Ecole doctorale Sciences pour l'Ingénieur N°ordre: 40340

Vers une suivi en temps réel de la position de la tête et du visage

THÈSE

présentée et soutenue publiquement le 24 Septembre 2010

pour l'obtention du

Doctorat de l'Université Lille1 Sciences et Technologies (spécialité informatique)

par

Haibo Wang

Composition du jury

Président :	Qing Yang	Chercheur á CASIA
Rapporteurs:	Vincent Lepetit	Chercheur principal à l'EPFL
	Hongbin Zha	Chercheur à l'Université de Pékin
Directeurs de thèse:	Christophe Chaillou	Professeur á l'USTL
	Chunhong Pan	Professeur á CASIA
Examinateur:	Franck Davoine	Chercheur au CNRS

Laboratoire d'Informatique Fondamentale de Lille - CNRS 8022

Abstract

Towards Online and Real-Time 3D Head Pose Tracking

Monocular 3D head tracking is a core technique for designing intelligent computer-human interfaces. Over the last decade, long-term tracking in complex environments remains a challenging problem. In this thesis, we investigate this problem by presenting two alternative frameworks and exploit its potential applications in computer-human interactions.

The first framework is a robust implementation of the differential tracking approach along with a 3D ellipsoid for geometric reasoning. It recursively estimates the head pose over time from prior prediction and dynamically updates its template model built beforehand. This makes it robust to appearance changes and lead to smooth estimations. However, they also bring two severe problems: the system can only handle slowly moving targets and the need for updating the model makes it prone to drift, which together make tracking over long periods of time impossible.

To avoid these important limitations, the second part of this thesis turns to a novel tracking by detection approach. Compared to the last approach, it requires an offline modeling and learning procedure, and performs tracking without the motion prior and dynamic updating. Tracking is actually made by matching features detected from the input images with the reference ones via a novel multi-view learning scheme. The learning relies on face texture synthesis to produce training examples, stable class detection and multi-view selection that are executed within a simple head modeling system. Extensive experiments show that this prevents drift while successfully tracking natural head motions. To further improve the performances, we also integrate optical-flow correspondences to enforce temporal consistency and incorporate color prior to identify possible outlier features. Finally, fusing all these components leads to a system that can be used for computer-human interactions.

Lastly, we present two applications of the proposed 3D head tracking system. The first estimates the user's gaze direction in the presence of natural head rotations. The second transfers facial expressions from a user to his online avatar. As a result, integrating these two functions into a virtual collaborative system greatly improves the communications of two remote partners.

Résumé

Vers une suivi en temps réel de la position de la tête et du visage

Le suivi en 3d des mouvements de la tête avec une seule caméra est un sujet important et difficile pour concevoir des interfaces informatiques. Dans cette thèse nous proposons deux logiciels et étudions leurs applications à l'interface Homme machine.

Le premier logiciel propose un suivi différentiel d'un ellipsoïde en 3D. Il estime récursivement la position de la tête avec des prédictions et mise à jour du modèle. Cette solution, bien que robuste, souffre de deux limites : les mouvements de la tête doivent être petits et il y a une importante dérive temporelle. D'autres pistes doivent être envisagées.

La deuxième partie de cette thèse se tourne vers une nouvelle solution à détection. La nouveauté est de conduire conjointement la modélisation, l'apprentissage et le suivi. Le suivi de position est réalisé en associant un suivi dynamique avec un apprentissage hors ligne des textures de visage utilisant un modèle 3D élémentaire de la tête. De nombreuses expériences valident que les mouvements de la tête sont suivi sans dérives. Pour améliorer les performances, nous avons ajouté un algorithme de flux optique pour renforcer la cohérence temporelle en éliminant les valeurs aberrantes.

Dans la dernière partie, nous présentons deux applications des algorithmes. La première vise à estimer le regard à partir de la rotation de la tête. La seconde permet d'afficher des avatars réalistes en plaquant la texture du visage extraite de la vidéo.

Acknowledgements

There are so many people who more or less contribute to making this thesis possible and to whom I owe deep gratitude.

My primary thanks are dedicated to my advisors, Prof. Christophe Chaillou and Prof. Chunhong Pan. They together figure out a perfect blend of big thinking and gave me the opportunity to work closely with them. Prof. Chaillou's deep insights in computer-human interactions guide me into this great research and many chats at his office impresses me with his humor, friendliness and thoughtfulness. Prof. Pan, with his deep understanding of computer vision, has always been there to guide me through the vision problems that obstruct me. I sincerely appreciate all your support, generosity, and, above all, patience.

I also like to say thank you to my thesis reviewers: Dr. Vincent Lepetit and Dr. Hongbin Zha. You suggestions greatly improve this document.

I have enjoyed working with Dr. Franck Davoine since I learned so much from your carefulness and your patience. I especially thank you, Dr. Vincent Lepetit, for the numerous insightful discussions and the elaborative paper corrections. The way that you do research teaches me a lot. I am also grateful to my colleagues, Li Ding and Jixia Zhang, for co-authoring papers and having many valuable discussions. Moreover, I would thank all my french collaborators, Damien Fournier, Jeremy Ringard, Vincent Enjalbert, Victor Gabillon, Guillaume Boulay, Frediric Roy, Julien Cantin, Md. Haidar Sharif and El Ghini, who are now also my good friends. I have been enjoying the special french dinner offered by Vincent Enjalbert and Victor Gabillon when I was going to leave Lille.

The most wonderful thing in my Ph.D. life is living in two cities with different cultures, Lille and Beijing, which enables me to make innumerable friends. I would like to express my gratitude to these friends who are either teachers or students at USTL or CASIA. I have been enjoying the amazing time in Lille with Quan Xu, Ling Shang, Tao Zeng, Xiaoli Su, Zheng Dai and Ling Peng for being foreigners together. I also would like to sincerely thank all my friends in our IGIT group: Ying Wang, Wei Li, Shiming Xiang, Chunlei Huo, Gaofeng Meng, Yiyi Wei, Jiangyong Duan, Lingfeng Wang, Shaoguo Liu, Xin Gu, Huxiang Gu and Jun Bai, for helping me and sharing your knowledge with me.

I give special thanks to my parents and my brother, for your so many years of love, care and encouragement. I especially thank my wife, Fengmei Chu, who is always on my side and comforts me whenever I got depressed or frustrated over the last five years. Without you, this thesis could not be finished.

Lastly, I gratefully acknowledge the French Embassy in China for financing my wonderful studies in USTL.

List of Figures

Contents

Chapter 1
Introduction

Understanding and reproducing human gestural communications is crucial for designing intelligent computer-human interfaces. Among the various semantic abilities, head gesture is a multi-functional one in that it takes charge of indicating human attentions and expressing semantic behaviors, such as head nodding and shaking. In the context of computer vision, 3D head pose/gesture tracking is to infer the 6 Degrees-of-Freedom (DOF) poses of a person's head relative to the view of a camera. In particular, the problem we will address refers to estimating continuous 6-DOF head poses from a monocular video sequence.

3D head pose tracking is intrinsically linked with two other tracking issues. On one hand, it is coupled with eye gaze estimation, which characterizes one's gaze direction and focus target. Physiological experiments have demonstrated that gaze behavior is the coordination result of eye and head orientations. On the other hand, it is fundamental to track non-rigid facial expressions. However, face appearance results the combination of a rigid pose and simultaneous non-rigid facial variations. In this sense, head pose tracking is indispensable to deliver gaze contact and drive facial animation of an avatar embedded in a virtual environment, such as the Second Life or 3D Poker. Such an example is illustrated in Figure 1.1, which is referred as *mixing video and avatar* remote collaboration environment. Thanks to head and gaze tracking, gaze focus and facial expression are simultaneously transferred from a natural human user to a distant avatar, which enhances remote communications in a collaborative virtual environment.

Before going further, we must clarify that the addressed 3D head tracking problem is not identical as several other related problems. First, it differs from estimating head poses on static images. Since there is no temporal prior, the latter has to depend on a strong prior model that is sufficiently trained beforehand. Second, it is also different from 2D face detection which at most have four in-plane parameters to estimate. Third, it is less challenging than facial action tracking since we have six pose variables to estimate while the latter has more non-rigid unknowns. By these comparisons, we may conclude that the addressed problem is a reasonable trade-off between 2D face tracking and 3D non-rigid face tracking. To solve this problem, learning from one instance (a reference images) is usually adequate and temporally coherent observations can be relied on, which are often incomplete and correlated with each other.

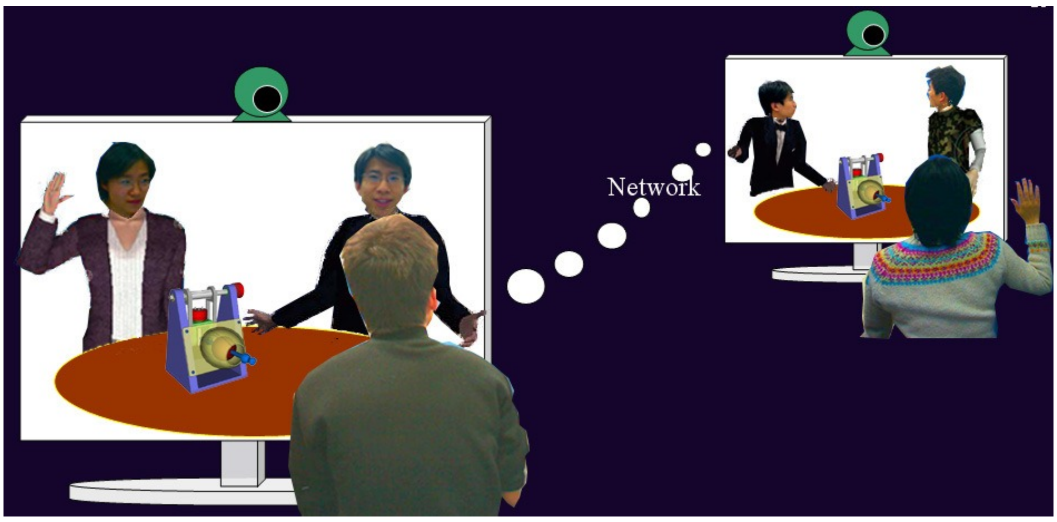


Figure 1.1: A *mixing video and avatar* collaborative virtual environment. In the environment, three distant partners are co-designing a vehicle component via network transmission. For better collaboration, the facial expressions and gaze information are transferred from each human being to his/her avatar by using 3D head tracking pose techniques.

1.1 3D Head Tracking Formulation

We start with addressing the formal definition of 3D head pose tracking from a monocular video sequence. A video sequence is a dynamic mirror of the real world with a camera as its eye. Suppose we have means to tell the computer a real object in world coordinate and its correspondent in video frame, the goal of head tracking is to recover the underlying structure that generates this correspondence. If we further suppose the camera's knowledge is known, the goal degenerates to estimating the motion mode of the target head from a geometric perspective. Mathematically, the problem can be formulated in the following compact way

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix}}_{\textit{Intrinsic matrix } \mathbf{A}} \underbrace{\begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}}_{\textit{Motion matrix } \mathbf{M}=[\mathbf{R}|\mathbf{T}]} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1.1}$$

where $\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^{\mathbf{T}}$ and $\mathbf{x} = \begin{bmatrix} u & v & w \end{bmatrix}^{\mathbf{T}}$ are a pair of 3D-to-2D correspondence in 3D and 2D homogeneous coordinates, respectively. Matrix $\mathbf{A}$ contains the known intrinsic camera parameters, i.e. the focal length f_u and f_v and the principal point (u_c, v_c). Matrix $\mathbf{M}$ is the motion matrix that we want to estimate, which consists of a 3×3 rigid rotational matrix $\mathbf{R}$ and a 3×1 translational matrix $\mathbf{T}$.

The image correspondents of target head are rendered as a set of pixels. In presence of image noise, the reprojected image point $\mathbf{A}[\mathbf{R}|\mathbf{T}]\mathbf{X}$ usually does not accurately equal to its corresponding pixel $\mathbf{x}$. Considering this, with a set of approximate correspondences, we estimate the pursing $\mathbf{M}$ via minimizing the so-called reprojection error

$$\arg\min_{\mathbf{R},\mathbf{T}} \sum_i \|\mathbf{A}[\mathbf{R}|\mathbf{T}]\mathbf{X}_i - \mathbf{x}_i\|^2. \tag{1.2}$$

which is a typical problem that has been well studied in 3D geometric vision. It is usually solved in two stages: first establish the 3D-to-2D correspondences and then estimate a 3D pose that best aligns these correspondences. In the following section, we will first present the state of the art.

1.2 State of the Art

3D head pose tracking has been an active area of research in computer vision for more than 20 years. Many approaches have been proposed to solve this problem. However, pure 3D head pose tracking approach is not easily found because pose tracking is always coupled with other problems, such as face detection, face alignment, facial action recognition and even face recognition.

Recently a very complete survey was presented in [Murphy-Chutorian 2009], which covers not only works on head detection and tracking but also the literature of techniques that try to solve some related subproblems. In contrast, we provide a more compact state-of-the-art review and do the classification more flexibly.

1.2.1 Model Flow Regularization Methods

This type of methods aims at interpreting image flow in terms of the motion parameters of a 3D model known in advance. DeCarlo & Metaxas [Decarlo 2000] treated optical flow as a hard constraint on the extracted motion of a 3D model in order to regularize 3D head pose as well as non-rigid motion tracking. Brand and Bhotika [Brand 2001] tracked the flows of the densely sample vertices from a 3D model to estimate rigid and nonrigid head motion in a recursive Bayesian framework. The flows are the measured optical flow that are regularized by a flexible 3D model. Zhang and Kambhamettu [Zhang 2002] addressed regularizing optical flow with a 3D model in the presence of partial occlusions. In this paper, they introduced a superquadric method to localize occluded areas, which is based on the estimated residual error filed.

The main limits come directly from the assumptions made during flow field computation. Most obvious is the assumption of brightness constancy during sparse optical flow computation. Major lighting changes can cause tracking failure. In addition, model flow cannot explain image flow when the image is rather noisy.

1.2.2 Template Matching Methods

Template matching takes the entire face appearance as a template and registers it with input ones to recover 3D pose. Since all template pixels are considered, it turns to be more robust than the flow regularization ones. La Cascia *et al.* [Cascia 1999] proposed to use a combined template of warped appearance texture and illumination basis. Similarly, Brown [Brown 2001] treated the texture mapped on a cylinder model as its appearance template. The confidence on each appearance pixel is also incorporated. Xiao *et al.* [Xiao 2003] presented a similar appearance-based method but applied iteratively re-weighted least squares (IRIS) to accommodate non-rigid motion and occlusion. Paterson and Fitzgibbon [Paterson 2003] casted tracking 3D pose with a 3D morphable model as a non-linear optimization problem such that various numerical methods can be applied. Dornaika and Ahlberg [Dornaika 2004] proposed a faster local search method to fit 3D morphable model for head pose tracking. Morency [Morency 2010] developed an adaptive appearance-based 3D head tracking system. To remove drift caused by the adaptation, keyframe selection scheme is integrated in the hope that drift-free static template can be chosen occasionally. Wu and Toyama [Wu 2000] used Gabor wavelet features to replace the direct appearance template for head pose estimation.

The representative template methods are the family of Active Appearance Models (AAMs), which couple appearance and face shape in a unified Principle Component Analysis (PCA) model. While the original AAMs was proposed for shape fitting [Cootes 2001], Matthews and Baker extended it to 2D face tracking by replacing the slow forward search with a more efficient Inverse Compositional (IC) alignment framework [Matthews 2004a]. To make AAMs handle global 3D head poses, Cootes *et al.* [Cootes 2000] introduced View-based AAMs to simultaneously estimate head orientations ahead of AAMs fitting. Alternatively, Xiao *et al.* [Xiao 2004] proposed a combined 2D+3D AAMs that treated the 3D modes as a constraint to regularize the general AAMs fitting. Both of these pseud 3D models may lead to ambiguous pose and shape estimates that are impractical in 3D space. In contrast, the 3D Morphable Model (3DMMs) [Blanz 1999a] are drawn from real 3D space and always produce plausible estimates. To make 3DMMs possible for tracking, its fitting speed can be also improved via IC alignment [S. 2003]. More recently, Dornaika and Davoine [Dornaika 2008a] took advantage of a simplified 3DMMs and achieved real-time 3D pose and facial action tracking performances within the framework of particle filter. The thorough comparisons between AAMs and 3DMMs can be found in [Matthews 2007].

Template matching estimates pose with simple observations. When face appearance is used as the template, updating it over time is cheap but could make the template more reasonable to interpret current face. On the other hand, since the template contains every pixel over the entire face region, matching it is usually performed within a local parameter space that is predicted by a standard prediction tool, such as Kalman filtering or Particle filtering.

1.2.3 Pure Learning Methods

Training neural networks for head pose tracking was popular thanks to its strengthes that are proven by many applications. Niyogi and Freeman [Niyogi 1996] learned a tree-structured vector quantizer (TSVQ) classifier for nearest-neighbor example indexing. Once learned, the classifier is no longer modified.Rae and Ritter [Rae 1998] proposed to train three neutral networks to enhance the robustness of head orientation estimation. Voit *et al.* [Voit 2006] also trained neural networks for determining 3D head poses in the scenes of meeting rooms.

In terms of the popularity of manifold learning, training a manifold that is embedded in view subspaces turns to a mainstream direction in 3D pose tracking. Morency *et al.* [Morency 2003b] learned intensity and depth view-based subspaces for 3D head pose tracking via SVD decomposition. Raytchev *et al.* [Raytchev 2004] learn a nonlinear manifold of a large set of training faces by using Isomap embedding method. Alternatively, Chen *et al.* [Chen 2003] proposed to learn the pose manifold by using the Fisher Discriminant Analysis (FDA). Liao and Medioni [Liao 2008] proposed to iteratively infer 3D pose and facial actions by decomposing the fields of facial expressions into a basis of eight 1D manifolds. Wu and Trivedi [Wu 2008] tested the PCA and Fisher LDA subspaces and their corresponding non-linear forms, KPCA and KDA for 3D head pose tracking.

The two learning methods have two common limitations. First is that once learnt, at testing stage, the model will be not adapted due to their complexity and high time consumption. Thus, it usually has a weak generalization capability in the sense that major unseen changes can seriously affect the query results. Second, the training data must be well collected and carefully labeled. When the data is rather noisy or the labeling is inaccurate, the learnt model is prone to being severely biased.

1.2.4 Local Feature Matching Methods

3D head pose tracking can be equivalently casted as the problem of estimating camera pose. The first way is to track and match online detected features to form a Perspective-n-Point problem to recover 3D head pose. Strom [Strom 2002] tracked online feature points and recovered 3D pose via Structure from Motion (SfM) by propagating covariance matrices in the framework of Kalman filter. Ohayon and Rivlin [Ohayon 2006] formulated head tracking as a wide-baseline feature matching and camera pose estimation problem. Jang *et al.* [Jang 2008] employed the efficient SIFT descriptor for this pose tracking task. Lepetit *et al.* [Lepetit 2004] integrated offline acquired feature points and online detected ones to perform head tracking via a keyframe selection scheme at each frame.

The alternate is to localize local semantic features to establish correspondences. Wang and Sung [Wang 2001] localized eye-lines and mouth-line and infer 3D head poses by detecting the vanishing points of these two lines. Robustness is achieved if the vanishing points were stably found. Ji and Hu [Ji 2002] localized the two pupil centers from special IR images to fit an ellipse to target face such that head orientations are estimated from the geometric distortions of the ellipse. Horprasert *et al.* [Horprasert 1996] tracked five semantic points (four eye corners and one nose tip) and employed projective invariance of the cross-ratios of the eye corners and anthropometric statistics to estimate head orientations. Hu *et al.* [Hu 2004] detected facial components to make use of the heuristic relation between facial appearance asymmetry and 3D geometry to recover 3D poses.

Combining semantic and online features can yield more robust results. Wang *et al.* [Zhang 2008] proposed to combine the annotated semantic features in 3DMMs and online features for deformable face tracking. The flexibility of online matches and stability of semantic matches are naturally combined to complement each other via an automatized RANSAC selection scheme.

1.2.5 Stereo Vision Methods

The aforementioned methods are all based on a single monocular camera. The nature of its illposedness can be partially offset by considering stereovision information. Seemann *et al.* [Seemann 2004] trained a neural network using the depth information acquired from a stereo camera to estimate head pose. Ruddarraju *et al.* [Ruddarraju 2003] used a multiple camera system to first localize eye region and then estimate 3D head poses. Harville *et al.* [Darrell 1999] derived a new depth constraint equation for 3D pose tracking by integrating depth information obtained from stereo cameras into the traditional linear brightness constraint equations. Moreno *et al.* [Moreno 2002] resided on fusing depth information gather from stereo vision and color histograms to track 3D poses. Yang and Zhang [Yang 2002b] made use of the additional epipolar constraint from the stereo image pair to improve 3D model-based head tracking results. By removing the stray points that violate this constraint, the accuracy of 3D/2D registration is improved significantly. Morency [Morency 2003a] presented a differential 3D head tracking approach to remove drift with stereo cameras.

Stereovision provides more information that are useful to 3D head pose tracking. The cost is also obvious that it must pay more to buy a stereo camera. Additionally, the stereo camera must be carefully calibrated such that the yielded depth information is accurate.

1.2.6 Hybrid Methods

Hybrid methods are those that combine at least two of the aforementioned methods to estimate 3D pose. This category aims to extract the merits of the elementary methods such that they are complementary to each other. Sung *et al.* [Sung 2008] combined AAM and cylinder models for both facial action and face pose tracking. The outcomes of cylinder flow-based tracking is used to initialize an AAM. Nikolaidis and Pitas [Nikolaidis 2000] integrated curve detection, template matching, active contour models and geometric information into a unified framework to determine 3D head poses. Sherrah [Sherrah 2001] proposed a head tracking framework that fuses multiple visual cues and multiple tracking modules. Murphy- Chutorian and Trivedi [Murphy-chutorian 2008] presented a hybrid head tracking system for driver assistance. To be real-time, the sampling in particle filter are implemented in GPU. Malciu and Preteux [Malciu 2000] combined 3D/2D appearance matching and optical flow measuring to track 3D head pose with a parametric model. Huang and Trivedi [Huang 2004] jointed head detection and pose tracking in a multiple visual cue integrated framework. Vogler *et al.* [Vogler 2007] combined the strengths of both 3D deformable models and ASMs to better handle non-rigid actions during 3D head pose tracking. There are also some methods that cannot be classified to any of the aforementioned categories. For example, Breitenstein *et al.* [Breitenstein 2008] proposed to estimate 3D head pose of an unseen face from a single range image.

Integrating many methods is a promising direction to explore since each elementary method can only take charge of one of the many challenges during 3D head pose tracking. However, building such a hybrid system needs to carefully investigate the intrinsic links between the components and usually do many implementations in parallel. Otherwise, the high complexity and source cost of a hybrid system would offset the benefits it brings and thus make it hardly usable in practice.

1.3 Challenges and Motivations

Recovering 3D pose from monocular video sequence is ill-posed since only 2D observations are given. These observations are incomplete as target face is projected from 3D to 2D space. In practice, more challenging is that these incomplete observations are disturbed by many noises:

- **Illumination variations.**
- **Simultaneous non-rigid facial expressions.**
- **Occlusions and self-occlusions.**
- **Cluttered or ambiguous background.**

To face these difficulties, there have reached a widely accepted framework after many years' explorations. But the uncertainties of these challenges bring three new dilemmas to our implementation. The pair of factors in each dilemma is collaborative yet competitive and balancing them amounts to playing a seesaw, which is illustrated in Figure 1.2:

- **Dilemma 1: Offline Learning vs. Online Adaptation**
- **Dilemma 2: Neighboring Search vs. Full-scale Search**
- **Dilemma 3: Global Feature vs. Local Feature**

In Dilemma 1, offline learning means learning a model ahead of tracking while online adaptation is to dynamically update this model. The more complicated the learnt model is, the less adaptable it is, e.g. a planar template is much easier to update than the AAMs. Comparatively, learning-biased tracker has weak generalization ability while adaptation-biased one is prone to model drift. In Dilemma 2, neighboring

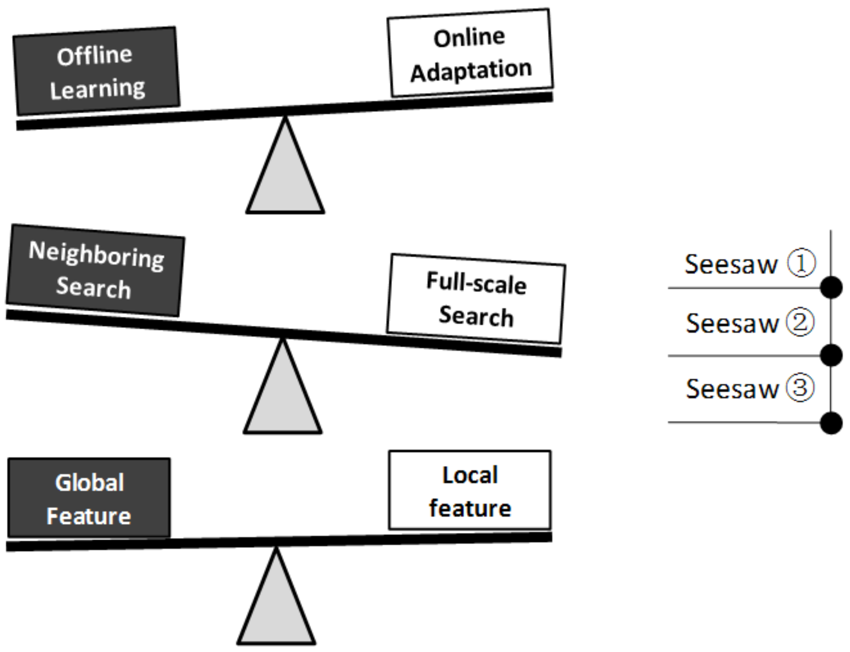


Figure 1.2: The three dilemmas that a 3D head tracking algorithm has to balance.

search specifies searching a subset of input image while full-scale search represents searching the entire image. By definition, neighboring search seeks a local minimum and is fast while a full search aims at the global optimum and therefore needs more computation time. In tracking, the neighboring region is usually predicted from the temporal prior knowledge. In Dilemma 3, global feature is extracted from the whole face while local feature specifies local semantic feature or online detected one. Comparatively, global feature is computationally cheaper, occlusion-sensitive while local one is more robust and accurate. These three dilemmas compromises our so-called technical challenges that any head tracking system must be confronted with. The work in this thesis proceed around how to overcome these dilemmas.

1.4 Thesis Contributions

Guided by the above analysis, we aim at developing a usable 3D head tracking system that automatically provides an individualized head model with few training images, and yields accurate, real-time and robust tracking results with a monocular camera. By doing so, we can not only use it for many vision applications, but also attempt to showcase the possibilities of our used technologies to solve this head tracking problem such that future investigators can learn the experiences. In more detail, our contributions can be described as follows:

- We first present a robust implementation of the differential-based tracking framework that relies on a global feature, instant online adaptation and prediction-based neighboring search. We start with a generic ellipsoid model for 3D geometric reasoning and take its mapped texture at the first frame as the template. For each successive frame, 3D head pose estimate starts with the value estimated at previous frame and then is iteratively refined by minimizing template discrepancy with the steepest descent optimization approach. An extensive number of experiments were conducted to reveal the strong and weak points of this framework, which motivate our next work.

- We develop an interactive modeling system to generate individualized head model in expecting better geometric reasoning than the ellipsoid model. It is simple in that we adapt a generic head model to a specific person via a unified Moving Least Squares (MLS) interpolation scheme constrained by a set of anchor points interactively placed by users. The MLS is limited to be as rigid as possible to allow the integrity of mesh triangles. The surface smoothness is also guaranteed via a carefully selected smooth kernel. Our experiments show both geometrically and photometrically realistic textured head models.

- We propose a new Tracking-by-Detection (TbD) framework based on matching local keypoints. By attaching a descriptor to each keypoint, our key idea is treating the matching as a classification problem, where all possible views of such a descriptor constitutes a single class. In our framework, the 3D head tracking is preceded by 1) learning a classifier via view set synthesis, 2) solving for 3D-to-2D correspondences via keypoint classification, and 3) using the correspondences to estimate an optimal 3D transform. Given only one instance of each class, we synthesize more training data under the simulations of 3D rigid pose, non-rigid shape and illumination variations with texture mapping. With these view set, the classifier is trained to be robust to shape, pose and illumination changes. Extensive experiments show that this framework successfully gets rid of the drift problem while successfully tracking natural motion. Then, to remove motion jitters, we also establish an additional set of 3D-to-2D smooth correspondences by measuring sparse optical flow and incorporate them as a smoothness constraint. This leads to an improved TbD solution that is different from the differential tracker by using temporal prior as a soft aid rather than a hard prediction. Finally, in order to reduce outlier correspondences, we learn a color prior from a reference image and compare it with the color information of each detected keypoints. To make it discriminative, we make use of the covariant color representation of a local keypoint and the stored posteriors of a powerful Random Projection (RP) trees.

- Lastly, we present two applications of 3D head pose tracking. First is estimating eye gaze in the presence of natural head rotations. Our major contribution lies in deriving a simplified linear relation between head orientation and eye gaze within a pinhole camera model. The second is transferring facial expressions from natural human being to an online avatar within a prototyping collaborative virtual environment.

1.5 Thesis Organization

There are seven chapters in the thesis that systematically introduce our work from idea inspiration, algorithm implementation to system application. The remaining six chapters are organized as follows. In Chapter 2, we present the recursive tracking framework and focus on analyzing its drawbacks given a set of experimental results. Chapter 3 introduces the human-assisted head modeling system. Chapter 4 presents the proposed wide-baseline tracking by detection scheme with the help of an individualized head model. Chapter 5 describes our methods to enforce temporal consistency by incorporating optical-flow correspondences and reject a large number of outlier correspondences by considering color as a discriminative prior. Chapter 6 presents two possible applications of our tracking system - estimating eye gaze in the presence of head rotations and performing facial expression transfer in a network-mediated collaborative environment. Chapter 7 gives a summary of the mentioned work, discusses the remaining limitation and forecasts the potential directions for further improvements.

CHAPTER 2

A Model-Based Short-Baseline Solution

3D head pose tracking aims at continuously recovering the six degrees of freedom of a human head. Based on motion continuity in video sequence, the short-baseline framework is the most popular solution even up to today. It transfers the estimated poses at previous to the next one to provide pose or measurement predictions. In this chapter, we develop such a short-baseline solution by deriving technical components from state-of-the-art systems. Before tracking, we use a generic ellipsoid model to mimic target head and then employ an adaptive appearance-based scheme to track head pose in monocular sequence. In addition, we also integrate robust techniques to handle outliers and occlusions occurring to face appearance.

In the following, we start with formulating our solution in the popular Bayesian tracking framework. Then we describe the proposed solution in detail and finally analyze its efficiencies and drawbacks by doing a set of experiments on realistic videos.

2.1 Bayesian Formulation

The visual tracking problem is usually formulated in terms of a dynamical process, aiming at estimating the density of the state sequences $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$ based on the corresponding image measurements $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$. At a discrete time t, the goal is to estimate the state $\mathbf{x}_t$ from the available observations $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$. From a Bayesian perspective, tracking is to recursively calculate some degree of belief in the state $\mathbf{x}_t$ at time t, given the measurements $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$ up to time t. To do this, it is required to construct the probability density function $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ and estimate it via Bayes's rule:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1}). \tag{2.1}$$

where $p(\mathbf{z}_t|\mathbf{x}_t)$ is the likelihood function and $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is the prior probability, from which the density propagation rule (2.1) starts. Based on the assumption of Markov process, $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ can be calculated from the Chapman-Kolmogorov equation

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}, \tag{2.2}$$

where $\int$ is the integration over the set of possible values for the previous state $\mathbf{x}_{t-1}$. The term $p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ is the previously estimated density function and

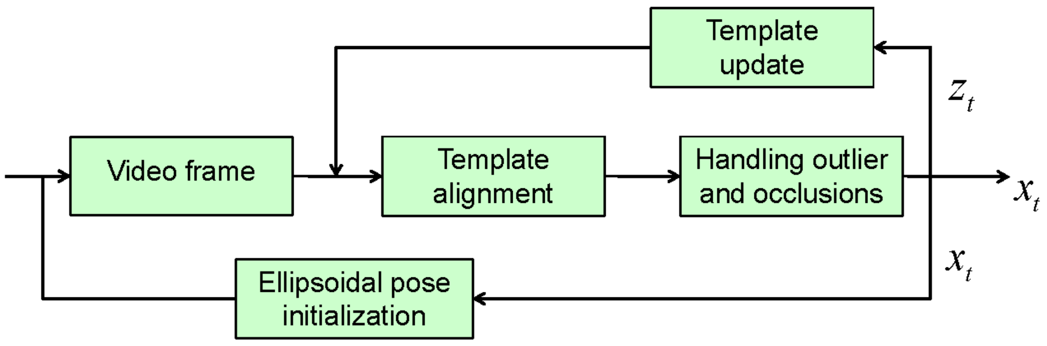

Template update
z_t
Video frame
Template alignment
Handling outlier and occlusions
x_t
x_t
Ellipsoidal pose initialization

Figure 2.1: The model-based short-baseline tracking framework. By starting with the pose estimated in previous frame, the approach proceeds by 1) aligning the template and the face candidates at current image, 2) handling outliers and occlusion and 3) updating the template.

$p(\mathbf{x}_t|\mathbf{z}_{t-1})$ represents the prediction probability of applying a certain motion model. Therefore, this term $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is a propagation of its previous density and the motion prediction. In this sense, $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be interpreted as a prediction density weighted by the observation density $p(\mathbf{z}_t|\mathbf{x}_t)$, which allows for taking into account the features observed in the incoming image.

This Bayesian formulation provides a statistically well-grounded manner for visual tracking. If we only consider Gaussian distributions, it leads to the well-known Kalman Filter and the Extended Kalman Filter. For more general distributions, it yields the so-called Particle Filters. Each particle actually represents a hypothesis of the state vector.

2.2 Framework Overview

3D head pose tracking is a specific problem in visual tracking. It is thus natural to cast it into the Bayesian formulation. To simplify our solution, we ignore the prior density $p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$ is uniform and only consider the likelihood one $p(\mathbf{z}_t|\mathbf{x}_t)$. To enforce temporal consistency, the predicted $\mathbf{x}_t$ is supposed be to around the previously estimated one $\widehat{\mathbf{x}}_{t-1}$. Therefore, the optimal $\widehat{\mathbf{x}}_t$ is determined via the Maximum-Likelihood (ML) estimate

$$\widehat{\mathbf{x}}_t = \arg\max p(\mathbf{z}_t|\mathbf{x}_t). \tag{2.3}$$

When we consider the Gaussian distribution

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto \mathrm{N}(\mu, \mathbf{\Sigma}), \tag{2.4}$$

the ML estimator is equivalent to minimize the following Mahalanobis distance

$$\widehat{\mathbf{x}}_t = \arg\min(\mathbf{z}_t(\mathbf{x}_t) - \mu_t)^T \mathbf{\Sigma}^{-1}(\mathbf{z}_t(\mathbf{x}_t) - \mu_t). \tag{2.5}$$

where μ_t and $\mathbf{\Sigma}$ are the mean vector and covariance matrix of image measurements, respectively. We suppose that they have been known before tracking and are time-varied. In this sense, our problem turn to a standard template matching process, from which pose can be recovered via regression.

In the context of 3D head tracking, the state vector $\mathbf{x} = [x, y, z, \alpha, \beta, \gamma]^T$ and the measurements $\mathbf{z}$ is the frame image. The entire framework is summarized in Figure 2.1, which is composed of two stages. During the offline stage, we build a parametric 3D model to mimic head and adapt it in the beginning of the tracking. During the online stage, we perform the above template matching to produce head pose and update the template parameters dynamically.

2.3 A Parametric Ellipsoid Model

Since image is the 2D projection of our 3D world, tracking 3D head pose from it is a ill-conditioned problem. To solve it, we build a 3D ellipsoid model to mimic the head target and cast our problem as estimating the motion vector $\mathbf{x} = [x, y, z, \alpha, \beta, \gamma]^T$ of this model in each frame. As shown in Figure 2.4, the model is parameterized by a set of triangles, whose vertices are equally sampled from the model surface. Figure 2.2 shows the 3D motion flow fields of an ellipsoid. Their different appearances in 2D view make 3D head tracking possible by finding the best motion parameters that yields a given appearance.

To each individual, the ellipsoid to mimic his head must be specified, i.e. its three major axes and center coordinates must be given. To do this, we use two separate operations. Before tracking, we capture two views of a human face and adapt a generic ellipsoid according to the approximate shapes of his face shown at the two view. At the same time, some feature points are automatically detected using SIFT keypoints [Lowe 2004a]. Subsequently, in the beginning of tracking, if we assume the face is at frontal view, the above fitted parameters can be adapted by matching the learned and the newly detected SIFT feature points.

2.4 The Tracking Algorithm

2.4.1 2D Appearance Measurements

With a built ellipsoid, our head tracking objective turns to recovering its motion vectors $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\}$ from a video sequence $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_n\}$ depicting a moving face. To estimate a discrete pose $\mathbf{x}_t$, all the available observations $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$ should be used to provide information as more as possible. We regard a full input frame as $\mathbf{z}$. In detail, we first project the ellipsoidal model onto the 2D image plane to get a 2D triangular mesh and then synthesize the texture by grabbing pixels from the face region in input image covered by 2D triangular mesh using a piece-wise affine warping function $\mathbf{\Psi}$. Mathematically, the warping function $\mathbf{\Psi}$ applied to an input image $\mathbf{z}_t$ is denoted by:

$$\mathbf{y}_t = \mathbf{y}(\widehat{\mathbf{x}}_t) = \mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_t) \tag{2.6}$$

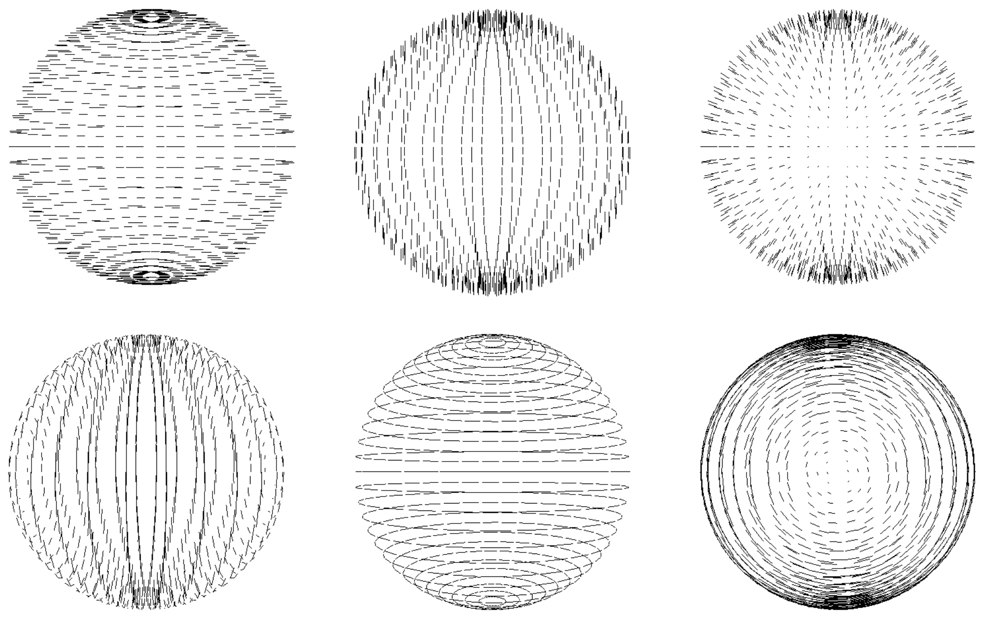


Figure 2.2: The 3D motion flow fields of an ellipsoid. The upper row shows the translational flows along x, y, and z axes, respectively. Similarly, the lower displays the rotational flows around x,y and z axes.

where $\mathbf{y}_t$ represents the view-free texture vector, which is warped from input image and mapped on the ellipsoid model at frontal view and $\mathbf{x}_t$ is the estimated head pose at time t. To use $\mathbf{y}$ as a template, we denote $\mathbf{A}_t$ as an adaptive view-free texture that models all the view-free textures $\mathbf{y}$ up to time $t-1$.

