\dfrac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial t_z\beta}\\
\dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \omega_j}=\dfrac{\dfrac{\partial Y_{C,k}}{\partial \omega_j}}{1+Z_{C,k}\beta}+\dfrac{\partial Z_{C,k}\beta}{\partial \omega_j}\cdot\dfrac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial t_z\beta}
\end{cases}
\qquad (6-39)
$$

where j stands for *x* or *y* or *z*, and

$$
\frac{\partial}{\omega_j}\begin{pmatrix} X_{C,k}\\ Y_{C,k}\\ Z_{C,k}\end{pmatrix}=\begin{pmatrix}1&&\\&1&\\&&\beta\end{pmatrix}\frac{\partial \vec{R}_{\delta k}}{\partial \omega_j}\vec{R}_{k-1}\begin{pmatrix}X\\Y\\Z\end{pmatrix}
\qquad (6-40)
$$

The partial derivatives of the incremental rotation matrix with respect to ω_j are as follows:

$$\frac{\partial \vec{R}_{\delta k}}{\partial \omega_x} = \begin{pmatrix} 0 & \frac{2\omega_y\gamma + \omega_x\omega_z}{4\gamma} & \frac{2\omega_z\gamma + \omega_x\omega_y}{4\gamma} \\ \frac{2\omega_y\gamma - \omega_x\omega_z}{4\gamma} & -\omega_x & \frac{\omega_x^2}{4\gamma} - \gamma \\ \frac{2\omega_z\gamma + \omega_x\omega_y}{4\gamma} & -\frac{\omega_x^2}{4\gamma} + \gamma & -\omega_x \end{pmatrix} \qquad (6-41)$$

$$\frac{\partial \vec{R}_{\delta k}}{\partial \omega_y} = \begin{pmatrix} -\omega_z & \frac{2\omega_x\gamma + \omega_y\omega_z}{4\gamma} & -\frac{\omega_y^2}{4\gamma} + \gamma \\ \frac{2\omega_x\gamma + \omega_y\omega_z}{4\gamma} & 0 & \frac{2\omega_z\gamma + \omega_x\omega_y}{4\gamma} \\ \frac{\omega_y^2}{4\gamma} - \gamma & \frac{2\omega_z\gamma + \omega_x\omega_{yz}}{4\gamma} & -\omega_y \end{pmatrix} \qquad (6-42)$$

$$\frac{\partial \vec{R}_{\delta k}}{\partial \omega_z} = \begin{pmatrix} -\omega_z & \frac{\omega_z^2}{4\gamma} - \gamma & \frac{2\omega_x\gamma + \omega_y\omega_z}{4\gamma} \\ -\frac{\omega_z^2}{4\gamma} + \gamma & -\omega_z & \frac{2\omega_y\gamma + \omega_x\omega_z}{4\gamma} \\ \frac{2\omega_x\gamma - \omega_y\omega_z}{4\gamma} & \frac{2\omega_y\gamma + \omega_x\omega_z}{4\gamma} & 0 \end{pmatrix} \qquad (6-43)$$

Here, $\gamma = \sqrt{1-\varepsilon}$ and ε is given in Equation (6-31). $\vec{R}_{\delta k}$ is the interframe rotation matrix and $\vec{R}_k$ is the global rotation matrix, *i.e.* the rotation of the face from the first frame to the current frame. $\vec{R}_k$ is updated every frame by multiplying the interframe and global rotation matrices. $\vec{R}_{k+1} = \vec{R}_k \vec{R}_{\delta k}$. Therefore, the final results are:

$$\left.\begin{aligned}
\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \omega_x} &= \frac{Y_k c_2 + Z_k c_5}{1+Z_{C,k}\beta} - \frac{X_{C,k}\beta(Y_k k_2 - Z_k c_0)}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_x(\vec{\Xi}_n)}{\partial \omega_y} &= \frac{-X_k c_2 + Z_k k_0}{1+Z_{C,k}\beta} - \frac{X_{C,k}\beta(-X_k k_2 - Z_k c_3)}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_x(\vec{\Xi}_n)}{\partial \omega_z} &= \frac{-X_k c_5 - Y_k k_0}{1+Z_{C,k}\beta} - \frac{X_{C,k}\beta(X_k c_0 - Y_k c_3)}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_y(\vec{\Xi}_n)}{\partial \omega_x} &= \frac{Y_k c_1 - Z_k k_1}{1+Z_{C,k}\beta} - \frac{Y_{C,k}\beta(Y_k k_2 - Z_k c_0)}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_y(\vec{\Xi}_n)}{\partial \omega_y} &= \frac{X_k c_1 + Z_k c_4}{1+Z_{C,k}\beta} - \frac{Y_{C,k}\beta(-X_k k_2 - Z_k c_3)}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_y(\vec{\Xi}_n)}{\partial \omega_z} &= \frac{X_k k_1 + Y_k c_4}{1+Z_{C,k}\beta} - \frac{Y_{C,k}\beta(X_k c_0 - Y_k c_3)}{(1+Z_{C,k}\beta)^2}
\end{aligned}\right\} \qquad (6-44)$$

Similarly, the partial derivatives with respect to β are:

$$\left.\begin{aligned}
\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \beta} &= \frac{\beta u_k}{1+Z_{C,k}\beta} - \frac{X_{C,k}}{(1+Z_{C,k}\beta)^2} \\
\frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \beta} &= \frac{\beta v_k}{1+Z_{C,k}\beta} - \frac{Y_{C,k}}{(1+Z_{C,k}\beta)^2}
\end{aligned}\right\} \qquad (6-45)$$

The partial derivatives with respect to α_k are:

$$\left.\begin{aligned}\frac{\partial C_{x,k}(\vec{\Xi}_n)}{\partial \alpha_k} &= \frac{k_0\beta u_k - c_5\beta v_k + c_2}{1+Z_{C,k}\beta} - \frac{X_{C,k}\beta\,(-c_3\beta u_k + c_0\beta v_k + k_2)}{(1+Z_{C,k}\beta)^2} \\ \frac{\partial C_{y,k}(\vec{\Xi}_n)}{\partial \alpha_k} &= \frac{c_4\beta u_k + k_1\beta v_k + c_1}{1+Z_{C,k}\beta} - \frac{Y_{C,k}\beta\,(-c_3\beta u_k + c_0\beta v_k + k_2)}{(1+Z_{C,k}\beta)^2}\end{aligned}\right\} \qquad (6-46)$$

6.3 Implementation, result and analysis

6.3.1 Initialization

6.3.1.1 Initialization of PLS-Kalman

Since we have no concrete *a priori* knowledge about the covariance matrices $\vec{Q}$ and $\vec{R}$, we make the assumption that all the vector components are uncorrelated to each other

$$\vec{Q} = \begin{bmatrix} \delta^2_{\omega,1} & & & & \\ & \delta^2_{\omega,2} & & & \\ & & \delta^2_{\omega,3} & & \\ & & & \delta^2_{\omega,4} & \\ & & & & \delta^2_{\omega,5} \end{bmatrix}$$

and

$$\vec{R} = \begin{bmatrix} \delta^2_{\varepsilon,1} & & & \\ & \delta^2_{\varepsilon,2} & & \\ & & \ddots & \\ & & & \delta^2_{\varepsilon,m} \end{bmatrix}$$

For the $\delta^2_{\varepsilon,i}$, we usually choose a constant value c because we assume that the deviation of the measurement error is the same for all points. In $\vec{Q}$, we usually set $\delta^2_{\omega,1} = \delta^2_{\omega,2} = a$ and $\delta^2_{\omega,3} = \delta^2_{\omega,4} = \delta^2_{\omega,5} = b$. This yields the same estimation sensitivity among the translational parameters and among the rotational parameters. Then the results only depend upon the ratios a/b and a/c. Smolie *et al* (1999) have found experimentally that for a significant change of the estimation results, the ratios have to be changed by a factor of at least 10.

In experiments, we set $\delta^2_{\varepsilon,i} = 0.1$ and $\delta^2_{\omega,i} = 0.01$.

6.3.1.2 Initialization of SFM

Because the state vector in SFM is more complicated than that of PLS-Kalman, it is difficult to find a rule to initialize the covariance values in the Kalman filter. In addition, the results of SFM are very sensitive to initialization. Therefore, we merely initialize it by trying. In the experiments, the covariance values in $\vec{Q}$ and $\vec{R}$ are set to 0.01.

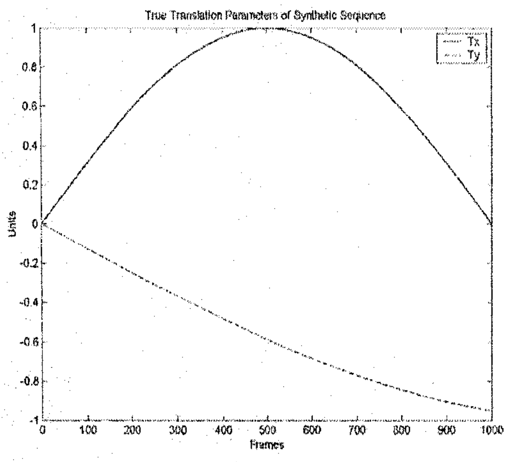
6.3.2 Synthetic experiments

In the synthetic experiments, we specify six reference points. Each of these points is located within [-1, 1] in x, y and z coordinates. The measurement points in 1000 frames are generated by applying (6-4) to the reference points with the generated motion parameters.

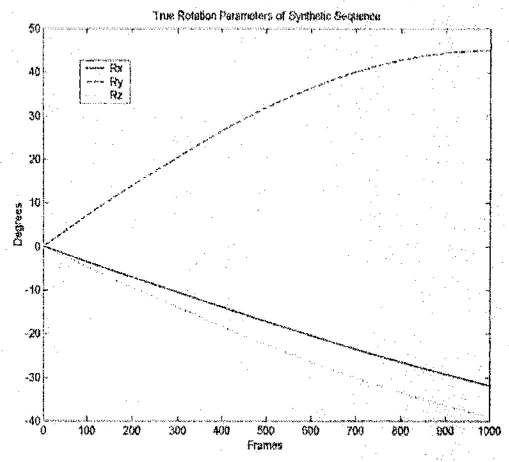
Figure 6.4 shows the motion parameters that we use to generate the synthetic sequences. Figure 6.5, Figure 6.7 and Figure 6.9 are motion parameters estimated by using PLS, PLS-Kalman and SFM, respectively. In order to show the differences between Figure 6.4 and Figure 6.3, Figure 6.4 and Figure 6.7, we draw Figure 6.6 and Figure 6.8 by using the estimated motion parameters of the first 100 frames.

Normal distribution noise is added to the measurement points to test if the algorithms are robust to noise. The mean of this noise is zero and the deviation is 0.01. Figure 6.10, Figure 6.11 and Figure 6.12 show the estimated motion parameters from the noisy measurement points by using PLS, PLS-Kalman, and SFM, respectively.

We have to mention here that, in SFM, the estimated translational parameters in Figure 6.9 and Figure 6.12 are scaled by a constant in order to compare with the true values.