The way to compute the piecewise affine warp is illustrated in Figure 2.4. Consider the pixel $(s,t)^T$ in the triangle $(s_1,t_1)^T$,$(s_2,t_2)^T$ and $(s_3,t_3)^T$ in the texture plane. This pixel is uniquely expressed as:

$$(s,t)^T = (s_1,t_1)^T + \alpha[(s_2,t_2)^T - (s_1,t_1)^T] + \beta[(s_3,t_3)^T - (s_1,t_1)^T] \tag{2.7}$$

where $(\alpha,\beta)^T$ is the barycentric coordinates of $(s,t)^T$ and satisfies $0 \leq \alpha \leq 1$, $0 \leq \alpha \leq 1$. Then the piecewise affine warping function is written as:

$$\Psi(\mathbf{z},\mathbf{x}) = (u,v)^T = (u_1,v_1)^T + \alpha[(u_2,v_2)^T - (u_1,v_1)^T] + \beta[(u_3,v_3)^T - (u_1,v_1)^T] \tag{2.8}$$

where $(u_1,v_1)^T$,$(u_2,v_2)^T$ and $(u_3,v_3)^T$ are the vertices of the corresponding triangle in $\mathbf{z}$. The barycentric coordinates of each pixel can easily be computed from Equation 2.7. Once they are known, we could store them to perform a simple lookup during tracking. But this occupies a large number of memory storages. Alternatively, as described in [Iain 2004], we could only compute the affine warp coordinates of each vertex and re-compute the barycentric coordinates of each pixel via simple affine multiplications and additions during tracking.

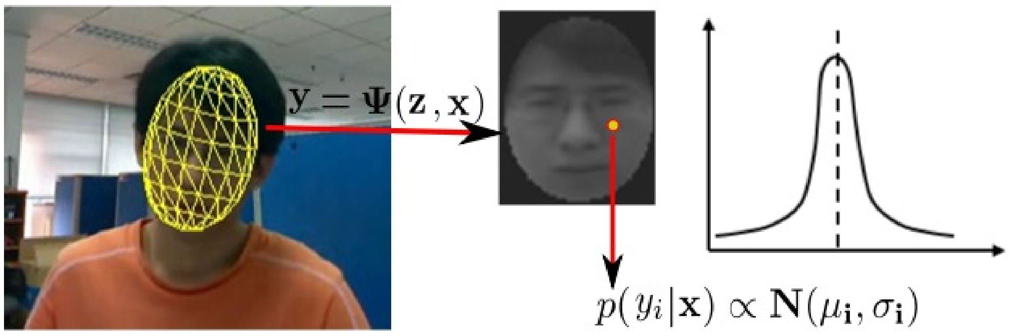


Figure 2.3: The definition of the appearance measurements. The view-free texture is the warping outcome of $\mathbf{\Psi}$. The intensity of each pixel within it is assumed to follow an independent Gaussian distribution.

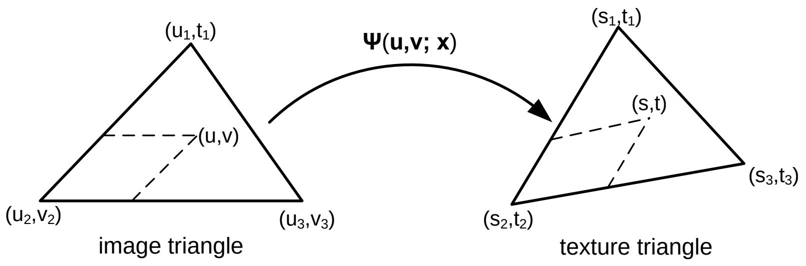


Figure 2.4: Computing the warping function $\mathbf{\Psi}$ in a triangle. The vertex correspondences are established via model projection while the inside ones are computed via piece-wise linear interpolation.

To simplify future computation, the graylevel of each pixel within $\mathbf{y}_t$ are assumed to be mutually independent and follows a single Gaussian distribution: each pixel y_i is modeled by a mean value μ_i and a deviation value σ_i. Such a Gaussian assumption can partially explain the appearance variations caused by facial animation, illumination changes and other factors. Thus, a multivariate Gaussian is used to represent the view-free texture vector $\mathbf{y}_t$

$$P(\mathbf{z}_t|\mathbf{x}_t) = \prod_{i=1}^{n} N(y_i : \mu_i, \sigma_i)_t, \tag{2.9}$$

where $N(y_i : \mu_i, \sigma_i)_t$ is the single Gaussian distribution and n is the size of $\mathbf{y}_t$. As we will show later, both μ_i and σ_i are dynamically updated as new image measurements are known. For simplicity, we will use $\mathbf{m}$ to represent the mean vector $\mu_{1:n}$ and $\mathbf{\Sigma}$ for the diagonal covariance matrix of $\sigma^2_{1:n}$.

2.4.2 Gaussian-Newton Image Alignment

Given the Gaussian assumption, the cost function to stand for the discrepancy between our template $(\mathbf{m}_t, \mathbf{\Sigma})$ and an input texture $\mathbf{y}_t$ is given as their *Mahalanobis* distance

$$\Delta \mathbf{x}_t = \arg\min(\mathbf{y}_t - \mathbf{m}_t)^T \mathbf{\Sigma}^{-1}(\mathbf{y}_t - \mathbf{m}_t) \tag{2.10}$$

where $\mathbf{\Sigma}$ is the diagonal covariance matrix. This equation is a non-linear problem as the appearance vector $\mathbf{y}_t = \mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_t)$ has an unknown form. To make the problem tractable, we may assume $\mathbf{y}_t$ as a linear function of the estimated $\widehat{\mathbf{x}}_{t-1}$ and use its first-order Taylor expansion to replace it. That is, we expand $\mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_t)$ around $\widehat{\mathbf{x}}_{t-1}$ to a first-order Taylor expansion and obtain

$$\mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_t) \simeq \mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_{t-1}) + \mathbf{J}_t \Delta \mathbf{x}_t. \tag{2.11}$$

where $\mathbf{J}_t$ is the Jacobian matrix of Ψ and $\Delta \mathbf{x}_t$ is the pose displacement of $\mathbf{x}_{t-1}$. With this simplification, the objective function becomes

$$\arg\min[\mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_{t-1}) + \mathbf{J}_t \Delta \mathbf{x}_t - \mathbf{m}_t]^T \mathbf{\Sigma}^{-1}[\mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_{t-1}) + \mathbf{J}_t \Delta \mathbf{x}_t - \mathbf{m}_t]. \tag{2.12}$$

By taking derivatives with respect to $\Delta \mathbf{x}_t$ and setting it to zero we have

$$\mathbf{J}^T \mathbf{\Sigma}^{-1} \mathbf{J} \Delta \mathbf{x}_t = \mathbf{J}^T \mathbf{\Sigma}^{-1}[\mathbf{m}_t - \mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_{t-1})], \tag{2.13}$$

which is also known as the normal equation. Solving it yields the following closed form solution

$$\Delta \mathbf{b_t} = (\mathbf{J}^T \mathbf{\Sigma}^{-1} \mathbf{J})^{-1} \mathbf{J}^T \mathbf{\Sigma}^{-1}[\mathbf{m}_t - \mathbf{\Psi}(\mathbf{z}_t, \mathbf{x}_{t-1})]. \tag{2.14}$$

We can see that it corresponds to solve a weighted least-squares problem. In practice, the matching error E of this linearized formulation is not always the minimum one. To reach a minima, we run a steepest searching process via gradient-descent optimization. Starting with $\mathbf{x}_0 = \widehat{\mathbf{x}}_{t-1}$, we derive an initial shift $\Delta \mathbf{x}$ and a matching error $E(\mathbf{x})$ from Equations (2.14) and (2.10). Following it, we update the motion vector and compute a new matching error according to

$$\begin{cases} \mathbf{x}' = \mathbf{x} + \rho \Delta \mathbf{x} \\ E' = E(\mathbf{x}') \end{cases} \tag{2.15}$$

where ρ is a positive real. If $E' < E$ we accept this update and repeat the template matching until converged; Otherwise, a smaller ρ will be used and iterate the operations (2.15) until $E' < E$. If the matching error cannot be improved anymore or a certain iteration number is reached, convergence is declared and the search stops.

Since the above search is performed around the neighborhoods of $\widehat{\mathbf{x}}_{t-1}$, the solution usually falls into a local minimum instead of the global one. In this sense, this template matching is a compromise between fast computation and global optimum. It can be close to the global minima only if the linearization is a good approximate.

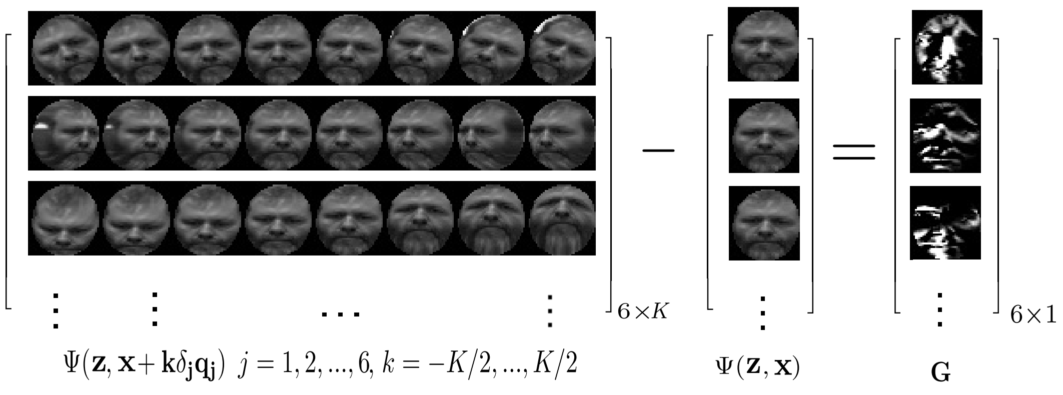

$6\times K$
6×1
$\Psi(\mathbf{z},\mathbf{x}+\mathrm{k}\delta_{\mathrm{j}}\mathbf{q}_{\mathrm{j}})$ $j=1,2,...,6,\ k=-K/2,...,K/2$
$\Psi(\mathbf{z},\mathbf{x})$
$\mathbf{G}$

Figure 2.5: Illustrating the numerical way to compute the gradient matrix $\mathbf{J}$.

To enhance its robustness, the Jacobian matrix $\mathbf{J}_t$ is computed via a set of its numerical lists. Suppose the j^{th} column of $\mathbf{J}_t$ is defined as

$$\mathbf{J}_j = \frac{\partial \Psi(\mathbf{z}_t, \mathbf{x}_t)}{\partial \mathbf{x}_j} \tag{2.16}$$

and its difference equation is given by

$$\mathbf{J}_j = \frac{\Psi(\mathbf{z}_t, \mathbf{x}_t + \delta \mathbf{q}_j) - \Psi(\mathbf{z}_t, \mathbf{x}_t)}{\delta}, \tag{2.17}$$

where δ is a suitable step size and $\mathbf{q}_j$ is the j^{th} unit vector of a $\dim(\mathbf{x}_t) \times \dim(\mathbf{x}_t)$ identity matrix. By using several steps around the current value $\mathbf{x}_t$, a more robust estimate of $\mathbf{J}_j$ is expressed as the average

$$\mathbf{J}_j = \frac{1}{K} \sum_{k=-K/2, k\neq 0}^{K/2} \frac{\Psi(\mathbf{z}_t, \mathbf{x}_t + k\delta_j \mathbf{q}_j) - \Psi(\mathbf{z}_t, \mathbf{x}_t)}{k\delta_j}, \tag{2.18}$$

where δ_j is the smallest perturbation associated with $\mathbf{x}_j$ and K is the size of this numerical spreading. A good set is $K = 8$ to trade off robustness and computational efficiency.

Even with a moderate K, e.g. 8, the time cost of this numerical computation remains a huge burden for real-time performance. Especially, with the gradient-descent search, as $\mathbf{x}$ is varied at each iteration, $\mathbf{J}_t$ has to be re-computed as well. This will lead to a occupation of the 80% tracking time. In practice, we do not update $\mathbf{J}_t$ during the iterative search. For further efficiency, as suggested in [Dornaika 2008a], $\mathbf{J}_t$ can be only re-computed by every five frames.

For better understanding, the visualizations of the numerical way to compute $\mathbf{J}_t$ and the composition of the first-order solution are shown in Figure 2.5.

2.4.3 Confidence Map

On warping the input image into the texture plane, not all pixels have equal confidences due to the nonuniform density of pixels. As proposed in [Cascia 1999], an approximate measure of the confidence can be derived in terms of the ratio of a triangle's area in the video image $\mathbf{z}$ over the triangle's area in the normalized texture. The underlying idea is that the amount of information carried by a triangle is directly proportional to the number of pixels it contains in the input image $\mathbf{z}$. Consider a triangle whose vertices in image coordinates are (u_1, v_1), (u_2, v_2) and (u_3, v_3), and in texture coordinates are (s_1, t_1), (s_2, t_2) and (s_3, t_3). The ratio is easily given as

$$c = \frac{\sqrt{(u_2 - u_1)(v_3 - v_1) - (v_2 - v_1)(u_3 - u_1)}}{\sqrt{(s_2 - s_1)(t_3 - t_1) - (t_2 - t_1)(s_3 - s_1)}}. \tag{2.19}$$

Parts of the texture corresponding to the nonvisible part of ellipsoid surface would have zero confidence as they contribute no pixels. In practice, since only a 180° part of the ellipsoid is visible at any instant, we could simply render a half ellipsoid as our head model.

If we reform the confidence map into a diagonal matrix $\mathbf{W}(\mathbf{s}, \mathbf{t})$ to weight the contribution of each texture pixel, the original cost function in equation (2.10) turns to a weighted *Mahalanobis* distance measure

$$\Delta\mathbf{x}_t = \arg\min(\mathbf{y}_t - \mathbf{m}_t)^T \mathbf{W}\boldsymbol{\Sigma}^{-1}\mathbf{W}(\mathbf{y}_t - \mathbf{m}_t), \tag{2.20}$$

where $\mathbf{W}$ is the diagonal weight matrix, with the j^{th} diagonal element corresponding to the confidence value c_j of the j^{th} pixel of $\mathbf{m}_t$. Correspondingly, the weighted least-squares solution becomes

$$\Delta\mathbf{x}_t = (\mathbf{J}^T\mathbf{W}\boldsymbol{\Sigma}^{-1}\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}\boldsymbol{\Sigma}^{-1}\mathbf{W}[\mathbf{m}_t - \boldsymbol{\Psi}(\mathbf{z}_t, \widehat{\mathbf{x}}_{t-1})]. \tag{2.21}$$

2.4.4 Handling Outliers and Occlusion

To partially offset the influence of outliers that are not modeled by the current template, we adopt a simple robust statistics function. For the i^{th} pixel x_i, if $|\frac{x_t - \mu_t}{\sigma_t}| < c$, then it will be announced as an outlier; otherwise, it is retained as an inlier. For the outliers, we don't use them to update the mean vector and the covariance matrix. For the inliers, we update them by equation (2.24). In our experiments, a proper value for c is 3.

As the prediction-based tracking approach always suffers from the drifting problem, to define a strategy to correct it is indispensable. Iain Matthews [Matthews 2004b] have proposed a simple strategy to partially correct this problem. While the registration error $\mathbf{e}_m$ is bigger than a threshold value $\mathbf{e}_{drift}$, we say that drifting happens and correct the dynamic template by the initial one. As our template is a multivariate Gaussian distribution, the mean pixel vector $\mu_{1:n}$ in current appearance template $\mathbf{A}_t$ is corrected as follows:

$$\mathbf{A}^*(\mu_{1:n})_t = (1 - \theta)\mathbf{A}(\mu_{1:n})_t + \theta\mathbf{A}(\mu_{1:n})_0 \tag{2.22}$$

$\mathbf{A}(\mu_{1:n})_0$ is the initial template and θ is a weight value. The accumulative covariance matrix is reset to be an identity matrix. Then the template matching will be run again with the above corrected template. The re-estimated head pose is denoted as $\mathbf{x}_t^*$, which will replace the previous $\mathbf{x}_t$.

Occasionally, due to full occlusions or many other factors, the tracking may completely fail. In these cases, the tracking must be re-initialized. We do this by assigning a big threshold E_m to bound the template matching error. If it exceeds the given threshold, the tracking is probably failing and will have to be re-initialized by matching SIFT keypoints.

2.4.5 Model Update

The estimated pose displacements $\Delta\mathbf{x}_t$ can be directly used to derive the current pose:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta\mathbf{x}_t. \tag{2.23}$$

Meanwhile, once the above template matching is finished, we can use the derived $\mathbf{x}_t$ to synthesize a new view-free texture to update our Gaussian template. Similar to the updating of Online Appearance Models (OAMs), our updating scheme is simply as:

$$\mu_{i(t+1)} = (1-\alpha)\mu_{i(t)} + \alpha_{i(t)} \tag{2.24}$$

$$\sigma^2_{i(t+1)} = (1-\alpha)\sigma^2_{i(t)} + \alpha(x_{i(t)} - \mu_{i(t)})^2 \tag{2.25}$$

where $\alpha = 1 - exp(-\log(2/n_h))$ is a forgetting factor, which weights past observations in recursive filters, and n_h represents the half-life of the envelope in frames. This links our approach to the recursive filter in online signal processing. In practice, equation(12) is not used until a stable variance σ^2 is reached, e.g. after the first 50 frames. Conversely, in the first 50 frames, a naive $\alpha = a/t$ is set to update the variance (a makes sure $\alpha \in (0.01, 0.1)$). Thus, the first 50 frames are for both tracking and learning. Note that pose variations are restricted during this learning stage for reliability.

2.5 Experiments and Discussions

We only conducted qualitative experiments to evaluate the presented head tracking approach. There are in total five video clips captured for this purpose, namely, ‘Shangling’ ‘Haibo1’ ‘Fredric’ ‘Haibo2’ and ‘Jeremey’. The clip ’Shangling’ is captured via a hand-held camera while all the others are grabbed by a Logitech webcam. Their respective resolutions are 640×480, 320×240, 640×480, 640×480 and 640×480. On experiments, our unoptimized C/C++ code runs at about 15 frames/second on a Xeon 3.0GHz, 1.5G RAM computer, with setting $K = 8$ and the template resolution to 40×42. Almost 80% of the tracking time is occupied by computing the numerical gradient matrix at each iteration.

Figure 2.6 shows two tracking snapshots with the ’Shangling’ clip. The built ellipsoid is not well-shaped since his face is rather planar. Faces are successfully tracked most of the time. Figure 2.7 exhibits several tracking shots sampled from the low-resolution 'Haibo1' sequence. The used global template is rather robust to the blurring face observations. On the other hand, the subject moves very slowly to ensure the correctness of the motion prediction.

In Figure 2.8, the tracking approach is evaluated in facing the challenge of out-of-plane rotations. The first two rows corresponds to nice results with negligible pose drifts, while the last manifests visible drifts. For better illustration, Figure 2.9 reveals the appearance drift caused by self-occlusions and facial expressions. When the drifted appearance is temporally integrated into the template, template matching will produce an error-prone pose estimate, which in turn introduces a new erroneous appearance. Finally, when the error of this loopy template-pose drift accumulates to a certain degree, obvious tracking failures are observed as happens in the cases of the last row in Figure 2.8.

Figure 2.10 shows the shots on tracking three challenging cases, that is, occlusions, putting off glasses and moderate illumination variations. Since no large head motions simultaneously happens, the faces remains being tracked. Similar results are also demonstrated in Figure 2.11 and Figure 2.12 on the 'Jeremey' clip.

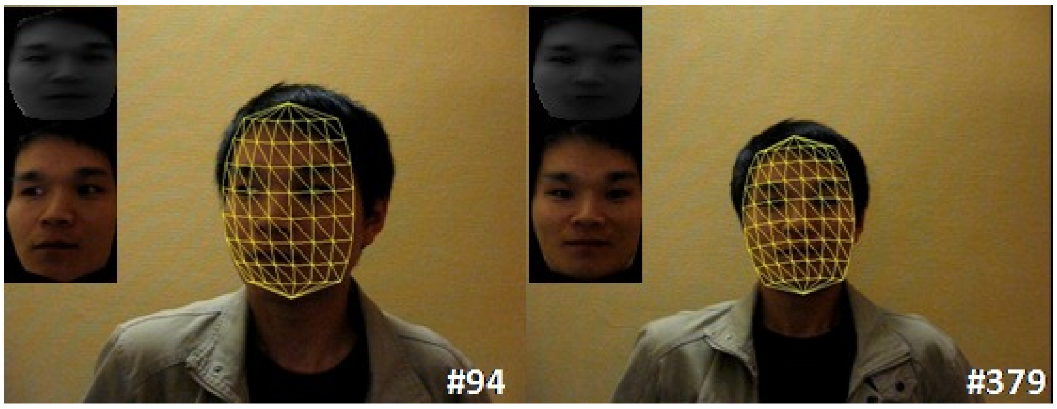


Figure 2.6: Two tracking shots on the 'Shangling' sequence. The appearance template in gray scale and the current appearance feature in color scale are shown on top left corner.

2.5.1 Discussions

The presented approach deals with the three dilemmas (Chapter 1) in a traditional way. Verified by the experiments, its drawbacks can be summarized as follows:

First, the approach is biased for online adaptation while against offline learning. The only fixed information is the template extracted at the first frame. The dynamic template updating makes the tracking error accumulated and comes back the anchoring template only if pose was at neutral view again. In addition, the online thresholding method to reject outlier and occlusions by its nature cannot overcome

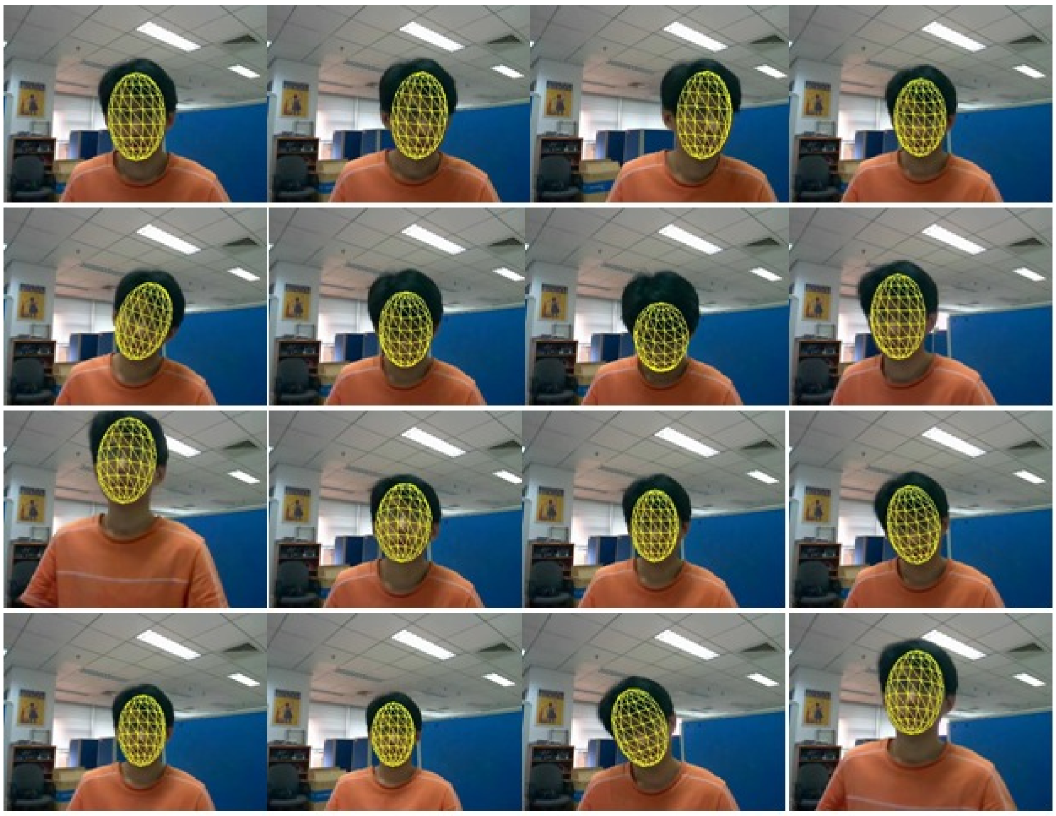


Figure 2.7: Tracking snapshots on the 'Haibo1' sequence. The sequence is captured at a lower 320×240 resolution.

the difficulty.

Second, the approach realizes neighboring search based on the predictions from temporal prior knowledge. This significantly limits the allowed speed of the tracking target and raises the possibility of drifts. In front of a camera, user must move very smoothly, which makes this method nearly unusable in practice. The better prediction scheme, e.g. Kalman filter or Particle filter, with a high-order dynamic equation would not solve this problem. A promising direction is to perform full-scale search without no longer depending on temporal predictions.

Third, despite its robustness, the global appearance feature is less accurate and occlusion-suffering. Some pixels or subregions within it may be less reliable, but make equal contributions to the pose estimate in the global-based scheme. This brings many difficulties to error measurements. Local features, decomposed from the global one, must be considered in the first place such that global feature can provide good reasoning to accept or reject certain local-based results.

In order to overcome these drawbacks, in the following chapters, we will present a totally different tracking approach that employs an individualized head model, relies more on offline learning, substitutes global template with local robust feature and

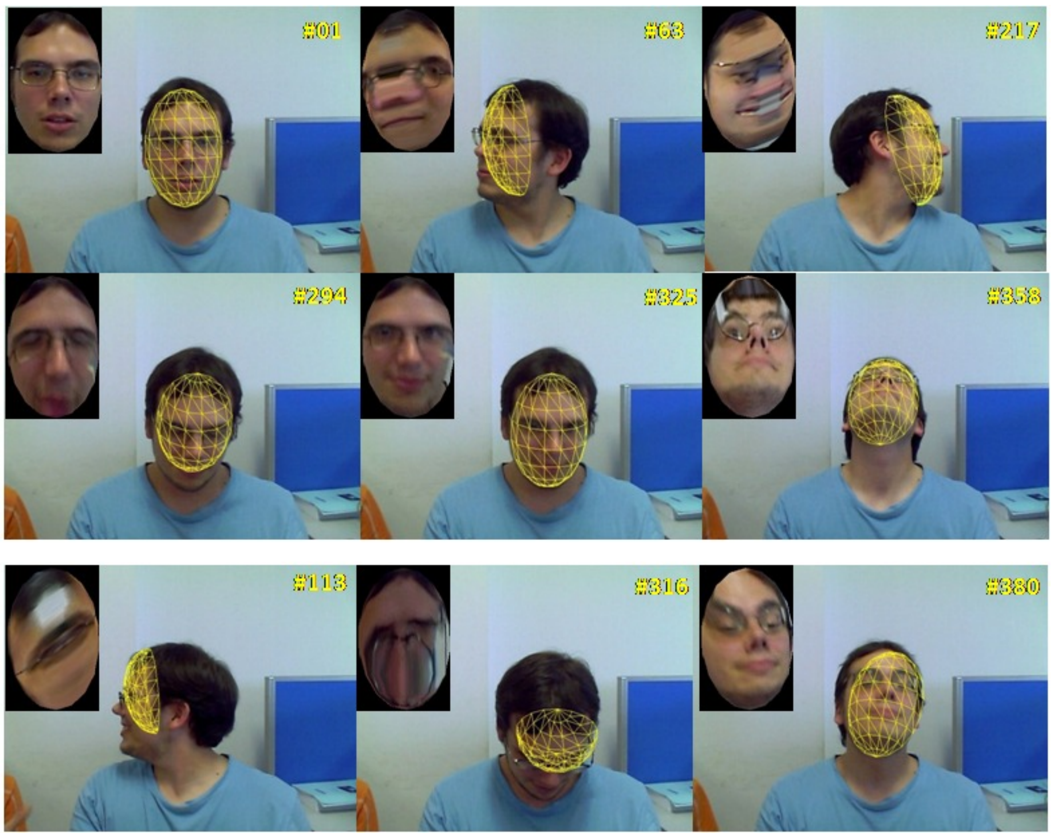


Figure 2.8: Tracking snapshots on the 'Fredric' sequence. Large out-of-plane rotations are the main challenge in this sequence. The first two rows correspond to nice tracking images while the last row shows the bad ones. The sub-image on top left corner is the appearance feature of current frame.

most importantly, gets rid of the dangerous prediction-based neighboring search.

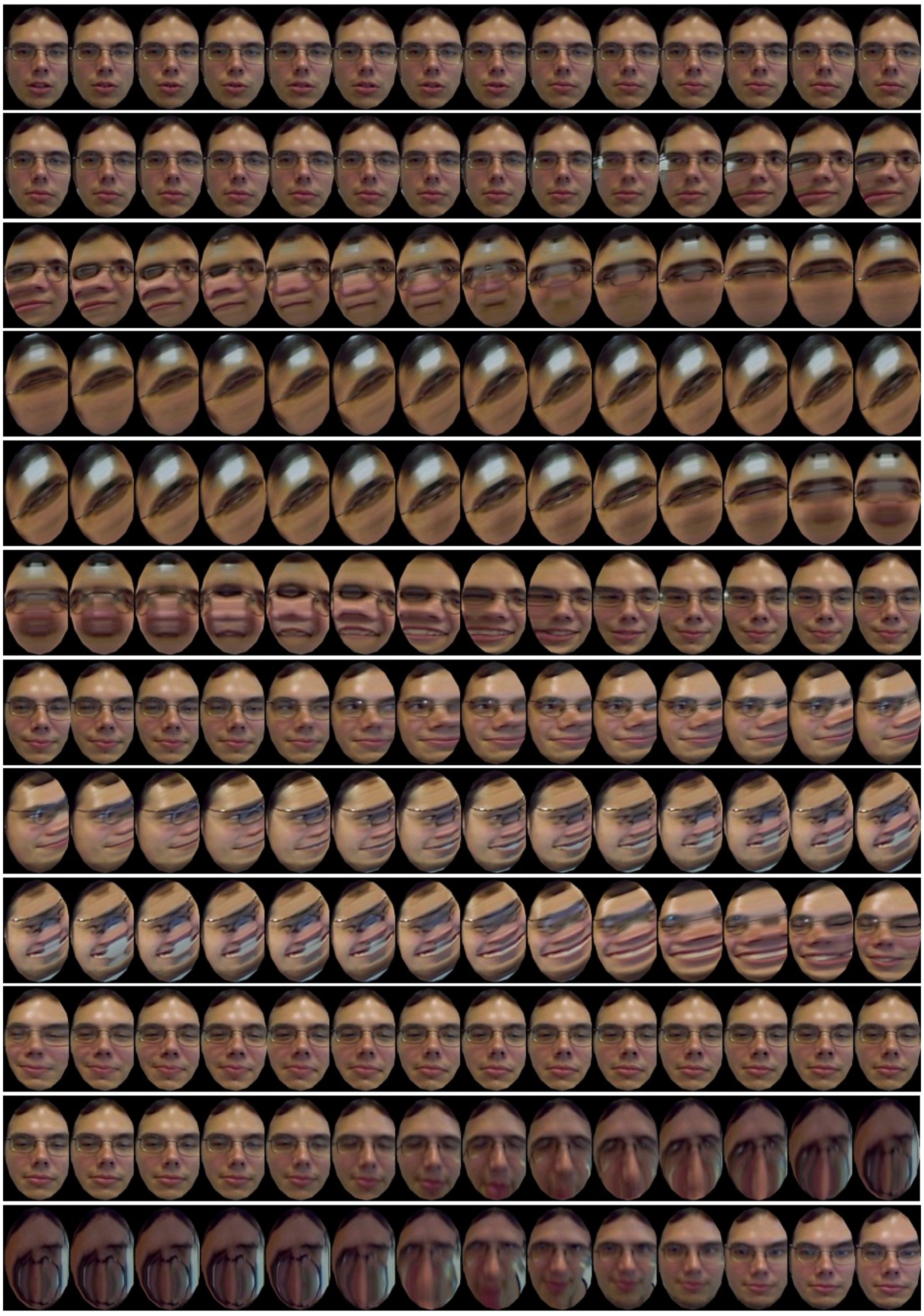


Figure 2.9: The color face appearances of the first 168 frames on the 'Fredric' sequence. The drift concept is clearly seen due to self-occlusions or facial expressions.

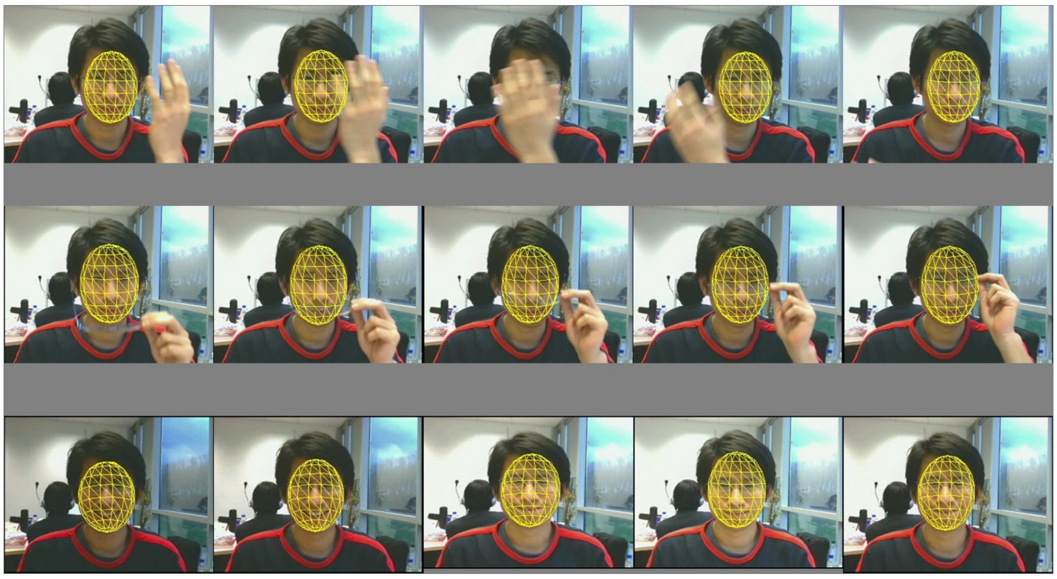


Figure 2.10: Tracking snapshots on the 'Haibo2' sequence. From top to bottom each row represents one challenging cases: partial occlusion, putting off glasses and illumination variations.

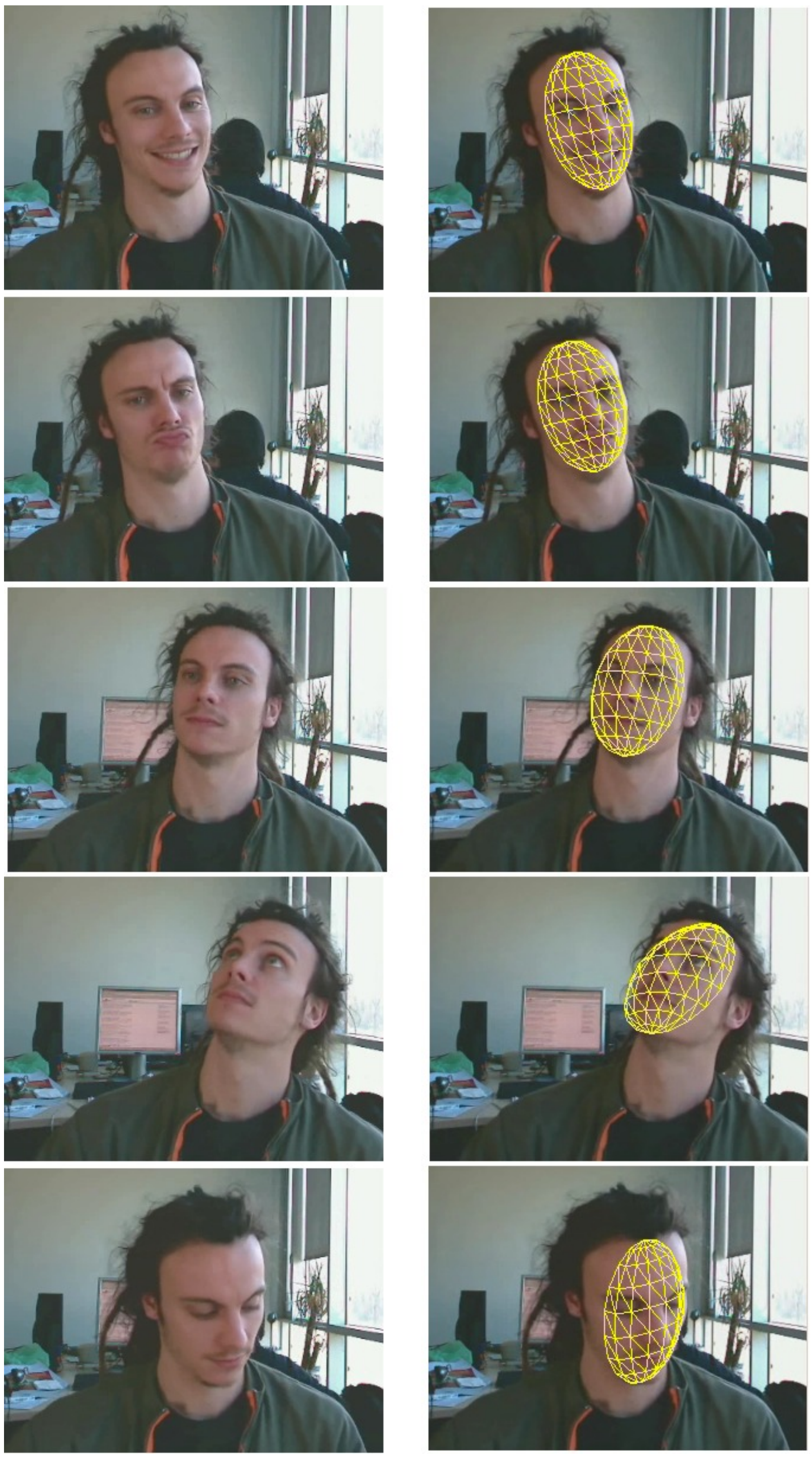


Figure 2.11: Tracking results on the 'Jeremey' sequence. The tracker deals well with moderate facial expressions and large in-plane head orientations.