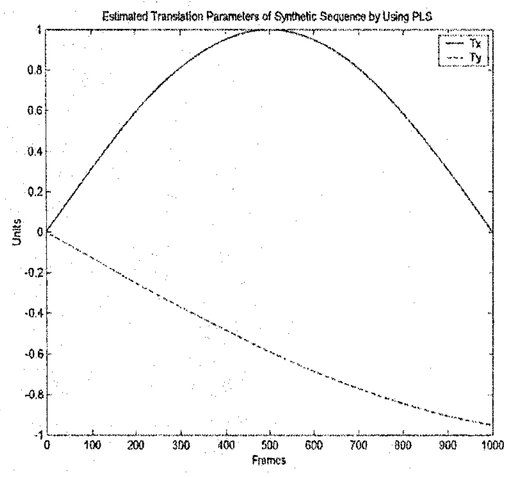


(a) Translation

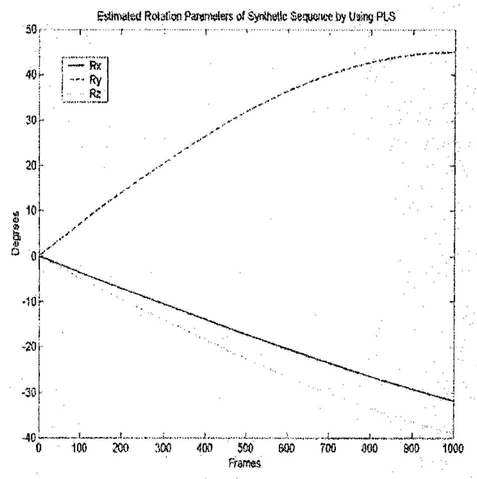


(b) Rotation

Figure 6.4 True motion parameters of the synthetic sequence

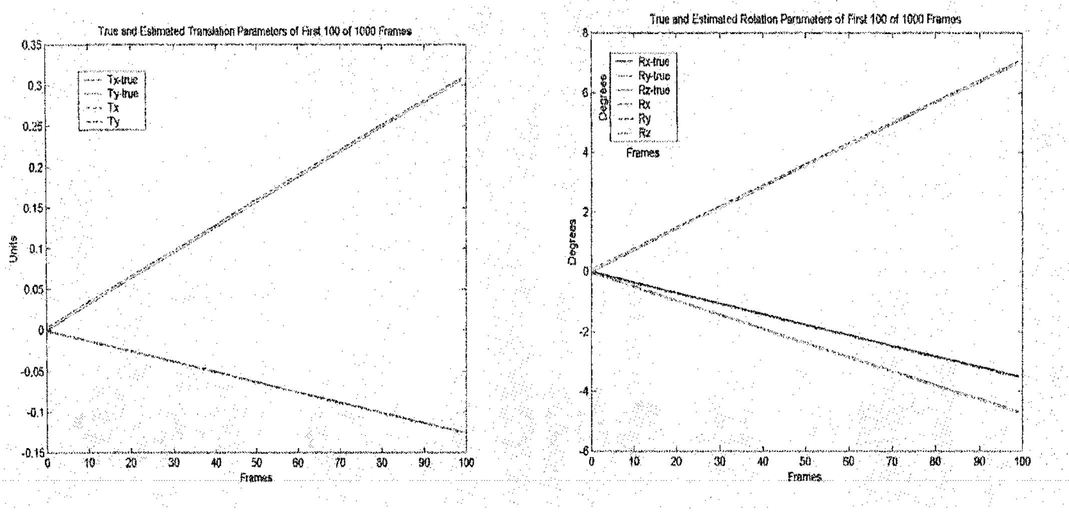


(a) Translation



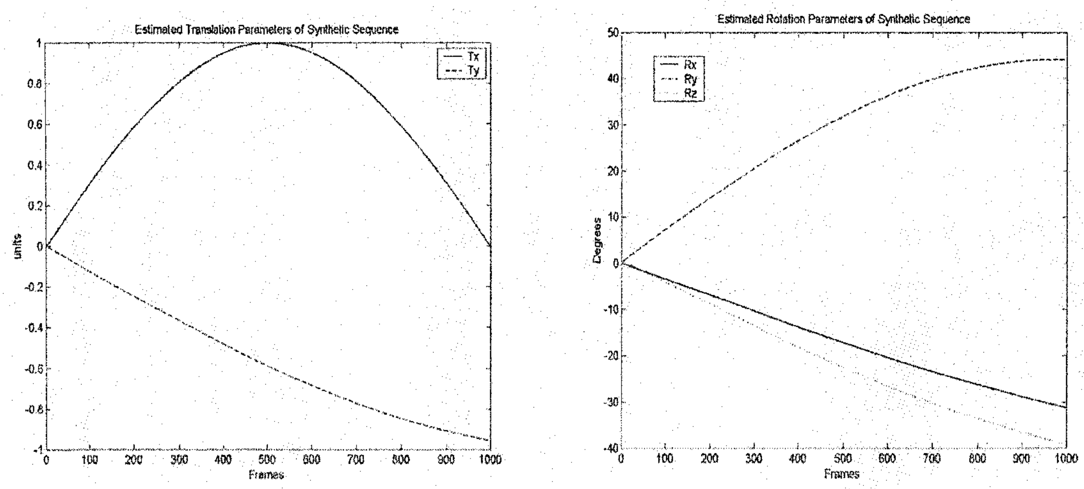
(b) Rotation

Figure 6.5 Estimated motion parameters of the synthetic sequence by PLS



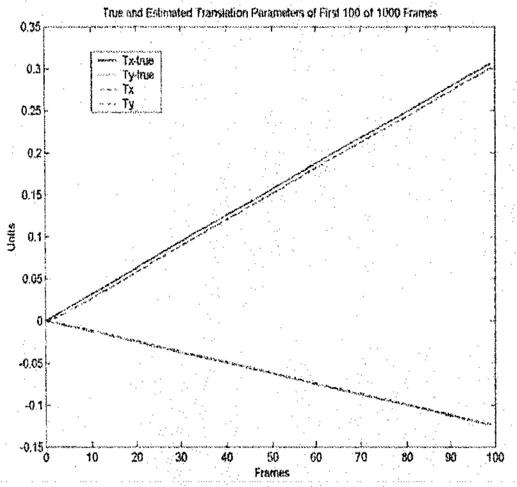
(a) Translation (b) Rotation

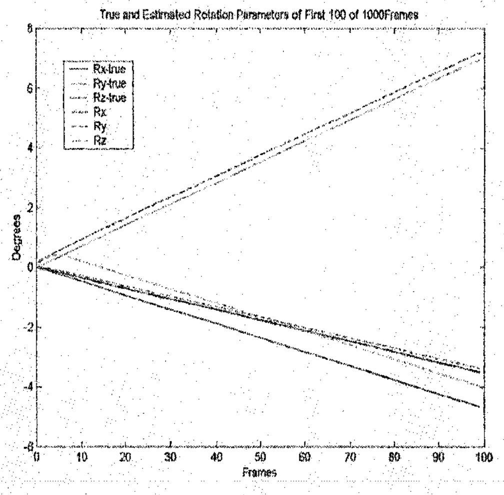
Figure 6.6 True and estimated motion parameters in the first 100 frames by PLS



(a) Translation (b) Rotation

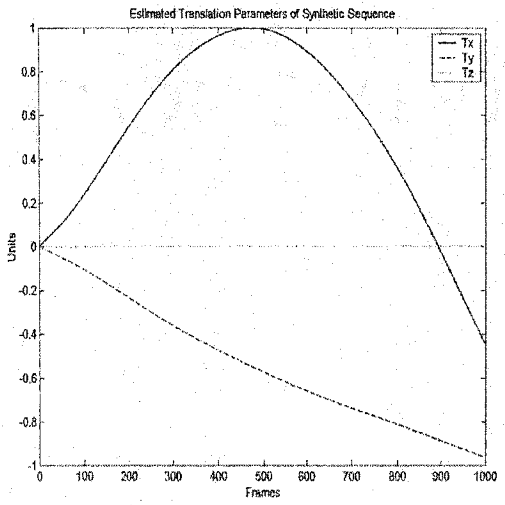
Figure 6.7 Estimated motion parameters of the synthetic sequence by PLS-Kalman

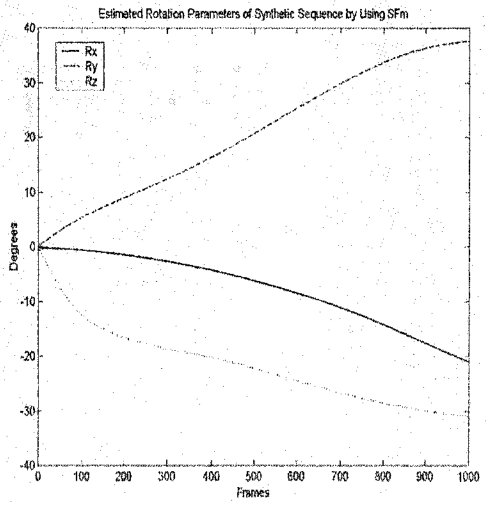




(a) Translation (b) Rotation

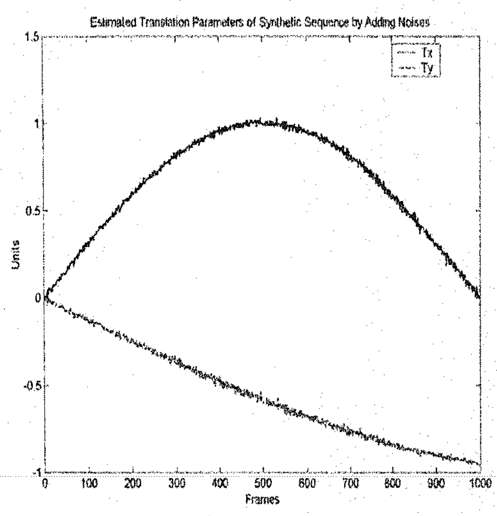
Figure 6.8 True and estimated motion parameters in the first 100 frames by PLS-Kalman

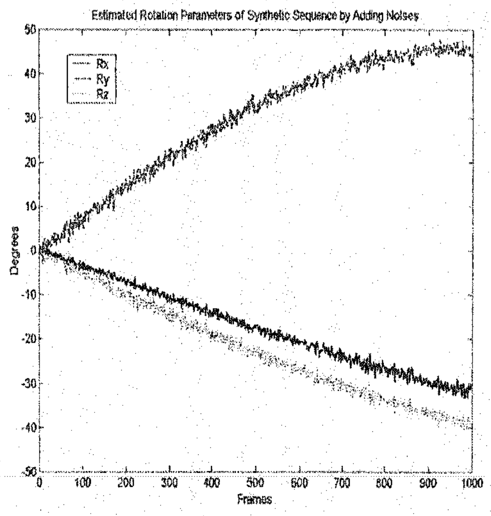




(a) Translation (b) Rotation

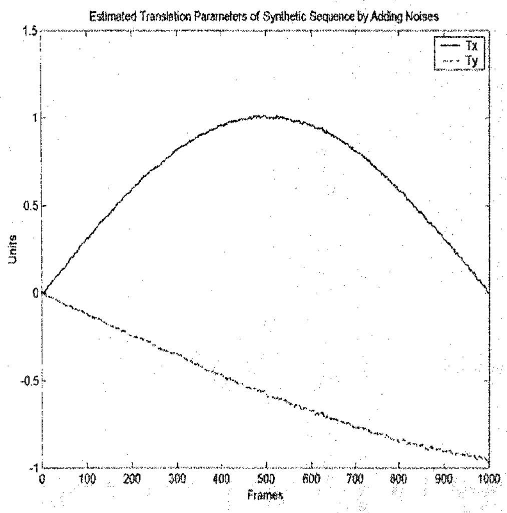
Figure 6.9 Estimated motion parameters of the synthetic sequence by SFM

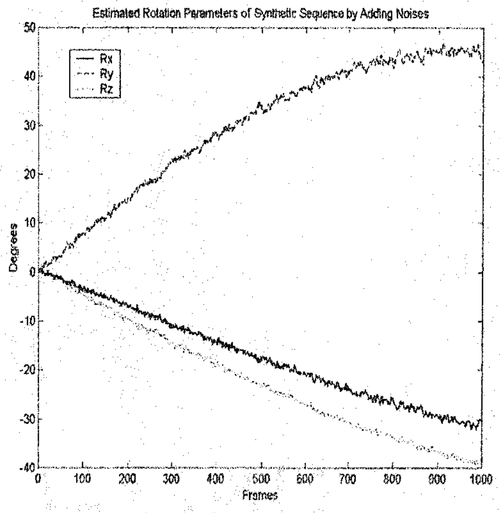




(a) Translation (b) Rotation

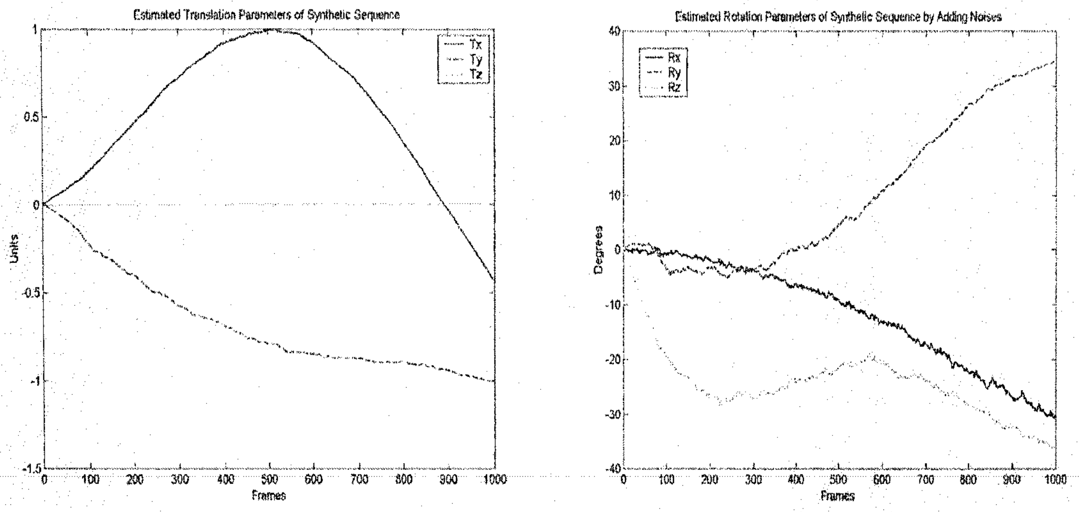
Figure 6.10 Estimated motion parameters of the synthetic sequence by PLS with noise





(a) Translation (b) Rotation

Figure 6.11 Estimated motion parameters of the synthetic sequences by PLS-Kalman with noise



(a) Translation (b) Rotation

Figure 6.12 Estimated motion parameters of the synthetic sequences by SFM with noise

6.3.3 Experiments on real images

The face sequences we used in the test were obtained in our lab. A head sensor was used to measure the true motion parameters of the moving head. The speed of the whole system is about 5 frames per second on a P-2 PC. Most of the time is spent on the point tracking, motion estimation, and model adaptation.

Figure 6.13 (a) shows the face images in the face sequence niu_1.avi. The frame indexes of these face images are 1, 40, 73, 106, 122, 167, 178, and 198. The face models, which are adapted by the motion parameters obtained by the PLS, PLS-Kalman and SFM algorithms, are shown in column (b), (c), and (d), respectively.

Figure 6.14 shows the true motion parameters obtained by using the head sensor in our lab. The estimated motion parameters by using the PLS, PLS-Kalman, and SFM algorithms are shown in Figure 6.15, 6.16, and 6.17, respectively.

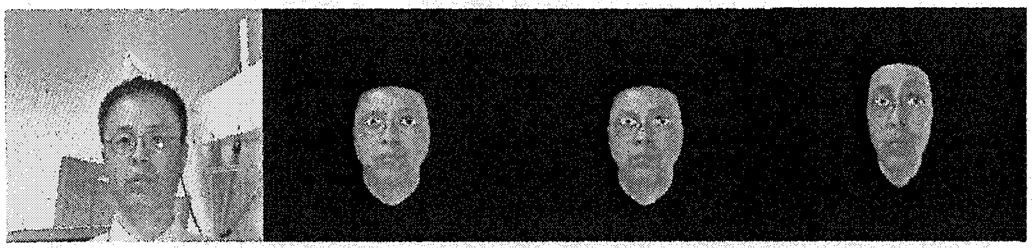
Appendix VI shows the rigid motion estimation results of sequence Vam1.avi, which is downloaded from Image and Video computing Group of Boston University (http://cs-www.bu.edu).

6.3.4 Analysis

From the experiments, we can see that the PLS-Kalman algorithm shows the best results. The trajectories of the parameters obtained by the PLS-Kalman algorithm are much smoother and more accurate than those obtained by the other two algorithms. Moreover, it is more robust to noise. Consequently, the PLS-Kalman algorithm was selected as the motion estimator in our videoconference system.

In the following, we will analyse first what cause the estimation errors in the three algorithms. Next, the advantages and disadvantages will be discussed.

Frame 1:



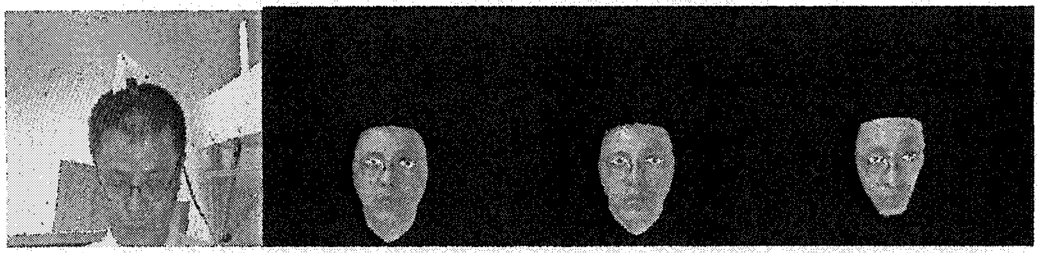
Frame 40:

Frame 73:

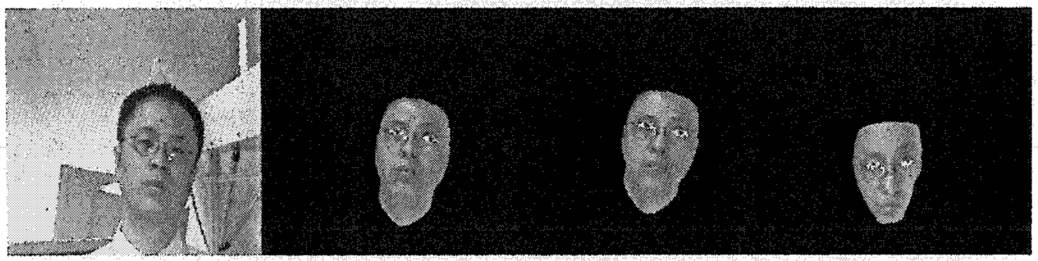
Frame 106:

(continue)

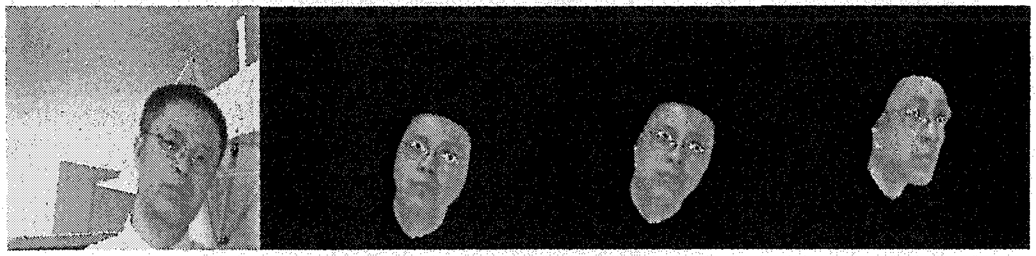
Frame 122:



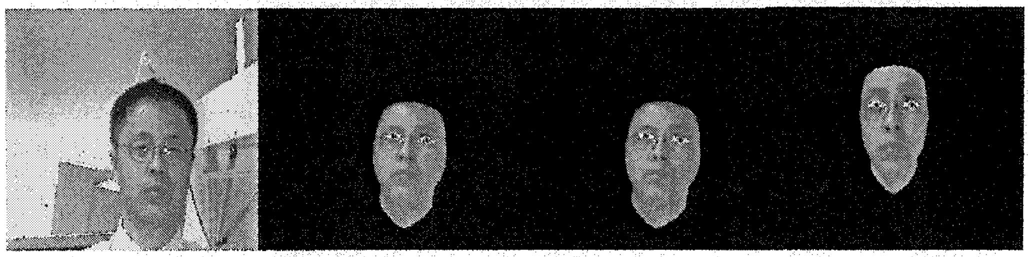
Frame 167:



Frame 178:

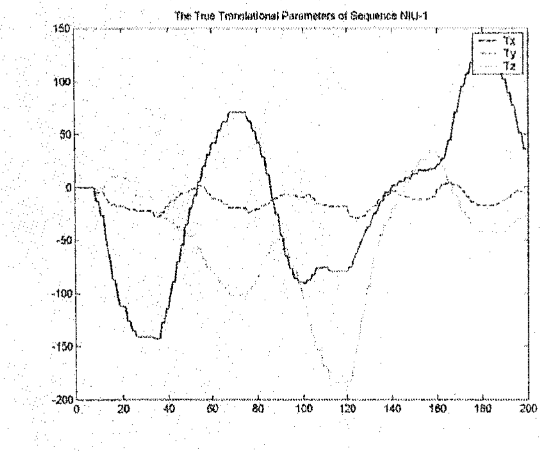


Frame 198:

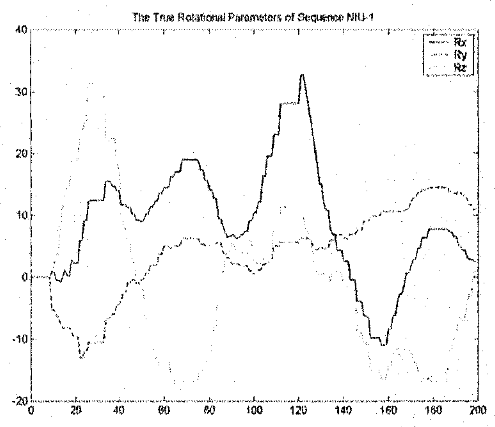


(a) face images (b) PLS algorithm (c) Kalman-PLS (d) SFM

Figure 6.13 Face images in the sequence Niu_1.avi and adapted face models

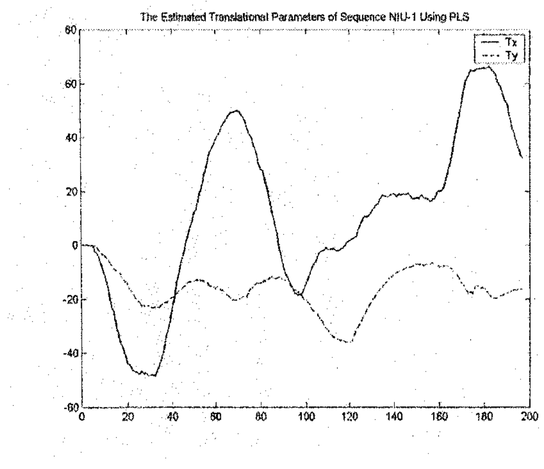


(a) Translational parameters

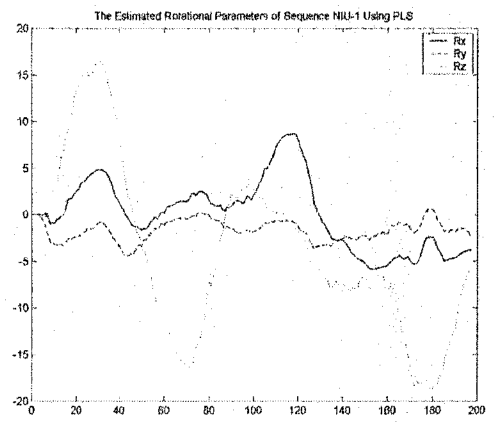


(b) Rotational Parameters

Figure 6.14 True motion parameters of the sequence Niu_1

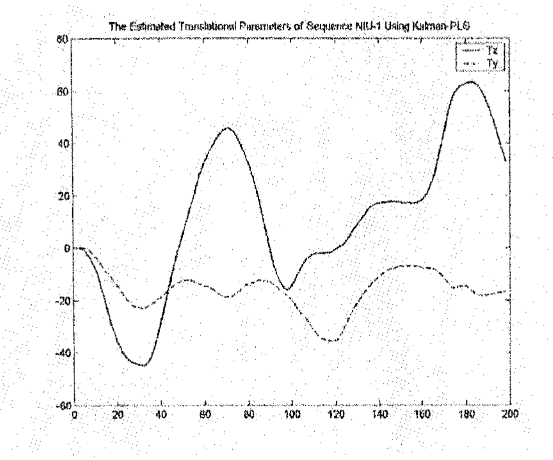


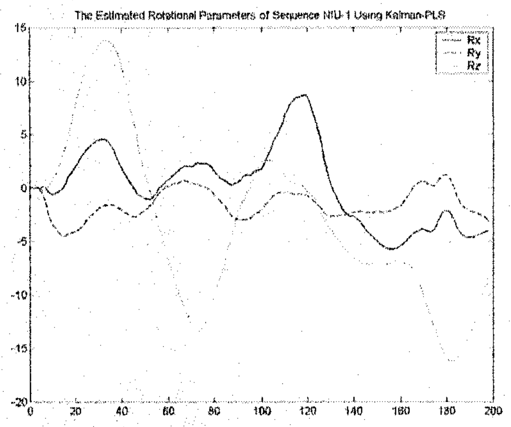
(a) Translational parameters



(b) Rotational Parameters

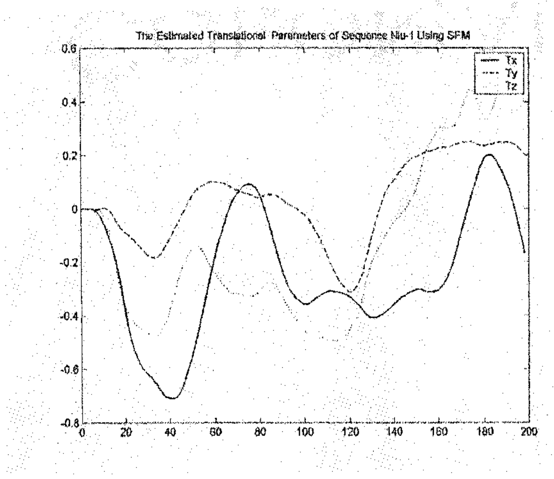
Figure 6.15 Estimated motion parameters of the sequence Niu_1 using PLS