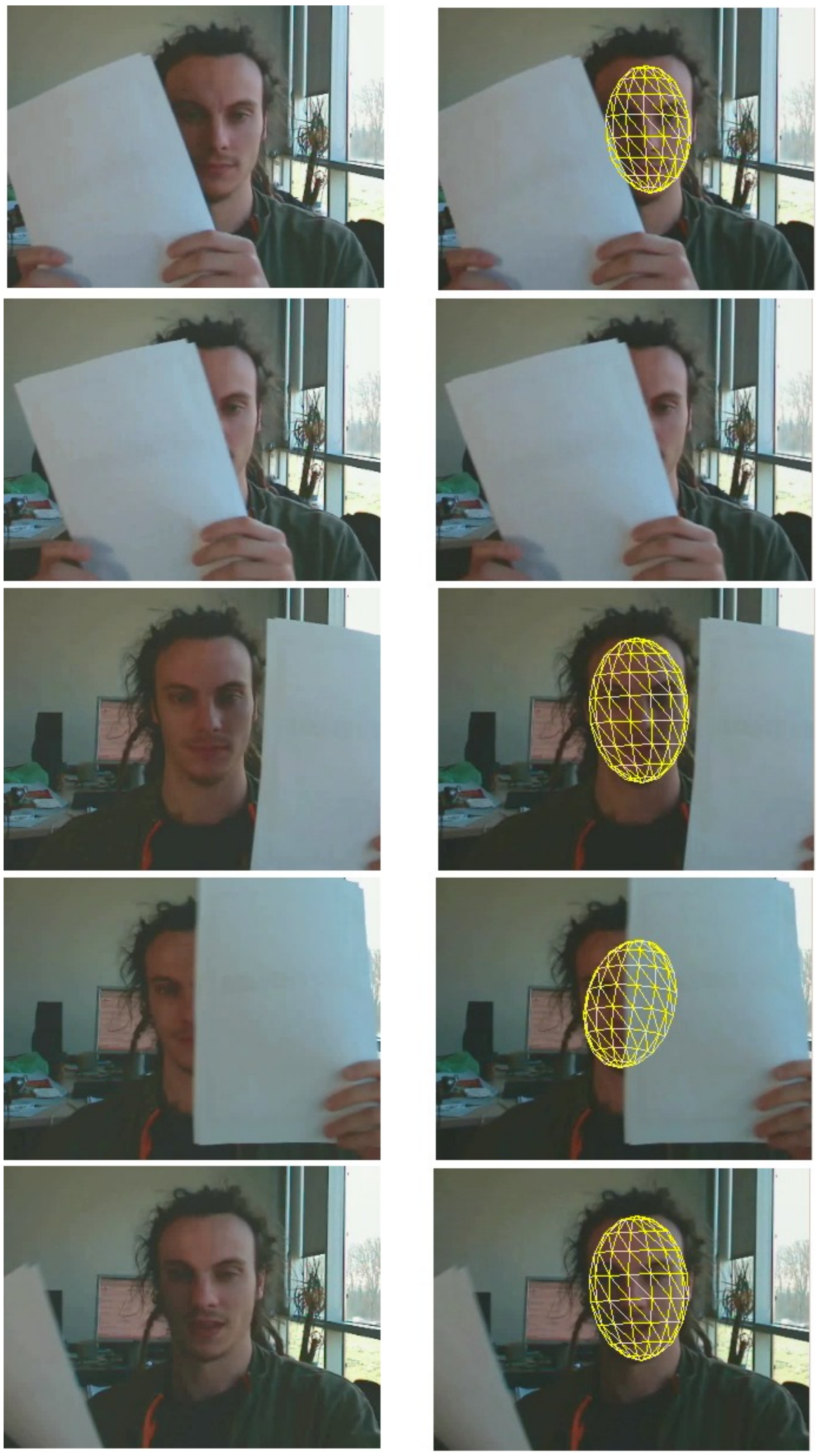


Figure 2.12: More tracking results on the 'Jeremey' sequence. Occlusions is handled by an automatic recovery procedure.

Chapter 3

A Human-Assisted 3D Head Modeling System

The first improvement over the solution in Chapter 2 is replacing the generic ellipsoid model with a person-specific one. To do this, we develop a human-assisted modeling system that is based on mesh editing technique with a Moving Least Squares (MLS) function. The basic philosophy is involved with two steps: the first step requires user to adjust the large-scale parameters of a generic model and fit its semantic features to their corresponding locations on two given photos; the second step is the automatic deformation of mesh vertices following the interpolation results of Moving Least Squares. To be realistic, we adopt the concept of as-rigid-as-possible during the MLS computation such that surface smoothness and shape locality can be preserved, as happens with physical objects when deformation is applied to them.

The following sections introduce two alternative MLS methods, analyze the specialities of the as-rigid-as-possible MLS, describe the modeling implementation with MLS, show experimental results and forecast its possible uses to our tracking tasks.

3.1 Introduction to Moving Least Squares

The Moving Least Squares (MLS) was first proposed by Lancaster and Salkauskas [Lancaster 1981] to tackle scattered data interpolation. It moves an arbitrary fixed point over the entire parameter domain with a weighted least squares fitting result. Due to its simplicity and flexibility in forms, it has attracted much attentions in the field of mesh editing and shape deformation [Levin 1998, Schaefer 2006, Cuno 2007, Fleishman 2005].

3.1.1 2D Affine Moving Least Squares

The Moving Least Squares method is capable of providing smoothing interpolations and approximations for scattered data [Levin 1998]. Its power to support smoothing deformations was successfully introduced in [Schaefer 2006]. In order to build surface deformations based on the MLS, we have to define a set of anchor points with which the user controls the deformations. Denote **p** as the control points set and **q** to be their deformed positions. Given an arbitrary surface point vector **v**, we solve for

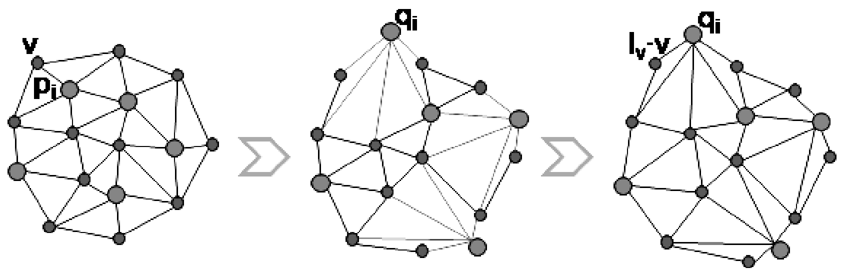


Figure 3.1: The schematic illustration of using Moving Least Squares for surface deformation.

the best local transformation function $L_{\mathbf{v}}(\mathbf{x})$ that minimizes

$$\sum_i w_i \| L_{\mathbf{v}}(\mathbf{p_i}) - \mathbf{q_i} \|^2 \tag{3.1}$$

where $\mathbf{p_i}$ and $\mathbf{q_i}$ are row vectors and w_i is a smooth kernel that weighs the distance $\|\mathbf{p_i} - \mathbf{v}\|^{2\alpha}$.

Since the smooth kernel w_i depends on the surface point v, we will derive a specific transformation $L_{\mathbf{v}}(\mathbf{x})$ for each point. In the case of 2D surface, we may simply consider the general affine transformation, which consists of a linear deformation and translation matrices:

$$L_{\mathbf{v}}(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{T} \tag{3.2}$$

Replacing the transformation function in Equation (3.1) gives us a specific objective function and solving it by using partial derivatives yields closed-form solutions

$$\mathbf{M} = (\sum_i \hat{\mathbf{p}}_i^T w_i \hat{\mathbf{p}}_i)^{-1} \sum_j w_j \hat{\mathbf{p}}_j^T \hat{\mathbf{q}}_j \tag{3.3}$$

$$L_{\mathbf{v}}(\mathbf{x}) = \mathbf{M}(\mathbf{v} - \mathbf{p}_*) + \mathbf{q}_* \tag{3.4}$$

As we can see, only the matrix $\mathbf{M}$ needs to be derived due to the fact that the translation matrix $\mathbf{T}$ could be discarded by a simple transform. It was also shown in [Schaefer 2006] that most of the calculations in Equation (3.3) could be done ahead of the deformation such that real-time performances were yielded. This illustrates the power of the MLS in real-time surface deformations, which also gives the reason of our choosing it as our main approach for head model deformation.

3.1.2 3D As-Rigid-As-Possible Moving Least Squares

It is natural to apply the MLS algorithm to surface deformation in 3D space. If the transformation is limited to be rigid, the resulted cost function would be a complicated constrained quadratic formula when the 3×3 rotation matrix is used. Recently, Cuno *et al.* [Cuno 2007] proposed to employ the exponential map of the rotation matrix, which in turn makes the problem solvable.

Suppose the transformation is rigid in 3D space $\mathrm{L}_{\mathbf{v}}(\mathbf{x}) = \mathbf{R}\mathbf{x} + \mathbf{T}$. In its exponential map, $\mathbf{R}$ is expressed by a rotation of an angle α around an axis $\mathbf{e}$:

$$\mathbf{R}_{\mathbf{e},\alpha}(\mathbf{v}^{\mathbf{T}}) = \mathbf{e}^{\mathbf{T}}\mathbf{e}\,\mathbf{v}^{\mathbf{T}} + \cos(\alpha)(\mathbf{I} - \mathbf{e}^{\mathbf{T}}\mathbf{e})\mathbf{v}^{\mathbf{T}} + \sin(\alpha)\mathbf{e} \times \mathbf{v} \tag{3.5}$$

The resulted cost function of this change is a constrained quadratic maximization problem:

$$\arg\max_{\mathbf{e},\alpha} \mathbf{e}\,\mathbf{M}\,\mathbf{e}^{\mathbf{T}} + \cos(\alpha)(E - \mathbf{e}\,\mathbf{M}\,\mathbf{e}^{\mathbf{T}}) + \sin(\alpha)\mathbf{V}\,\mathbf{e}^{\mathbf{T}} \tag{3.6}$$

$$\text{s.t.} \qquad \| \mathbf{e} \| = 1 \;\text{ and }\; \cos(\alpha)^2 + \sin(\alpha)^2 = 1 \tag{3.7}$$

where the detailed expressions of $\mathbf{M}, E$ and $\mathbf{V}$ are given in [Cuno 2007]. As a optimization problem, its Kuhn-Tucker conditions must be satisfied, which leads to the following eigenvalue system:

$$(\mathbf{M} + \mathbf{M}^{\mathbf{T}} + a\,\mathbf{V}^{\mathbf{T}}\mathbf{V})\mathbf{e}^{\mathbf{T}} = \lambda\mathbf{e}^{\mathbf{T}} \tag{3.8}$$

where the unknown rotation axis $\mathbf{e}$ corresponds to the eigenvector of matrix $\mathbf{M} + \mathbf{M}^{\mathbf{T}} + a\,\mathbf{V}^{\mathbf{T}}\mathbf{V}$ associated with the eigenvalue λ. Directly investigating the eigenvalue λ amounts to solving this large linear system. Rather than doing so, the authors made use of the characteristic polynomial of $\mathbf{M} + \mathbf{M}^{\mathbf{T}} + a\,\mathbf{V}^{\mathbf{T}}\mathbf{V}$ and derived a depressed quadratic equation which has a closed-form root. By assigning λ as the largest real root of this polynomial, all the unknowns $\mathbf{e}$, $\sin(\alpha)$ and $\cos(\alpha)$ are then determined correspondingly.

3.2 Prior Work in Head Modeling

Head modeling has been investigated for several decades. Roughly speaking, the existing approaches can be classified into two categories, depending on the amount of necessary interactive manipulations.

The first category requires no or few human interactions with the aid of image processing or vision techniques. Face modeling from a single image was made possible by adapting a pre-learned 3D morphable model to the image face [Blanz 1999b, Blanz 2003]. The only supervision is to provide an initial pose to the model. Automatic face modeling from video sequence becomes feasible by fusing the vision techniques of stereo calibration, model fitting and temporal flow tracking [Zhang 2001, Xin 2005]. With both photometric and geometric cues integrated, this modeling process is able to reconstruct 3D faces without any supervision. However, on achieving automation, the modeling accuracy and robustness are sacrificed. With the sampled image or video slightly changed, the modeling results might be far away from the original ones.

The second category needs comparatively more human manipulations. The concerned methods usually start with a parameterized model and then adapt it to an individual's face driven by certain physical laws or captured data. Particularly, the physical-driven modeling approaches have been thoroughly investigated for decades [Lee 1995, Terzopoulos 2004]. These approaches emphasize the modeling realism for which human supervision are indispensable. One of their important motivations is to drive facial animations in game, film and arts [DeCarlo 1998, Pei 2007, Haber 2004]. The main drawback of this category is its long-term processing time and near-expert experience request. Thus, rapid modeling for non-expert users are difficult. Consequently, the recent trend turns to fusing graphics and vision. Chai *et al.* [Chai 2003], for example, fused video tracking and motion capture data to create arbitrarily realistic yet online available face avatars.

Moreover, many researchers tend to use a generative modeling tool to edit face model, *ImageModeler*[1] as an example. Nevertheless, for a non-expert user, the modeling process may be full of pain and takes longer time. Comparatively, our modeling approach is in between of the two categories in that we perform automated surface deformation with moderate supervision of anchor points fitting.

3.3 Using Moving Least Squares for Head Modeling

Using this MLS algorithm to build a head modeling system is encouraged by its good performances in image and 3D shape deformations [Schaefer 2006, Cuno 2007]. We aim at such a system that requires the manual operations of a non-expert user as few as possible. To do this, we mark 35 semantic vertices in face model as the anchor points for MLS. But only half of them need to be adjusted due to face symmetry.

3.3.1 Choosing An Appropriate Moving Least Squares

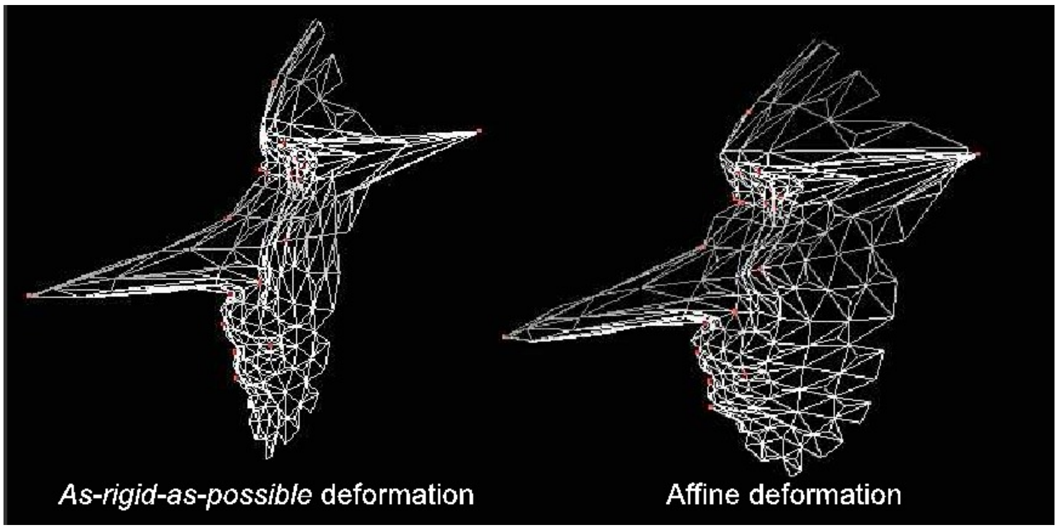


Figure 3.2: The comparisons of the 3D rigid and 2D affine Moving Least Squares functions.

[1] http://usa.autodesk.com/

At the first sight, both a 2D affine and a 3D rigid MLS might be suitable for our modeling purpose. In both cases, the 2D position $\mathbf{v}'$ in image plane of a 3D vertex $\mathbf{V}$ is respectively written as

$$\mathbf{v}' = \mathbf{M}(\mathbf{PV}) + \mathbf{t} \text{ and } \mathbf{v}' = \mathbf{P}(\mathbf{RV} + \mathbf{T}),$$

where $\mathbf{P}$ is the 3D-to-2D projective matrix. These two formula possess the same six degrees of freedom and thus can lead to the same deformation range. However, a simple analysis will show that the 3D rigid method is a more reasonable choice. Denote the three vertices of a mesh triangle as $\mathbf{v_1}, \mathbf{v_2}$ and $\mathbf{v_3}$. When the deformation remains small, the three vertices have approximately the same transformation. In the 2D affine case we have

$$\begin{cases} \mathbf{v}'_1 = \mathbf{MPV_1} + \mathbf{t} \\ \mathbf{v}'_2 = \mathbf{MPV_2} + \mathbf{t} \\ \mathbf{v}'_3 = \mathbf{MPV_3} + \mathbf{t} \end{cases} \Rightarrow \begin{cases} \mathbf{v}'_2 - \mathbf{v}'_1 = \mathbf{MP}(\mathbf{V_2} - \mathbf{V_1}) \\ \mathbf{v}'_3 - \mathbf{v}'_1 = \mathbf{MP}(\mathbf{V_3} - \mathbf{V_1}) \end{cases}$$

Similarly in the 3D rigid case we have

$$\begin{cases} \mathbf{v}'_1 = \mathbf{PRV_1} + \mathbf{PT} \\ \mathbf{v}'_2 = \mathbf{PRV_2} + \mathbf{PT} \\ \mathbf{v}'_3 = \mathbf{PRV_3} + \mathbf{PT} \end{cases} \Rightarrow \begin{cases} \mathbf{v}'_2 - \mathbf{v}'_1 = \mathbf{PR}(\mathbf{V_2} - \mathbf{V_1}) \\ \mathbf{v}'_3 - \mathbf{v}'_1 = \mathbf{PR}(\mathbf{V_3} - \mathbf{V_1}) \end{cases}$$

The deformations of the two edges, $\mathbf{v}'_3 - \mathbf{v}'_1$ and $\mathbf{v}'_2 - \mathbf{v}'_1$, are respectively embodied in $\mathbf{M}$ and $\mathbf{R}$ in the two cases. The affine motion $\mathbf{M}$ implicitly possesses a non-isotropic scaling and a shearing distortion. This may destroy or even rend the triangle. Conversely, the 3D rotation $\mathbf{R}$ imposes a rigid constraint in the two edges to maintain the integrity of the triangle. If the number of triangular facets is large enough, as 414 in our head case, the model plasticity can be guaranteed despite this local rigidity constraint.

As a matter of fact, this as-rigid-as-possible paradigm has been widely realized in shape deformation. How to integrate this paradigm into different optimization framework has also been investigated in the literature [Alexa 2000, Sorkine 2007, Cuno 2007]. As we illustrate in Figure 3.3, the major advantage of this paradigm is that the integrities of all mesh triangles are always preserved no matter how the surface is distorted.

3.3.2 Choosing A Smooth Kernel

The weighting function determines the amount of influence that an anchor point gives to its neighbors. It must be differential everywhere such that the MLS functional is differential as well. In this chapter, we experiment with three functions. The first tested function is the Gaussian function,

$$w(d) = e^{-\frac{d^2}{h^2}} \tag{3.9}$$

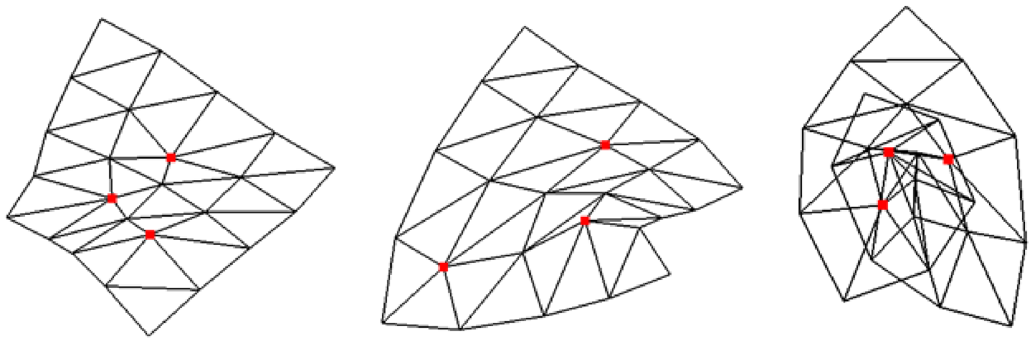


Figure 3.3: Illustrating the as-rigid-as-possible property that triangular cells are always preserved no matter how the surface is deformed.

where d is the distance between interpolated point and anchor point and, h is a spacing parameter to smooth out small features. The second function is the Wendland function [Wendland 1995]:

$$w(d) = \begin{cases} (1 - \frac{d}{h})^4(\frac{4d}{h} + 1) & \text{if} \quad d \in [0, h] \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

which is well defined in the interval $[0, h]$ to satisfy the smoothing conditions. The third one is simply the Euclidean function,

$$w(d) = \frac{1}{d^2 + \varepsilon^2} \tag{3.11}$$

where ε is a non-zero value to avoid singularity at $d = 0$.

Their respective influences are shown in Figure 3.4. The brighter the color is, the larger the degree of influence is. For the Gaussian and Wendland cases, we also show the influence of the bandwidth h. The larger it is, the heavier the influence is. Our need is that facial anchor points merely affect their corresponding semantic region (e.g. eye anchor points only affect the eye region.). As can be seen, two cases are most likely to meet this wish: the Gaussian with $h = 0.2$ and the Wendland with $h = 0.5$. Their color maps appear to be very similar. In our implementation, we adopt the former case without any specific reason. One can also refer to [Dey 2005] to investigate an appropriate way to automatically adapt this bandwidth.

3.3.3 Modeling Details

We use a generic model consisting of 414 triangular facets, formed by 229 vertices sampled from facial feature domains. From these vertices, we choose 32 visible anchor points at frontal view and 10 of them as anchor points at profile view. The 32 anchor points are then partitioned into five groups. The group depicting face contour is allowed to deform the entire surface while all the others, including the Eye,

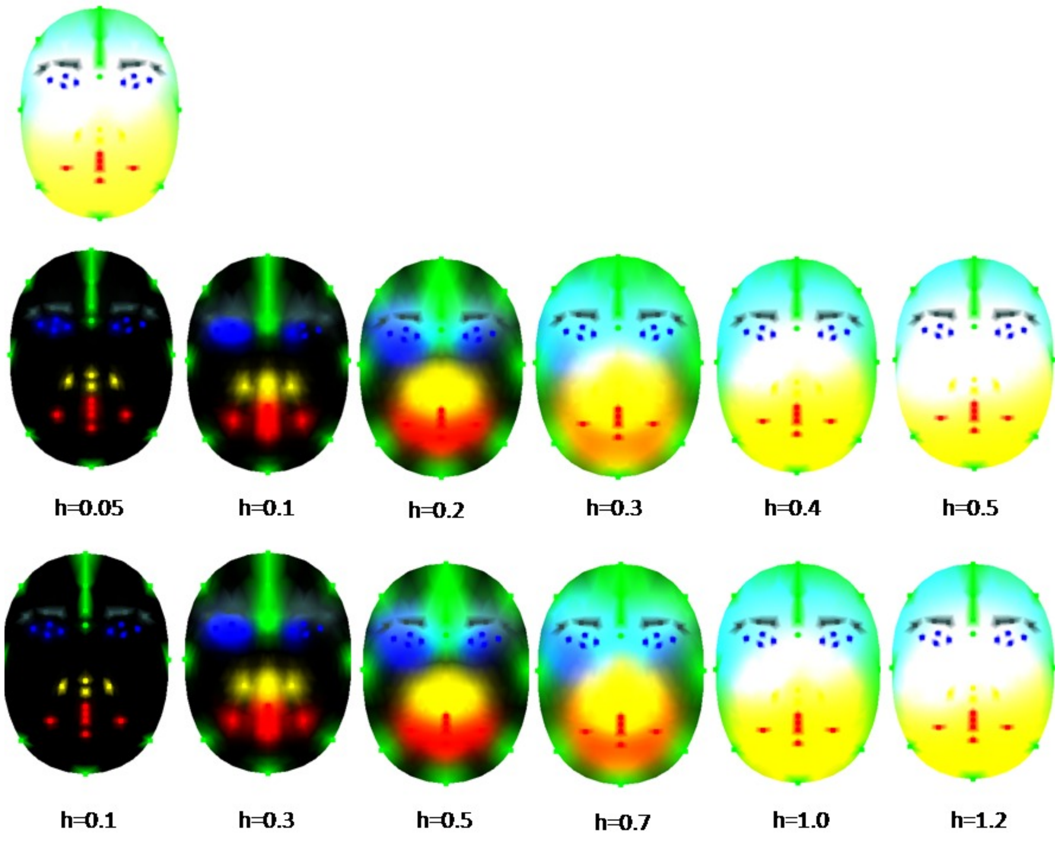


Figure 3.4: The influence fields of different weight functions. The first row shows the case of Euclidean distance. The second and third rows correspond to the cases of Gaussian and Wendland distances, as varying the local window size h.

Eyebrow, Nose and Mouth groups, are restricted to influence only their neighboring vertices. The used model and the feature groups are depicted in Figure 3.5.

Our modeling system first captures two roughly orthogonal photos, one at frontal view and another at profile view. In such a way, for a certain control point $\mathbf{v} \in \mathbb{R}^3$, we can only adjust its x, y coordinates at frontal view and y, z coordinates at profile view. As a result, its 3D position is easily derived by only normalizing the two different y values. Then all the vertices that are not control points will follow this deformation to a new place smoothly interpolated by the MLS computation.

3.3.4 Texture Synthesis and Mapping

The texture mapping procedure is schematically illustrated in Figure 3.6. In order to synthesize realistic modeling results, the cylindrical projection [Pighin 1998] is adopted, which will first project mesh vertices on a cylinder plane and then un-roll the cylinder to get planar facial texture by mapping the frame pixels into the plane.

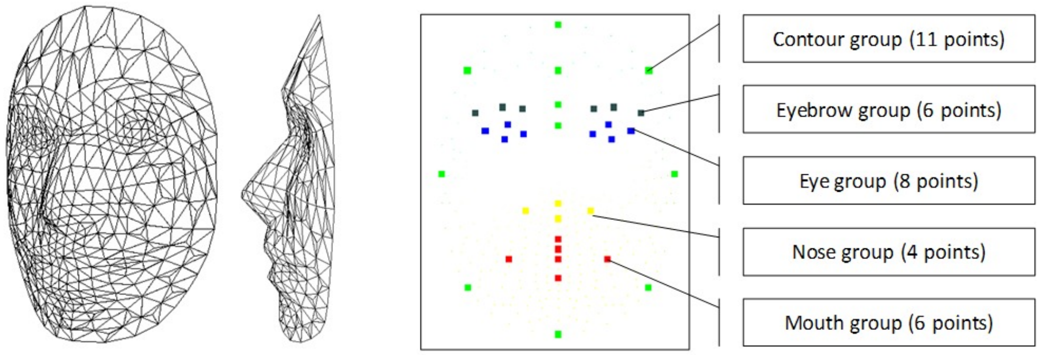


Figure 3.5: Illustrating the used generic model and the definition of control points and their five groups.

To save computing time, the barycentric coordinates are used to correspond texture pixels to image ones such that the pixel color is assigned. Suppose the head model is already well fitted to the face regions at the captured frame, the texture synthesis can be then proceeded as follows:

(i) Project the model facets onto the cylinder plane to yield a list of planar triangles (we assume the basic facet is triangle within the head model). The projected position of each vertex is regarded as its texture coordinate;

(ii) Un-roll the cylinder to get a planar texture. For each texture pixel, compute its barycentric coordinates and mark it as an inlier. Otherwise, if no coordinates found, then the pixel is regarded as an outlier;

(iii) Back-project each inlier pixel to the model surface to find the corresponding frame pixel. Then use bilinear interpolation around the found frame pixel to fill the color of the inlier pixel.

As the last step, the interpolated texture is smoothed by a 3×3 Gaussian Filter such that interpolation noises may be eliminated. Once the texture synthesis is completed, it can be easily mapped onto head model to render realistic face. The vertex positions on cylinder plane is re-used as texture coordinates.

3.4 Experiments & Results

We select 32 out of the 229 mesh vertices as anchor points. Then on making use of face symmetry, only 23 out of the 32 anchor points must be manually fitted. The used MLS scheme meets the requirements of smooth deformation and constraint enforcement in a natural way. The non-iterative MLS scheme makes the entire modeling procedure nearly real-time on a common modern PC.

The deformation ability of the proposed scheme is specifically examined here. Individual variation across the three example faces in Figure 3.7 encompass a range of features: clear differences are found in the zoomed regions of eye, nose and mouth. The clear differences of the overall shapes of modeled samples can be seen

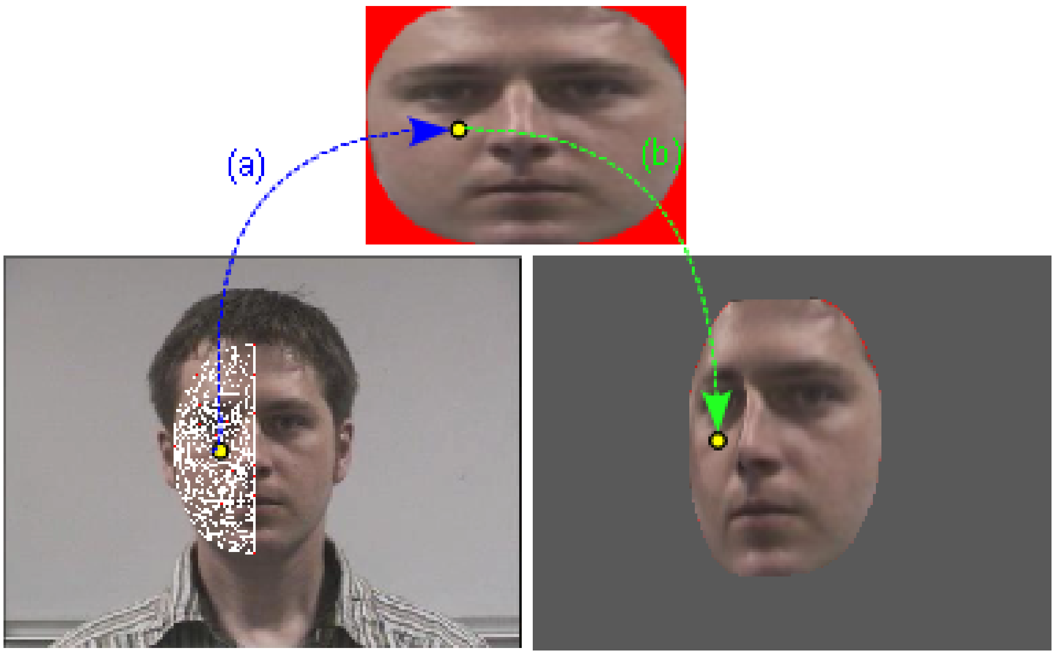


Figure 3.6: Illustrating the procedures of model texture synthesis and mapping. Procedure (a) represents texture synthesis while procedure (b) stands for mapping the texture onto the built model.

in Figure 3.8.

We do not quantify the comparison between modeling samples and ground truth, but measure its qualities based on human perceptions. Figure 3.9 shows four modeling samples by using the static photos in the INRIA database [Nicolas 2004]. One can easily see the strong similarities between the realistic pictures and the synthesized ones in the core proportions of eye sockets, nose and mouth. Of course, there are also visible differences due to insufficient deformation forces. One example is the depth of the eye sockets, which is either deeper or lighter than the realistic one as no anchoring points are assigned around it. Another observable problem is that since the texture is only extracted from the realistic face at frontal view, its profiling view appears to be different from that of the ground truth.

Figure 3.10 shows four more modeling results by using live-captured images. The textures look less realistic than those based on the static images, mainly due to the non-uniform face reflections. Another problem happens with the glasses, which is not virtually realized and thus looks obviously unrealistic at profiling view. Despite these flaws, the generated model is suitable for the model-based head tracking framework given its high efficiency. Even being roughly realistic, it is able to synthesize reality-similar textures at multiple views and provide model constraint and geometric reasoning (such as reasoning for self-occlusion).

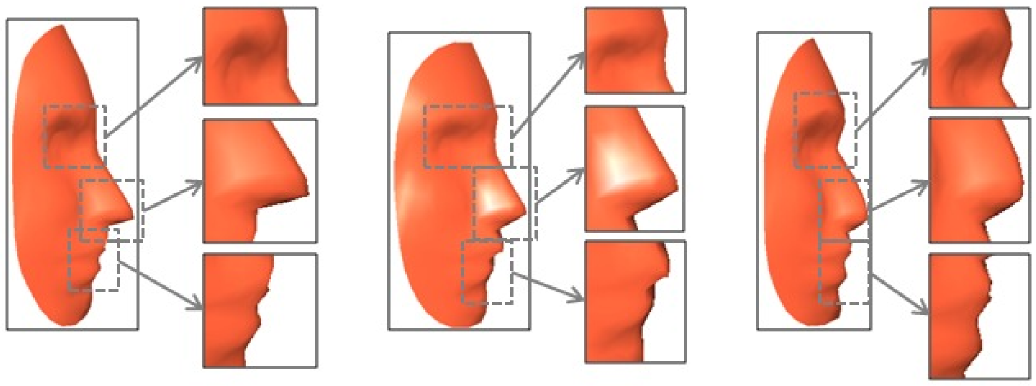


Figure 3.7: Illustrating the individual variations across three face models deformed from the same prototype one. The slopes of eye, nose and mouth are zoomed to demonstrate their topological differences.

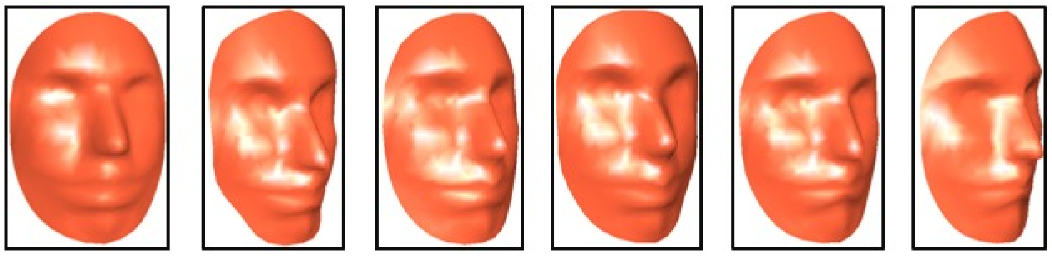


Figure 3.8: Six sample face models that are derived from the same prototype model.

3.5 Conclusions and Discussions

This chapter presents a head modeling system requiring simple interactive operations of a non-expert user. The core idea relies on casting modeling as a surface deformation process, which is implemented via the as-rigid-as-possible 3D MLS. The suitable choices of the MLS form and the smooth kernel are empirically verified in the above descriptions. Experimental results are also shown on a set of given photos.

Developing this modeling system is to provide a person-specific model to facilitate head tracking. In the next chapter, we will show that the built model is used to render a set of face textures at various viewpoints, which are then applied to provide many interest points. The possible improvements over this system will be made in two aspects. First, the current constraints to MLS computation are the movements of the assigned semantic vertices placed by the user. However, even such placing work may be still tedious for users. Since many online face dataset are available, it is possible to perform facial feature localizations with auto detections. Second, only point constraints are not adequate to model some regional features. More sketched constraints, such as reserving line or gradients, can be added to yield more accurate

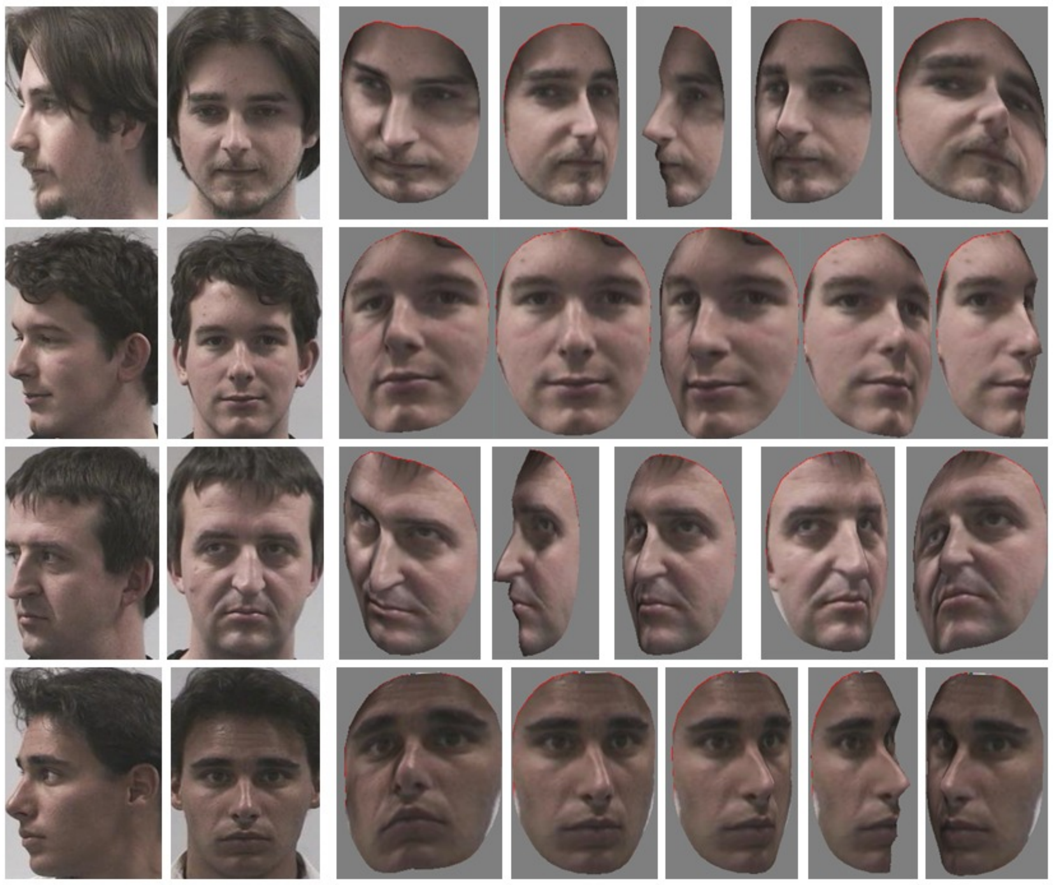


Figure 3.9: The modeling faces based on the photos given in dataset [Nicolas 2004]. At each row, the first two pictures are the ground truth at frontal and profile views while the left are the synthesized textures.

faces. Moreover, relighting problem should be tackled to create various photometric scenes.

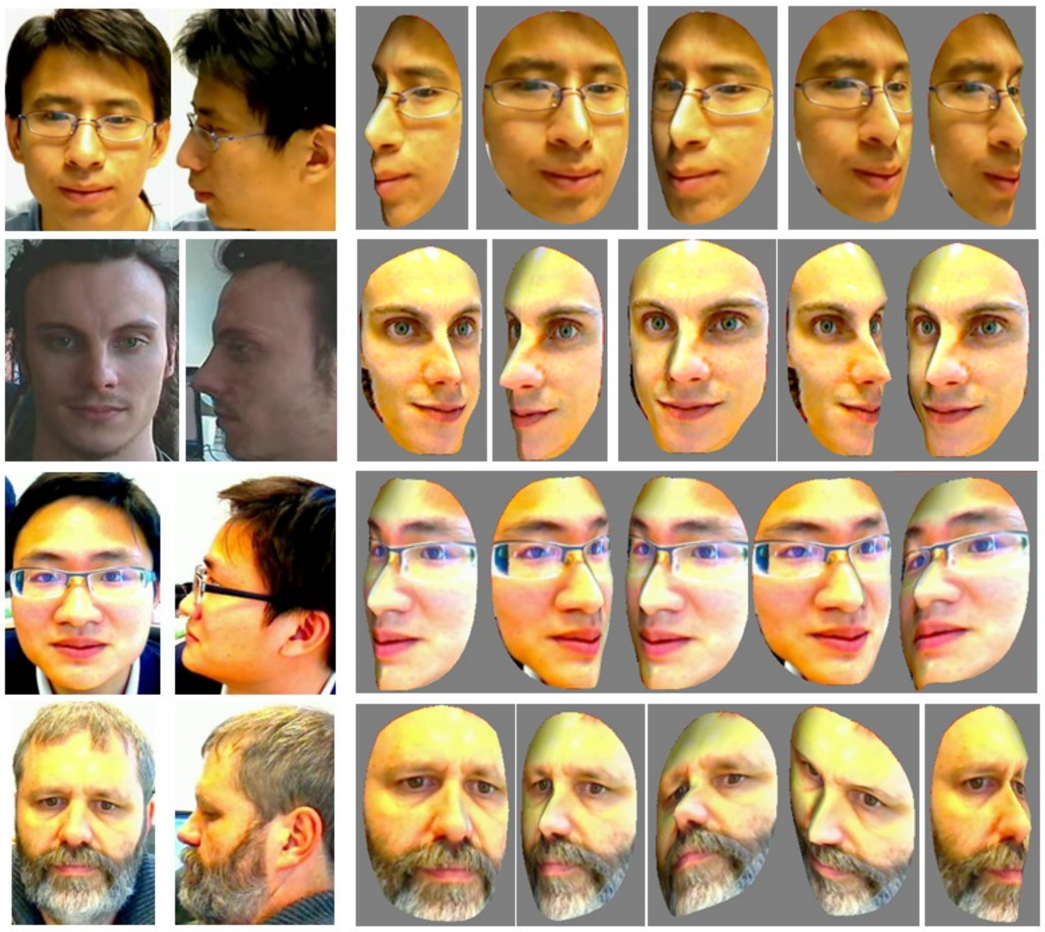


Figure 3.10: The built models based on four live-captured photos. At each row, the first two pictures are the ground truth at frontal and profile views while the left are the synthesized textures.

Chapter 4

Tracking by Detection - A Wide-Baseline Approach

The nature of the differential-based approach in Chapter 2 is recursively applying a temporal prior to the pose estimate for the next frame. The merit is to make efficient local pose estimate reasonably feasible. Nevertheless, this merit can become deathful when the prior is unreliable after a complete occlusion of the head or an incorrect estimate. Such a weakness leads to the widely recognized drifting issue that obsesses all tracking algorithms. Thus, re-initialization is indispensable once the system gets lost. In this chapter, we employ a totally different approach that estimates pose by independently detecting it in the entire input image. It is termed as tracking by detection as temporal prior is no longer considered [Lepetit 2005]. Although the efficiency of local estimate is lost, we benefit more from being far away from its damage, e.g. obtaining higher reliability and robustness.

A 3D model is known to greatly facilitate the tracking task, if it is available. The traditionally model-based methods were more aware of its value to partially offset the illedposeness of the tracking problem during tracking [Basu 1996, Cascia 1999, Morency 2003a, Dornaika 2008b]. Then recently, with the popularity of learning, more researchers realized that a 3D model is also greatly helpful during the tracker training [Lepetit 2004, Liebelt 2010]. As initial trials, their proposals did not uncover the full potentials of a 3D model for learning due to lack of a supportive system. With our MLS-based modeling system, we propose an extended scheme to take full advantage of a 3D model in tracking by detection .

In implementing tracking by detection, we consider matching only the interest points that are distinctive and repeatable out of the entire image pixels. Since the viewpoints of the input images can be very different from those of the training images, wide-baseline matching is performed in essence. To address it, we develop a multi-view local feature learning paradigm such that the matching can tolerate moderate view deviation from the training ones.

In the following, we start by addressing the bayesian formulation of tracking by detection and then present the proposed approach hierarchically.

4.1 The Bayesian Formulation

Tracking by detection is a special case of the recursive tracking in terms of the Bayesian formulation. The sequences of states of the tracked object and image

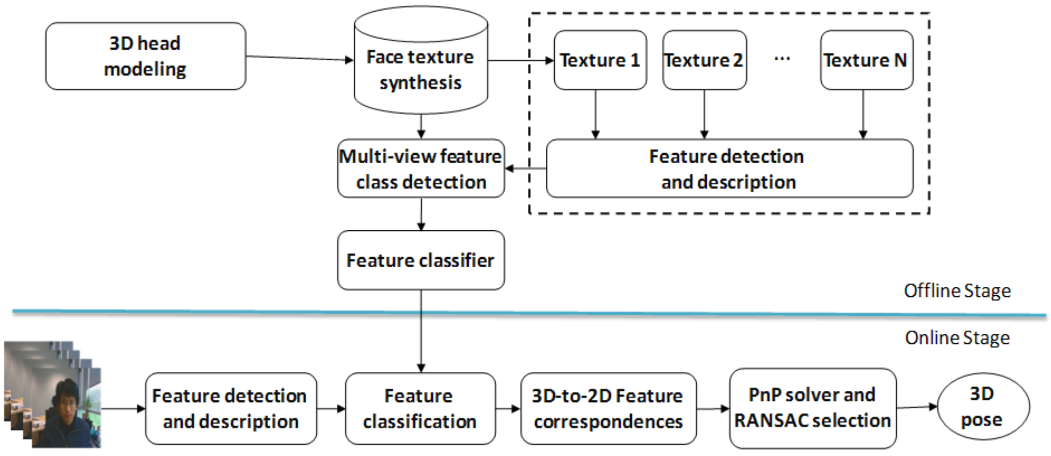

3D head modeling
Face texture synthesis
Texture 1
Texture 2
...
Texture N
Multi-view feature class detection
Feature detection and description
Feature classifier
Offline Stage
Online Stage
Feature detection and description
Feature classification
3D-to-2D Feature correspondences
PnP solver and RANSAC selection
3D pose

Figure 4.1: The framework of our tracking approach. It involves two stages. The offline stage consists of interactive modeling, face texture synthesis and stable feature selections to provide a feature classifier. The online stage performs tracking by matching feature points and solving a Perspective-n-Point problem.

observations up to time t are denoted by $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$ and $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$, respectively. The goal of tracking at time t is to estimate the state $\mathbf{x}_t$ from the available observations $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$.

In short-baseline tracking, estimating state $\mathbf{x}_t$ starts with an initial vector that is predicted from its previous state $\mathbf{x}_{t-1}$. However, the dynamical probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ may be unreliable after an abrupt motion or if $\mathbf{x}_{t-1}$ is estimated incorrectly. Moreover, only local convergence can be reached since $\mathbf{x}_t$ is estimated only in the neighboring range of the predicted state. Alternatively, tracking-by-detection is to perform state estimation independently at each frame, which amounts to regarding the density of $\mathbf{x}_t$ as uniform and setting $p(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto constant$. From the Bayes rule, we can write

$$p(\mathbf{x}_t = \gamma|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t = \gamma), \tag{4.1}$$

where γ is the independently estimated state. Unlike the recursive tracking, its initial value is independent of the pose estimated at the previous frame. Then we consider more state candidates and attain the one with maximum measure probability

$$\hat{\gamma} = \arg\max p(\mathbf{z}_t|\mathbf{x}_t = \gamma). \tag{4.2}$$

Since consecutive motion cue is neglected here, the observation $\mathbf{z}$ must be reliable enough to recover the right $\mathbf{x}$.

4.2 Approach Overview

Our tracking-by-detection approach builds on matching interest points on each frame to the model ones. The key ingredient is casting matching as a point classification problem. A class c is defined as all the possible views of the patch around a feature point $\mathbf{k}$. Suppose we set up a set of classes $\mathbf{C} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_n}\}$ and a compact classifier $\mathfrak{R}$ during training. Given an input patch $\mathbf{p}$, the classification goal is to assign it a class label $\widehat{Y} = \mathfrak{R}(\mathbf{p})$ such that $P(\widehat{Y} \neq Y)$ is smallest.

We develop a system to implement the approach. There are two separate stages as demonstrated in Figure 4.1 in the system. The offline stage aims at setting up the class set $\mathbf{C}$ and growing the compact classifier $\mathfrak{R}$ via 3D modeling, texture synthesis and class view selection. The online stage performs the actual classification of all online patches so as to provide us a set of 3D-to-2D correspondences. Following it, we solve a Perspective-n-Point problem to recover 3D poses with all these correspondences. Since the classification is matched independently at each frame, tracking-by-detection is actually employed.

4.3 Multi-View Class Detection with a 3D Model

4.3.1 Face Texture Synthesis and Stable Patch Selection

Once the 3D model building is completed, we can generate a large number of face views via random geometric transforms as if many realistic training data are available. In detail, the geometric sample space is defined as

$$\{\theta, \phi, \lambda_1, \lambda_2, t_u, t_v, \alpha, \beta, \gamma\}, \tag{4.3}$$

where $\{\theta, \phi, \lambda_1, \lambda_2, t_u, t_v\}$ are the 2D affine transform parameters and $\{\alpha, \beta, \gamma\}$ denote the 3D orientation angles. We first generate a set of perspective views with the textured model by running 3D random rotation transforms by n_1 times

$$\mathcal{T}_1 = R_\alpha R_\beta R_\gamma. \tag{4.4}$$

On each perspectively synthesized image, we then perform the random affine transform by n_2 times to simulate planar or small non-planar changes

$$\mathcal{T}_2 = R_\theta R_{-\phi} diag(\lambda_1, \lambda_2) R_\phi. \tag{4.5}$$

We set $n_1 = 4 \sim 10$ and $n_2 = 10000$ to trade off between training time and view number. To be more realistic, random lighting variations are imposed during the $\mathcal{T}_1$ transform and random white noises are inserted into the $\mathcal{T}_2$ transform.

Not all the synthetic interest points will appear during tracking. To be efficient, we propose to select the most stable ones by extending the way as developed in [Lepetit 2006]. For a single feature point $\mathbf{k}$, we test its stability by counting how often it is found throughout a large number of random $\mathcal{T}_1$ and $\mathcal{T}_2$ transforms. By denoting the respective finding frequencies under the two transforms as $P(\mathbf{k}|\mathcal{T}_1)$ and $P(\mathbf{k}|\mathcal{T}_2)$, we rank all the detected feature points according to $P(\mathbf{k}|\mathcal{T}_1)P(\mathbf{k}|\mathcal{T}_2)$ and only retain the 100 ones with the highest values.

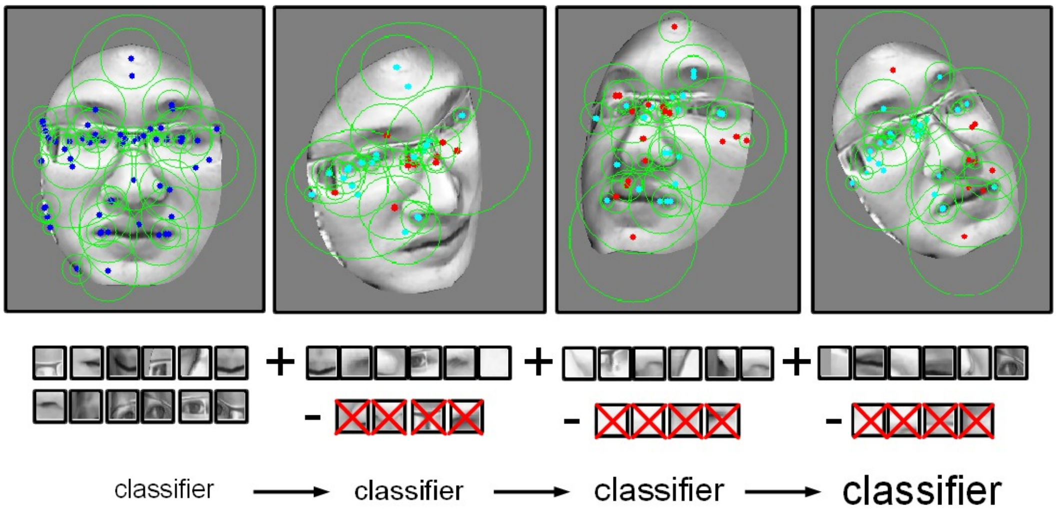


Figure 4.2: Illustrating multi-view class detection and classifier building. On the frontal view, all features are accepted as new classes (marked in blue color) to train a basic classifier. As a new synthetic view is available, we identify new class views and new classes (marked in sky-blue color) and discard the outliers (marked in red color). In such a way, the basic classifier is incrementally updated to be stronger and more robust. The green circle implies the scale at which the feature point is found. The discarded patches are shown with an X on the upper line.

4.3.2 Multi-View Patch Class Detection

Figure 4.2 depicts this multi-view class detection process. With the beginning frontal texture, we treat all the detected feature points as our classes, except the very edging ones. The classifier $\mathfrak{R}_0$ built with these classes are named as the *basic* classifier. As a perspectively synthesized texture is available, we detect new feature points and extract their surrounding patches. Typically, the patch around each feature point belongs to one of the three categories:

1. the new view of an existing class,
2. the first view of a new class and
3. a flawed patch due to modeling or texture-mapping faults.

To identify them, we first back-project all detected feature points onto the 3D model and match them by looking for the closest back-projected 3D positions to get c. Then we use the current $\mathfrak{R}$ to classify the point with a matched class c'. The one whose classified label c satisfies $c = c'$ is retained as the new view of the existing class c. The point having $c' \neq c$ is likely to be mis-classified and thus is discarded for safety purpose. However, for the left point without matched classes, finding its category is rather difficult. We observe that it is possible to treat them together by computing

Algorithm 1: Multi-view patch class detection

```
Input: a set of $\mathcal{T}_1$-transformed face views $\{\mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_t\}$ and the basic
       classifier $\mathfrak{R}$
Output: an augmented classifier $\mathfrak{R}^+$
for $\mathbf{T}_i = \mathbf{T}_1$ to $\mathbf{T}_t$ do
    1. Detect feature points $K = \{k_1, k_1, .., k_m\}$ on $\mathbf{T}_i$.
    2. Back-project each $k_j$, $j = 1, 2.., m$, to 3D model and match it with a
    class $c_j$ by searching the nearest 3D position. All these with matched
    classes form a subset $K_r$ of size $m_r$.
    3. Each point in $K_r$ is classified with the current $\mathfrak{R}$ to find a $c'_j$. If
    $c'_j = c_j$, $k_j$ is regarded as a new view of class $c'_j$. Otherwise, abandon it.
    4. If $m_r/m < \delta$, the set $K - K_r$ are retained as new classes; Otherwise,
    abandon them.
    5. Update $\mathfrak{R}$ with $N$ $\mathcal{T}_2$-transformed new patch views of all the new class
    views and stable new classes.
```

the ratio of the new view number m_r to the number of detected points m. If m_r/m is smaller than a threshold, the global view tends to be partially self-occluded and hence we accept all of them as new classes. Otherwise, we give them up. After the above feature categorization, we perform $\mathcal{T}_2$ transformations to synthesize new homographic views of new class views and new classes, check the stability of new classes via small $\mathcal{T}_1$ transformations and utilize the stable new classes and all new views to update the current $\mathfrak{R}$. The new classes are simply imposed into the $\mathfrak{R}$ while new views are used to update their corresponding classes by either updating the mean descriptor at this view or the corresponding posterior probabilities. A summary of the procedures is presented in Algorithm 1.

Searching the 3D position of a feature point is implemented via an efficient color-filling technique. As shown in Figure 4.3, we render a color map at the same viewpoint of the synthetic texture beforehand. Each triangular facet is coded by a unique color. For an interest point $\mathbf{u}$, its belonging facet $\hat{f}$ is indexed by decoding the color at the same $\mathbf{p}$ point of the color map. Denote $(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3)$ as the three vertices of $\hat{f}$ in 3D space and $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ their respective projections in image plane. As $\mathbf{u}$ is inside of triangle $\hat{f}$, it can be expressed as a weighted sum of these projections

$$\mathbf{u} = \alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2 + (1 - \alpha_1 - \alpha_2)\mathbf{u}_3 \tag{4.6}$$

where (α_1, α_2) is the unique barycentric coordinates. By assuming the same relation holds in the 3D facet plane, we also write

$$\hat{\mathbf{V}} = \alpha_1 \mathbf{V}_1 + \alpha_2 \mathbf{V}_2 + (1 - \alpha_1 - \alpha_2)\mathbf{V}_3 \tag{4.7}$$

where this $\hat{\mathbf{V}}$ is the 3D position of $\mathbf{u}$. When the size of facet $\hat{f}$ is small, the aliasing effect in rasterizing may cause offset of its filled-in color such that the chosen facet $\hat{f}$ is incorrect. To deal with it, we iteratively search every facet in the range of

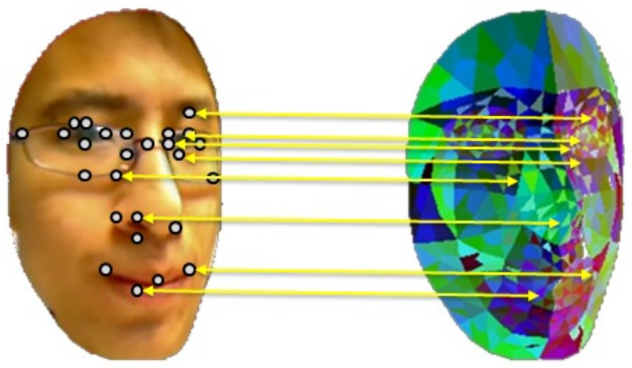


Figure 4.3: Illustrating the color-filling technique for fast 3D position assignment.

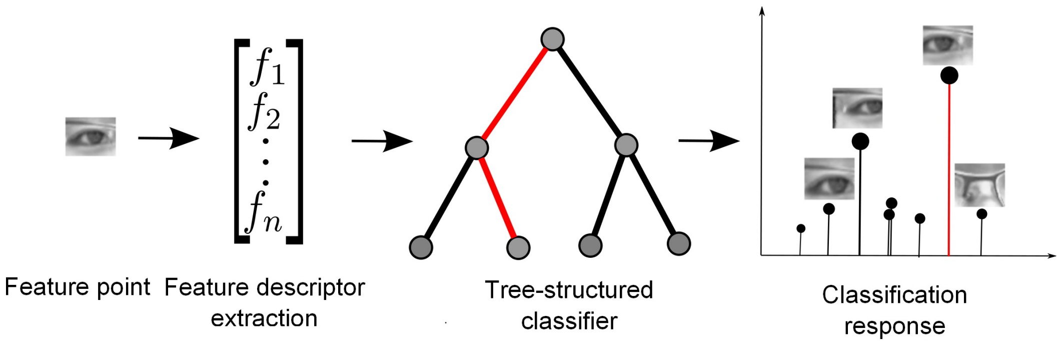


Figure 4.4: The pipeline of feature matching via classification.

$[\hat{f}-8, \hat{f}+8]$ and stop until the barycentric coordinates (α_1, α_2) that satisfy $0 \leq \alpha_1 \leq 1, 0 \leq \alpha_2 \leq 1$ are found.

4.4 Testing Feature-Classifier Combinations

3D Pose is recovered from a set of 3D-to-2D correspondences that associate 3D positions to interest image points that ought to be repeatable and distinctive over different views. The local patch around such a point is extracted to represent its specific properties. The correspondences are established by matching query patches to the model ones processed during training. In this section, we test two alternative combinations that detect, describe and match local patches in two different directions. Particularly, both the classifiers are tree-structured but one emphasizes speed while the other favors accuracy. Figure 4.5 shows a generic pipeline.

4.4.1 Harris Detector, Random Binary Feature and Ferns classifier

4.4.1.1 Harris Detector

A local feature is such an image pattern that differs from its immediate neighborhood. Among existing feature types, Local interest point, typically either a corner or a junction point, is the most popular one due to its easy description. To localize interest points in images, a local neighborhood of pixels needs to be analyzed. Those whose neighborhood is repeatable and distinctive are the most favorable [Mikolajczyk 2004]. The Harris detector localizes corner points as the local maxima of a directional variance measure [Harris 1988]. This measure relies on a second moment matrix of the neighborhood computed with Sobel derivatives and a Gaussian window

$$\mathbf{M} = \sigma_D^2 G(\sigma_I) * \begin{bmatrix} I_x^2(x,\sigma_D) & I_x I_y(x,\sigma_D) \\ I_x I_y(x,\sigma_D) & I_y^2(x,\sigma_D) \end{bmatrix} \tag{4.8}$$

Given the two eigenvalues of this matrix, a corner is then decided via a function comprising its determinant and trace

$$cornerness = det(\mathbf{M}) - \lambda trace(\mathbf{M}) \tag{4.9}$$

where a typical value for λ is 0.04. The cornerness corresponds to both eigenvalues being large. Specifically, adding the second terms with the trace suppress the responses of strong straight contours that only have one big eigenvalue. Moreover, such a measure is much less demanding than actually performing the eigen decomposition as both its determinant and trace can be computed without knowing the eigenvalues explicitly.

To be scale-invariant, the Harris-Laplace detector is usually used, which starts with a multi-scale Harris detector and determine the characteristic scale that attains an extremum over its scale space. The Laplacian operator is known to be best for this scale selection, after which the detector is named.

4.4.1.2 Random Binary Features

The random binary feature captures both the textural and spatial information of a target region in a simple way. It is composed of binary sequences, each of which is assigned by comparing the intensities of two randomly picked pixels $\mathbf{m}_1$ and $\mathbf{m}_2$ of a local patch I:

$$f_i = \begin{cases} 1, & I(\mathbf{m}_1) \leq I(\mathbf{m}_2) \\ 0, & otherwise \end{cases} . \tag{4.10}$$

In order to attain robustness, the features are computed over example patches seen from different viewpoints. This is achieved via random affine synthesis of the given patch as described in the last subsection. At training stage we may need a long sequence of binary features

$$\{f_1, f_2, f_3, ..., f_N\} \tag{4.11}$$

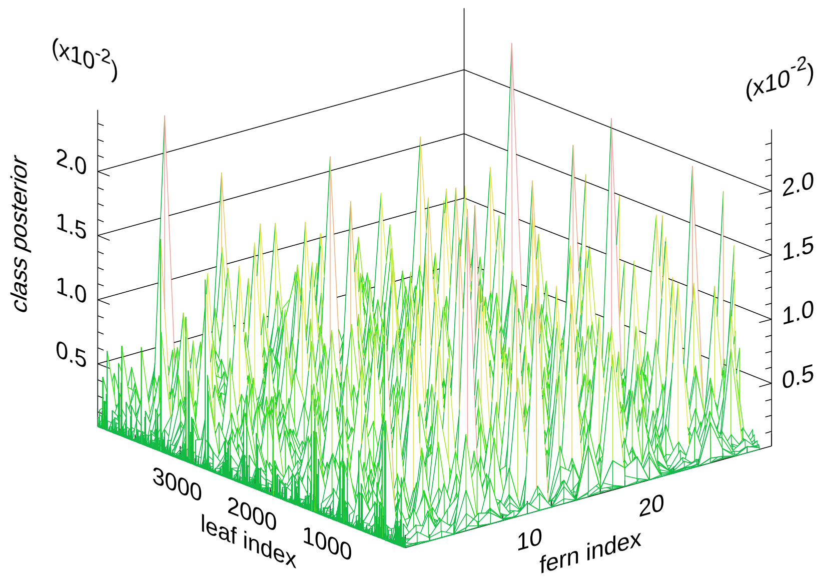


Figure 4.5: The conditional distributions of an example class over all the leaves of all the ferns.

where a suitable choice is $N = 10^4$ for a 32×32 patch. Contrarily, at run-time, the sequence length needs to be proper to trade off efficiency and preciseness, for example, 300 for a 32×32 testing patch.

4.4.1.3 Ferns Classifier

The Ferns classifier is a derivation of the powerful Randomized Trees [Ozuysal 2009]. It is non-hierarchical and building on the fact that combining groups of binary features rather than the tree structure itself allows improved classification rates. The random binary features are used in such a way that each node corresponds to a single binary test. Suppose there are H patch classes extracted from training data. For each feature patch $\mathbf{p}_i, i = 1, ..., H$, its class label is found by

$$\widehat{c_i} = \arg\max_{c_i} P(C = c_i | f_1, f_2, ..., f_N) \tag{4.12}$$

$$\approx \arg\max_{c_i} P(f_1, f_2, ..., f_N | C = c_i) \tag{4.13}$$

Here we assume the Bayesian prior are uniform and thus neglect it. Given the N-length binary sequence of a training patch, we assume partial independence and partition them into M independent groups of size $S = \frac{N}{M}$,

$$P(f_1, f_2, ..., f_N | C = c_i) = \prod_{k=1}^{M} P(F_k | C = c_i) \tag{4.14}$$

Such a random group $F_k = \{f_{\sigma(k,1)}, f_{\sigma(k,2)}, ..., f_{\sigma(k,S)}\}$ is a fern and $\sigma(k, j)$ is a random permutation function with range $1, ..., N$. Each conditional probability $P(F_k|C = c_i)$ is attained by repeating the random grouping many times such that every leaf of each fern possesses a probability belonging to each class

$$P(F_m = k|C = c_i) = \frac{N_{k,c_i} + N_r}{N_{c_i} + K \times N_r} \tag{4.15}$$

where N_r is a uniform Dirichlet prior to avoid zero probability. As a class view is found, we simply update the probabilities stored in the leaves that are reached. More simply, new classes are added by creating new conditional probabilities of each leave of each fern.

Correspondingly, for a query feature patch p, it is labeled by dropping its compact features $\{f'_1, f'_2, .., f'_n\}$ down each fern to certain leaves and multiplying these reached conditional probabilities

$$\begin{aligned} \hat{c}_i &= \arg\max_{c_i} P(C = c_i|f'_1, f'_2, .., f'_n) && (4.16)\\ &\approx \arg\max_{c_i} \prod_{j=1}^{M} P(L(F_j, p)|C = c_i), && (4.17) \end{aligned}$$

where $P(L(F_j, p)|C = c_i)$ is the posterior probability stored in the leaf $L(F_j, p)$ of fern F_j reached by the random features. If 30 ferns of size $S = 30$ are used, the query features $\{f'_1, f'_2, .., f'_n\}$ needs also to be randomly partitioned into 30 groups of size 30. The simplicity of binary features determines the computational efficiency of this patch classification.

4.4.2 SURF Detector, Descriptor and Nearest-Neighbor Classifier

4.4.2.1 SURF Detector

The alternative interest point to corners is the blob-like point. The effective way for its localization is the Hessian detector. It explores the second order derivatives of a local window (the Hessian matrix) and returns a blob-like structure that corresponds to the maxima determinant of this matrix

$$\mathbf{H} = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma_D) & I_{xy}(\mathbf{x}, \sigma_D) \\ I_{xy}(\mathbf{x}, \sigma_D) & I_{yy}(\mathbf{x}, \sigma_D) \end{bmatrix} \tag{4.18}$$

where I_{xx} etc. are second-order Gaussian image derivatives. Similar to the Harris detector, the Hessian detector is usually extended to be scale-invariant by using the laplacian of Gaussian to select the scale. To efficiently compute it, Bay *et al.* [Bay 2006] proposed the Speeded-Up Robust Features (SURF) which roughly approximates the Hessian matrix by using a set of box-type filters. The filters are built by cropping the gaussian second-order partial derivatives over continuous parts such that integral images, independent of filter size, can be used. Moreover, the determinant of the approximated Hessian is given by

$$\det(\mathbf{H}_{approx}) = \mathbf{D}_{xx}\mathbf{D}_{yy} - (0.9\mathbf{D}_{xy})^2 \tag{4.19}$$

with smallest estimation error to compensate the error introduced by the approximated values of $\mathbf{D}_{xx}$, $\mathbf{D}_{yy}$ and $\mathbf{D}_{xy}$. This efficient implementation leads to much faster yet similarly accurate blob localizations.

4.4.2.2 SURF Descriptor

SURF also provides a specific way to describe the detected interest points. It first estimates a reproducible orientation based on the wavelet responses of a circular region and align a square region to the orientation. Then, it splits up the region into 4×4 square sub-regions and describe each sub-region with the sum of its wavelet responses,

$$\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{4.20}$$

where d_x etc. the Haar wavelet response in the x direction. $||$ denotes the absolute operation. Thus, each sub-region yields a four-dimensional descriptor and a descriptor vector for all 4×4 sub-regions of length 64. As a result, such a vector shows robustness against both geometric deformations and illumination offsets.

4.4.2.3 K-D Nearest-Neighbor Classifier

Labeling a query SURF descriptor $\mathbf{V}_q$ is to find its nearest neighbor from the training feature database

$$\hat{c}_i = \arg\min_{c_i} \parallel \mathbf{V}_q - \mathbf{V} \parallel, \tag{4.21}$$

During training, we extract SURF descriptors from interest points and adopt a k-d tree as our classifier. As each new view arrives, we renew the existing descriptor to be a weighted mean.

Practically, the k-d tree classifier is replaced with the optimized trees implemented in the public FLANN library [Muja 2009]. It first uses cross-validation to identify the best algorithm among many modified k-d trees, e.g. k-means trees and randomized kd-trees. Meanwhile, it also uses cross-validation to search for optimal configurations that best trade off given query and memory costs. In addition, the nearest-neighbor classification is speeded up via the approximate best-bin-first search strategy [Beis 1997].

4.5 3D-to-2D Correspondences Setup

The actual classification is made in a simple discriminative way. Let us denote the largest and second largest classification responses of a query patch k as $\mathcal{L}_k(\hat{c}_1)$ and $\mathcal{L}_k(\hat{c}_2)$ respectively. The acceptance correspondence is the one that satisfies

$$\frac{\mathcal{L}_k(\hat{c}_1)}{\mathcal{L}_k(\hat{c}_2)} \geq \sigma_1 \tag{4.22}$$

where c_i represents the estimated class label. To be symmetric, there should be a similar measure for each training class. Thus, the acceptance condition becomes

$$\frac{\mathcal{L}_k(\hat{c}_1)}{\mathcal{L}_k(\hat{c}_2)} \geq \sigma_1 \ \ and \ \ \frac{\mathcal{L}_c(\hat{k}_1)}{\mathcal{L}_c(\hat{k}_1)} \geq \sigma_2, \tag{4.23}$$

where $\mathcal{L}_c(\hat{k}_1)$ is the largest response to training class c. The values of δ_1 and δ_2 are varied in the range of $0.49 \sim 0.59$, depending on the number of available correspondences. As each class has a known 3D position, feature matching actually establishes a set of 3D-to-2D correspondences that allows us to derive 3D poses from them.

4.6 3D Pose Recovery

Given all the 3D-to-2D correspondences via feature matching, the objective is to recover the underlying 3D positions of orientations of a camera. It forms a typical Perspective-n-Point problem or PnP in short as so-named in computer vision field. Many solutions to this problem have been proposed, which can been broadly classified into iterative and non-iterative ones. Among the non-iterative methods, the Direct Linear Transformation (DLT) algorithm [Hartley 2004] is the popular one that efficiently handles the cases of arbitrary n. However, it is such simple that generally the estimate result is less stable. A recent algorithm that has the complexity of $O(n)$ was proposed in [Moreno-Noguer 2007a]. It leads to pose estimates as accurate as iterative ones yet much faster to compute. Therefore, We adopt this algorithm to avoid that large number of correspondences slow tracking speed down. We now briefly introduce how it works.

Suppose we have a set of correspondences with known 3D coordinates $\{\mathbf{P}_i\}$ and known 2D image projections $\{\mathbf{u}_i\}$, $i = 1, 2, ..., n$. The algorithm attempts to seek four control points such that all others can be expressed as a weighted sum of them. That is, each reference point is expressed as

$$\mathbf{P}_i = \sum_{j=1}^{4} \alpha_{ij}\mathbf{c}_j^w \quad \text{with} \quad \sum_{j=1}^{4} \alpha_{ij} = 1. \tag{4.24}$$

where the α_{ij} are homogeneous barycentric coordinates, that can be uniquely computed and $\mathbf{c}_j^w$ is the jth control point in the world coordinate system. Similarly in the camera coordinate system, the same relation holds and we also write

$$\mathbf{P}_j^c = \sum_{j=1}^{4} \alpha_{ij}\mathbf{c}_j^c \tag{4.25}$$

where $\mathbf{P}_j^c$ and $\mathbf{c}_j^c$ are the respective correspondents of $\mathbf{P}_j$ and $\mathbf{c}_j^w$ in the camera coordinate system. By choosing the $\{\mathbf{c}_j^w\}$ properly, we have the following relationship

$$w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{P}_i^c = \mathbf{A}\sum_{j=1}^{4} \alpha_{ij}\mathbf{c}_j^c \tag{4.26}$$

where

$$\mathbf{u}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{c}_j^c = \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{4.27}$$

In this equation, the unknown variables are the control camera coordinates $\mathbf{c}_j^c$, $j = 1, 2, 3, 4$. By concatenating the equations for all the reference points, we have a linear system of the form

$$\mathbf{Mx} = 0, \quad \text{with } \mathbf{x} = [\mathbf{c}_1^{cT}, \mathbf{c}_2^{cT}, \mathbf{c}_3^{cT}, \mathbf{c}_4^{cT}]^T \tag{4.28}$$

It is easy to see that the solution $\mathbf{x}$ belongs to the null space of $\mathbf{M}$, or equivalently the null space of matrix $\mathbf{M}^T\mathbf{M}$ in the case of $2n > 12$

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{v}_i \tag{4.29}$$

where $\mathbf{v}_i$ is the ith null eigenvectors of $\mathbf{M}^T\mathbf{M}$. In theory the value of N ought to vary from 1 to 4, corresponding to the projection model changing from the perspective projection to the orthogonal one. The final solution corresponds to the one with the smallest reprojection error

$$\min_{\mathbf{R},\mathbf{T}} \sum_i \|\mathbf{A}[\mathbf{R}|\mathbf{T}]\mathbf{M_i} - \mathbf{m_i}\|^2, \tag{4.30}$$

subject to the distance constraints between pairs of control points. The N is assigned as the one that yields the smallest reprojection error.

As usual, the above closed-form solutions are refined using iterative Gauss-Newton optimization. Specifically, it is only performed with respect to the four coefficients $\beta = [\beta_1, \beta_2, \beta_3, \beta_4]^T$. As a result, this step brings higher estimate accuracy, with negligible computational burdens. More details about how to determine the value of N is thoroughly described in the original paper.

4.7 Experimental Results

We integrated all the above components into a C/C++ system that comprises head modeling, unsupervised learning and online tracking modules. To test its performance, we used a USB camera to capture five video sequences at a resolution of 640×480. Their representative frames are shown in Figure 4.6 while detecting interest points at their frontal training images is shown in Figure 4.7. For each tracking, we render six perspective textures at alternative viewpoints by drawing γ uniformly from the interval $[-60°, 60°]$, and α, β from the interval $[-45°, 45°]$. For the affine transforms, θ and ϕ are drawn uniformed from the interval $[-\pi; +\pi]$, and λ_1 and λ_2 from the interval $[0.6; 1.5]$. This range of scales in conjunction with the multi-scale selection by interest point detection can handle sufficiently large scale changes of the target head. The tracking quality is measured in terms of the number

Figure 4.6: From left to right, snapshots of the test videos 'v1', 'v2', 'v3', 'v4' and 'v5'.

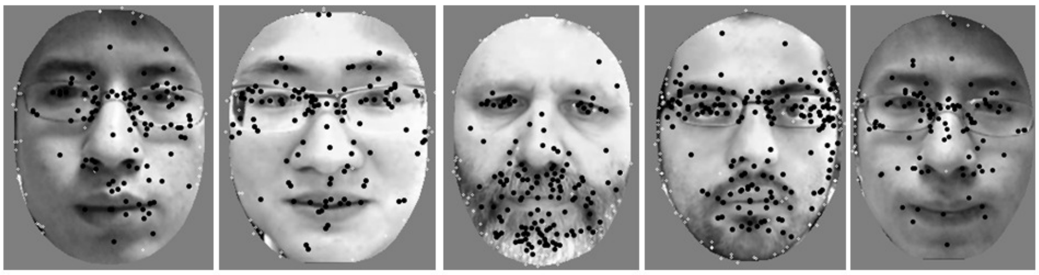


Figure 4.7: The feature point detections at the frontal view of each tester. Their distributions spread over the entire faces with major focus on salient regions, e.g. glasses, mouth and moustache. Note that the SURF detector is used here.

of inliers n_{in} and the reprojection error e. If $e < 3.0$ *pixels* and $n_{in} > 5$ are satisfied, we say that the estimated head pose is correct at this time.

The first experiment tested the better combinations between the local binary feature + ferns classifier (FERNS) and the SURF descriptor + flann K-D Trees (SUNN). The test sequence 'v1' was used with the length of 500 frames. During the training, the SUNN learns about 323 model feature points while FERNS learns only 167 ones. The results are shown in Figure 4.9. We see that the SUNN has a higher pose recognition rate, which reflects the better discriminative ability of the SURF descriptor. On the other hand, the FERNS outperforms the SURF in terms of the tracking speed. A more precise comparison on the sequence 'v1' is presented in Table 4.7. For accuracy purpose, we adopt the SUNN in all the subsequent experiments. Practical applications can switch to the FERNS for real-time performance.

The merit of multi-view class detection is illustrated in Figure 4.8. Clearly, it leads to matching more interest points. Figure 4.11 shows a set of representative frames. Faces are successfully tracked in spite of the challenges of fast motion, facial expressions, partial occlusions and large illumination changes.

The tracking accuracy of our system was also assessed by conducting experiments on the ground-truth database from Boston University [Cascia 1999]. The testing sequences were first resized from 320 $\times$ 240 to 640 $\times$ 480 to increase the number of feature points. Figure 4.12 depicts the recovered 3D head poses on the 'ssm1'

	$e(pixels)$	n_{in}	$t_d(ms)$	$t_m(ms)$	$t_p(ms)$
FERNS	2.07	14.9	15.67	45.52	2.11
SUNN	2.27	26.7	80.82	12.0	1.24

Table 4.1: Precise comparisons in the case of 100 model feature points. From left to right, the respective items are the reprojection error, the number of inliers, the feature extracting time, feature matching time and the pose recovery time. Obviously, the FERN reduces the time cost by shifting the burden of feature extraction to feature classification. This however leads to a loss of almost half feature inliers.