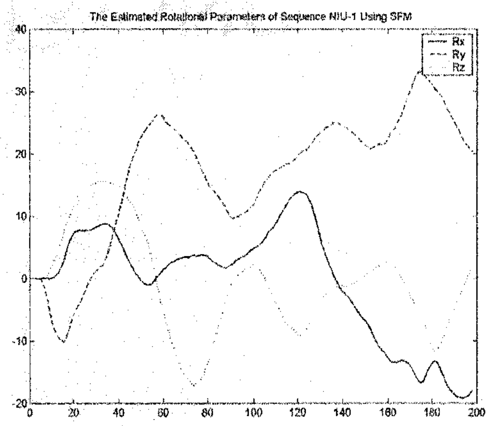




(a) Translational parameters (b) Rotational Parameters

Figure 6.16 Estimated motion parameters of sequence Niu_1 using Kalman-PLS





(a) Translational parameters (b) Rotational Parameters

Figure 6.17 Estimated motion parameters of the sequence Niu_1 using SFM

The estimation errors in the PLS and PLS-Kalman algorithms are caused by the following:

- The inaccuracies in the point depth values that were obtained from the adapted face model. The face model in our project is adapted only at the global level. Therefore, there are shape differences between the real face and the adapted face model. In addition, we cannot avoid depth error, even if we adapt the face model at the local level, because of the limited model adaptation in all existing algorithms;
- The tracking errors;
- The errors caused by the assumption of zero translation in the z direction. If the object has large translations in the z direction, the results will be inaccurate;
- The errors caused by linearization of the rotation matrix in Equation (6-9).

In addition, the PLS-Kalman algorithm has the following aspects that can cause inaccurate estimation:

- The inaccuracies caused by the Kalman initialization. In fact, the covariance values in Kalman filter control the sensitivity of the different types of motions;
- Linearization errors in the Kalman filter.

The resulting motion parameters in the SFM algorithm are affected by:

- The tracking errors;
- The errors caused by the Kalman initialization;
- The inaccuracies caused by the Kalman linearization;
- The errors caused by the interactions among rotations, translations and focal length.

From the experiment results and the above analysis, we can compare the advantages and disadvantages of the different algorithms.

The advantages of PLS and PLS-Kalman algorithms are that they use a simple motion model, which requires less computation compared with SFM. As a result, the Jacobian matrix in the PLS-Kalman algorithm can be obtained easily. Because PLS-Kalman uses depth information, it is robust to different initial settings. The disadvantages are that the PLS and PLS-Kalman assume zero translation of the face in the z direction and therefore require depth information. If the face undergoes large translations in the z direction, the results of motion estimation become worse. The inaccurate depth information can cause estimation errors.

The advantage of the SFM algorithm is that no depth information is used in the motion estimation. It is an essential advantage for the application of monocular sequences. The disadvantages of the SFM algorithm are that it uses a complicated state vector and more computation. The complicated state vector gives us difficulties in calculating the Jacobian matrix. Also, SFM is sensitive to the initial settings. It diverges easily if the initial settings are not appropriate. In the experiments, we can see that PLS-Kalman is more robust to noise than the SFM, possibly because SFM has no *a priori* knowledge about the object at all. If SFM does not have enough point positions, it is difficult to obtain accurate structures for sure. In such a situation, SFM can possibly explain the translations to rotations and vice versa.

CHAPTER VII
CONCLUSION

In the thesis, we proposed a real-time model-based coding system for videoconference. The system includes robust face detection, facial feature localization, tracking-point selection and feature tracking, model initialization and motion estimation. Three current rigid motion estimation methods, PLS, PLS-Kalman and SFM are compared in the experiments.

In the face detection component, the skin-color regions are segmented successfully from the background in YCrCb color space by selecting the appropriate range values of Cr and Cb. The histogram equalization and histogram fitting method proposed can reduce the luminance effects on the face color ranges in YCrCb color space. Density regularization and some image analysis techniques are used to efficiently eliminate skin-color regions with small areas. The elliptic shape correction method can eliminate skin-color regions with large areas in the background.

The facial feature localization module uses first a luminance verification method, based upon the assumption that the facial features have lower luminance values than parts of the face. A knowledge-based method is used to select the facial features among all candidate regions. The measurements used in this method are selected by using knowledge of the detected face and the ideal face proportions. Multiple measurements in the facial feature selection can make the system more stable. All the measurements are approximated by normal distribution models which represent the probabilities of the candidate regions to be the facial features.

The face detection and facial feature localization methods have been tested on the AR face database, which has 133 face images in total, including 75 men and 58 women. In the experiment, 125 face images were successfully detected and in 9 images the method failed, corresponding to a success rate 94.98%. For comparison, we manually select the three facial feature positions as the true feature positions. The average differences between detected feature positions and the manually selected positions are 1.79 pixels in x coordinates and 1.58 in y coordinates. In the experiments, we can process about one image per second and much of the time is spent in changing the RGB color space to YCrCb color space. Since face detection and facial localization are needed only in the first frame of the video sequence, the time required is not critical to the system.

The size of the face model is globally adapted at the beginning of the face model initialization stage. Then, the detected face texture is mapped onto the face model. Here we propose a method that separates the face region into 18 smaller regions to map 3D nodes of the face model to 2D face texture pixels. Five control points are used to keep the original shapes for all the facial features. In the experiments, the initialized face model can represent the individual faces successfully.

Good features for tracking are selected in the face region by using a feature selection criterion. Most of the time, the features for tracking are selected around the corners of eyes, eyebrows and mouth. Block matching and tracking as well as monitoring methods are used to track the selected features in the following frames.

By using the feature positions between two neighbor frames, three recursive methods, Predictive Least-square (PLS), PLS-Kalman, and Structure from motion (SFM), are compared at the rigid motion estimation stage. Then, the motion of the face model can be controlled by the motion parameters. The PLS and PLS-Kalman methods use the depth information from the adapted face model. The SFM method uses the Extended Kalman filter to recover the motion, pointwise structure and focal length for arbitrary image sequences without depth information. Among the three motion estimation methods, the PLS-Kalman method provides the best results in the experiments and is robust to noises as well as different initial settings.

The speed of the whole system is about 4-5 frames per second on a P-2 300 MHz PC computer. It is not difficult to achieve real-time if we move the system to a faster computer.

The main contributions of this thesis are shown in the following:

- It proposes the histogram equalization and histogram fitting method to reduce the luminance effects of face color in YCrCb color space.

- It proposes using the elliptical shape correction method to eliminate the skin-color regions with large areas on the density map.
- It proposes a luminance verification method to enhance the candidate regions for facial features.
- It proposes robust knowledge-based selecting measurements for facial features in facial feature localization.
- It proposes a normal distribution approximation method for the selecting measurements, which can show the probabilities of the candidate regions to be the facial features.
- It proposes a registration method for mapping 3D nodes of the face model onto the 2D face textures.
- It proposes the integration method for the real-time system.

In the system, many aspects are still to be improved. First, the current system does not include non-rigid motion estimation, *i.e.* the extraction of the facial movements in order to reproduce them on the 3-D model at the other end of the system. Secondly, more stable and robust face detection can be obtained if we can combine the edge or contour information from the face images in the system. Thirdly, only global adaptation is performed on the face model in the model initialization stage. Local adaptation on the face model can improve the accuracy of individual representations and the depth values used in the rigid motion estimation. And lastly, a robust feature tracking algorithm needs to be developed to deal with the rotation and error accumulation.

REFERENCES

AVIDAN, Shai, SHASHUA, Amnon. 1998. ‹‹Novel view synthesis by cascading trilinear tensors››. IEEE Transactions on visualization and computer graphics. 4:4. 293-306.

AZARBAYEJANI, Ali, PENTLAND, Alex P. 1997. ‹‹Recursive estimation of motion, structure and focal length››. IEEE Transactions on pattern analysis and machine intelligence. 19:7. 562-575.

BRUNELLI, Roberto, POGGIO, Tomaso. 1993. ‹‹Face recognition: features versus templates››. IEEE Transactions on pattern analysis and machine intelligence. 15:10. 1042-1052.

CASTLEMAN, Kenneth R. 1996. *Digital Image Processing*. Prentice Hall.

CHAI, Douglas, NGAN, King N. 1999. ‹‹Face segmentation using skin-color map in videophone applications››. IEEE Transactions on Circuits and Systems for Video Technology. 9:4. 551-564.

CROWLEY, James L., SCHWERDT, Karl. 1999. ‹‹Robust tracking and compression for video communication››. Proceedings of the international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG_RTS '99), IEEE Computer Society. 2-9.

DONATO, Giabluca, BARLETT, Marian Steward, HAGER, Joseph C., EKMAN, Paul, SEJNOWSKI, Terrence J. 1999. ‹‹Classifying facial actions››. IEEE Transactions on pattern analysis and machine intelligence. 21:10. 974-989.

EISERT, Peter, GIROD, Bernd. 1998. ‹‹Analyzing facial expressions for virtual conferencing››. IEEE Computer graphics and applications. 70-78.

EISERT, Peter, STEINBACH, Eckehard, GIROD, Bernd. 2000. ‹‹Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views››. IEEE Trans. on circuits and systems for video technology. 10:2. 261-277.

ESSA, Irfan A., PENTLAND, Alex P. 1997. ‹‹Coding, analysis, interpretation and recognition of facial expressions››. IEEE Transactions on pattern analysis and machine intelligence. 19:7. 757-763.

FENG, G. C., YUEN, Pong C., LAI, J. H. 2000. ‹‹Virtual view face image synthesis using 3-D spring-based model from a single image››. Proceedings of the Forth International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 530-535.

FRIGOLA, M., AMAT, J., CASALS, A. 1999. ‹‹Stereoscopic system for human body tracking in natural scenes››. IEEE International workshop on modeling people, IEEE Computer Society. 70-78.

GARCIA, Christophe, TZIRITAS, Georgios. 1999. ‹‹Face detection using quantized skin color regions merging and wavelet packet analysis››. IEEE Transactions on multimedia. 1:3. 264-277.

GONZALEZ, R. C., WOODS, R. E. 1992. *Digital image processing*. Addison-Wesley Publishing Company (New York).

GRAF, Hans Peter, COSATTO, Eric, EZZAT, Tony. 2000. ‹‹Face analysis for the synthesis of photo-realistic talking heads››. Proceedings of the Forth International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 189-194.

HAVALDAR, Parag, LEE, Mi-suen, MEDIONI, Gerard. 1997. ‹‹Synthesizing novel views from unregistered 2-D images››. Computer graphics forum. 16:1. 65-73.

HERPERS, R., VERGHESE, G., DERPANIS, K., MCCREADY, R., MACLEAN, J., LEVIN, A., TOPALOVIC, D., WOOD, L., JEPSON, A., TSOTSOS, J. K. 1999. ‹‹Detection and tracking of faces in real environments››. Proceedings of the international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG_RTS '99). 96-104.

JEBARA T., PENTLAND, A. 1997. ‹‹Parameterized structure from motion for 3-D adaptive feedback tracking of faces››. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97). 144-150.