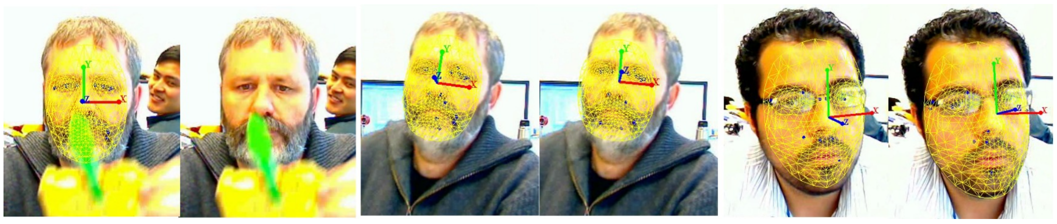


Figure 4.8: Comparing the tracking performances of single-view with multi-view class detection. The left image of each pair is the single-view based performances. Both tracking inaccuracy and failures are reduced due to the fact that more class views lead to more matches.

sequence. As can be seen, the estimated curves are consistent with the ground truth most of the time except the small shifts in the angle graphs that are caused by the different rotation axes of our head model.

Finally, we compare our system with the popular Watson system [1] that is a representative of the recursive tracking algorithms. For authenticity, we directly run the provided Watson API with our test videos to capture snapshots. The same intrinsic parameters are used via a camera calibration. Several snapshots are shown in Figure 4.13. The comparison is not fully fair in that we use a person-specific model while Watson uses a generic ellipsoid model. But one can still see the significant difference that Watson suffers from drifting while ours does not.

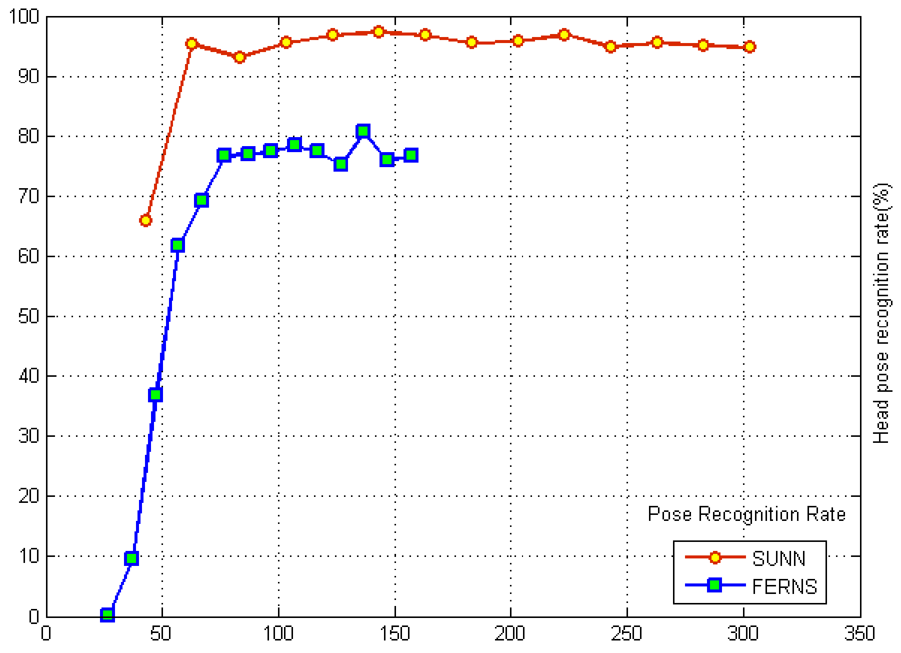
4.8 Conclusion

This chapter presents a 3D head tracking approach that requires a short-term learning stage and relies on tracking by detection. To handle wide-baseline matching, we propose a multi-view class detection scheme by bridging modeling and learning. The role of a 3D model during learning is fully investigated by texture and view synthesis, stable interest point selection and multi-view class detection. The derived system succeeds in getting rid of the repetitive tracking-failure-reinitialization

[1]http://groups.csail.mit.edu/vision/vip/watson/index.htm

module as in the traditionally recursive tracking systems.

Pure wide-baseline matching brings two problems. Without temporal consistency constraint makes the estimated poses jittering. Moreover, the phenomena of large ratio of outlier matchings and insufficient matchings makes our system lost in certain frames. The next chapter will address how we try to solve these two problems.



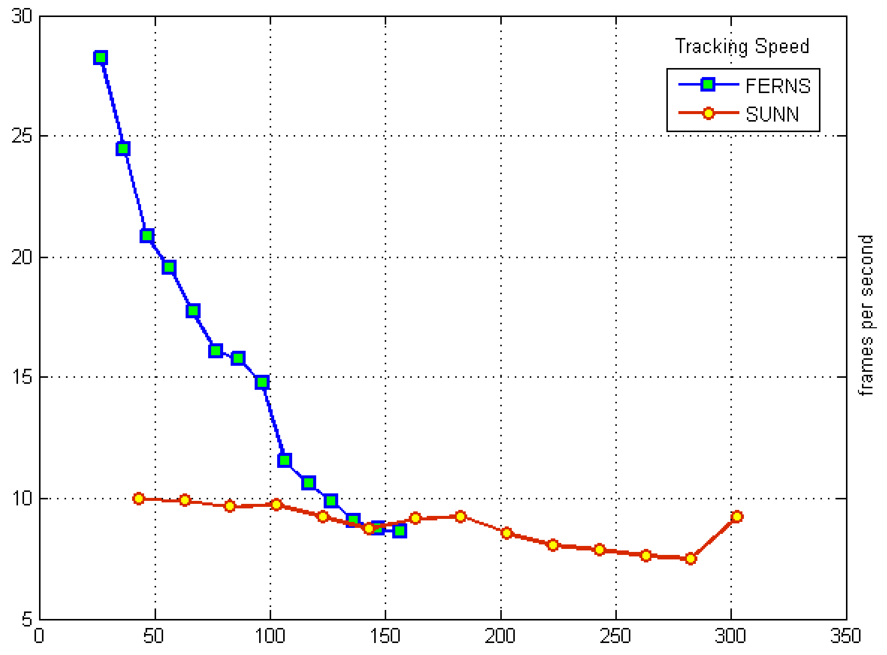


Figure 4.9: Comparing FERN and SUNN with respect to the pose recognition rate and tracking speed. We show the averages over the 500-frame-long video sequence as varying the number of model feature points. We say that head pose is recognized if its depicting frame is tracked successfully.

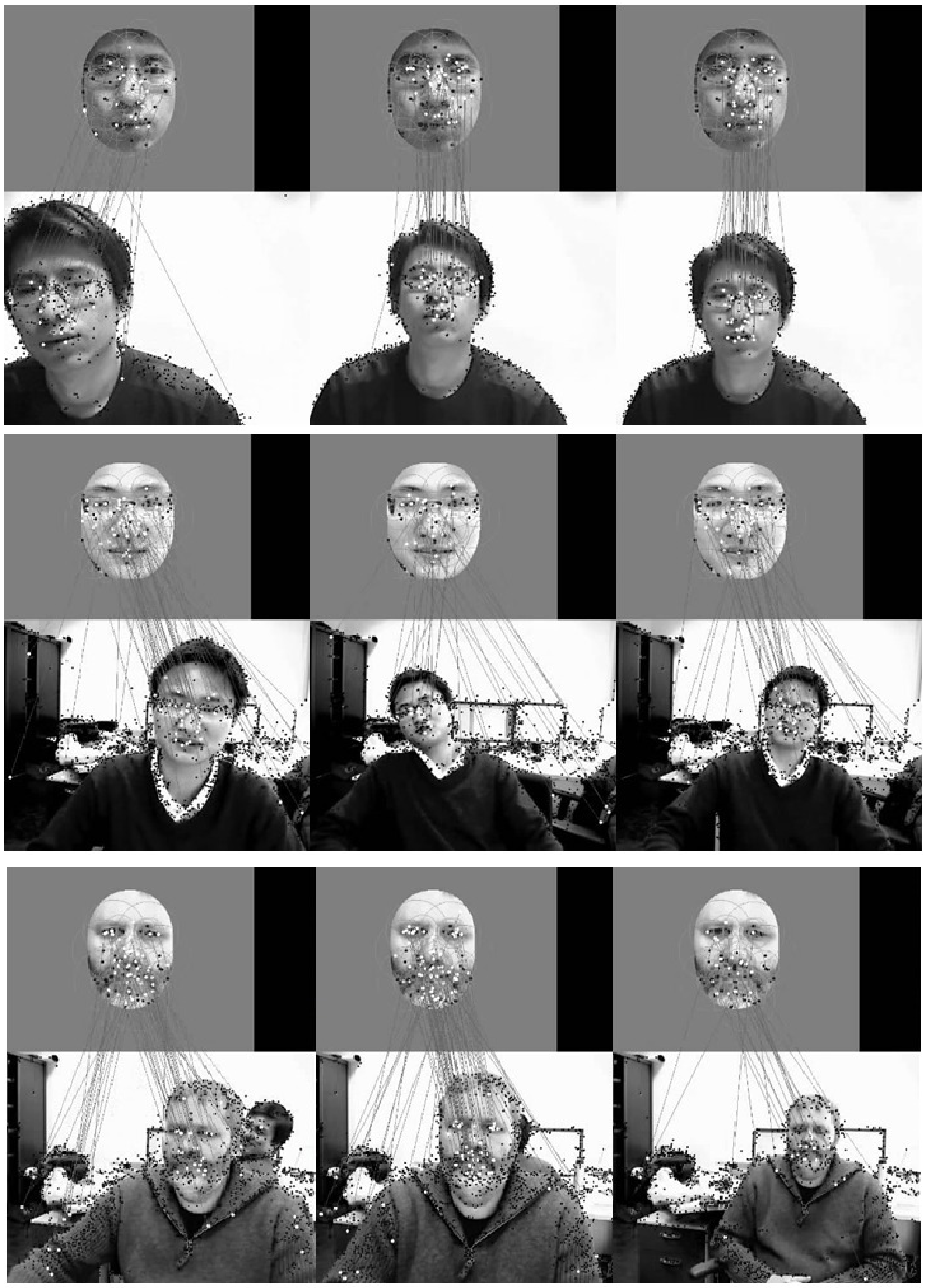


Figure 4.10: SUNN-based feature matchings. At the first row, fewer outliers happen since the background is fairly flat while the left rows contain many mismatches due to the cluttered background scene. Note that poses are correctly estimated for all these frames with the help of RANSAC.

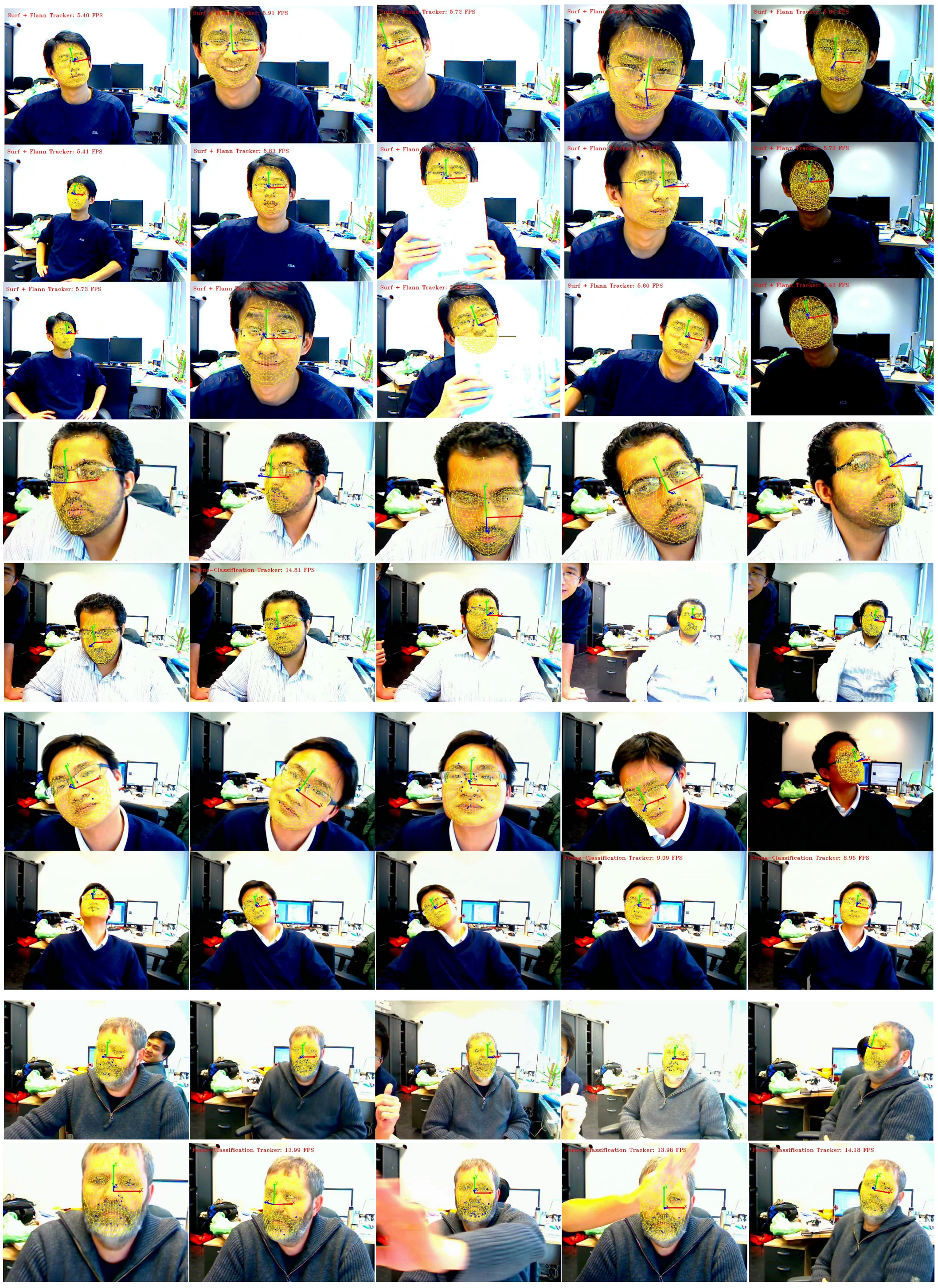


Figure 4.11: The first three rows show tracking of a face in the challenging video ‘v5’. From left to right, each column represents one of the following challenges: low face resolution, facial expressions, partial occlusions, large pose variations and illumination changes. The left three rows show snapshots of tracking the videos ‘v2’ ‘v3’ and ‘v4’, respectively.

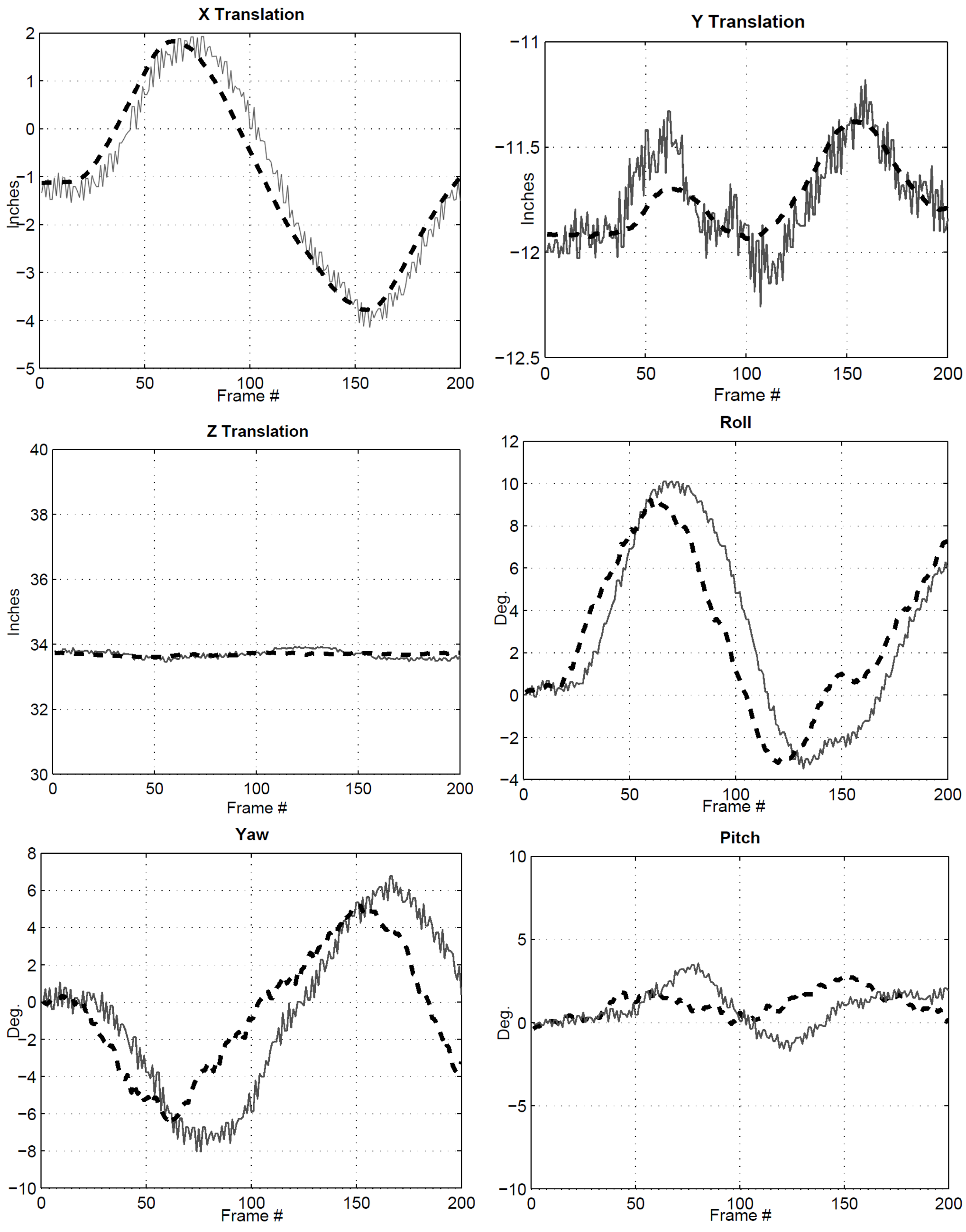


Figure 4.12: Illustrating the tracking accuracy of our system on the sequence 'ssm1' from Boston University [Cascia 1999]. In all graphs, the dashed broad curve depicts the estimated poses and the solid curve depicts the ground truth. The shift in the orientations is due to that our rotation origin (defined with our model) differs from that of the ground truth.

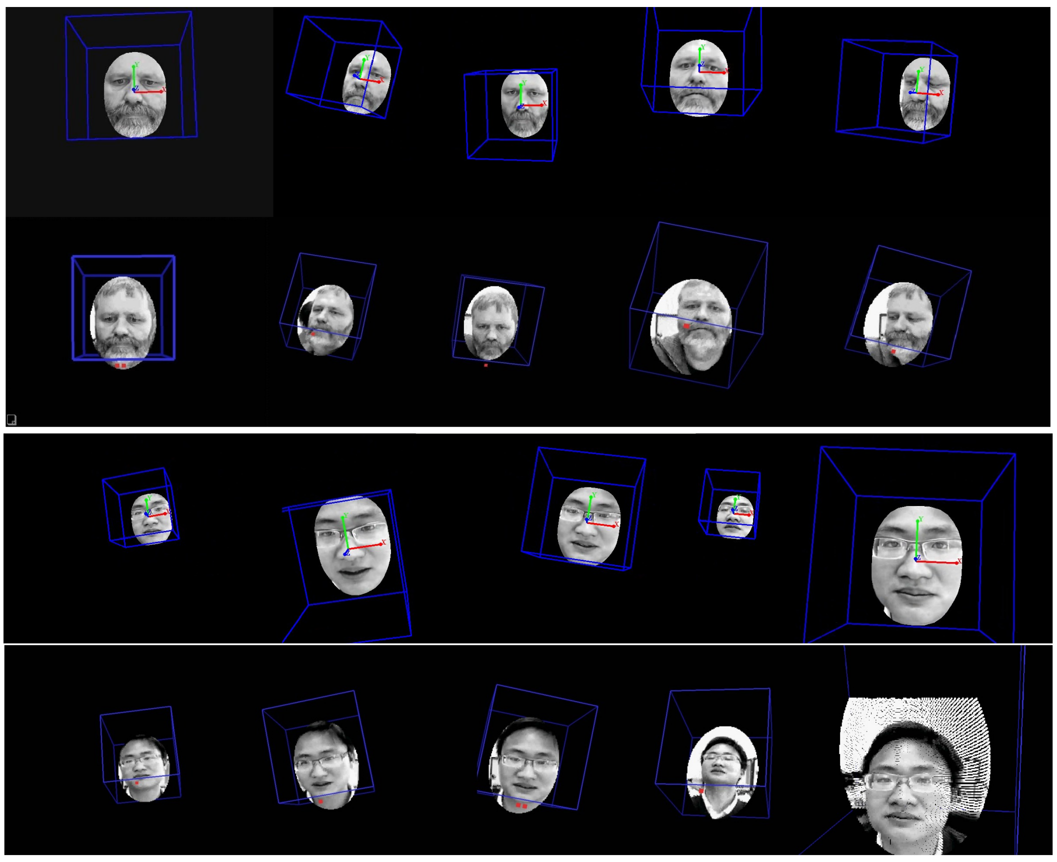


Figure 4.13: Comparison with the Watson system on the videos 'v3' and 'v2'. The bounding box as well as the axes indicate the estimated 3D pose. In each group, the upper row shows the results of our system while the lower depicts the results of the Watson system. Due to being insensitive to drift, the bounding boxes of our system fit the human faces more precisely.

Chapter 5
Temporal Consistency and Discriminative Color Prior

In the last chapter, we have presented a 3D head tracking solution based on pure wide-baseline feature matching. It overcomes the drifting difficulty in differential tracking and leads to robust and stable pose estimates. However, it suffers from two new problems that severely affect its performances. On one hand, temporal coherence is not considered and the motion discrepancy between consecutive frames can be large, thus yielding jittering. On the other hand, the used spatial descriptor is not sufficiently discriminative such that many mismatches are introduced to reduce tracking speed and even cause tracking failures. In this chapter, we present our approaches that fuse more visual cues to tackle them.

5.1 Temporal Consistency

Motion smoothing is one of the fundamental topics in visual tracking. A smoothing filter was proposed to smooth the outputs of particle filter in the forward-backward scheme [Isard 1998]. This approach however can not be transferred to tracking-by-detection as temporal prediction is still propagated in it. Alternatively, the simple way is to add a smoothing regularization term in the cost function composed of image detection correspondences. But defining an appropriate term is not trivial. Too tight a term would over-smooth the raw motion while too loose one would hardly take effect. The usually feasible method is to assign an empirical term via cross-validation and repeat the process in a new scenario. Recently, many researchers realize that transient image features can be used as smoothness constraint. One typical example is in face tracking, e.g. Vacchetti *et al.* [Vacchetti 2004] proposed using cross-reprojected features to penalize the motion estimated from wide baseline matching. However, the currently used features either are complicated to compute [Vacchetti 2004] or require semantic knowledge for guiding local search [Zhang 2008]. There is a need to discover a much simpler yet efficient feature.

Full-scale based tracking can be smoothed by neighboring search based method. Our idea is to establish a set of optical flow correspondences as smoothness constraints. This mainly comes from the temporal consistency attribute of optical flow. The relevant work is in 3D pose tracking. In [Brox 2006], optical flow correspondences are fused with other wide-baseline feature correspondences by assigning a relative weight empirically. However, finding an optimal weight is often time-consuming or even infeasible and there is no support to justify its optimality for all experiments. An alternative related work is in [Decarlo 2000], which treats optical flow as a hard constraint to serve edge and shape correspondences. The problem is that it cannot handle large target motion as flow smoothness is over emphasized as such a constraint. To overcome these limits, we assign no relative weights but adapt the number of flow correspondences to that of the feature ones to automatically adjust their contributions. We also associate flow and feature correspondences by sampling a set of image points that have both feature and flow correspondences. By biasing for one while against the other, this association allows us to further tune their relative contributions in small scale. Furthermore, without any prediction, we treat all correspondences equally and adopt a robust algorithm to select inliers that are consistent with a global geometric model. In such a way, the dilemma of fusing large motion displacements in the feature correspondences and local motion smoothness in the flow ones are naturally solved.

5.1.1 The Bayesian formulation

A typical tracking problem is usually formulated in terms of a dynamical process. The sequences of states of the tracked object and image observations up to time t are denoted by $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$ and $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$, respectively. The goal of tracking at time t is to estimate the state $\mathbf{x}_t$ from the available observations $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$. Given this notation, temporal consistency is to restrict the derivative of any two consecutive states to a small range

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2 < \delta, \tag{5.1}$$

where δ is a small positive threshold. This consistency is often valid as long as video capture rate is not too low. We will describe its roles in visual tracking in the following subsection.

The tracking-by-detection framework does not incorporate temporal consistency and thus leads to jittering or even trembling results. To solve it, one solution is to associate $\mathbf{x}_t$ with the estimated value $\widehat{\gamma_{t-1}}$ of the last state $\mathbf{x}_{t-1}$. For example, we can take $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ to be

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto \begin{cases} K(-\|\mathbf{x}_t - \widehat{\gamma_{t-1}}\|_2), & \text{if } \widehat{\gamma_{t-1}} \text{ is valid} \\ constant, & \text{otherwise} \end{cases} \tag{5.2}$$

where $K(\cdot)$ is a kernel function, e.g. a Gaussian kernel. Consequently, we should consider a new objective pdf $p(\mathbf{x}_t = \gamma, \mathbf{x}_{t-1} = \widehat{\gamma_{t-1}}|\mathbf{z}_{1:t})$. From the Bayes' rule, we can derive that:

$$p(\mathbf{x}_t = \gamma, \mathbf{x}_{t-1} = \widehat{\gamma_{t-1}}|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t = \gamma) \tag{5.3}$$

Since $p(\mathbf{x}_{t-1} = \widehat{\gamma_{t-1}}|\mathbf{z}_{1:t-1})$ is independent of γ, the estimator to select the best γ becomes

$$\widehat{\gamma_t} = \arg\max_{\gamma} p(\mathbf{z}_t|\mathbf{x}_t = \gamma)p(\mathbf{x}_t = \gamma|\mathbf{x}_{t-1} = \widehat{\gamma_{t-1}}), \tag{5.4}$$

which simultaneously maximizes $p(\mathbf{z}_t|\mathbf{x}_t = \gamma)$ and minimizes the relative state discrepancy as introduced in $p(\mathbf{x}_t = \gamma|\mathbf{x}_{t-1} = \widehat{\gamma_{t-1}})$, when $\widehat{\gamma_{t-1}}$ is valid. Therefore, in

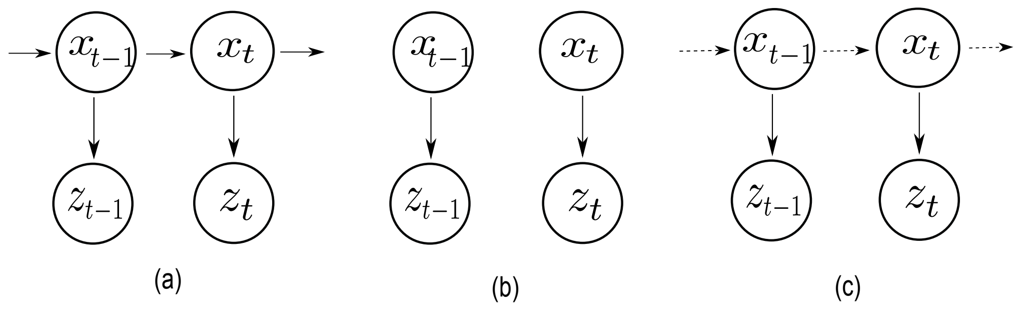


Figure 5.1: Illustrating the three different tracking frameworks. (a) The Bayesian tracking framework under a Markovian chain model; (b) The independent tracking-by-detection framework; (c) The tracking-by-detection framework with temporal consistency constraints. In this graph, the dashed line implies that $\mathbf{x}_{t-1}$ affects $\mathbf{x}_t$, but unlike the solid line, $\mathbf{x}_{t-1}$ does not necessarily cause $\mathbf{x}_t$.

such a framework (as shown in Figure 5.1 (c)), temporal consistency is considered as a smoothness *constraint* to regularize large relative state estimations.

5.1.2 Incorporating Optical-Flow Correspondences

Jitters can be reduced by minimizing the relative pose discrepancy between consecutive frames [Lepetit 2005]. Among many eligible features, we find that optical flow is particularly suitable due to its easy computation and low cost. In addition, many mature optical flow algorithms provide us a lot of options to choose in facing different situations. To simplify our problem yet without loss of generality, we neglect the routine of forward-backward smoothing [Isard 1998] and restrict ourselves to forward smoothing, meaning that we merely smooth the raw state $\mathbf{x}_t$ based on the previous ones $\{\mathbf{x}_{t-1}, ..., \mathbf{x}_1\}$. For the same reason, only $\mathbf{x}_{t-1}$ is involved. Consequently, our problem is degenerated to smooth the raw output $\mathbf{x}_t$ from the knowledge of $\mathbf{x}_{t-1}$, $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$.

Our idea is to use sparse optical flow that is calculated locally as the additionally smoothing correspondences. Among the plentiful methods, the Lucas-Kanade algorithm provides an efficient way to track such sparse optical flow [Lucas 1981]. Within a small window, this algorithm estimates the flow velocity at a pixel i in the image I via

$$\mathbf{A}\mathbf{v}_i = \mathbf{b} \tag{5.5}$$

where

$$\mathbf{A} = \begin{bmatrix} \sum_w I_x^2 & \sum_w I_x I_y \\ \sum_w I_x I_y & \sum_w I_y^2 \end{bmatrix}, \quad \mathbf{v}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} -\sum_w I_x I_t \\ -\sum_w I_y I_t \end{bmatrix}, \tag{5.6}$$

where I_x and I_y are the spatial derivatives and I_t is the time derivatives. u_i and v_i are the components of the flow velocity. w is the size of the local window. We

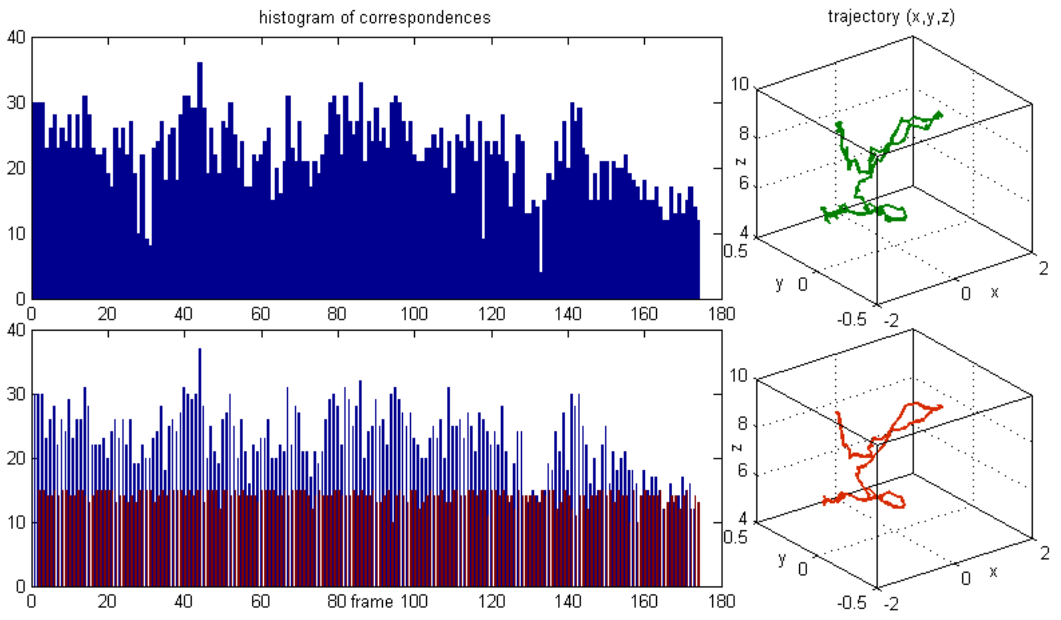


Figure 5.2: Histogram of the number of inlier correspondences and tracking trajectory. The upper row shows the case of feature correspondences (blue bar) and the estimated (x, y, z) trajectory. The lower row shows the case of combined feature and optical-flow (red bar) correspondences and the estimated trajectory as well. Fifteen flow points are randomly sampled at each frame. Note that the lower trajectory appears to be smoother than the upper one.

employ the iterative multi-resolution implementation proposed in [Bouguet 2002] to achieve robust performance.

To illustrate its smoothness role, let us consider the general case of tracking 3D motion based on a sufficient set of 3D-to-2D correspondences, which is posed as a Perspective-n-Point (PnP) problem. Suppose there are N_f pairs of feature correspondences $\mathcal{C}_f$ and N_s pairs of optical-flow correspondences $\mathcal{C}_s$. As $\mathbf{x}_t$ is valid, we aim at estimating $\mathbf{x}_{t+1}$ by minimizing the reprojection errors of these given correspondences

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}_{t+1}}\left(\sum_{i=1}^{N_f} \| \Phi(\mathbf{P}^i, \mathbf{x}_{t+1}) - \mathbf{p}^i_{t+1} \| + \sum_{j=1}^{N_s} \| \Phi(\mathbf{P}^j, \mathbf{x}_{t+1}) - \mathbf{p}^j_{t+1} \|\right), \quad (5.7)$$

where Φ is the 3D-to-2D projection function. Typically the solution is a linear combination of the eigenvectors of a null space spanned by $\mathcal{C}_f$ and $\mathcal{C}_s$ [Hartley 2004, Moreno-Noguer 2007a]. As $\mathcal{C}_f$ comes from feature matching via full-scale search, it implicitly introduces large displacements in the null eigenbasis. Alternatively, $\mathcal{C}_s$ arises from short-baseline tracking and thus pullbacks the large-displaced eigenbasis to small scale. An experiment with and without $\mathcal{C}_s$ is depicted in Figure 5.2.

To make $\mathcal{C}_f$ and $\mathcal{C}_s$ collaborate properly, there remains several tightly coupled issues to deal with. The first issue is how to select optical-flow seeds and establish their 3D-to-2D correspondences. The second is to balance the relative contributions of large displacements in $\mathcal{C}_f$ and temporal consistency in $\mathcal{C}_s$. Meanwhile, the issue of discarding outliers in both $\mathcal{C}_f$ and $\mathcal{C}_s$ should also be considered. The following describes how we handle these issues.

5.1.3 Balancing the Two Worlds

The selected optical flow seeds should satisfy two conditions: having solvable flow equations (5.5) and having clearly known 3D positions to establish $\mathcal{C}_s$. To this end, we consider two sparse sources: A - image features matched at previous frame; B - a subset of randomly drawn vertices. If we denote their respective optical-flow correspondences as $\mathcal{C}_s^{(A)}$ and $\mathcal{C}_s^{(B)}$, then we have $\mathcal{C}_s = \mathcal{C}_s^{(A)} \cup \mathcal{C}_s^{(B)}$. Besides the smoothness role, $\mathcal{C}_s^{(B)}$ and $\mathcal{C}_s^{(B)}$ have two additional functions. First, $\mathcal{C}_s^{(A)}$ might have both a feature and an optical-flow correspondences (termed as 'competitive' correspondences). This links the two worlds to allow us to flexibly bias for one while against the other. Second, $\mathcal{C}_s^{(B)}$ provides complementary correspondences in the cases of inadequate feature correspondences.

Balancing the contributions of $\mathcal{C}_f$ and $\mathcal{C}_s$ and rejecting outliers are simultaneously considered. Since $\mathcal{C}_f$ and $\mathcal{C}_s$ come from two different worlds, the cost function (5.7) leaves us a bi-criterion optimization problem. The conventional approach simply assigns a relative weight to each term [Brox 2006] or treats optical flow as a constraint [Decarlo 2000]. The shortcomings are that the two worlds are regarded independently and the empirical weights are as varied as each experiment. Instead of doing so, we assign no relative weights but equally minimize their reprojection errors with the help of a robust Progressive Sample Consensus (PROSAC) [Chum 2005] algorithm. This has two advantages: First, only these correspondences in $\mathcal{C}_f$ and $\mathcal{C}_s$ are selected if they are consistent with a global geometric model. Second, too large feature and too small flow correspondences are naturally discarded as outliers.

Still in favor of tracking-by-detection, we take the following simple actions to encourage feature correspondences. The first action is setting the respective size of A and B at current frame as adaptive as the N_f matched in the previous frame: $N(\mathcal{C}_s^{(A)}) = N_f$, $N(\mathcal{C}_s^{(B)}) = N_f$. The insight is that more feature correspondences need more flow correspondences to smooth themselves, and vice versa. Thus, N_s is given by $N_s = inlier(N(\mathcal{C}_s^{(A)})) + inlier(N(\mathcal{C}_s^{(B)}))$, where $inlier()$ denotes the number of inliers of an entry. The second action is that for the element in $\mathcal{C}_s^{(A)}$, if it has a pair of 'competitive' correspondences, its flowed position is biased for its matched feature position. To summarize, the following operations take place to combine the two worlds:

At frame $t-1$, with an estimated $\widehat{\mathbf{x}_{t-1}}$,

1. add the reprojected positions of all the n matched feature correspondences under $\widehat{\mathbf{x}_{t-1}}$ as optical flow seeds;

2. randomly draw n visible vertices of the 3D model and add their reprojected positions under $\widehat{\mathbf{x}_{t-1}}$ as optical flow seeds.

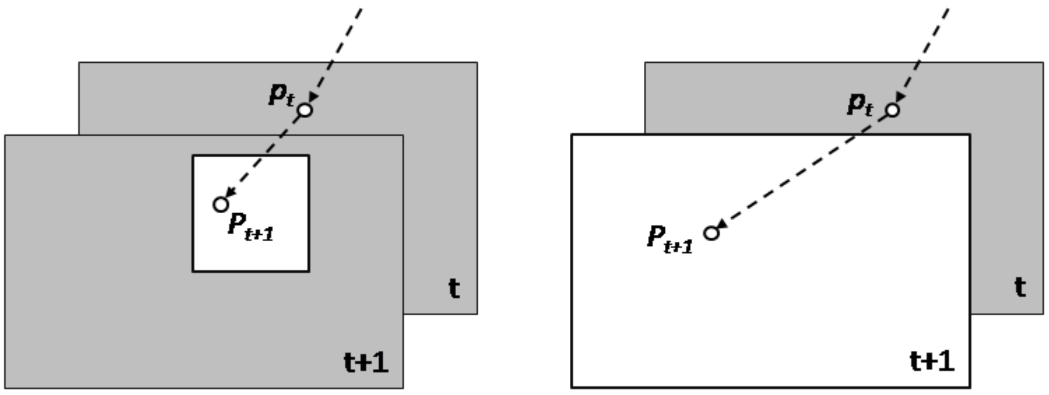


Figure 5.3: Starting from the same original point $\mathbf{p}_t$ at frame t, optical flow seeks its new position $\mathbf{p}_{t+1}$ in a neighboring window around $\mathbf{p}_t$ (the left figure) based on its normal equation while feature feature looks for its new position $\mathbf{p}_{t+1}$ among the feature points detected in the entire frame and acquires the one that is best matched (the right figure). Motion smoothness is naturally embedded in the neighboring search of optical flow computation.