KIM, Sang-hoon, KIM, Hyoung-Gon. 2000. ‹‹Face detection using multi-modal information››. Proceedings of the Forth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 14-19.

KOCH, Reinhard. 1993. ‹‹Dynamic 3-D scene analysis through synthesis feedback control››. IEEE Transactions on pattern analysis and machine intelligence. 15:6. 556-568.

KOTROPOULOS, C., TEFAS, A., PITAS, I. 2000. ‹‹Morphological elastic graph matching applied to frontal face authentication under well-controlled and real conditions››. Pattern Recognition: The Journal of the Pattern Recognition Society. 33 (2000). 1935-1947.

KUMAR Vinay P., POGGIO, Tomaso. 2000. ‹‹Learning-based approach to real time tracking and analysis of face››. Proceedings of the Forth International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 96-101.

LANITIS, Andreas, TAYLOR, Chris J., COOTES, Timothy F. 1997. ‹‹Automatic interpretation and coding of face images using flexible models››. IEEE Transactions on pattern analysis and machine intelligence. 19:7. 743-756.

LEE, Won-Sook, MAGNENAT-THALMANN, Nadia. 1999. ‹‹Generating a population of animated faces from pictures››. IEEE International workshop on modeling people, IEEE Computer Society. 62-69.

LIN, Chun-hung, WU, Ja-ling. 1999. ‹‹Automatic facial feature extraction by genetic algorithms››. IEEE Transactions on image processing. 8:6. 834-845.

LI, Haibo, ROIVAINEN, Pertti, FORCHEIMER, Robert. 1993. ‹‹3-D motion estimation in model-based facial image coding››. IEEE Transactions on pattern analysis and machine intelligence. 15:6. 545-555.

LIU, Jin, PRZEWOZNY, David, PASTOOR, Siegmund. 2000. ‹‹Layered representation of scenes based upon multi-view image analysis››. IEEE Transactions on circuits and systems for video technology. 10:4. 518-529.

MALCIU, Marius, PRETEUX, Francoise. 2000. ‹‹A robust model-based approach for 3-D head tracking in video sequences››. Proceedings of the Forth International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 169-174.

MATSUMOTO, Yoshio, ZELINSKY, Alexander. 1999. ‹‹Real-time stereo face tracking system for visual human interfaces››. Proceedings of the international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG_RTS '99), IEEE Computer Society. 77-82.

MOEZZI, Saied, KATKERE, Arun, KURAMURA, Don Y., JAIN, Ramesh. 1996. ‹‹Reality modeling and visualization from multiple video sequences››. IEEE Computer graphics and applications. 58-63.

MOEZZI, Saied, TAI, Li-cheng, GERARD, Philippe. 1997. ‹‹Virtual view generation for 3-D digital video››. IEEE Multimedia. 18-26.

OKADA, Kazunori, AKAMATSU, Shigeru, MALSBURG, Christoph Von Der. 2000. ‹‹Analysis and synthesis of pose variations of human faces by a linear PCMAP model and its application for pose-invariant face recognition system››. Proceedings of the Forth

IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 142-149.

PARKE, Frederic I., WATERS, Keith. 1996. *Computer facial animation*. publisher: A. K. Peters, 1996.

REINDERS, M. J. T., BEEK, P. J. L. Van, SANKUR, B., LUBBE, J. C. A. Van Der. 1995. *Facial feature location and adaption of a generic face model for model-based coding*. Delft University of Technology.

ROWLEY, Henry A., BALUJA, Shumeet, KANADE, Takeo. 1998. ‹‹Neural network-based face detection››. IEEE Transactions on pattern analysis and machine inteligence. 20:1. 23-38.

SATOH, Shinichi, NAKAMURA, Yuichi, KANADE, Takeo. 1999. ‹‹Name-It: naming and detecting faces in news videos››. IEEE Multimedia. 22-35.

SCHUBERT, ‹‹Detection and tracking of facial Features in real time using a synergistic approach of spatio-temporal models and generalized Hough-transform techniques››.

Proceedings of the Forth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 116-121.

SCHWERDT, Karl, CROWLEY, James L. 2000. ‹‹Robust face tracking using color››. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 90-95.

SENGUPTA, Kuntal, WANG, Shiqin, KO, C. C., BURMAN, Prabir. 2000. ‹‹Automatic face model from monocular image sequences using modified non parametric regression and affine camera model››. Proceedings of the Forth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 524-529.

SHI, Jianbo, TOMASI, Carlo. 1994. ‹‹Good features to track››. IEEE Conference on Computer Vision and Pattern Recognition (CVPR94, Seattle). 593-600.

SMOLIE, Aljoscha, MAKAI, Bela, SIKORA, Thomas. 1999. ‹‹Real-time estimation of long-term 3-D motion parameters for SNHC face animation and model-based coding applications››. IEEE Transactions on Circuits and Systems for Video Technology. 9:2. 255-263.

SOBOTTKA, K., PITAS, I. 1997. ‹‹Looking for faces and facial features in color images››. Pattern Recognition and Image Analysis. 7:1. 124-137.

SUNG, Kah-Kay, POGGIO, Tomaso. 1998. ‹‹Example-based learning for view-based human face detection››. IEEE Transactions on pattern analysis and machine intelligence. 20:1. 39-51.

TERRILLON, Jean-Christophe, SHIRAZI, Mahdad N., FUKAMACHI, Hideo, AKAMATSU, Shigeru. 2000. ‹‹Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images››. Proceedings of the Forth International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 54-61.

TERZOPOULOS, Demetri, WATERS, Keith. 1993. ‹‹Analysis and synthesis of facial images sequences using physical and anatomical models››. IEEE Transactions on pattern analysis and machine intelligence. 15:6. 569-579.

TIAN, Ying-Li, KANADE, Takeo, COHN, Jeffrey F. 2000. ‹‹Dual-state Parametric Eye Tracking››. Proceedings of the Forth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 110-115.

VETTER, Thomas, POGGIO, Tomaso. 1997. ‹‹Linear object classes and image synthesis from a single example image››. IEEE Transactions on pattern analysis and machine intelligence. 19:7. 733-742.

WEBER, Frank, HERNANDEZ, Alfredo Herrera. 1999. ‹‹Face location by template matching with a quadratic discriminant function››. Proceedings of the international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG_RTS '99), IEEE Computer Society (Cofu). 10-13.

WU, Haiyuan, CHEN, Qian, YACHIDA, Masahiko. 1999. ‹‹Face detection from color images using a fuzzy pattern matching method››. IEEE Transactions on pattern analysis and machine intelligence. 21:6. 557-563.

ZARIT, Benjamin D., SUPER, Boaz J., QUEK, Francis K. H. 1999. ‹‹Comparison of five color models in skin pixel classification››. Proceedings of the international workshop on recognition, analysis, and tracking of faces and gestures in real-time systems (RATFG_RTS '99), IEEE Computer Society (Cofu). 58-63.

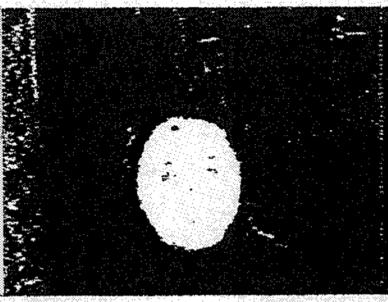
ZHANG, Chongzhen, COHEN, Fernand S. 2000. ‹‹Face shape extraction and recognition using 3-D morphing and distance mapping››. Proceedings of the Forth IEEE International Conference on Automatic Face and Gesture Recognition (Grenoble, France). 28-33.

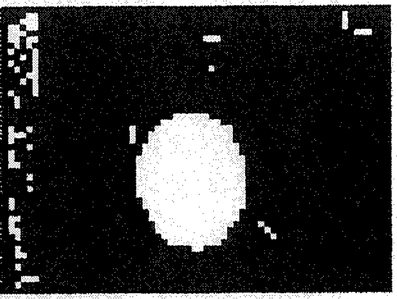
ZHANG, Liang. 1998. ‹‹Automatic adaptation of a face model using action units for semantic coding of videophone sequences››. IEEE Transactions on Circuits and Systems for Video Technology. 8:6. 781-795.

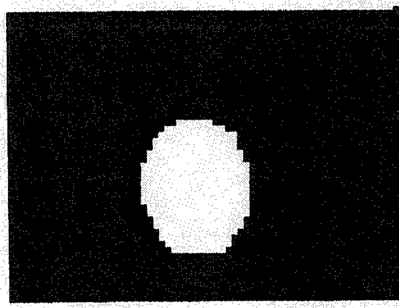
ZHENG, Jiang Yu. 1994. ‹‹Acquiring 3-D models from sequences of contours››. IEEE Transactions on pattern analysis and machine intelligence. 16:2. 163-178.

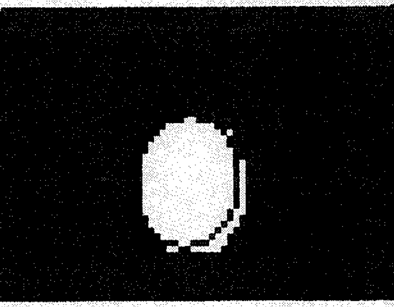
APPENDIX I An example of face detection and facial feature localization

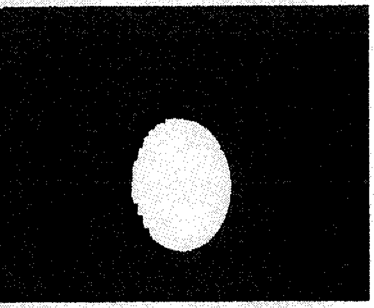
(a)


(b)

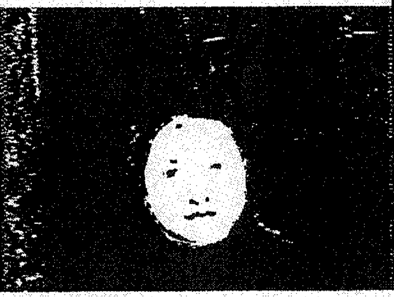

(c)


(d)

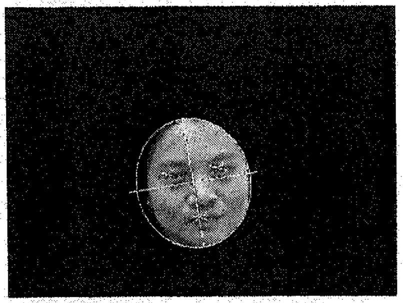

(e)


(f)

(g)


(h)

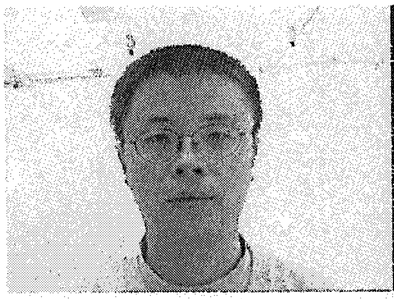
(i)

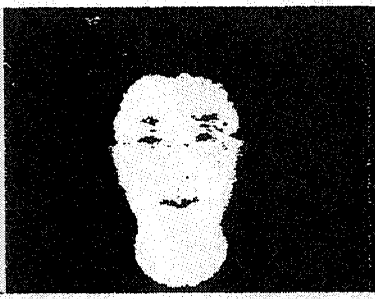


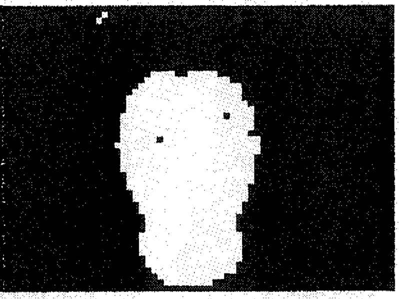
(j)

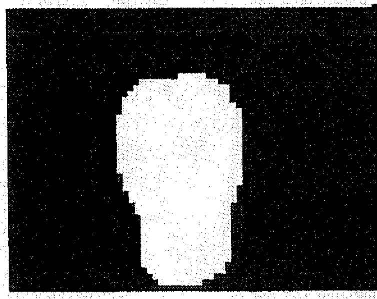
(a) face image (b) processed face map after color verification (c) density map after density regularization (d) density map after geometric correction (e) density map after ellipse shape correction (f) face map after contour extraction (g)original face map after luminance verification (h)original face map after geometric correction (i)contour of candidate regions (j)last results