At frame t, after feature matching and optical flow tracking,

1. select inliers from the pool of feature and optical-flow correspondences with PROSAC and optimize equation (5.7) to compute a raw $\widetilde{\mathbf{x}}_t$ with these inliers;

2. for a point having 'competitive' correspondences, if its optical-flow correspondence is chosen as an inlier, the flowed position is tuned to be the mean of the flowed and feature-matched positions $(u_o^{'}, v_o^{'}) = (\frac{u_f+v_o}{2}, \frac{u_f+v_o}{2})$;

3. repeat Step 1 to select new inliers and compute a feature-favored yet optical-flow-smoothed pose $\mathbf{x}_t$ by optimizing equation (5.7).

5.1.4 Evaluations

We conducted experiments on tracking 3D rigid head pose to verify the approach proposed for motion smoothness.

We first compared the proposed smoothing approach with the 'Lowess' smoothing algorithm implemented in Matlab (The other algorithms are also tested but perform a little worse). Specifically, we apply it in both an offline manner (Lowess) and a sequential manner (SLowess). In Figure 5.4, all the three smoothing methods yield similar trajectories. But the proposed approach outperforms both Lowess and SLowess, in that it smoothes out most of the small peaks while the others do not. A typical example is at frame 49, where jittering is successfully filtered merely by our approach. The respective model-mapped faces are also shown there, in the same order of legend display. The remaining two peaks at frames 76 and 118 are caused by erroneous optical-flow measures due to sudden movements.

Several representative frames that justify the smoothness role of incorporating

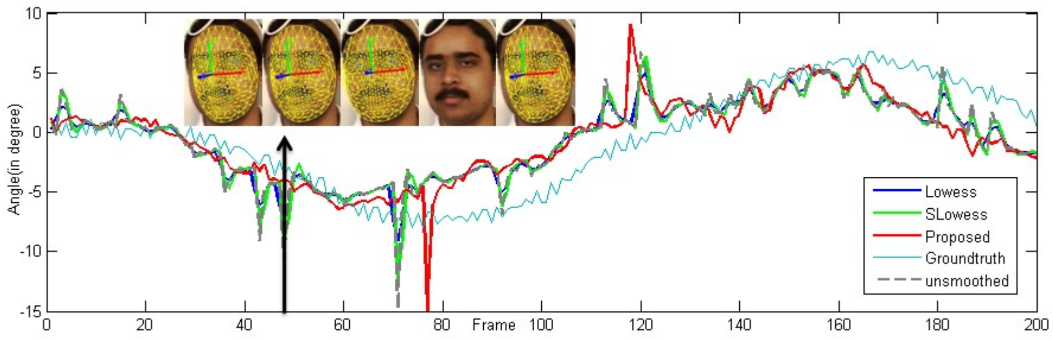


Figure 5.4: Comparing the different smoothing approaches on the roll angle. The results on other motion components are similar and not shown here.

optical-flow correspondences are shown in Figure 5.5. Large jittering are corrected, especially at the moments of 165, 171 and 181.

The last experiment compared the tracking performances of different methods. The experimental data are the same as used in last chapter. The pose tracking rates of four corresponding methods are specified: the combined local feature+optical-flow method (SUNN+OP), local feature matching only (SUNN), 70 optical-flow correspondences only (OP-Only-70), and 35 optical-flow correspondences only (OP-Only-35). As shown in Table 1, the SUNN+OP outperforms others in all the cases. One interesting thing is that OP-Only-70 performs worse than OP-Only-35, which reflects that more samples may be not helpful.

	v1	v2	v3	v4	Average
Video Length	500	626	2010	518	913.5
OP-Only-35	71.6%	79.6%	80.5%	74.7%	76.6%
OP-Only-70	69.4%	77.2%	79.2%	73.2%	74.8%
SUNN	96.0%	87.2%	86.2%	90.0%	89.9%
SUNN + OP	**100.0%**	**94.4%**	**89.6%**	**92.9%**	**94.2%**

Table 5.1: Pose tracking rates on the ‘v1’ ‘v2’ ‘v3’ and ‘v4’ test sequences.

5.2 Discriminative Color Prior

In cluttered scenes, many incorrect correspondences are introduced. How to effectively reduce these outliers remains a challenging problem in matching-based technologies. As for tracking, the estimated pose from previous frame is usually employed as a temporal-prior term to output a prediction region [Vacchetti 2004]. The matches that are out of the region are identified as outliers in either a hard or a soft manner. A simple heuristic variant was also introduced in [Wang 2006].

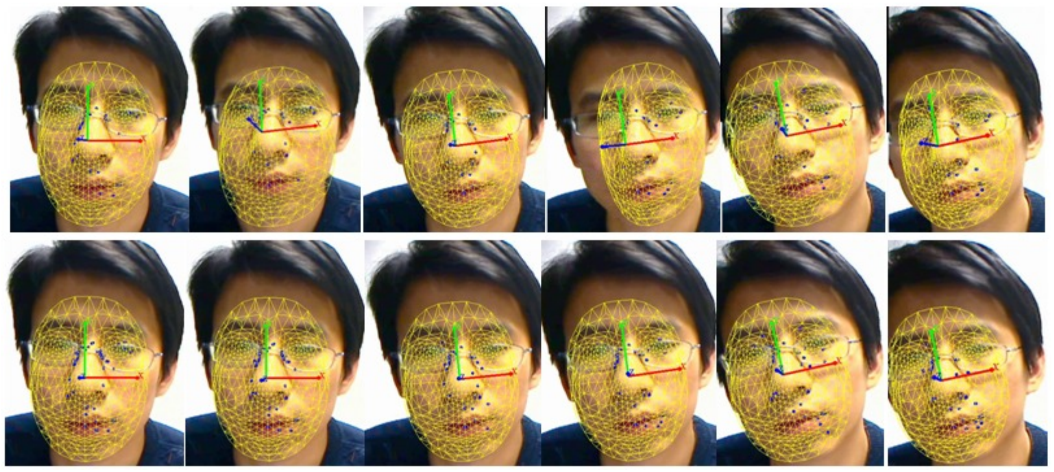


Figure 5.5: Example tracking frames 163, 165, 168, 171, 175, 181 from the 'v1' sequence. The upper row depicts the results without optical-flow correspondences while the lower depicts the results with them.

The main problems with these approaches are that the temporal prior may be not reliable if the prediction is far away from the true position.

Face color, on the other hand, is a stabler cue to indicate rough face localization. In earlier years, color-based methods are widely applied to localize human faces. See [Yang 2002a] for the literature survey. Most of these works focus on finding such a color space and its distribution function that are generative enough to model skin color. Many color spaces have been explored while the Mixture Of Gaussians (MoG) is the most popular function to model its histogram distribution. Unfortunately, neither a perfect color space nor a perfect distribution model has been found. Inspired by feature selection, discriminative color spaces start to be investigated [Collins 2005]. One example is the so-called Fisher color space that best separates foreground and background [Moreno-Noguer 2007b] under the criterion of Fisher discriminant. Such a space is usually sensitive to light illumination disturbances and thus must be dynamically updated.

Our objective is to incorporate color information as a prior term to alleviate outliers as more as possible. Unlike the above mentioned efforts, we attempt to investigate the possibility of using a discriminative distribution model with a simple color space. The underlying idea is that a strongly discriminative model is more likely to create a powerful color representation than a strongly discriminative color space. Our specialities lie in two points: (i) Similar to the spatial descriptor, the color information is represented by its compact covariance descriptor (RGB Sigma Set). (ii) Color prior is modeled as the conditional posterior distributions stored in Random Projection (RP) based classification trees, which are trained with both positive and negative samples. Recently, Yin and Collins [Yin 2009] and Zhou *et al.* [Zhou 2010] have proposed to perform color-based segmentation to create a region prior for the posterior tracking. In contrast, we only considers the colors of the local regions around feature points. The advantage is that some redundant colors are abandoned such that time cost is reduced and precision is improved. As demonstrated later, without any dynamic space or model update, the prior created from the first image is robust enough to recognize inlier features over the sequences that contain sharp illumination changes.

5.2.1 Color Prior Formulation

In feature-based head tracking, the key step is to establish a set of geometric 3D-to-2D correspondences by matching individual features to a database of features learnt from reference images. The feature is computed from a local patch around a keypoint in graylevel space. For the i sample $\widetilde{\mathbf{k}}_i$ in our keypoint database $\left\{\widetilde{\mathbf{k}}_1 \ldots \widetilde{\mathbf{k}}_m\right\}$, denote $\mathbf{f}(\widetilde{\mathbf{k}}_i)$ as its graylevel feature and $\mathbf{U}_i$ its associated 3D position. At run-time, given the keypoint set $\mathbf{K} = \{\mathbf{k}_1 ... \mathbf{k}_n\}$ from a video frame, each geometric correspondence $\mathbf{U}_i \leftrightarrow \mathbf{v}_j$ is determined by the feature matching with highest probability,

$$\arg \max_{\mathbf{U}_i \leftrightarrow \mathbf{v}_j} P(\mathbf{f}(\widetilde{\mathbf{k}}_i) | \mathbf{f}(\mathbf{k}_j)) \tag{5.8}$$

where $\mathbf{v}_j$ is the 2D position of $\mathbf{k}_j$. In practice, this matching can cause outlier correspondences from background clutter, mostly because the used feature descriptor is not discriminative enough. If there were excessive erroneous correspondences, it will be impossible to find the consistent pose parameters. In observation, color information is usually distinctive enough to distinguish face from background clutter. Denote $\mathbf{c}(\mathbf{k}_j)$ as the color feature of keypoint $\mathbf{k}_j$. To integrate color, we change the above decision function to be

$$\arg \max_{\mathbf{U}_i \leftrightarrow \mathbf{v}_i} P(\mathbf{f}(\widetilde{\mathbf{k}}_i) | \mathbf{f}(\mathbf{k}_j)) f(\mathbf{c}(\mathbf{k}_j)) \tag{5.9}$$

where $f(\mathbf{c}(\mathbf{k}_j))$ is a binary function that stands for the inlier/outlier attribute of $\mathbf{k}_j$ given its color feature $\mathbf{c}(\mathbf{k}_j)$. Since it gives a *priori* confidence to feature matching from color measurement, we call it color prior. In the following, we will describe how to yield and use this prior.

5.2.2 Creating the Prior

The color prior is defined in terms of a new color descriptor and a discriminative classifier based on Random Projection (RP) trees. Fig. 5.6 illustrates the pipeline. Color feature $\mathbf{c}(\mathbf{k})$ is represented as a novel RGB Sigma Set vector that captures covariant relationship of color channels. The prior comes from the classification responses by dropping $\mathbf{c}(\mathbf{k})$ down the RP trees based classifier. Specifically, the classifier is trained with labeled color descriptors $\{\{\mathbf{c}(\mathbf{k}_1), y_1\} \ldots \{\mathbf{c}(\mathbf{k}_m), y_m\}\}$ from a reference image, where $y \in \{+1, -1\}$, $+1$ is the label for face region and -1 the label for background clutter. Note that $\mathbf{c}(\mathbf{k})$ will be short as $\mathbf{c}$ in the following sections.

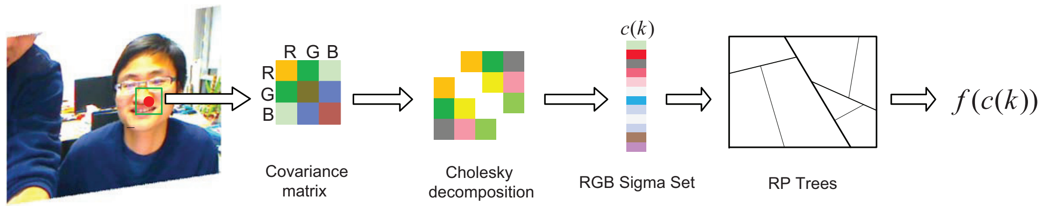


Figure 5.6: Pipeline to yield the color prior. The covariance matrix of RGB values in the local patch of keypoint $\mathbf{k}$ is calculated and the RGB sigma set descriptor $\mathbf{c}(\mathbf{k})$ is built on the Cholesky-decomposed lower triangular matrix of this covariance matrix. Then $\mathbf{c}(\mathbf{k})$ is classified by the trained RP based classification trees. Its discriminative responses are finally used to generate the binary color prior $f(\mathbf{c}(\mathbf{k}))$ of keypoint $\mathbf{k}$.

RGB Sigma Set - The sigma set is a second order statistics calculated from the covariance matrix. Here we adopt the RGB information of each pixel and calculate their covariance matrix. Thus we use a $d_p = 3$ dimensional vector $\mathbf{z}_{ij}, j = 1, 2, \dots, k$ to encode the RGB values of each pixel in one patch. The covariance matrix [Tuzel 2006] of these k pixels is given as

$$\mathbf{C}_i = \frac{1}{k-1} \sum_{j=1}^{k} (\mathbf{z}_{ij} - \mu)(\mathbf{z}_{ij} - \mu)^T \tag{5.10}$$

where μ is the mean vector of the k pixels inside the patch $\mu = \frac{1}{k} \sum_{j=1}^{k} \mathbf{z}_{ij}$. Note that the computational cost of the covariance matrix can be made independent of the size of the patch with the help of pre-computed integral images [Tuzel 2006].

For the covariance matrix does not lie on Euclidean space, its distance computations required for patch classification are usually computationally complex. We use an alternative second order descriptor derived from it, as proposed in [Hong 2009] [Kluckner 2009]. Let $\mathbf{C}_i = \mathbf{L}\mathbf{L}^\mathbf{T}$ be the Cholesky decomposition on the covariance matrix of the patch. A set of d_p-dimensional sigma points $S = \{\mu + \mathbf{l}_1, \mu + \mathbf{l}_2, \dots, \mu + \mathbf{l}_{d_p}, \mu, \mu - \mathbf{l}_1, \dots, \mu - \mathbf{l}_{d_p}\}$ are obtained from the lower triangular matrix $\mathbf{L}$. ($\mathbf{l}_i$ is the α-weighted i^{th} column of $\mathbf{L}$, with $\alpha = \sqrt{k}$) and μ is the mean vector talked beforehand. This sigma set represents the statistics of the patch up to second order, and allows simple similarity measurement on a Euclidean vector space. Then we define the feature vector $\mathbf{c}_j$ for one patch as the concatenation of these points which is a $d = d_p(2d_p + 1)$-dimensional vector and use it as the input of the following classification trees.

RP based Classification - RP tree is proposed for vector quantization [Freund 2007] and the efficiency comes from its particular binary splitting method: instead of dividing a given region into two along coordinate directions at the median, they divide along a random direction. In [Freund 2007], the authors consider two different types of splits at each node (or cell) of the tree: the first type is chosen such that it maximally decreases average squared interpoint distance in the cell and the second one is based on the distance from the mean of the cell and mainly used when the data in the cell is present at very different scales. Using these splits, the expected average diameter of cells is shown to be halved at every $O(d_I)$ levels, where d_I is the intrinsic dimensionality that is much lower than the high-dimensional extrinsic dimensionality.

In our discriminative model, each d-dimensional keypoint descriptor $\mathbf{c}$ is required to be classified. The label for each keypoint is denoted as y. That is, $\mathbf{c}$ is classified as belonging to face ($y = +1$) or background ($y = -1$). We propose our binary classifier based on random projections. This binary classification task is accomplished through an ensemble (a forest) of N classification trees. Since the trees are designed for classification not vector quantization, different split strategy is adopted from original random projection trees. As for classification, we intend to best separate data of two classes at each node. Our algorithm makes use of an entropy based score measure written as a particular normalization of the information gain:

$$S_c(L,T) = \frac{2\, I_{c,T}(L)}{H_c(L) + H_T(L)} \tag{5.11}$$

For the split T, $H_c(L)$ denotes the entropy of the classification in the leaf L, $H_T(L)$ denotes the split entropy and $I_{c,T}(L)$ denotes the mutual information of the split and the classification. Our classification trees split the training data at each node by a simple linear function given as:

$$\textit{if } \mathbf{b}^T\mathbf{c} + t \leq 0, \textit{split right; otherwise, split left} \tag{5.12}$$

where $\mathbf{b}$ is a d-dimensional random projection vector and t is a random scalar threshold.

To perform the classification task, the node parameters $\mathbf{b}$ and t should be learned beforehand. That is, training the trees is to select best $\mathbf{b}$ and t for each node. Here $\mathbf{b}$ is chosen from a preselected small dictionary of d-dimensional projection directions. The trees are constructed given the whole training set and projection direction dictionary. In [Bosch 2007], authors propose a 'feature cue selective' classifier, for example to be selective for shape or appearance of the objects to classify. They select specific cues by filling associated components of $\mathbf{b}$ with 0 or 1. We select the best parameters $\mathbf{b}$ and t in (5.12), by maximizing the entropy $S_c(L,T)$ like in [Moosmann 2008]. The recursive tree building algorithm stops when the node receive too few data or when its depth reaches a maximum. Once the forest of N trees is trained, all the leaves encode conditional probabilities for each class. We define $P^i_j(\mathbf{c}|y = +1)$ and $P^i_j(\mathbf{c}|y = -1)$ as the conditional probabilities of face and background respectively at j^{th} leaf of i^{th} tree. Hence, $P^i_j(\mathbf{c}|y = +1) = \frac{N^+_{ij}}{N^+}$ and $P^i_j(\mathbf{c}|y = -1) = \frac{N^-_{ij}}{N^-}$. N^+_{ij}, N^-_{ij} are the number of training samples of face and background at this leaf. While N^+, N^- denote the whole number of face and background training data.

At classification stage, an image patch represented as a vector $\mathbf{c}$ is dropped down the forest and reaches N leaves. Thus, the final conditional probability of $\mathbf{c}$ is the average of all the conditional probabilities of its reaching leaves:

$$P(\mathbf{c}|y) = \frac{1}{N}\sum_{i=1}^{N} P_{l_j}^{i}(\mathbf{c}|y) \tag{5.13}$$

where $P_{l_j}^{i}(\mathbf{c}|y)$ is the conditional probability at the reaching leaf l_j of i tree for $\mathbf{c}$. According to Bayes equation $P(y|\mathbf{c}) = \frac{P(\mathbf{c}|y)P(y)}{P(\mathbf{x})}$, we can obtain

$$P(y|\mathbf{c}) \approx P(\mathbf{c}|y) \tag{5.14}$$

by assuming equal prior.

5.2.3 Defining the Color Prior

Fig. 5.7 illustrates the definition of the color prior. It is given by comparing the two class posteriors $P(y = +1|\mathbf{c})$ and $P(y = -1|\mathbf{c})$, that is,

$$f(\mathbf{c}) = \begin{cases} 1 & P(y = +1|\mathbf{c}) > \alpha P(y = -1|\mathbf{c}) \\ 0 & otherwise \end{cases} \tag{5.15}$$

Here the prior $f(\mathbf{c})$ is simplified to be a 0-1 function. With its binary decision, we can declare a keypoint $\mathbf{k}$ as an outlier just before matching its graylevel feature, which saves plenty of time in practice. Note that we set $\alpha = 0.9$ that lowers the true negative rate as compared to $\alpha = 1.0$.

5.2.4 Evaluations

The experiments aims at showcasing the strength of the proposed way to incorporate color information. They were conducted with respect to two aspects. The first one validates the discriminative superiority of the RGB Sigma Set in comparison to various invariant color histograms. The second shows the significant reduction of outliers with the color prior integrated.

5.2.4.1 Color Histograms

The comparison with histogram is natural in view of their widespread use in modeling color distribution. In particular, we build four different color histograms - the *RGB* color histogram, *opponent* color histogram, *normalized* RGB color histogram and the *transformed* color histogram [van de Sande 2010], in order to show the strength of RGB Sigma Set. The derived color spaces are given in terms of the

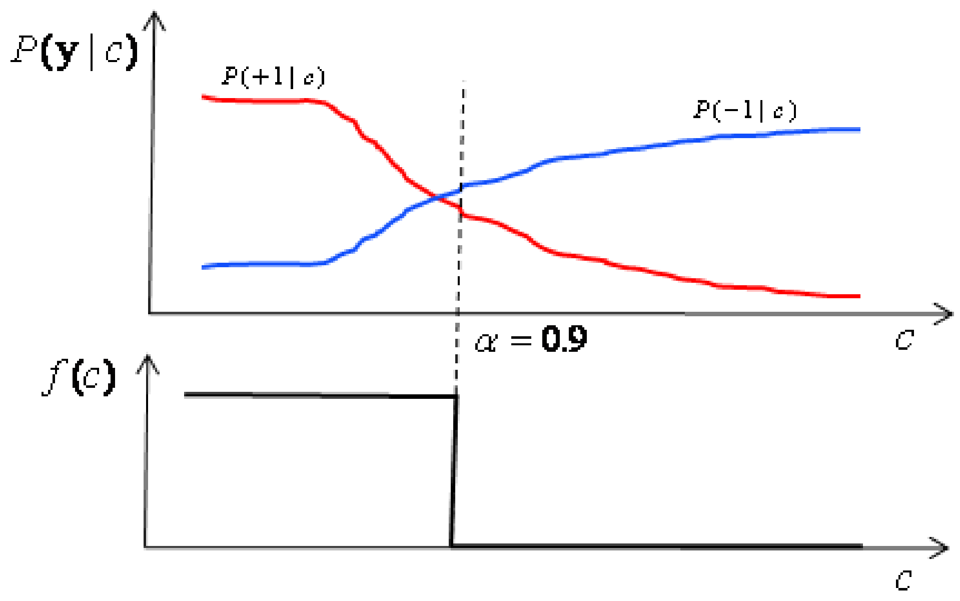


Figure 5.7: Defining the binary color prior $f(\mathbf{c})$. It outputs 1 when we have $P(y = +1|\mathbf{c}) > \alpha P(y = -1|\mathbf{c})$ and 0 otherwise. α is set to 0.9 to have a lower true negative rate.

original RGB space:

$$\begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}, \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix}, \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix} \tag{5.16}$$

The histogram bins are independently estimated in each color channel and placed together to form our color histogram. Each channel intensity is quantized into 15 bins with the interval of 17. By definition, the color histogram captures no spatial information but the statistics of the first-order color information.

5.2.4.2 Discriminability Comparison

The 'ssm1' sequence from Boston University [Cascia 1999] was used for this comparison. The training RGB Sigma Set descriptors as well as the four color histogram descriptors are collected from the first frame of this video. All these descriptors are then classified using the same classifier that compromises 20 RP trees with 200 preselected RP directions and 5-level depth. On testing each query feature, its belonging is determined according to Equation (??). The ground truth are automatically annotated with a manually selected rectangle that bounds face. The measure is the

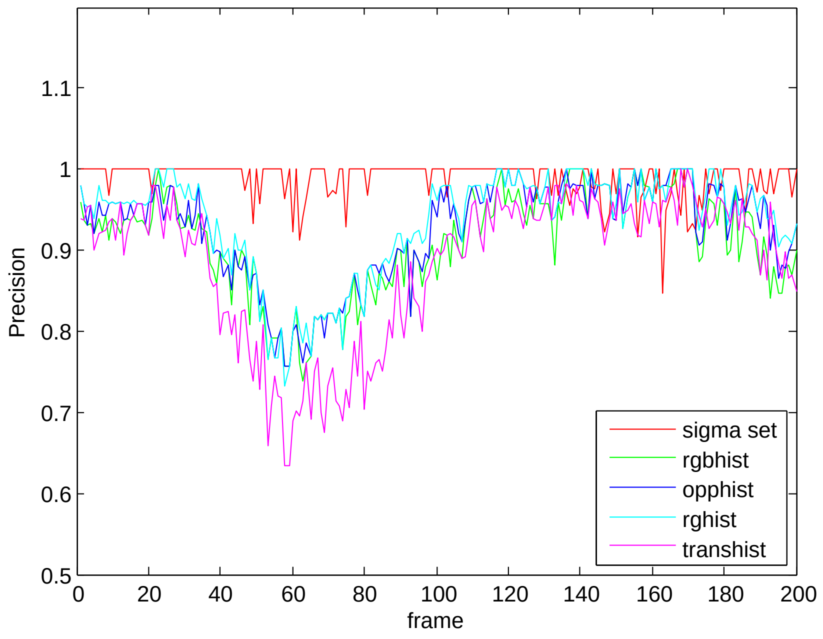


Figure 5.8: Comparing the classification precisions of five different descriptors.

RGB Sigma Set	RGB hist	Opponent hist	RG hist	Trans hist	RGB Sift
21	45	45	30	45	384
0.9882	0.9117	0.9252	0.9347	0.8843	0.6280

Table 5.2: Mean inlier recognition precision. The second row shows the length of each descriptor. The third row shows the precision.

standard classification precision defined as follows:

$$Precision = \frac{TP}{TP + TN} \tag{5.17}$$

where TP and TN stand for true positive and true negative, respectively. The precisions over the sequence are shown in Figure 5.8. The mean precisions as well as the length of each descriptor are shown in Table 5.2. As shown, the RGB Sigma set outperforms all the color histograms with a lower dimensionality (21). Therefore, covariance matrix is a compacter yet more discriminative way to represent features than the common histograms.

On the other hand, one may wonder why not to fuse color and spatial features. With the available ColorSift software [van de Sande 2010], we test the RGB Sift's performance in this binary classification task. The mean precision over the 'ssm1' sequence is also given in Table 5.2.4.2. Surprisingly, the precision (0.628) is much worse than any of the color-based descriptors although it dimensionality is extremely high (384). It reflects that color space is the dominant subspace that distinguishes face from background. This fused descriptor may be more suitable for a multi-class classification task. In future, we will test its performances to replace the SURF descriptor for 3D head tracking.

5.2.4.3 Outlier Reductions

This subsection assesses the roles of the proposed color prior in 3D head pose tracking. Only Sigma Set descriptor is used due to its better performance as verified in the last experiment. Both quantitative and qualitative experiments were conducted.

The quantitative experiment was conducted with the same 'ssm1' sequence. The measure is defined as the ratio of inliers to all the matched keypoints during pose tracking. This is reasonable since it is critical to the final PROSAC (and its original - RANSAC) selector. Fig. 5.10 depicts the ratio improvement from about 0.2 to about 0.3 after the color prior is imposed. As seen in the snapshots, the gain comes from the fact that many SURF model features are attracted to the cluttered background that many books and a bookshelf constitute. Consequently, this improvement saves the number of PROSAC iterations such that pose tracking is speeded up by about two times.

The qualitative experiments were carried out with live-captured video sequences that contain challenging situations. It is measured by observing how pose tracking is improved. Fig. 5.11 shows the comparative results with and without color prior. As can be seen, without color prior, a large percentage of matched features come from the cluttered background. This leads to wrong pose estimation or even tracking failures. The typical case is the moment when lighting condition suddenly switches as shown in rows 3 and 4 of Fig. 5.11. As it is becoming darker or brighter, without-color-prior tracker loses the target face with no mesh displayed in Fig. 5.11 while with-color-prior tracker still does the correct estimation. The results show that with the color priors, larger portion of inliers is reached and the tracker is more robust to large head movements and sudden changes of lighting conditions.

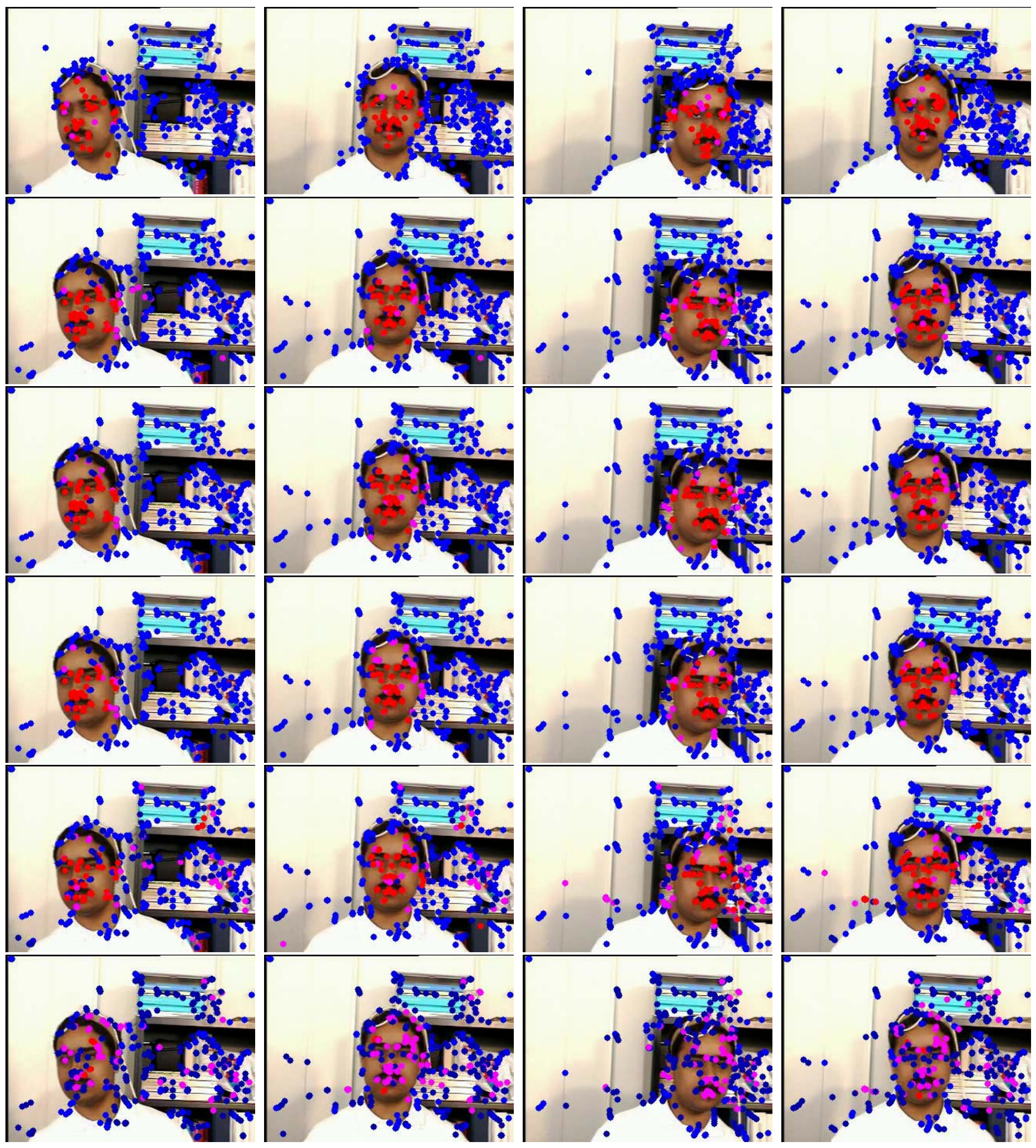
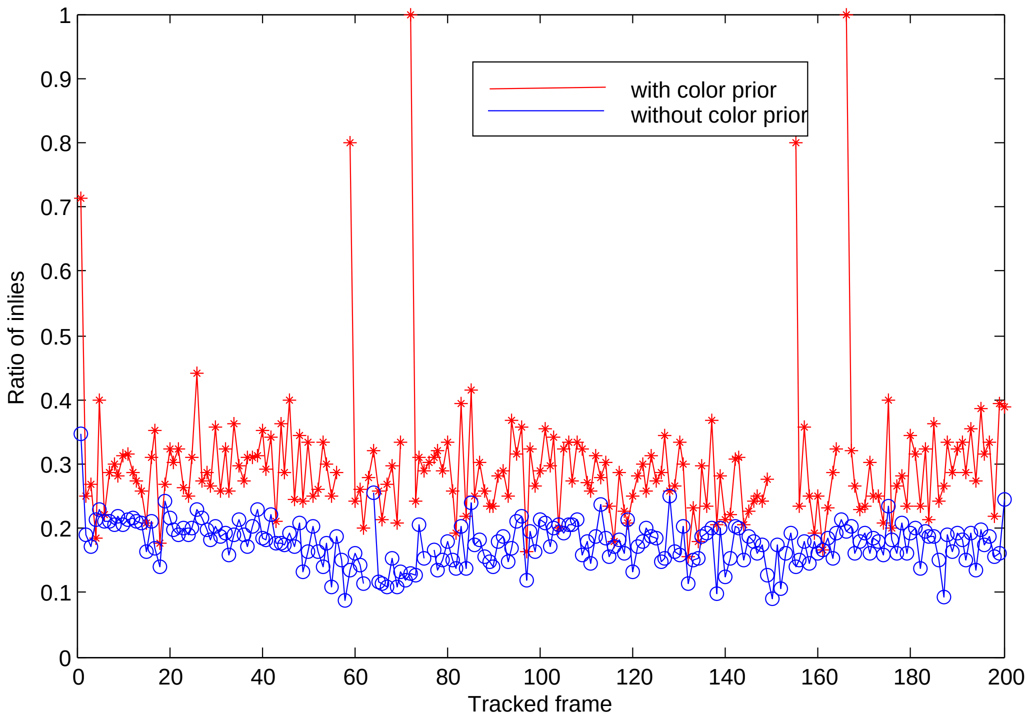


Figure 5.9: From top to bottom each row corresponds to the interest point classification of RGB Sigma Set, RGB histogram, Opponent histogram, RG histogram, Transformed color histogram and RGB sift. The color mapping for the interest points is defined as: $P(\hat{\mathbf{c}}|\omega_1)/P(\hat{\mathbf{c}}|\omega_2) > 3 : red, 3 \geq P(\hat{\mathbf{c}}|\omega_1)/P(\hat{\mathbf{c}}|\omega_2) \geq 1 : magenta, P(\hat{\mathbf{c}}|\omega_1)/P(\hat{\mathbf{c}}|\omega_2) < 1 : blue.$



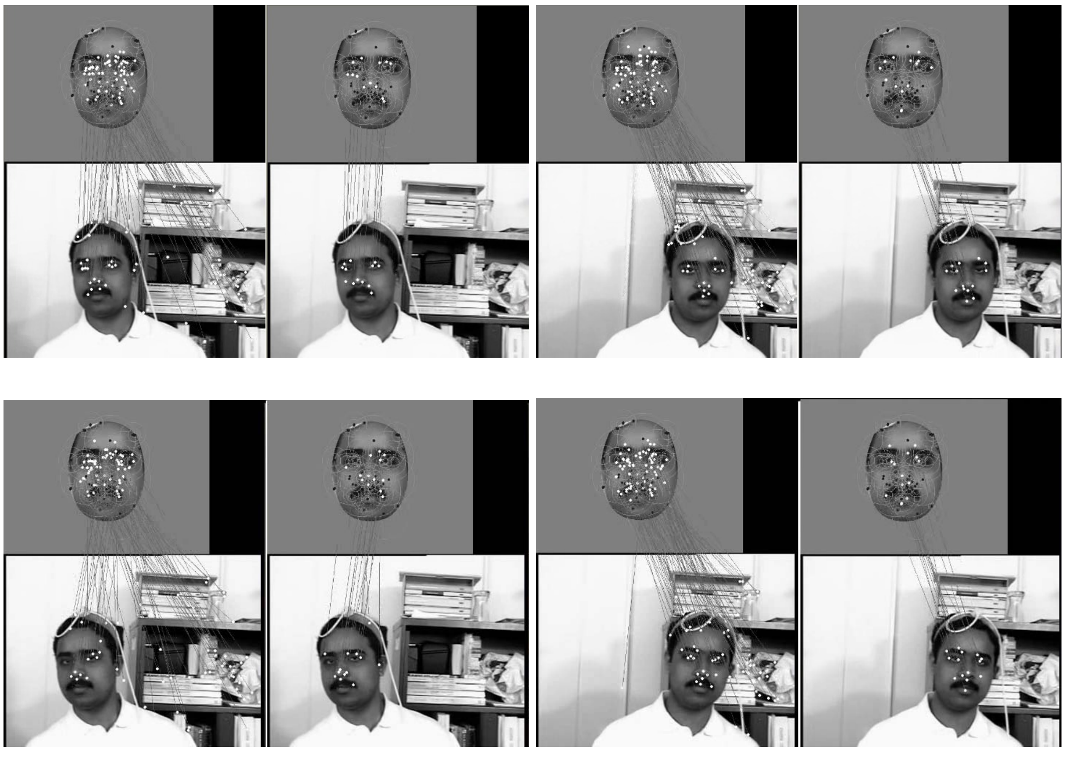


Figure 5.10: The plot compares the ratios of inliers with and without color prior. The image pairs are the matching examples without and with color prior.

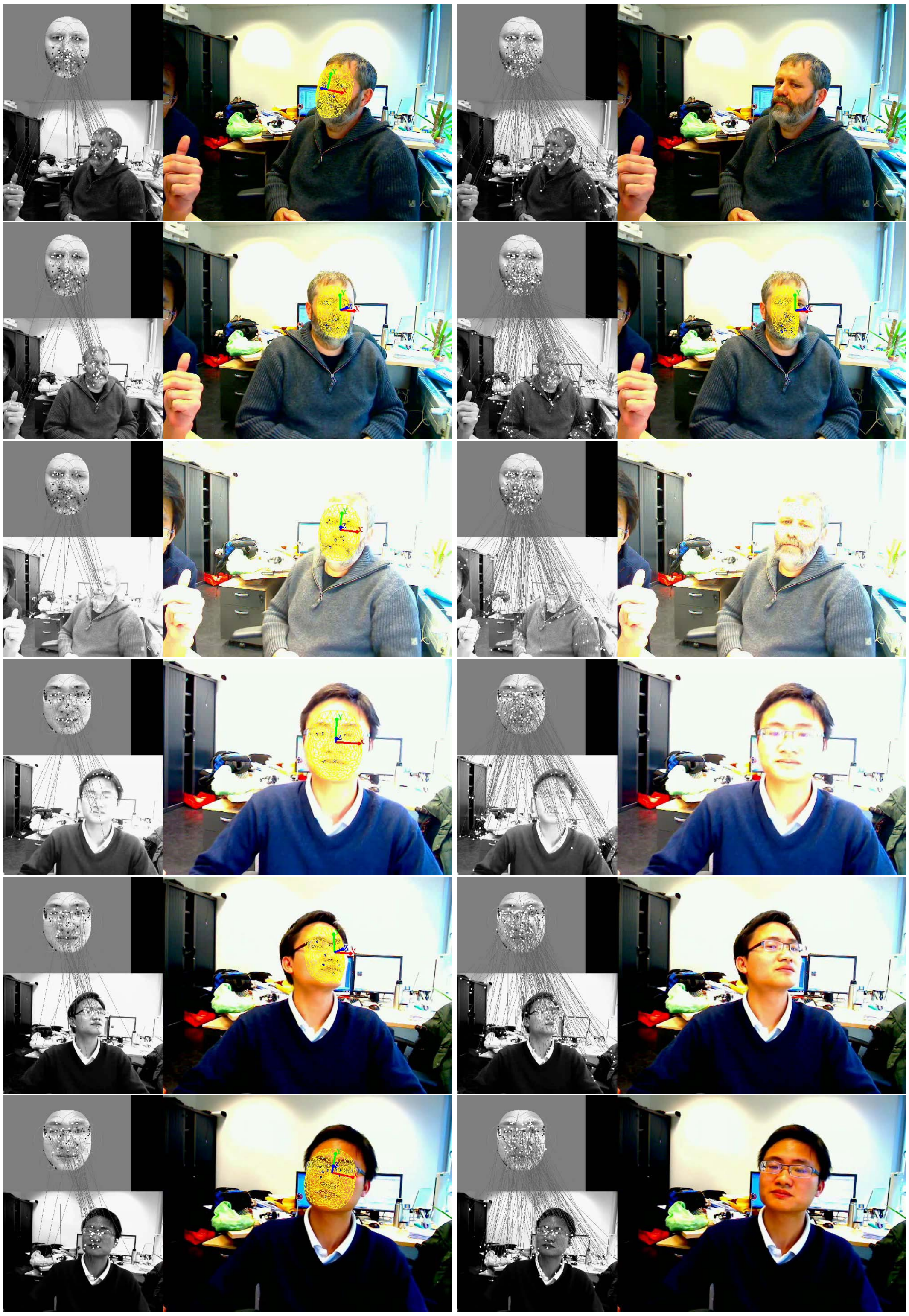


Figure 5.11: Selection of challenging frames to show the advantage of using color prior for tracking including strong illumination changes and pose variations. The left image of each row shows the tracking result with color prior while the right one without color prior.