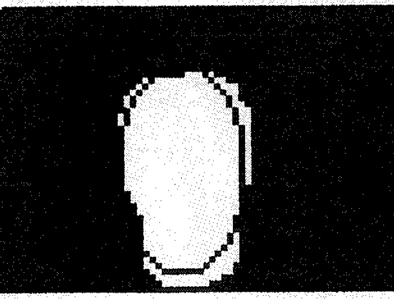
APPENDIX II An example of face detection and facial feature localization

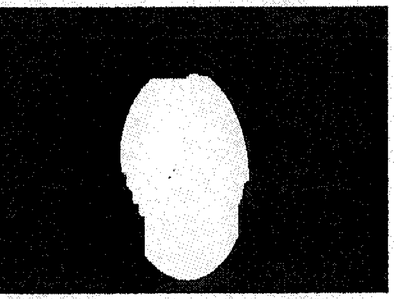

(a)

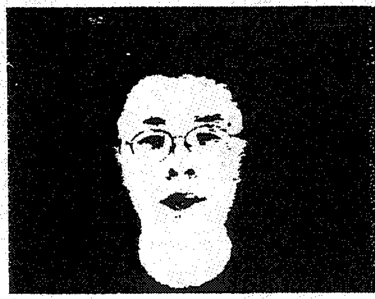

(b)

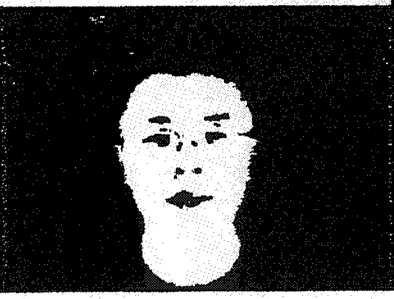

(c)

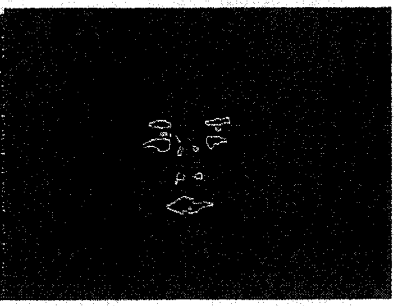

(d)

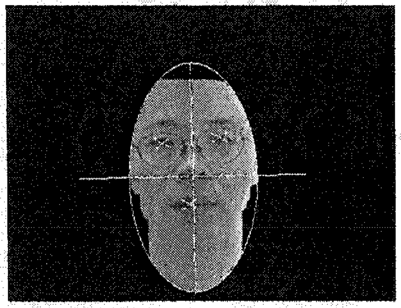

(e)


(f)


(g)


(h)


(i)



(j)

(a) face image (b) processed face map after color verification (c) density map after density regularization (d) density map after geometric correction (e) density map after ellipse shape correction (f) face map after contour extraction (g)original face map after luminance verification (h)original face map after geometric correction (i)contour of candidate regions (j)last results

Appendix III AR face database

m-001-01 m-002-01 m-003-01 m-004-01 m-005-01 m-006-01

m-007-01 m-008-01 m-009-01 m-010-01 m-011-01 m-012-01

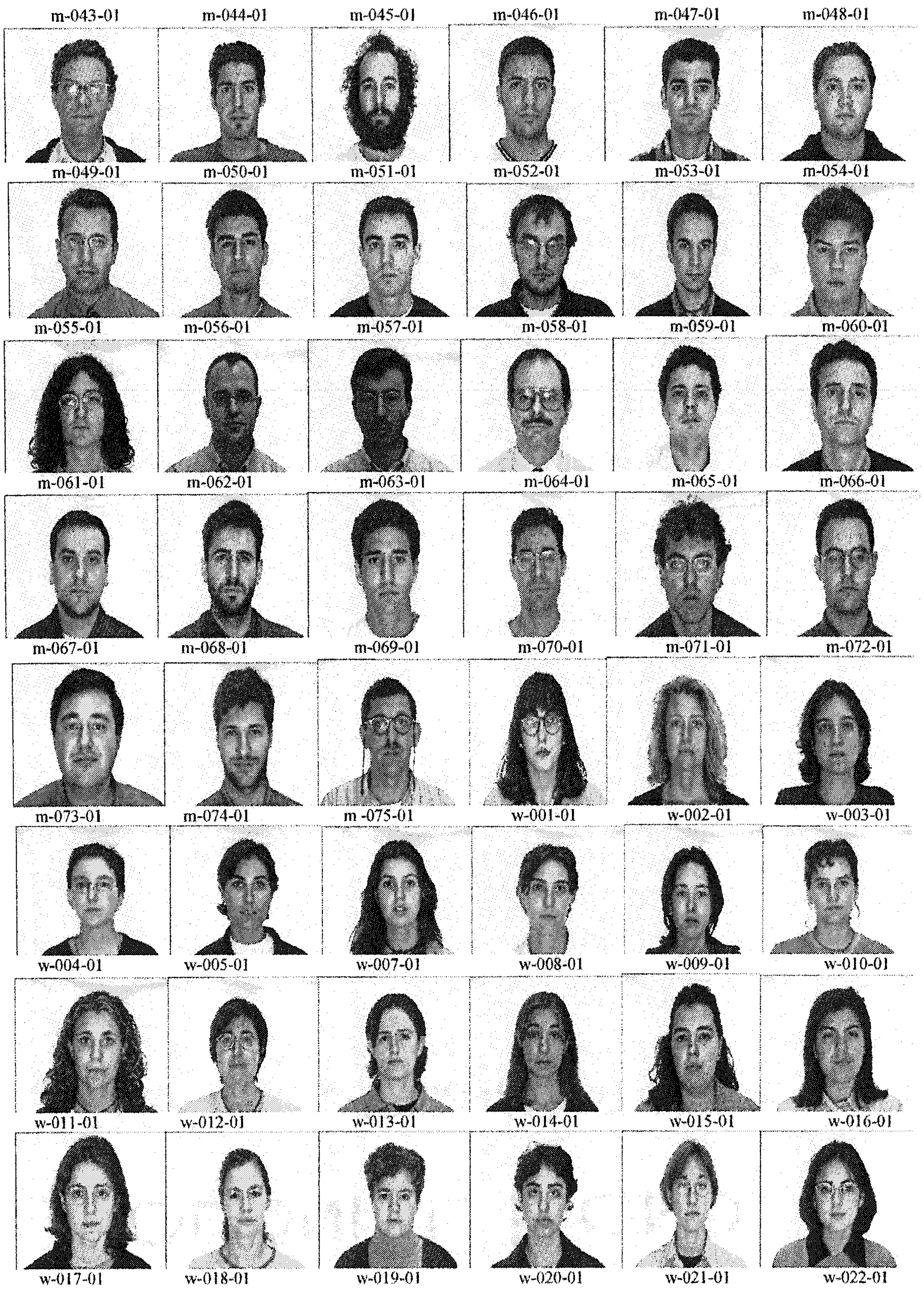
m-013-01 m-014-01 m-015-01 m-016-01 m-017-01 m-018-01

m-019-01 m-020-01 m-021-01 m-022-01 m-023-01 m-024-01

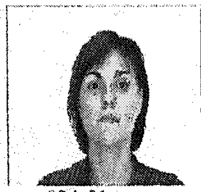
m-025-01 m-026-01 m-027-01 m-028-01 m-029-01 m-030-01

m-031-01 m-032-01 m-033-01 m-034-01 m-035-01 m-036-01

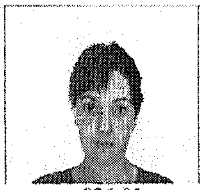
m-037-01 m-038-01 m-039-01 m-040-01 m-041-01 m-042-01


m-043-01
m-044-01
m-045-01
m-046-01
m-047-01
m-048-01
m-049-01
m-050-01
m-051-01
m-052-01
m-053-01
m-054-01
m-055-01
m-056-01
m-057-01
m-058-01
m-059-01
m-060-01
m-061-01
m-062-01
m-063-01
m-064-01
m-065-01
m-066-01
m-067-01
m-068-01
m-069-01
m-070-01
m-071-01
m-072-01
m-073-01
m-074-01
m -075-01
w-001-01
w-002-01
w-003-01
w-004-01
w-005-01
w-007-01
w-008-01
w-009-01
w-010-01
w-011-01
w-012-01
w-013-01
w-014-01
w-015-01
w-016-01
w-017-01
w-018-01
w-019-01
w-020-01
w-021-01
w-022-01