Chapter 6
Towards Applications

The motivation that drives our 3D head tracking research is a project whose goal is to develop a cheap yet efficient way to represent remote human beings who are collaborating in an online shared virtual environment. We call it a *mixing video and avatar* human representation. At its heart is the vision-based techniques to capture and reproduce three non-verbal functionalities - facial expressions, gaze and gesture expressions. Among them, 3D head pose tracking are crucial for the last two modules. In this chapter, we will present this *mixing video and avatar* idea with emphasis on the two head-pose-involved modules. For facial expression reproducing, we adopt a simple strategy, that is, extracting pose-normalized face textures from videos, transferring and directly mapping them on an avatar model. Then in terms of gaze channel, we specialize in the problem of simultaneous gaze and head pose estimations. We employ a simplified linear head-eye coordination model and measure gaze vector with only light calibrations.

6.1 Mixing Video and Avatar

Videoconferencing is a set of interactive telecommunication technologies which allow two or more locations to interact via two-way video and audio transmissions simultaneously [Joerg 2005]. It captures the 2D video mirrors of a local user and transmits the videos sequentially to his/her partners who are at physically remote places such that their conversations are like face-to-face. With the incrementing focus on interacting with 3D target objects, videoconferencing becomes insufficient considering its two-dimensional nature. In this context, many new tele-collaboration concepts have emerged, e.g. Collaborative Virtual Environment (CVE) or Computer-Supported Collaborative Workspace (CSCW) [Takemura 2002]. Although video remains a major media, the way to use it is changed so that the information lost in videos can be extracted and recovered as more as possible. In the following, we give a brief study of the emerging collaboration solutions.

Immersive Reality (IR), or shared reality or augmented reality, merges visual human embodiment into a physically realistic environment. In this solution, user is usually embodied by a video-based agent, in the form of either 2D images or extended 2.5D stereo streams [Leung 2000, Lee 2006, Regenbrecht 2003, Kauff 2002]. In principle, IR is able to generate impressive collaboration environment. The main obstacle is its dependencies on highly expensive and intrusive equipments, which makes it hardly usable in real applications. Virtual Reality (VR), works in a completely different way. It builds a shared virtual collaborative platform where avatars that represent remote partners are embodied. The avatar is usually a virtual agent that resembles the semantic behaviors of his human counterpart [Bailenson 2004]. To be realistic, avatar can be built via the intrusive and expensive equipments as in IR. But the pure video-based techniques are more economic. Hauber *et al.* [Joerg 2005] created a simple 3D 'TV' that is mapped by face videos and rotated it to indicate the possible gaze directions of users.

In this chapter, we propose a cheap yet efficient approach to represent remote human in IR or VR. It is featured in that video and avatar are mixed together to yield a functional and vivid human agent. In particular, our 3D head tracker is applied in two aspects. First, facial expression is directly transferred to a built avatar by extracting pose-normalized face from the tracked region. Second, head tracker helps to determine eye gaze in the way mentioned in the last section. In our approach, facial expression and gaze information (lost in video) are successfully recovered and embodied in avatar. In addition, a prototype system is also developed to test the functionalities of the proposed human agent in the task of distant collaboration.

6.1.1 Related Work

The earlier study of virtual humans representation went back to embodied agents that are digital models driven by computer algorithms [Cassell 2000a, Pelachaud 2002]. For example, the Embodied Conversational Agent (ECA) [Cassell 2000b] was built to have the conversational ability. Later, Helmut [Prendingery 2001, Helmut PRENDINGERY 2003] built his animated agents with the capability of affective communications. The basic elements of embodiments that determine the technical designs of these agents, such as degree of presence, identity and viewpoint, were summarized in [Benford 1995]. More recently, with the popularity of virtual reality, the concept avatar started to emerge thanks to the advancements of visualization techniques [Schroeder 2002]. Avatars can be defined as digital models of people that either look or behave like the users that they represent. A good overview of avatar was studied by Bailenson and Blascovich [Bailenson 2004], in which the determinant factors of an avatar is to be behavioral realism and photographic realism.

How to define the modalities of virtual human in the presence of mediated communications, e.g. CVE, CSCW, has attracted a great deal of interests from the communities of both computer and sociological sciences. Their principles are to augment the functional modalities and decrement the nonfunctional ones in remote communications. Since the 1990s, Steve Whittaker *et al.* [Whittaker 1991] had concentrated on the theoretical analysis of different modalities and the design and implementation of virtual humans in collaborative systems and mediated communications. His experiments revealed that linguistic cue conveys the richest information of communications while the visual cues, such as facial expressions and gestures, can substantially enhance the efficiency of verbal communications. According to this theory, our system considers expressing facial emotions and replicating semantic gestures. More specifically, we focus on the implementations of the two non-verbal modalities: facial expressions and semantic gestures.

There have been many previous attempts that tried to mix video and avatar by combining the merits of both. One commercial example is the Workspace3D [1] that simultaneously places a separate video window containing face and a virtual avatar in an identical interface. The similar idea was proposed by P. Quax *et al.* [Quax 2003] that simply replaced the avatar head with the video window. These easy ideas may seem weird as they do not bridge the gap between two-dimensional video and three-dimensional avatar. Additional techniques must be considered to mix video and avatar naturally. Schreer *et al.* [Schreer 2005] applied video processing to steer the gestures of a toy avatar. By comparison, we consider more modalities and implement the non-verbal ones by more intelligent tracking algorithms.

6.1.2 Mixing Video and Avatar

The MVA framework requires only two ordinary input devices, a webcam and an audio recorder, for capturing video and audio sources respectively. On a local computer, these two sources are processed by intelligent techniques to generate data to broadcast to remote terminal PC in the same network. All the data are packaged and encoded before emitting and unpackaged and decoded on receiving. The decoded data, including textures, postures and audio streams, are then used to drive corresponding parts of an avatar agent. In figure 6.1, the architecture summarizes the entire procedures. In more detail, each module and their major components are implemented as follows:

- Face expression transfer. We apply the 3D head tracking technique to track 3D poses from monocular video sequences and extract neutral-view face texture by image warping. The texture containing facial emotions as well as the tracked orientations $\{\alpha_h, \beta_h, \gamma_h\}$ are transmitted for avatar animation. Specifically, textures are compressed into JPEG format using an open DevIL library.

- Eye gaze estimation. The gaze direction of each eye $\{\alpha, \beta, \gamma\}$ is tracked using a combined iris-corner detection method (introduced in the next section). At the same time, the proposed 3D head tracking technique is fused to handle coordinated head rotations.

- Semantic gesture tracking. During the training stage, we learn six semantic behaviors from 20 testers and train a generative model by extracting features in the Curvature Scale Space (CSS) of body contour. At run-time, we match the detected CSS features with the trained model and declare its semantic category with highest matching score. For each frame, the three positions of all six joints are emitted to drive avatar gesture.

- Audio capture and transmission. Audio stream $\mathbf{A}$ is captured using the functions in the OpenAL library and to avoid network blocks, it is compressed using the Speex audio compressor.

[1] http://www.3d−test.com/interviews/tixeo$_1$.html

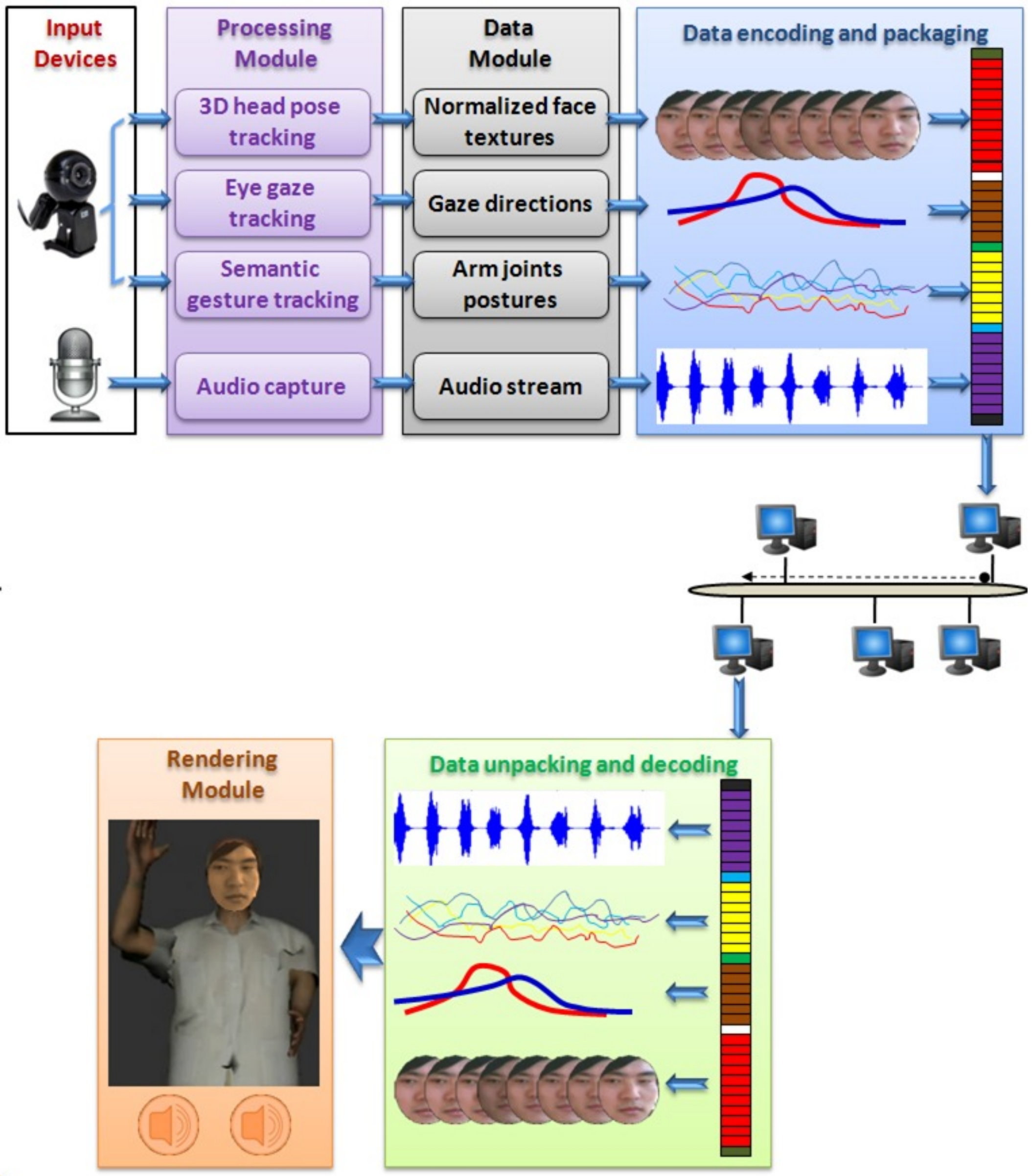


Figure 6.1: The framework of our remote human representation method that mixes video and avatar.

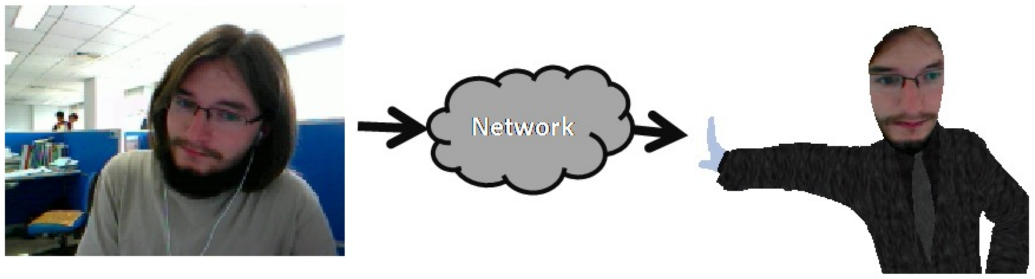


Figure 6.2: Illustrating the direct face transfer.

- Network strategy. The multicast strategy is employed to transmit data package. As compared to unicast and broadcast, multicast, in the communication-first mode, is a nice trade-off to avoid data blocks.

- Rendering module. On each remote PC, we build an avatar agent based on the H-anim standard. The received data stream needs first to be separate $\{\mathbf{I}_f, \{\alpha_h, \beta_h, \gamma_h\}, \{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{n=2}, \{x_i, y_i, z_i\}_{i=1}^{n=6}, \mathbf{A}\}$, where $\mathbf{I}_f$ is the neutral face texture, $\{\alpha_h, \beta_h, \gamma_h\}$ are the head orientations, $\{\alpha_i, \beta_i, \gamma_i\}_{i=1}^{n=2}$ are the gaze directions of two eyes, $\{x_i, y_i, z_i\}_{i=1}^{n=6}$ represents semantic gestures and $\mathbf{A}$ is the audio data block. Then each data term is used to drive its corresponding part of the avatar.

6.1.3 Primary Results

Testing the main components of the platform is still under construction. Here we only show the exemplar results of face transfer. Figure 6.2 illustrates the face transfer that extracts face texture from a live-captured frame and maps on an remote avatar agent after network transmission. Figure 6.3 and Figure 6.4 show the results with two different users. After head tracking and network transmission, the face textures are simply rendered at both the original and neutral viewpoints. All these results show that direct face transfer can reproduce facial emotions as compared to data-driven face animation.

To be as realistic as the original facial expression was, there are still many work to do. When re-mapping face texture at an alternative viewpoint, direct gaze contact is lost and gaze correction would be an indispensable rectification step as done in [Yang 2002c]. The skin color of the extracted face texture should be re-painted so as to be consistent with its virtual surrounding environment.

6.2 Natural Eye Gaze Estimation

Eye gaze estimation can applied to the design of next-generation computer-human interfaces, e.g. using it to replace mouse. Since the last decade, an extensive

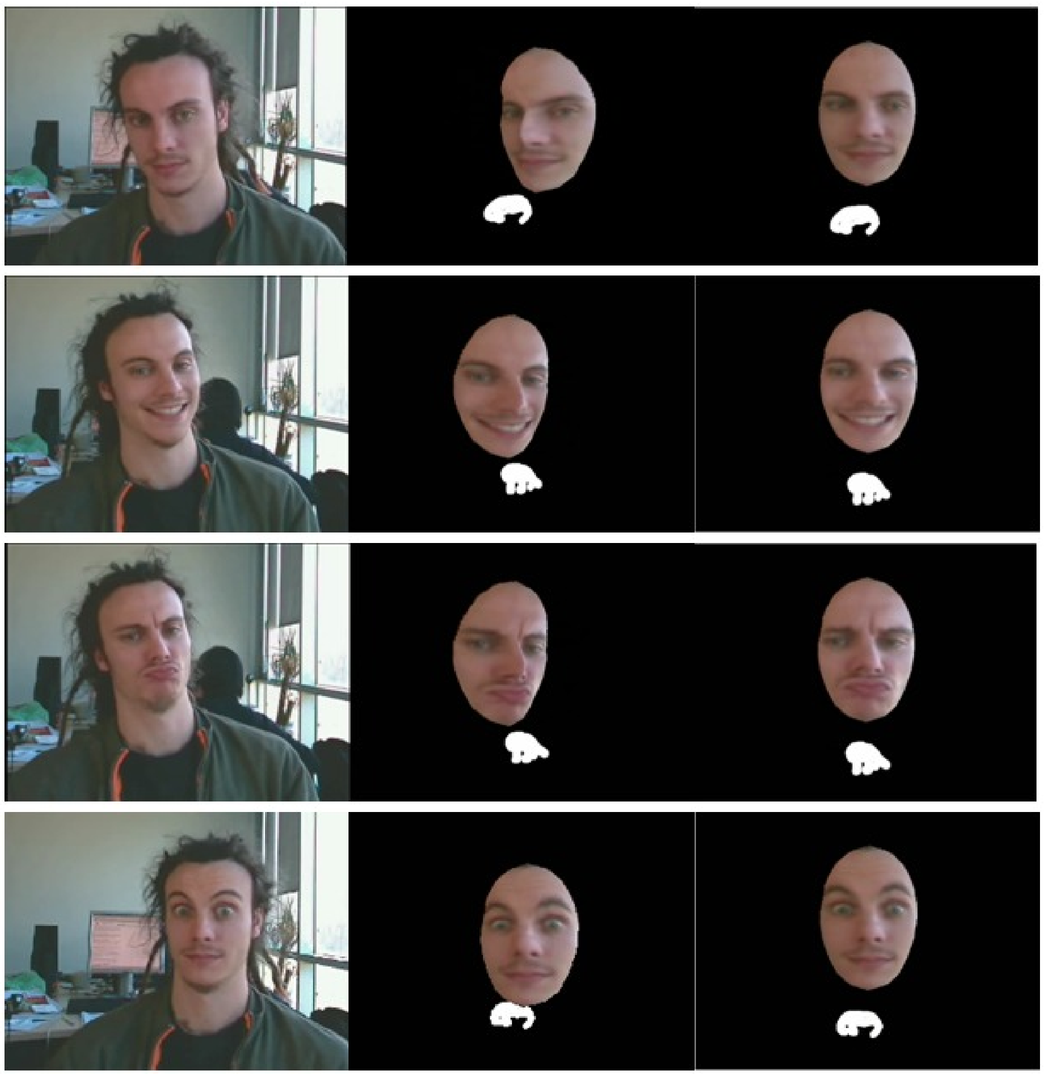


Figure 6.3: Face transfer examples. From left to right each column corresponds to live-captured frames, rendered face at original viewpoint and rendered face at neutral viewpoint.

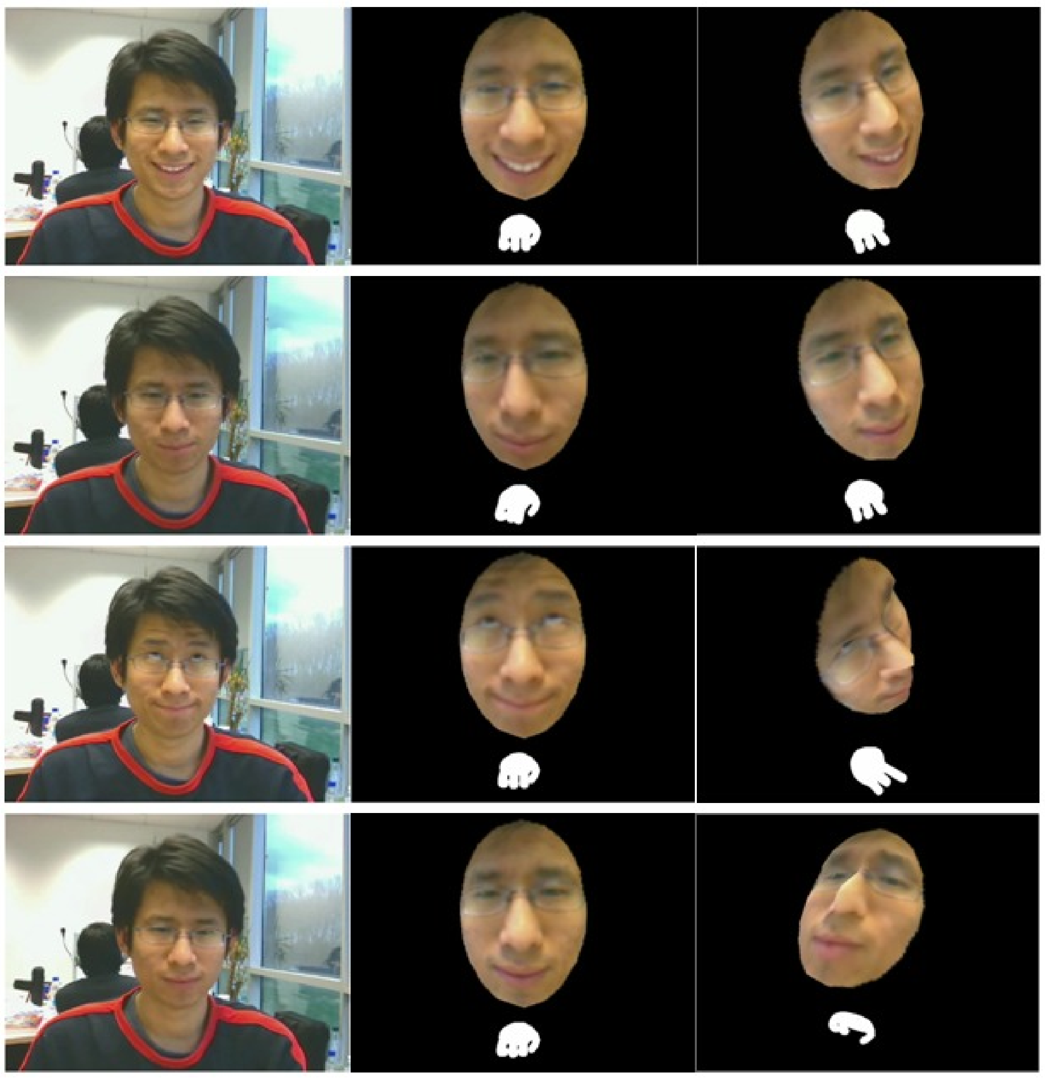


Figure 6.4: Another face transfer examples. From left to right each column corresponds to live-captured frames, rendered face at original viewpoint and rendered face at neutral viewpoint.

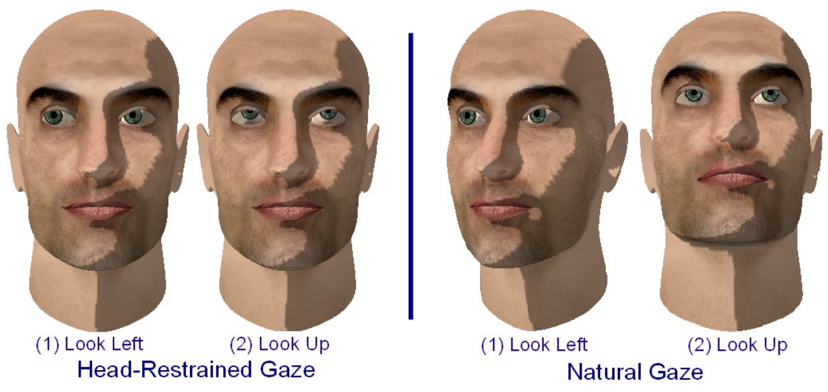


Figure 6.5: Illustrating the difference of head-restrained gaze and natural gaze. When head motion is restrained, only eye pose that moves within our orbite domain accounts for the gaze shifts. Conversely, when head is free to move, namely natural gaze, eye motion is accompanied by a concurrent head rotation in the same direction.

number of non-intrusive eye gaze systems have been proposed [Heinzmann 1998, Baluja 1993, Guestrin 2003, Beymer 2003, Wang 2003]. Although good performances have been achieved [Takahiro Ishikawa 2004], these earlier systems are limited in keeping head still while estimating eye gaze.

Recent efforts have attempted to incorporate head motion into eye gaze estimation. These attempts can be broadly clustered to be 3D gaze direction-based and 2D mapping-based [Zhu 2005]. The former method determines gaze point by recovering 3D gaze direction and simply intersecting with the scene. It allows free head motion, but can only estimate gaze direction rather than giving gaze targets [Yamazoe 2008, Kinoshita 2006]. On the contrary, the latter method encodes a set of eye-related vectors as inputs for a calibrated mapping function to determine gaze targets. This method is widely used due to its high tracking accuracy. One typical example is the pupil center and corneal reflections technique, a.k.a. PCCR [Guestrin 2003, Yoo 2002]. However, the classical PCCR suffers from a calibration-decay problem caused by head motion [Zhu 2005]. In order to overcome it, multiple stereo cameras or IR lights must be added in the PCCR systems [Guestrin 2003, Zhu 2002]. In this way, they are able to track eye gaze under fully natural head motion(6 DOF). The main drawbacks are that the camera sets need to be carefully calibrated and IR lights are expensive and usually inconvenient to be mounted.

Our objective is to track eye gaze under natural head motion with only an ordinary camera. However, Guestrin *et al.* [Guestrin 2003] proved that using a single camera was impossible to track eye gaze under natural head motion. On the other hand, neurophysiological studies [Hanes 2006, Freedman 2000, Freedman 2001] manifest that the free head motions along gaze shift are usually the rotations in the same directions. They specify this phenomenon as eye-head coordination of gaze shift. We therefore turn to explore the possibility of tracking gaze with one camera, while only considering the coordinated head rotations. This is proven feasible by developing a new tracking approach in this paper. Within this approach, the price we have to pay is two-time calibrations, although acceptable in practice. In particular, our approach follows the 2D mapping-based scheme [Zhu 2005] to directly output the targeted points. In order to compensate the affect of head rotations, we first derive a linear relationship between gaze vector and head rotations, and then develop a monocular vision-based framework to track gaze, which needs to detect iris-corner vector, track head rotations and warp facial images to extract a *head-normalized* iris-corner vector.

6.2.1 A Case Study

Naturally eyeball movements are always coupled with head orientation variations to gaze at a target. For empirical support, we conducted a study to investigate the frequencies of eye-head coordination in gaze behaviors. Based on the theory proposed by E.G. Freedman *et al.* [Freedman 2000, Freedman 2001] that the total head movement amplitude and gaze amplitude are linearly correlated in a certain range, we examine such linearity as the quantitative evidence of the occurrence of eye-head coordination.

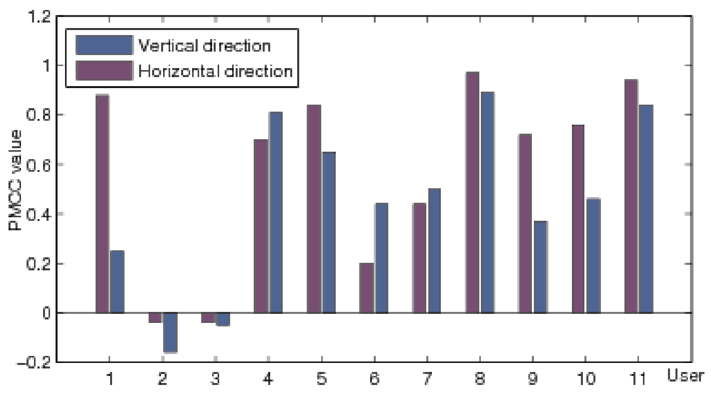


Figure 6.6: The PMCC values between head and gaze amplitude among 11 participants.

The study was conducted with 11 participants from our laboratory. Each participant is asked to stare at a visual sphere moving arbitrarily on a 19-in screen. Meanwhile, the trajectory of the sphere is recorded as gaze amplitude and the participant's head amplitude is tracked using the developed head pose tracker (See Section 4). The correlations between head rotation and gaze amplitude is measured in terms of the Pearson Product-Moment Correlation Coefficient (PMCC), which is shown in Figure 6.6. We can see that the PMMC is significantly different as different person. If PMMC>= 0.6 indicates a possible linear relationship, we find that there are seven possible linearities in both directions. We also assess the influence of the distance between participant and screen on eye-head coordination. It turns out that the farther participant stays away from the screen, the less likely head rotations are to occur. Therefore, we can conclude that the occurrence of eye-head coordination is very likely to occur in gaze behaviors, although it is circumstance-dependent and varied among different persons. This study validates our proposal that it is indispensable to incorporate head rotations in eye gaze tracking.

6.2.2 Problem Formulation

Let us consider tracking eye gaze in front of a monitor screen. The gaze directions are measured with respect to the horizontal and vertical ones of the monitor plane. We adopt some notations from Zhu and Ji [Zhu 2005] to describe our problem. First of all, via an interactive calibration, we are able to obtain a set of head-stationary gaze vectors $\{\mathbf{v'_1}, \mathbf{v'_2}, ..., \mathbf{v'_n}\}$ and its associated set of fixation positions$\{S_1, S_2, ..., S_n\}$ on the screen. The data are then used to compute the coefficients in a mapping function $\digamma$ such that given any head-stationary gaze vector $\mathbf{v}'$, its fixed position S can be yielded: $S = \digamma(\mathbf{v}')$. $\digamma$ is often a first or second order polynomial function. Then suppose at time t of tracking stage, we succeed in tracking head motion $\mathbf{b_h} = \{\beta_x, \beta_y\}$(only pitch and yaw rotations) and gaze vector $\mathbf{v_t}$. Since head motion $\mathbf{b_h}$ results in a change of the eye pose, the gaze vector $\mathbf{v_t}$ is no longer a correct input of the mapping function $\digamma$. In order to reuse $\digamma$, we need to compensate $\mathbf{v_t}$ for the head-caused bias to form an equivalently *head-compensated* gaze vector $\mathbf{v'_t}$: $\mathbf{v'_t} = \daleth(\mathbf{v_t}, \mathbf{b_h})$, where $\daleth$ stands for the *head compensation function.* This also means that $\mathbf{v'_t}$ are supposed to look at the same point S_t as $\mathbf{v_t}$ does, which yields:

$$S_t = \digamma(\mathbf{v'_t}) = \digamma(\daleth(\mathbf{v_t}, \mathbf{b_h})) \tag{6.1}$$

The key of our approach is thus to derive an appropriate $\daleth$ function so that the head rotation bias on iris-corner vector can be compensated. This can be done by projecting the iris-corner vector on image coordinate via a pinhole camera model.

6.2.3 A Head-Eye Coordination Model

To derive $\daleth$, we have to analyze the kinematics of eye-head coordination using the pictorial notations in Figure 6.7. In the figure, eyeball is modeled as a sphere with a gray iris inside, and eye direction is denoted as the ray shooting from eyeball center to the target point. E_1 represents the eyeball center position when head stays frontal. Initially, in order to look at the target point S, eyeball rotates by a α' angle. When head shifts by a β angle, moving the eyeball center from E_1 to a new position E_2, the eyeball has to re-rotate to a new angle α to still look at S. The iris center is defined as the intersection point of gaze ray and iris outer surface, which is denoted by P_1 and P_2 at the two different locations.

Suppose that the head rotates around its axis with the radius $R = OE_1 = OE_2$. From the geometrical relationships in $\triangle E_2PS$, $\triangle E_1O'S$ and $\triangle OE'E_2$, we can get

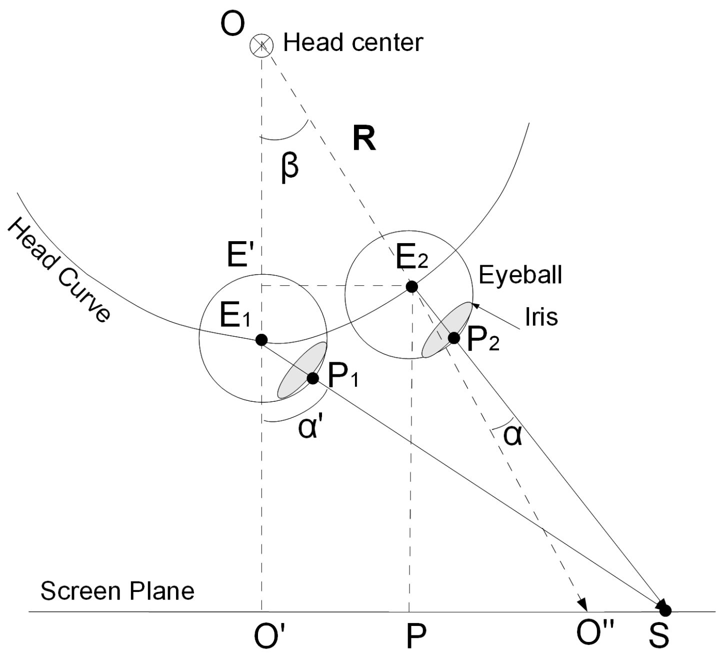


Figure 6.7: Eye-Head Coordination Model. The model is depicted in top-view.

$PS = E_2P\tan(\alpha+\beta)$, $O'S = E_1O'\tan\alpha'$ and $O'P = E'E_2 = R\sin\beta$. Then insert them into the equation $O'S = O'P + PS$ and we have:

$$E_1O'\tan\alpha' = E_2P\tan(\alpha+\beta) + R\sin\beta \tag{6.2}$$

In practice, the difference between E_1O' and E_2P is negligible. We thus replace them with $h = E_1O' = E_2P$ in equation (6.2):

$$\tan\alpha' - \tan(\alpha+\beta) = \frac{R}{h}\sin\beta \tag{6.3}$$

Since the radius R is usually much smaller than the distance term h, we can assume $R \ll h$ and plug it into (6.3) to get a simpler equation:

$$\alpha' \approx \alpha + \beta. \tag{6.4}$$

It can be seen that the total gaze amplitude is approximately the sum of eye and head amplitudes, which has also been empirically verified in neuroscience [Freedman 2000, Freedman 2001].

6.2.4 Coupling Head and Eye Pose Vectors

The next step is to explore the relationship between iris-corner and the concurrent head rotations, based on the derived sum principle. We assume that the iris-corner

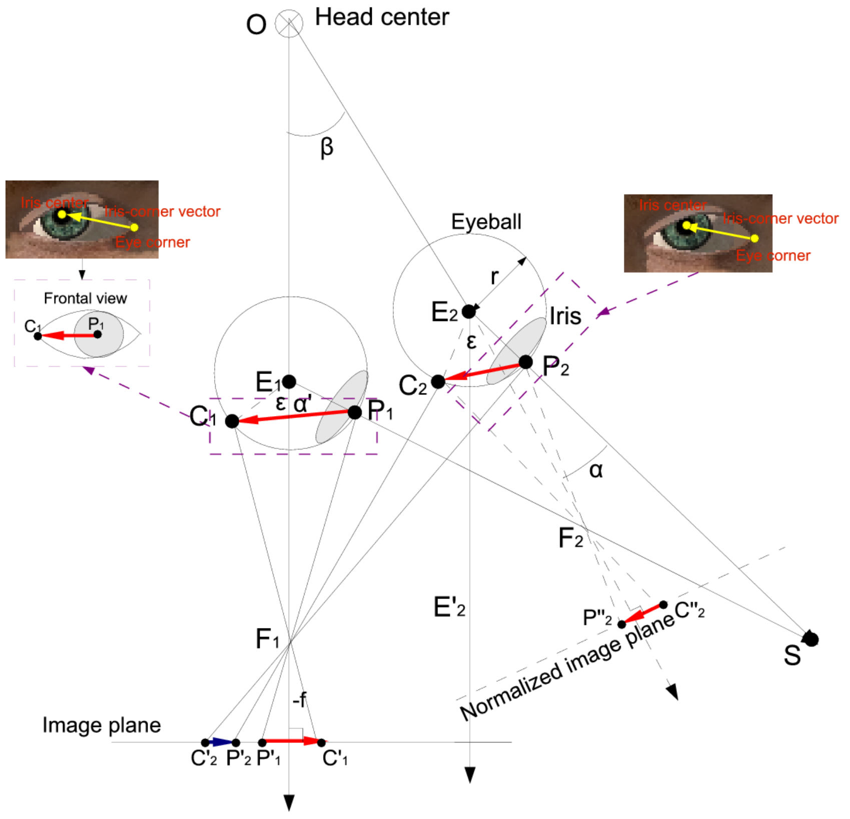


Figure 6.8: Illustration of projecting eyes onto visual image plane via the pinhole camera model(Top view).

vector is generated via a pinhole camera projection. As shown in Figure 6.8, we denote C_1 as an eye corner point, located in a face plane that is close to the eyeball surface. We assume that the face plane still remains stationary when eyeball rotates so that the corner point keeps a gaze-stationary point. As shown, when the head rotates by β angle, the corner point moves from C_1 to a new position C_2. Based on the geometrical relationship shown in the figure, we are able to derive a linear formula written in the matrix form as:

$$\mathbf{b_h} = \mathbf{C}\triangle\mathbf{v} \tag{6.5}$$

where

$$\mathbf{b_h} = \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} C_x & 0 \\ 0 & C_y \end{bmatrix} \text{ and } \triangle\mathbf{v} = \begin{bmatrix} x_{P'_1C'_1} - x_{P''_2C''_2} \\ y_{P'_1C'_1} - y_{P''_2C''_2} \end{bmatrix} \tag{6.6}$$

We name **C** the coefficient matrix. In order to determine it, we need a re-calibration procedure, during which the user is asked to look at the calibrated points with natural eye-head coordination. Thus, we can acquire a set of pairs of $\mathbf{b_h}$ and $\triangle\mathbf{v}$. All the calibrated pairs form a linear system, which can be solved using least squares solution to estimate the matrix **C**. Without loss of generality, if we assume the **C** is non-singular, the *head-compensated* gaze vector $\mathbf{v}'$ is simply calculated by

$$\mathbf{v}' = \mathbf{C}^{-1}\mathbf{b_h} + \mathbf{v}'' \tag{6.7}$$

Note, that we introduce a *head-normalized* gaze vector $\mathbf{v}''$ that accounts for normalizing the head rotations with the observable iris-corner vector $\overrightarrow{P_2'C_2'}$. In the following section, we will give the details about how to extract $\mathbf{v}''$ and track 3D head rotations $\mathbf{b_h}$ from grabbed images.