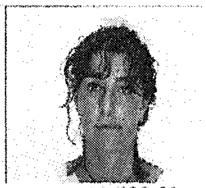
w-023-01


w-024-01

w-025-01


w-026-01

w-027-01


w-028-01

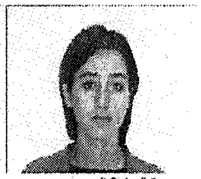
w-029-01

w-030-01

w-031-01

w-032-01

w-033-01

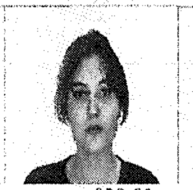

w-034-01

w-035-01

w-036-01

w-037-01

w-038-01


w-039-01

w-040-01

w-041-01

w-042-01

w-043-01

w-044-01

w-045-01

w-046-01

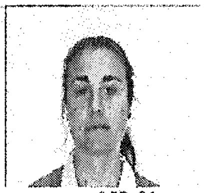
w-047-01

w-048-01

w-049-01

w-050-01

w-051-01


w-052-01

w-053-01

w-054-01

w-055-01

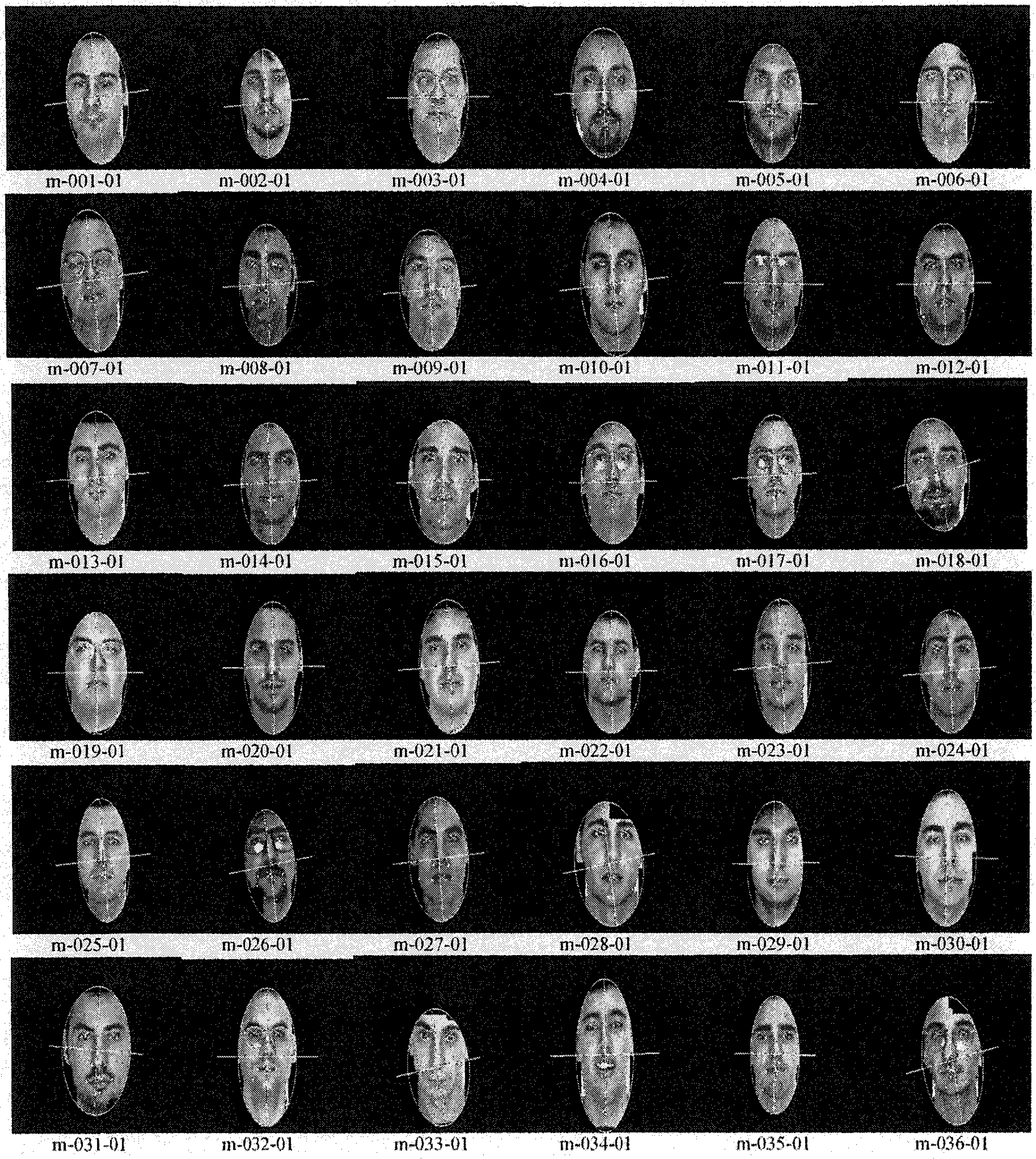
w-056-01

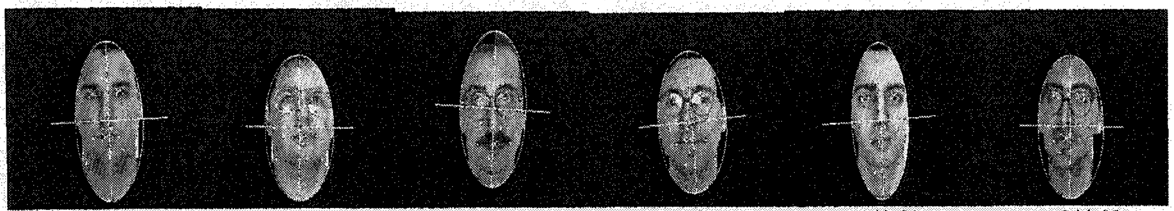
w-057-01

w-058-01

w-059-01

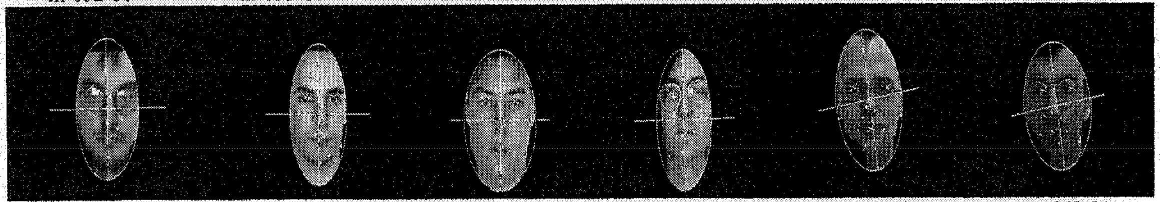
Appendix IV Results of face detection and facial feature localization using AR face database

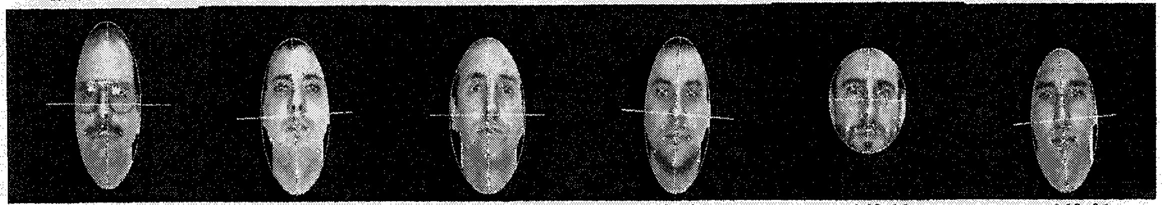


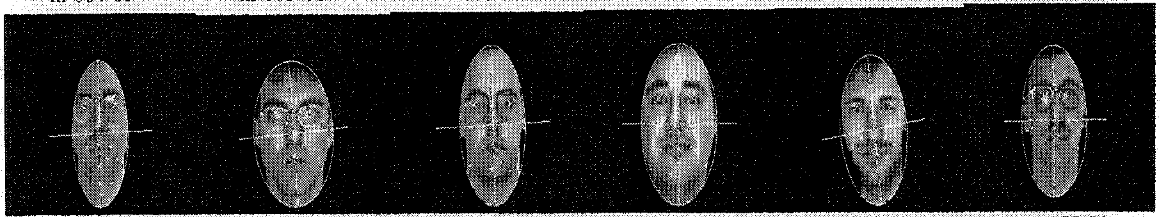

m-037-01 m-039-01 m-040-01 m-041-01 m-042-01 m-044-01

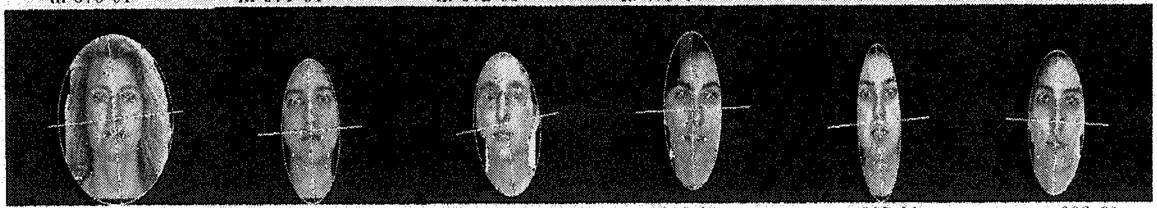
m-045-01 m-046-01 m-047-01 m-048-01 m-049-01 m-050-01

m-052-01 m-053-01 m-054-01 m-055-01 m-056-01 m-057-01

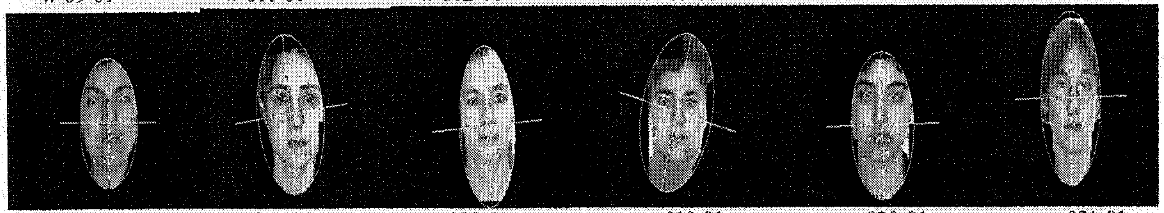

m-058-01 m-059-01 m-060-01 m-061-01 m-062-01 m-063-01

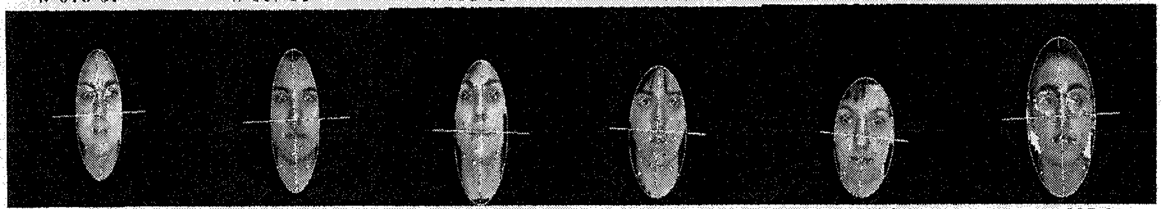

m-064-01 m-065-01 m-066-01 m-067-01 m-068-01 m-069-01

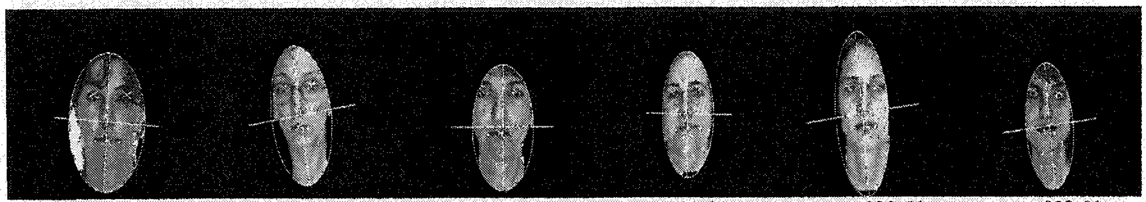

m-070-01 m-071-01 m-072-01 m-073-01 m-074-01 m-075-01

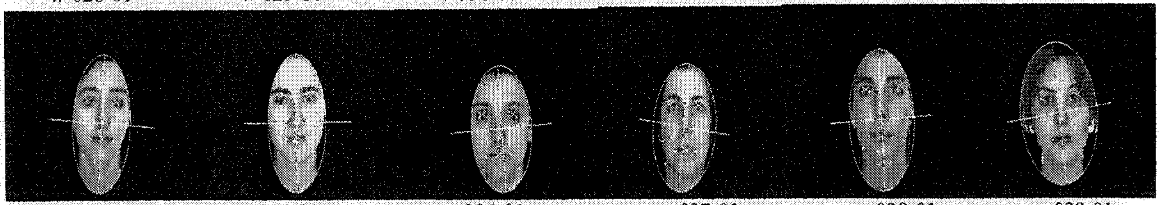

w-002-01 w-003-01 w-004-01 w-005-01 w-007-01 w-008-01

w-09-01 w-010-01 w-012-01 w-013-01 w-014-01 w-015-01

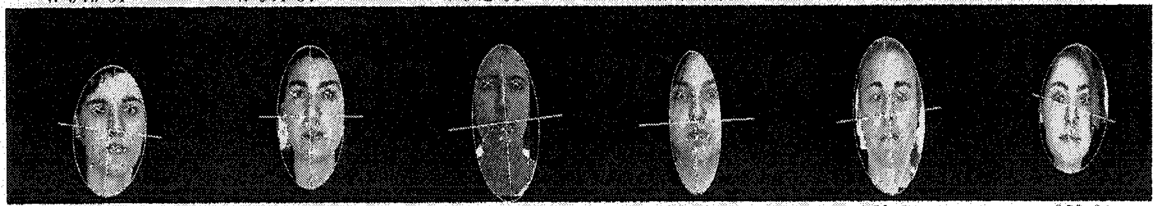

w-016-01 w-017-01 w-018-01 w-019-01 w-020-01 w-021-01

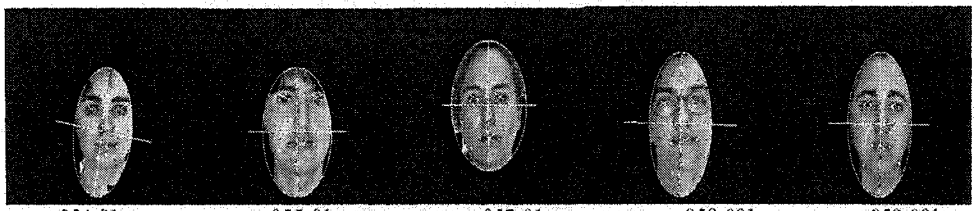

w-022-01 w-023-01 w-024-01 w-025-01 w-026-01 w-027-01


w-028-01 w-029-01 w-030-01 w-031-01 w-032-01 w-033-01


w-034-01 w-035-01 w-036-01 w-037-01 w-038-01 w-039-01

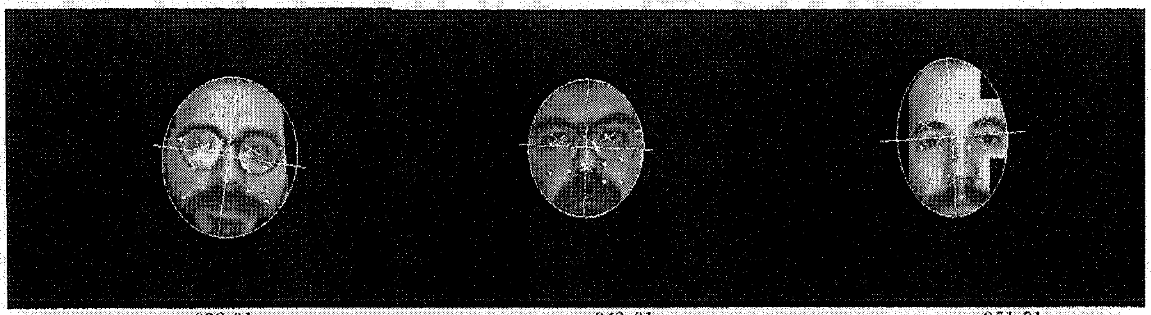
w-040-01 w-041-01 w-042-01 w-044-01 w-045-01 w-046-01

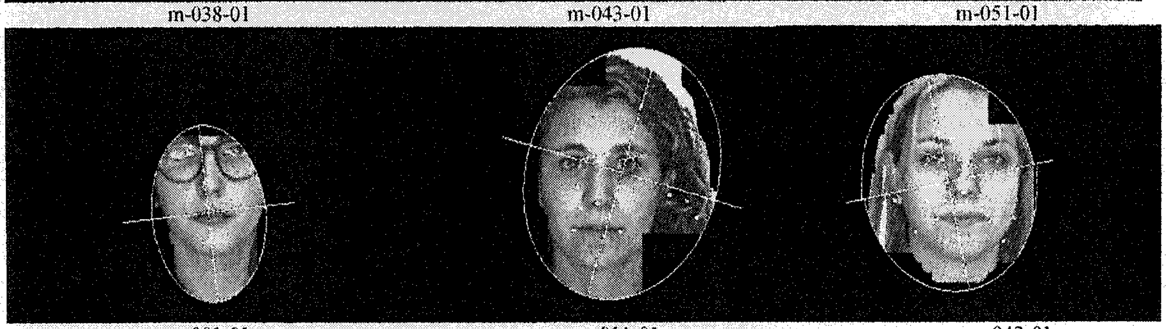

w-047-01 w-048-01 w-049-01 w-050-01 w-052-01 w-053-01

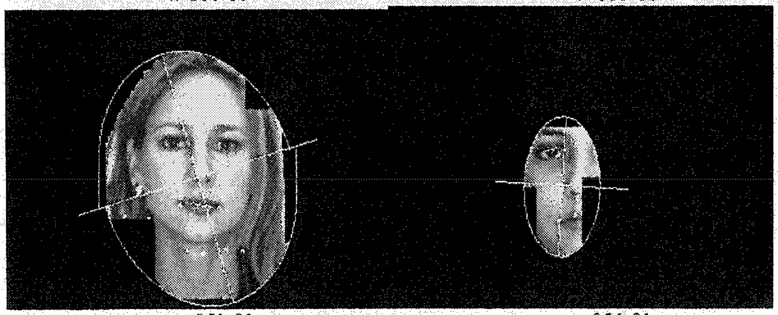


w-054-01 w-055-01 w-057-01 w-058-001 w-059-001

Appendix V Failed examples of face detection and facial feature localization in AR face database


m-038-01 m-043-01 m-051-01


w-001-01 w-011-01 w-043-01


w-051-01 w-056-01

Appendix VI Rigid motion estimation results using sequence Jam1.avi

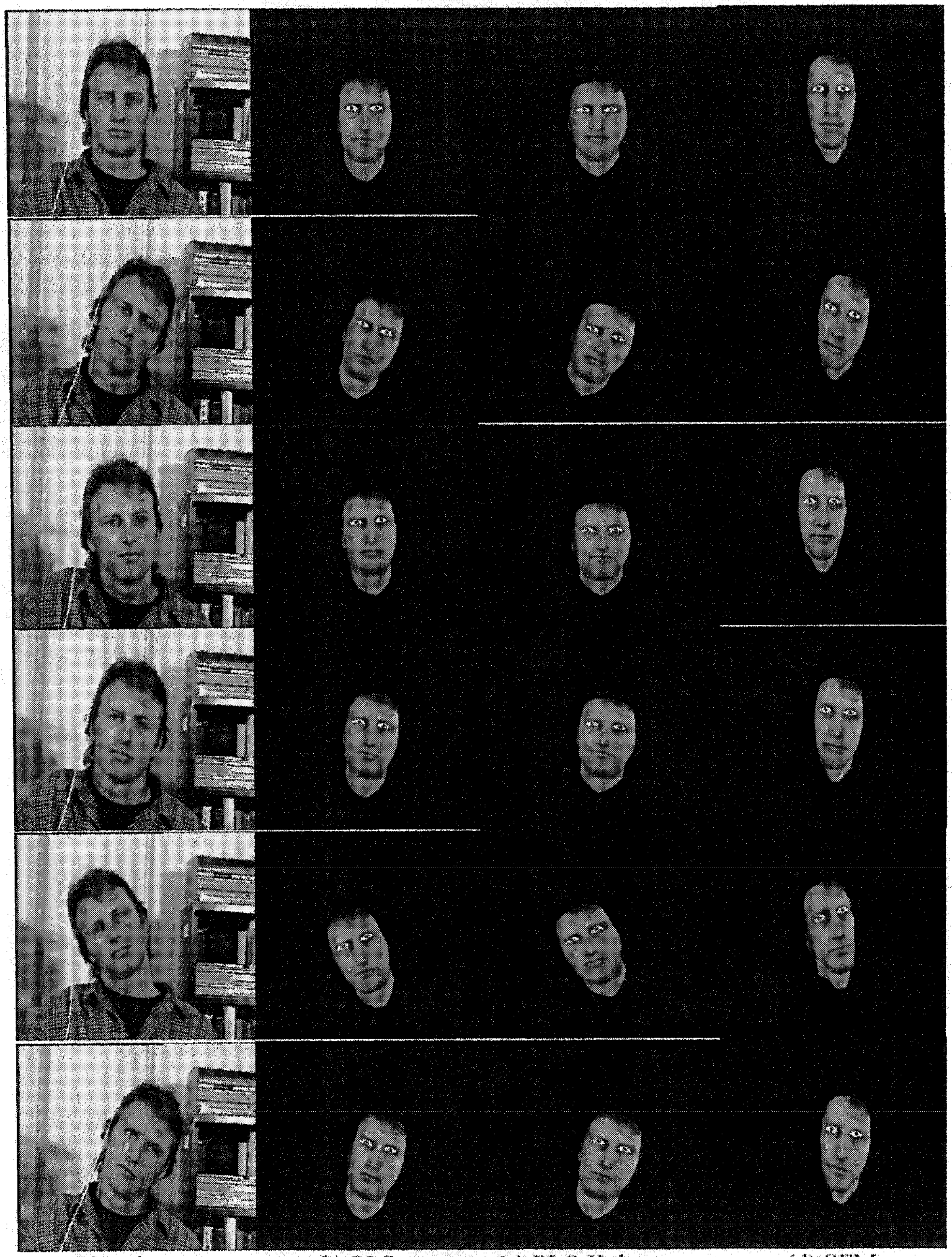
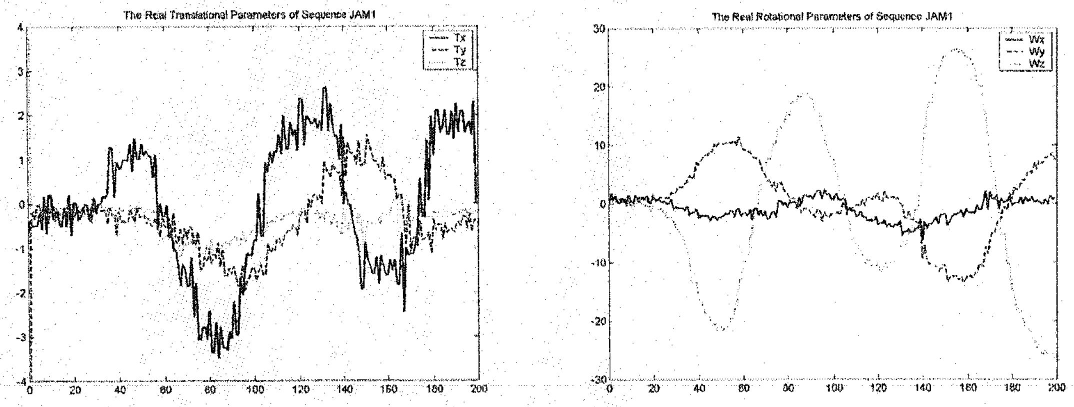
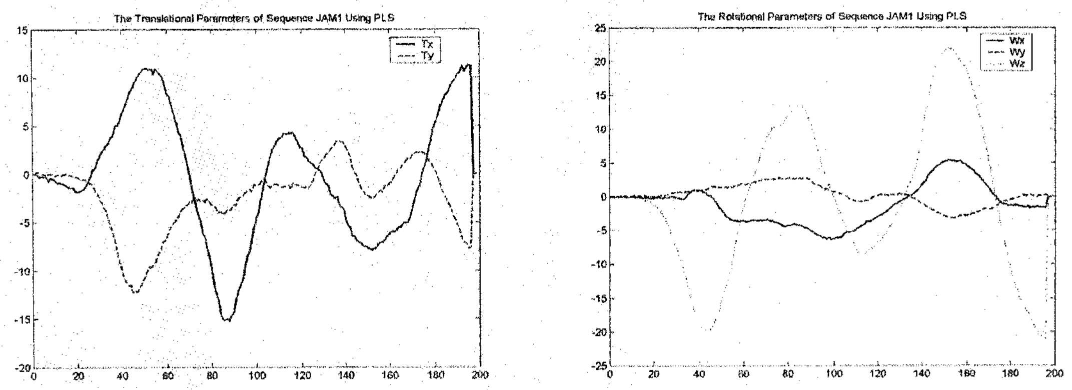


Figure appendix VI.1 The real face in the sequence Jam.avi and adapted face model



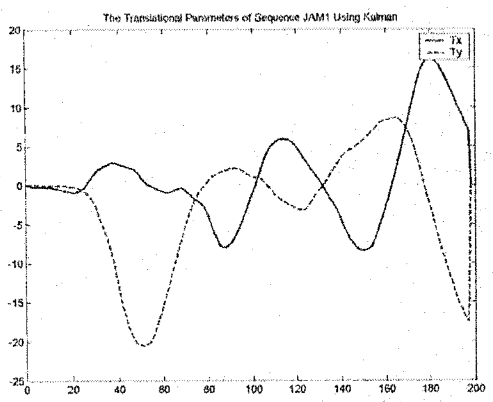
(a) translation (b) Rotation

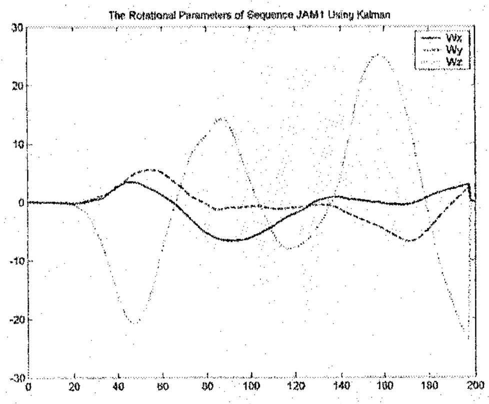
Figure appendix VI.2 The true motion parameters



(a) translation (b) Rotation

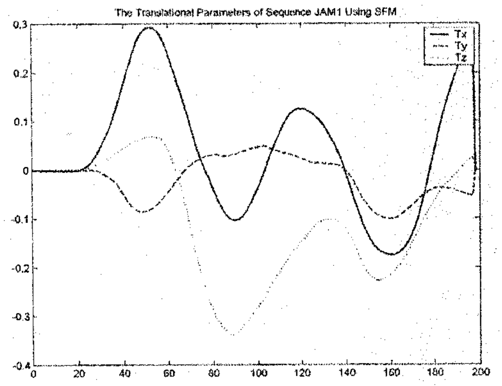
Figure appendix VI.3 The estimated motion parameters by PLS

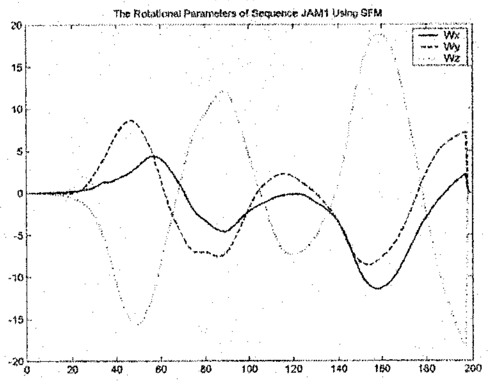




(a) translation (b) Rotation

Figure appendix VI.4 The estimated motion parameters by PLS-Kalman





(a) translation (b) Rotation

Figure appendix VI.5 The estimated motion parameters by SFM