6.2.5 Extracting Iris-Corner Vector

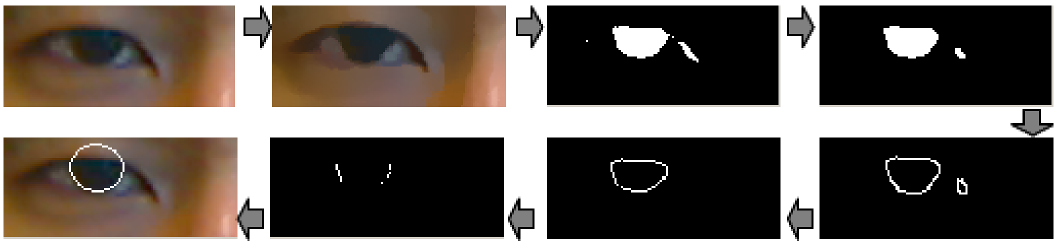


Figure 6.9: Illustration of iris ellipse fitting. Follow the arrow direction: Original eye image → Meanshift Segmentation → Histogram-based binary segmentation → 'Open' morphology → Canny edge operation → Largest contour finding → Keeping near vertical edge points → Fitted ellipse

We follow the technique in [Zhu 2002] to detect iris-corner vector as the eye gaze indicator. In our vector detection, a face detector is first employed to localize face region and then an eye detector is used to find the potential eye region on the localized face, both of which are based on the object detection cascade of boosted classifier [R.Lienhart 2002]. Subsequently, the iris and corner points are accurately found using the following method.

We combine the SIFT matching-based method and an ellipse fitting method to detect iris center. The SIFT was proposed by David Lowe [Lowe 2004b] to detect feature points in images. We use the SIFT codes provided by Rob Hess [1] in our system. The reference SIFT features for iris center is manually selected during calibration. Setting the ratio of Euclidean distances between two nearest neighbor features $r = 0.65$ performs best in our tests. When the SIFT-based method fails, an ellipse fitting approach will take it over. This happens when the SIFT features between successive frames are over the matching threshold. The ellipse fitting approach, illustrated in Figure 6.9, is similar to the one proposed in [Wang 2003]. On the other hand, the ellipse-fitted method may fail when the iris contour is unclear. In this case, the SIFT-based method will be re-evoked. Therefore, combining them enables us to acquire more robust iris detection results.

[1] http://web.engr.oregonstate.edu/~hess/

The SIFT-based method is also employed to detect the eye corner. Each single eye has two corners: inner corner and outer corner, but neither of them can be reliably detected over all the frames. Our strategy is to detect both of them and choose the one with smaller matching residual to form iris-corner vector. To gain robust performance, for both iris and corner points, their reference SIFT features are updated after each correct match. The SIFT-based detection is illustrated in Figure 6.10. Moreover, some distance and geometrical constraints, between the left and right eye, are added to eliminate the apparently wrong detections.

6.2.6 Extracting Normalized Gaze Vector

Extracting the *head-normalized* gaze vector $\overrightarrow{P_2''C_2''}$ (shown in Figure 6.8) consists of two steps: head normalization and iris-corner vector detection. The head normalization refers to normalizing the head rotations with the observable gaze vector $\overrightarrow{P_2'C_2'}$, which is implemented by an image warping operation. We define the warp operator $\mathbf{W} : \mathbb{R}^2 \to \mathbb{R}^3 \to \mathbb{R}^2$ as follows: a pixel point vector $\mathbf{x} = \{x, y\}^T$ on the face image plane is projected onto a 3D head surface(ellipsoidal in our case); then we transform the pose of head surface by $\triangle\mathbf{b_h}$ and back project the point from the 3D surface onto the image plane. For any given pixel $\mathbf{x}$, the warping function can yield its corresponding normalized point $\mathbf{W}(\mathbf{x}; -\mathbf{b_h})$. Denote $\mathbf{p_2'}$ as the point vector of P_2' and $\mathbf{c_2'}$ as the point vector of C_2'. Their normalized points $\mathbf{p_2''}$ and $\mathbf{c_2''}$ can be given by: $\mathbf{p_2''} = \mathbf{W}(\mathbf{p_2'}; -\mathbf{b_h}); \mathbf{c_2''} = \mathbf{W}(\mathbf{c_2'}; -\mathbf{b_h})$.

In practice, we have two alternative options to extract $\overrightarrow{P_2''C_2''}$. The first solution is to extract the points $\mathbf{p_2'}$ and $\mathbf{c_2'}$ on frame image $I(\mathbf{x})$ and then find the corresponding points $\mathbf{p_2''}$ and $\mathbf{c_2''}$. Contrarily, the second one is to first perform $\mathbf{W}$ to construct a *head-normalized* face image $I(\mathbf{W}(\mathbf{x}; -\mathbf{b_h}))$, and then run iris-corner extraction to locate $\mathbf{p_2''}$ and $\mathbf{c_2''}$ on $I(\mathbf{W}(\mathbf{x}; -\mathbf{b_h}))$. The essential difference between them is that the SIFT-based vector extraction will be applied before or after the head normalization. Although due to constructing $I(\mathbf{W}(\mathbf{x}; -\mathbf{b_h}))$, the second solution consumes more computational time, it's expected that applying the SIFT-based vector extraction on $I(\mathbf{W}(\mathbf{x}; -\mathbf{b_h}))$ will be superior to applying it on $I(\mathbf{x})$. Therefore, a comparative experiment is undertaken to evaluate the SIFT-based vector detection with and without head normalization. As shown in Figure 6.10, the results suggest that applying it after head normalization substantially outperforms the one before head normalization. We therefore adopt the second solution to extract $\overrightarrow{P_2''C_2''}$.

6.2.7 Experimental Results

Performance of the method extracting iris-corner vector was tested on real video clips containing eye gaze shift. The resolution of each video is 640×480 while the extracted eye region is about 85×40. Figure 6.11 shows some vector extraction results. We can see most of the iris-corner vectors are accurately extracted. In our tests, due to the usage of the combined detection methods, the vector detector is robust to facial variations and illumination changes to a certain degree. A full video

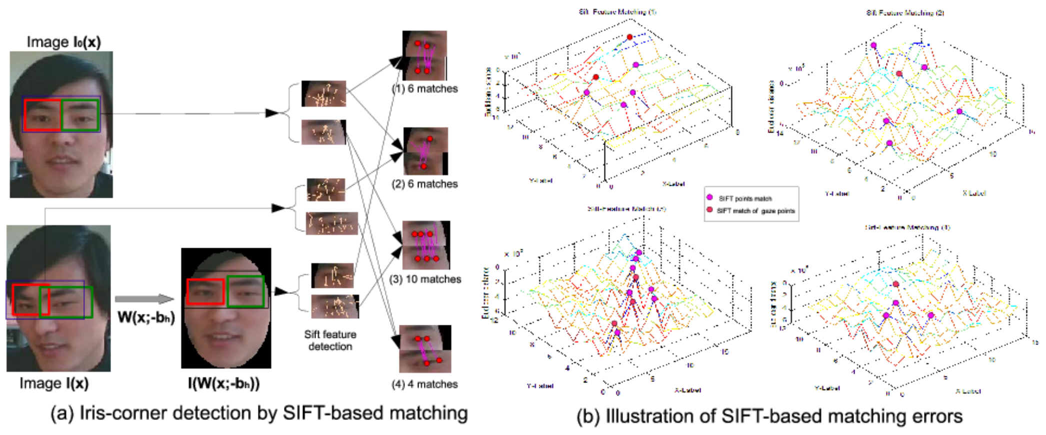


Figure 6.10: The comparisons of SIFT-based gaze detection before and after head normalization. **(a)** The reference SIFT features are automatically detected on the eye region of the image $I_0(\mathbf{x})$. Two eye sub-images, extracted from input frame $I_{\mathbf{x}}$ and the normalized image $I(\mathbf{W}(\mathbf{x}; -\mathbf{b_h}))$, serve as the instances to detect SIFT feature points and match with the reference ones. The correct gaze point matchings are marked in red color. We can see that (1)(3) outperforms (2)(4) in the number of correct matchings. **(b)** Plots (1)-(4) show the matching residuals in (1)-(4) of sub-figure (a) respectively. X-Label denotes reference feature points and Y-Label means the query feature points. It is shown that the differences between marked peaks and noisy peaks in (1)(3) are more distinguished than in (2)(4). The average residuals in (1)(3) are 7.3×10^4 and 3.9×10^4, which are smaller than the ones of (2)(4), i.e. 1.0×10^5 and 6.0×10^4.

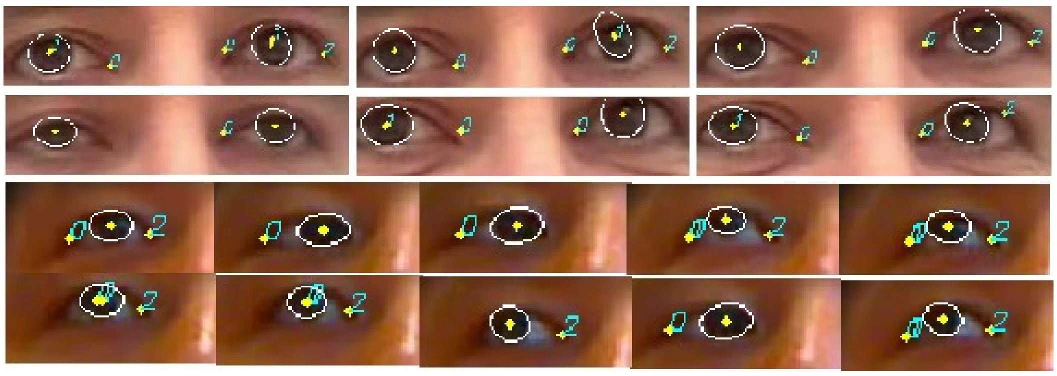


Figure 6.11: Extracting the iris-corner vectors. The yellow point denotes the SIFT-detected point and the one surrounded by an ellipse stands for the fitted ellipse center.

demo is shown in http://video.google.com/videoplay?docid= 6450216116737564885.

We have also applied the proposed framework to a gaze tracking system to test the accuracy of gaze estimation. The system was written in C/C++, using the renowned OpenCV and OpenGL library. In order to evaluate its accuracy, we test the system by collecting several image sequences with several testers. The tests are undertaken in front of a 19-in monitor with a video camera (640×480 pixels) mounted under it. There are two stages in the data collections. In the calibration stage, the testers are asked to stare at 12 evenly-distributed mark points in turn and yet maintain their heads stationary. In the tracking stage, the testers are allowed to freely rotate eye and head to stare the yellow-colored point that is randomly selected from the 12 calibrated points. The length of collected video for each tester is about 2 minutes.

We tested the accuracy of the gaze tracking with three actors. We first apply the system to extract calibrated iris-corner vectors, track head rotations and extract *head-normalized* gaze vectors. The maximum tracked head rotations in horizontal and vertical directions are $21.2°$ and $9.75°$ respectively. We then use these data to compute the matrix **C** and calculate the *head-compensated* vectors according to equation (6.5). The system accuracy is measured in the compensated errors, the pixel errors between the *head-compensated* vectors and calibrated vectors. For comparisons, we also measure the pixel errors between the observable vectors(such as $\overrightarrow{P_2'C_2'}$ in Figure 6.8) and calibrated vectors, called non-compensated errors. Figure 6.12 displays the statistics of the total accuracy measurements. As the four plots suggest, the compensated errors are substantially smaller than the non-compensated ones in horizontal direction while only slightly better in vertical direction. Since the right part in equation (6.5) actually equalizes to the gaze tracking angle, we use this to compute the gaze tracking error. The average head-compensated error is about $5.06°$ and the non-compensated one is about $8.30°$. We can see that compensating head rotation improves the gaze tracking accuracy by about $3.3°$, which is a consid-

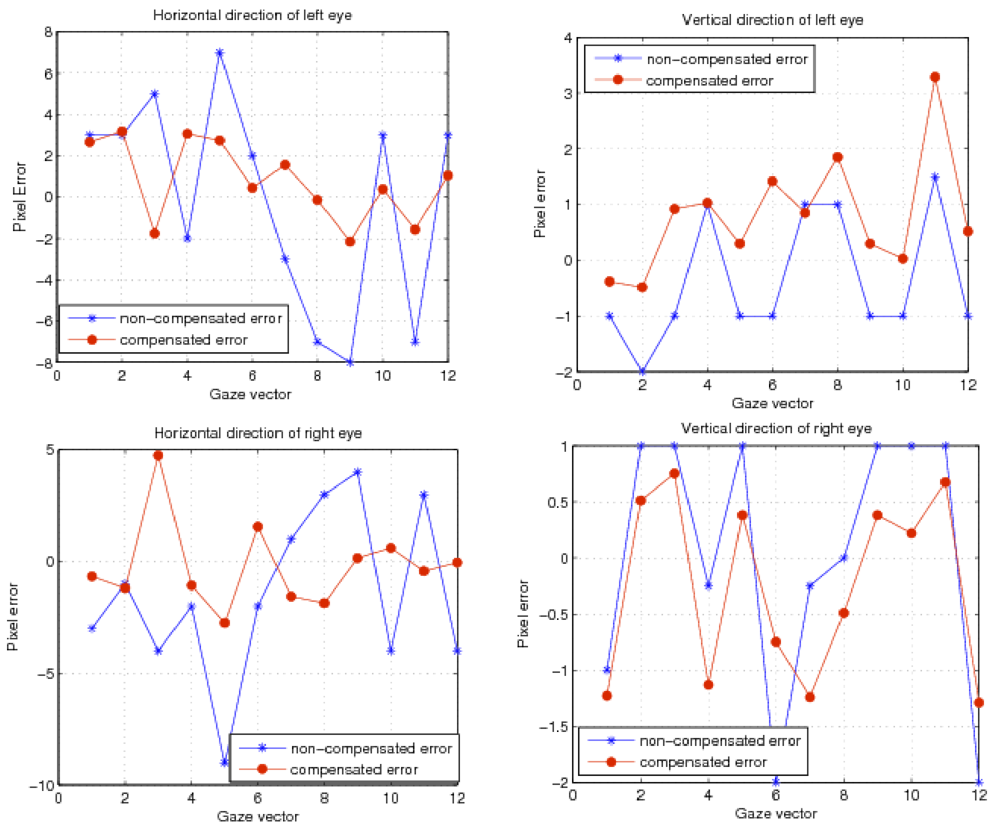


Figure 6.12: The average pixel errors of eye gaze tracking under coordinated head rotations.

erable value for a gaze tracking system. That is to say, ignoring the head rotations will significantly lower the accuracy of the gaze tracking system.

Chapter 7
Conclusion and Perspectives

7.1 Conclusion

In this dissertation, we have investigated the problem of tracking 3D head pose from a monocular video sequence. By studying state of the art, we have found out three basic dilemmas that any a head tracking system must face up with. Then our explorations proceed around how to handle these dilemmas in a practically usable tracking system and our findings lie in three aspects.

The short-baseline solution fully depends on model adaptation and has no priori learning except for a 3D model prior and the initialization at first frame (in Chapter 2) while the tracking-by-detection method is only involved with prior learning (in Chapter 4). Each has its own attributes and drawbacks, but comparatively the second is more feasible in practice since it does not suffer from the assumption of smooth head movements and model drifting although the first handles ever-changing environments more intrinsically. A sound consideration is to combine the proposed prior learning with its online adaptation for a stable yet flexible head tracker.

The fundamental difference between the short-baseline tracker (in Chapter 2) and the tracking-by-detection approach (in Chapter 4) is the scale of the search state space. The former seeks potential pose in a neighboring space that is predicted from the previous estimated one and proceeds along the gradient descent direction. Contrarily, the latter utilizes no prediction but intends to recover optimal pose in the entire state space. As illustrated, the former often reaches local minima which causes drifting and tracking failures in tandem with online update. On the other hand, the former seems more efficient by a local search, but in practice the deficiency of the latter paradigm is successfully overcome by only considering the detected feature points. Furthermore, we also succeed in incorporating prediction-based scheme (optical flow) to deal with motion smoothness in the latter paradigm (in Chapter 5). What we learn from this comparison is that for the sake of permanent tracking, prediction-based prior is better to work as a complementary factor rather than a determinant one.

The short-baseline solution makes use of global appearance feature (in Chapter 2) while the tracking-by-detection one combines local invariant features and global rigid model constraint during tracking (in Chapter 4). Compared to the local distinctive features, the global appearance is more general in the sense that local features are very few in less textured faces and low-resolution videos. However, its drawbacks are also obvious, i.e. containing many redundant pixels that can damage the tracking accuracy and prone to partial occlusions. Experimentally, as long as local features on face are sufficiently found, the combination of local invariant features and global model constraint outperforms the global appearance in most challenging cases, such as moderate lighting and non-rigid face variations.

Moreover, we have demonstrated two example applications of 3D head pose tracking to eye gaze estimation and facial expression transfer. Integrating them into a prototyping virtual collaborative platform has shown perceptually acceptable results. In future, we will quantify their complementary functions in enriching remote human-human collaborations by conducting extensive user studies.

7.2 Perspectives

Our future work will concentrate on the tracking-by-detection 3D head tracking algorithm until a usable system that is insensitive to any challenges is accomplished. To this ultimate end, future improvements will proceed towards the online adaptation of this framework and integrating more contextual information. On one hand, the proposed tracking-by-detection solution (in Chapter 4 and 5) matches online features with the model ones that are learnt before tracking. Their large view discrepancy often leads to insufficient feature matchings. On the contrary, the view discrepancy between consecutive frames is often small (when motion is smooth), and thus we can retain the features learnt from historical frames and incorporate them as additional model features to produce more correspondences. These pooled features can be simultaneously used or alternatively, we can make key-view selection as done in [Vacchetti 2004, Wang 2006] to keep the matching more efficient. On the other hand, currently we have only considered the feature points that are localized at frontal face view. This is problematic when out-of-plane rotations occur, e.g. head nodding or turning left by over 90°. In these cases, the profiling or top view of head turns to be visible and can be nice heuristic information to infer head poses. The way to learn these contextual information from given video examples [Ozuysal 2006, Yao 2010, H. Grabner 2010] usually cannot produce a generative model. Unlike them, we intend to do all the things in an online model to infer head poses by exploiting both structural and textural contexts.

Bibliography

[Alexa 2000] Marc Alexa, Daniel Cohen-Or and David Levin. *As-rigid-as-possible shape interpolation.* In SIGGRAPH '00, pages 157–164, New York, NY, USA, 2000. 31

[Bailenson 2004] J. N. Bailenson and J. Blascovich. *Avatars.* In W. S. Bainbridge (Ed.), Encyclopedia of human-computer interaction, pages 64–68, 2004. 78

[Baluja 1993] Shumeet Baluja and Dean Pomerleau. *Non-Intrusive Gaze Tracking Using Artificial Neural Networks.* AAAI Fall Symposium Series, Machine Learning in Computer Vision: What, Why and How?, 1993. 84

[Basu 1996] Sumit Basu, Irfan Essa and Alex Pentland. *Motion Regularization for Model-based Head Tracking.* In IEEE International Conference on Pattern Recognition, pages 611–616, Vienna, Austria, 1996. 39

[Bay 2006] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. *SURF: Speeded Up Robust Features.* In European Conference on Computer Vision, pages 404–417, Graz, Austria, 2006. 47

[Beis 1997] Jeffrey S. Beis and David G. Lowe. *Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1000–1006, 1997. 48

[Benford 1995] Steve Benford, John Bowers, Lennart E. Fahlen, Chris Greenhalgh and Dave Snowdon. *User embodiment in collaborative virtual environments.* In ACM SIGCHI conference on Human factors in computing systems, pages 242–249, 1995. 78

[Beymer 2003] D. Beymer and M. Flickner. *Eye gaze tracking using an active stereo head.* IEEE International Conference on Computer Vision and Pattern Recognition, vol. 2, pages 451–458, June 2003. 84

[Blanz 1999a] Volker Blanz and Thomas Vetter. *A morphable model for the synthesis of 3D faces.* In ACM SIGGRAPH'99, pages 187–194, New York, NY, USA, 1999. 4

[Blanz 1999b] Volker Blanz and Thomas Vetter. *A morphable model for the synthesis of 3D faces.* In SIGGRAPH'99, pages 187–194, 1999. 29

[Blanz 2003] Volker Blanz, Curzio Basso, Thomas Vetter and Tomaso Poggio. *Reanimating faces in images and video.* EUROGRAPHICS'03, vol. 22, 2003. 29

[Bosch 2007] A. Bosch, A. Zisserman and X. Muoz. *Image Classification using Random Forests and Ferns.* In IEEE International Conference on Computer Vision, 2007. 69

[Bouguet 2002] Jean-Yves Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the Algorithm.* Microprocessor Labs, Intel Corporation, 2002. 62

[Brand 2001] M. Brand and R. Bhotika. *Flexible Flow for 3D Nonrigid Tracking and Shape Recovery.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 315–322, 2001. 3

[Breitenstein 2008] M.D. Breitenstein, D. Kuettel, T. Weise, L.J. Van Gool and H. Pfister. *Real-Time Face Pose Estimation from Single Range Images.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008. 6

[Brown 2001] L.M. Brown. *3D Head Tracking Using Motion Adaptive Texture-Mapping.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 998–1003, 2001. 4

[Brox 2006] Thomas Brox, Bodo Rosenhahn, Daniel Cremers and Hans peter Seidel. *High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints.* In European Conference on Computer Vision, pages 98–111. Springer, 2006. 59, 63

[Cascia 1999] Marco La Cascia, Stan Sclaroff and Vassilis Athitsos. *Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texture-Mapped 3D Models.* IEEE International Conference on Computer Vision and Pattern Recognition, vol. 22, pages 322–336, 1999. vi, 3, 17, 39, 52, 57, 71

[Cassell 2000a] Justine Cassell. *Embodied conversational interface agents.* ACM Communications, vol. 43, no. 4, pages 70–78, 2000. 78

[Cassell 2000b] Justine Cassell. *Nudge nudge wink wink: elements of face-to-face conversation for embodied conversational agents.* Embodied conversational agents, pages 1–27, 2000. 78

[Chai 2003] Jinxiang Chai, Jing Xiao and Jessica K Hodgins. *Vision-based Control of 3D Facial Animation.* In ACM SIGGRAPH/Eurographics Symposium on Computer Animation, July 2003. 30

[Chen 2003] Longbin Chen, Lei Zhang, Yuxiao Hu, Mingjing Li and Hongjiang Zhang. *Head Pose Estimation using Fisher Manifold Learning.* In IEEE International Workshop on Analysis and Modeling of Faces and Gestures, page 203. IEEE Computer Society, 2003. 5

[Chum 2005] Ondrej Chum and Jiri Matas. *Matching with PROSAC - Progressive Sample Consensus.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 220–226, 2005. 63

[Collins 2005] Robert Collins, Yanxi Liu and Marius Leordeanu. *On-Line Selection of Discriminative Tracking Features.* IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 27, no. 1, pages 1631 – 1643, October 2005. 66

[Cootes 2000] T. F. Cootes, K. Walker and C.J. Taylor. *View-Based Active Appearance Models.* In IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 484–498. Springer, 2000. 4

[Cootes 2001] Timothy F. Cootes, Gareth J. Edwards and Christopher J. Taylor. *Active appearance models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 6, pages 681–684, 2001. 4

[Cuno 2007] Alvaro Cuno, Claudio Esperanca, Antonio Oliveira and Paulo R. Cavalcanti. *3D as-rigid-as-possible deformations using MLS.* In Computer Graphics International'09, Victoria, Canada, May 2007. 27, 28, 29, 30, 31

[Darrell 1999] Harville Rahimi Darrell, M. Harville, A. Rahimi, T. Darrell, G. Gordon and J. Woodfill. *3D Pose Tracking with Linear Depth and Brightness Constraints.* In IEEE International Conference on Computer Vision, pages 206–213, 1999. 6

[DeCarlo 1998] Douglas DeCarlo, Dimitris Metaxas and Matthew Stone. *An anthropometric face model using variational techniques.* In SIGGRAPH '98, pages 67–74, New York, NY, USA, 1998. ACM. 29

[Decarlo 2000] Douglas Decarlo and Dimitris Metaxas. *Optical Flow Constraints on Deformable Models with Applications to Face Tracking.* International Journal of Computer Vision, vol. 38, no. 2, pages 99–127, 2000. 3, 60, 63

[Dey 2005] Tamal K. Dey and Jian Sun. *An adaptive MLS surface for reconstruction with guarantees.* In Eurographics symposium on Geometry processing, page 43, Vienna, Austria, 2005. Eurographics Association. 33

[Dornaika 2004] Fadi Dornaika and Jorgen Ahlberg. *Fast And Reliable Active Appearance Model Search For 3d Face Tracking.* IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 34, pages 1838–1853, 2004. 4

[Dornaika 2008a] Fadi Dornaika and Franck Davoine. *Simultaneous Facial Action Tracking and Expression Recognition in the Presence of Head Motion.* International Journal of Computer Vision, vol. 76, no. 3, pages 257–281, 2008. 4, 17

[Dornaika 2008b] Fadi Dornaika and Franck Davoine. *Simultaneous Facial Action Tracking and Expression Recognition in the Presence of Head Motion.* International Journal of Computer Vision, vol. 76, no. 3, pages 257–281, 2008. 39

[Fleishman 2005] Shachar Fleishman, Daniel Cohen-Or and Cláudio T. Silva. *Robust moving least-squares fitting with sharp features.* In SIGGRAPH '05, pages 544–552, Los Angeles, California, 2005. 27

[Freedman 2000] Edward G. Freedman and David L. Sparks. *Coordination of the eyes and head: movement kinematics.* Experimental Brain Research, vol. 131, no. 1, pages 22–32, 2000. 84, 85, 87

[Freedman 2001] Edward G. Freedman. *Interactions between eye and head control signals can account for movement kinematics.* Biol. Cybern., vol. 84, no. 6, pages 453–462, 2001. 84, 85, 87

[Freund 2007] Y. Freund, S. Dasgupta, M. Kabra and N. Verma. *Learning the structure of manifolds using random projections.* In Neural Information Processing Systems, 2007. 68

[Guestrin 2003] E.D. Guestrin and M. Eizenman. *General theory of remote gaze estimation using the pupil center and corneal reflections.* IEEE Transactions on Biological Engineering, vol. 53, no. 6, 2003. 84

[H. Grabner 2010] L. Van Gool H. Grabner J. Matas and P. Cattin. *Tracking the invisible: Learning Where the Object Might be.* In IEEE International Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 2010. 96

[Haber 2004] Jorg Haber and Demetri Terzopoulos. *Facial modeling and animation.* In ACM SIGGRAPH 2004 Course Notes, 2004. 29

[Hanes 2006] Douglas A. Hanes and Gin McCollum. *Variables Contributing to the Coordination of Rapid Eye/Head Gaze Shifts.* Biol. Cybern., vol. 94, no. 4, pages 300–324, 2006. 84

[Harris 1988] C. Harris and M. Stephens. *A Combined Corner and Edge Detection.* In Proceedings of The Fourth Alvey Vision Conference, pages 147–151, 1988. 45

[Hartley 2004] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge University Press, March 2004. 49, 62

[Heinzmann 1998] Jochen Heinzmann and Er Zelinsky. *3-D facial pose and gaze point estimation using a robust real-time tracking paradigm.* IEEE International Conference on Computer Vision and Pattern Recognition, pages 142–147, 1998. 84

[Helmut PRENDINGERY 2003] Nonmember Helmut PRENDINGERY and Mitsuru ISHIZUKA. *Designing and evaluating animated agents as social actors.* IEICE Transactions on Information and Systems, vol. E86-D, no. 8, pages 1378–1385, 2003. 78

[Hong 2009] X. Hong, H. Chang, S. Shan, X. Chen and W. Gao. *Sigma Set: A Small Second Order Statistical Region Descriptor.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1802–1809, 2009. 68

[Horprasert 1996] T. Horprasert, Y. Yacoob and L.S. Davis. *Computing 3-D head orientation from a monocular image sequence.* In IEEE International Conference on Automatic Face and Gesture Recognition, pages 242–247, 1996. 5

[Hu 2004] Yuxiao Hu, Longbin Chen, Yi Zhou and Hongjiang Zhang. *Estimating Face Pose by Facial Asymmetry and Geometry.* IEEE International Conference on Automatic Face and Gesture Recognition, vol. 0, 2004. 5

[Huang 2004] Kohsia S. Huang and Mohan M. Trivedi. *Robust Real-Time Detection, Tracking, and Pose Estimation of Faces in Video Streams.* In IEEE International Conference on Pattern Recognition, pages 965–968. IEEE Computer Society, 2004. 6

[Iain 2004] Matthews Iain and Baker Simon. *Active appearance Models Revisited.* International Journal of Computer Vision, vol. 60, no. 2, pages 135–164, 2004. 14

[Isard 1998] Michael Isard and Andrew Blake. *A smoothing filter for Condensation.* In European Conference on Computer Vision, pages 767–781, 1998. 59, 61

[Jang 2008] Jun-Su Jang and Takeo Kanade. *Robust 3D Head Tracking by Online Feature Registration.* In IEEE International conference on Automatic Face and Gesture Recognition, Amsterdam, The Netherlands, September 2008. 5

[Ji 2002] Qiang Ji and Rong Hu. *3D face pose estimation and tracking from a monocular camera.* Image and Vision Computing, vol. 0, 2002. 5

[Joerg 2005] Hauber Joerg, Regenbrecht Holger, Hills Aimee, Cockburn Andrew and Billinghurst Mark. *Social Presence in Two- and Three-dimensional Videoconferencing.* In PRESENCE 2005: The 8th Annual International Workshop on Presence, pages 189–198, 2005. 77, 78

[Kauff 2002] Peter Kauff and Oliver Schreer. *An immersive 3D video-conferencing system using shared virtual team user environments.* In International conference on Collaborative virtual environments, pages 105–112, Bonn, Germany, 2002. ACM. 77

[Kinoshita 2006] Koichi Kinoshita, Yong Ma, Shihong Lao and Masato Kawaade. *A fast and robust 3D head pose and gaze estimation system*. IEEE International Conference on Multimodal Interfaces, pages 137–138, 2006. 84

[Kluckner 2009] S. Kluckner, T. Mauthner, P.M. Roth and H. Bischof. *Semantic Classication in Aerial Imagery by Integrating Appearance and Height Information*. In Asian Conference on Computer Vision, 2009. 68

[Lancaster 1981] P. Lancaster and K. Salkauskas. *Surfaces Generated by Moving Least Squares Methods*. Mathematics of Computation, vol. 37, no. 155, pages 141–158, 1981. 27

[Lee 1995] Yuencheng Lee, Demetri Terzopoulos and Keith Walters. *Realistic modeling for facial animation*. In SIGGRAPH '95, pages 55–62, 1995. 29

[Lee 2006] Sang-Yup Lee, Sang C. Ahn, Hyoung-Gon Kim and MyoTaeg Lim. *Realtime 3D video avatar in mixed reality: an implementation for immersive telecommunication*. Simul. Gaming, vol. 37, no. 4, pages 491–506, 2006. 77

[Lepetit 2004] V. Lepetit, J. Pilet and P. Fua. *Point matching as a classification problem for fast and robust object pose estimation*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 244–250, Washington, DC, 2004. 5, 39

[Lepetit 2005] Vincent Lepetit and Pascal Fua. *Monocular model-based 3D tracking of rigid objects*. Found. Trends. Comput. Graph. Vis., vol. 1, no. 1, pages 1–89, 2005. 39, 61

[Lepetit 2006] Vincent Lepetit and Pascal Fua. *Keypoint Recognition Using Randomized Trees*. IEEE Transactions on Pattern Recognition and Machine Intelligence, vol. 28, no. 9, pages 1465–1479, 2006. 41

[Leung 2000] Wing Ho Leung, B.L. Tseng, Zon-Yin Shae, F. Hendriks and T. Chen. *Realistic video avatar*. IEEE International Conference on Multimedia and Expo, vol. 2, pages 631–634, 2000. 77

[Levin 1998] David Levin. *The approximation power of moving least-squares*. Mathematics of Computation, vol. 67, no. 224, pages 1517–1531, 1998. 27

[Liao 2008] W.K. Liao and G. Medioni. *3D face tracking and expression inference from a 2D sequence using manifold learning*. In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008. 5

[Liebelt 2010] Jörg Liebelt and Cordelia Schmid. *Multi-View Object Class Detection with a 3D Geometric Model*. In IEEE International Conference on Computer Vision and Pattern Recognition, June 2010. 39

[Lowe 2004a] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of Computer Vision, vol. 60, pages 91–110, 2004. 13

[Lowe 2004b] David G. Lowe. *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision, vol. 60, pages 91–110, 2004. 89

[Lucas 1981] B. D. Lucas and T. Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision.* In IJCAI'81, pages 674–679, 1981. 61

[Malciu 2000] Marius Malciu and Françoise Prêteux. *A Robust Model-Based Approach for 3D Head Tracking in Video Sequences.* In IEEE International Conference on Automatic Face and Gesture Recognition, page 169, Washington, DC, USA, 2000. IEEE Computer Society. 6

[Matthews 2004a] Iain Matthews and Simon Baker. *Active appearance Models Revisited.* International Journal of Computer Vision, vol. 60, no. 2, pages 135–164, 2004. 4

[Matthews 2004b] Iain Matthews, Takahiro Ishikawa and Simon Baker. *The Template Update Problem.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, pages 810 – 815, June 2004. 18

[Matthews 2007] Iain Matthews, Jing Xiao and Simon Baker. *2D vs. 3D deformable face models: representational power, construction, and real-time fitting.* International Journal of Computer Vision, vol. 75, no. 1, pages 93–113, 2007. 4

[Mikolajczyk 2004] Krystian Mikolajczyk and Cordelia Schmid. *Scale & Affine Invariant Interest Point Detectors.* International Journal of Computer Vision, vol. 60, no. 1, pages 63–86, October 2004. 45

[Moosmann 2008] F. Moosmann, E. Nowak and F. Jurie. *Randomized Clustering Forests for Image Classification.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, pages 1632–1646, 2008. 69

[Morency 2003a] Louis-Philippe Morency, Ali Rahimi and Trevor Darrell. *Adaptive View-Based Appearance Models.* In IEEE International Conference on Computer Vision and Pattern Recognition, volume 1, pages 803–810, 2003. 6, 39

[Morency 2003b] Louis-Philippe Morency, Patrik Sundberg and Trevor Darrell. *Pose Estimation Using 3D View-Based Eigenspaces.* In IEEE Workshop on Analysis and Modeling of Faces and Gestures in Conjunction with ICCV'03, pages 45–52, 2003. 5

[Morency 2010] Louis-Philippe Morency, Jacob Whitehill and Javier Movellan. *Monocular head pose estimation using generalized adaptive view-based appearance model.* Image and Vision Computing, vol. 28, no. 5, pages 754–761, 2010. 4

[Moreno-Noguer 2007a] Francesc Moreno-Noguer, Vincent Lepetit and Pascal Fua. *Accurate Non-Iterative O(n) Solution to the PnP Problem.* In IEEE International Conference on Computer Vision, volume 0, pages 1–8, Rio de Janeiro, Brazil, 2007. 49, 62

[Moreno-Noguer 2007b] Francesc Moreno-Noguer, Alberto Sanfeliu and Dimitris Samaras. *Integration of deformable contours and a multiple hypotheses Fisher color model for robust tracking in varying illuminant environments.* Image and Vision Computing, vol. 25, pages 285–296, 2007. 66

[Moreno 2002] Francesc Moreno, Adria Tarrida, Juan Andrade-cetto and Alberto Sanfeliu. *3D Real-Time Head Tracking Fusing Color Histograms and Stereovision.* In IEEE International Conference on Pattern Recognition, pages 368–371, 2002. 6

[Muja 2009] Marius Muja and David G. Lowe. *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration.* In VISAPP'09, pages 331–340, Angers, France, 2009. 48

[Murphy-chutorian 2008] Erik Murphy-chutorian and Mohan Manubhai Trivedi. *HyHOPE: Hybrid Head Orientation and Position Estimation for Vision-based Driver Head Tracking.* In IEEE Intelligent Vehicles Symposium, pages 512–517, 2008. 6

[Murphy-Chutorian 2009] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. *Head Pose Estimation in Computer Vision: A Survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, pages 607–626, 2009. 3

[Nicolas 2004] Gourier Nicolas, Hall Daniela and Crowley James. *Estimating Face Orientation from Robust Detection of Salient Facial Features.* In ICPR'04 Workshop on Visual Observation of Deictic Gestures, Cambridge, UK, 2004. v, 35, 37

[Nikolaidis 2000] Athanasios Nikolaidis and Ioannis Pitas. *Facial feature extraction and pose determination.* Pattern Recognition, vol. 33, pages 1783–1791, July 2000. 6

[Niyogi 1996] S. Niyogi and W. T. Freeman. *Example-based head tracking.* In IEEE International Conference on Automatic Face and Gesture Recognition, pages 374–379, Washington, DC, USA, 1996. IEEE Computer Society. 4

[Ohayon 2006] Shay Ohayon and Ehud Rivlin. *Robust 3D head tracking using camera pose estimation.* In IEEE International Conference on Pattern Recognition, pages 1063–1066, 2006. 5

[Ozuysal 2006] Mustafa Ozuysal, Vincent Lepetit, Francois Fleuret and Pascal Fua. *Feature Harvesting for Tracking-by-Detection.* In European Conference on Computer Vision, pages 592–605, Graz, Austria, 2006. Springer. 96

[Ozuysal 2009] M. Ozuysal, M. Calonder, V. Lepetit and P. Fua. *Fast Keypoint Recognition Using Random Ferns.* IEEE International Conference on Computer Vision and Pattern Recognition, vol. PP, no. 99, pages 1–1, 2009. 46

[Paterson 2003] J.A. Paterson and A.W. Fitzgibbon. *3D head tracking using non-linear optimization.* In British Machine Vision Conference, 2003. 4

[Pei 2007] Yuru Pei and Hongbin Zha. *Stylized synthesis of facial speech motions.* Computer Animation and Virtual Worlds, vol. 18, no. 4-5, pages 517–526, 2007. 29

[Pelachaud 2002] Catherine Pelachaud and Isabella Poggi. *Multimodal embodied agents.* Knowl. Eng. Rev., vol. 17, no. 2, pages 181–196, 2002. 78

[Pighin 1998] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szelisk and David H. Salesin. *Synthesizing realistic facial expressions from photographs.* ACM SIGGRAPH'98, pages 75–84, 1998. 34

[Prendingery 2001] H. Prendingery and M. Ishizuka. *Simulating Affective Communication with Animated Agents.* In International Conference on Human-Computer Interaction, pages 182–189, 2001. 78

[Quax 2003] P. Quax, T. Jehaes, P. Jorissen and W. Lamotte. *A Multi-User Framework Supporting Video-Based Avatars.* In Proceedings of 2nd Workshop on Network and System Support for Games, pages 137–147, 2003. 79

[Rae 1998] R. Rae and H.J. Ritter. *Recognition of Human Head Orientation Based on Artificial Neural Networks.* IEEE Transactions on Neural Networks, vol. 9, no. 2, pages 257–265, March 1998. 4

[Raytchev 2004] Bisser Raytchev, Ikushi Yoda and Katsuhiko Sakaue. *Head Pose Estimation by Nonlinear Manifold Learning.* In IEEE International Conference on Pattern Recognition, volume 4, pages 462–466. IEEE Computer Society, 2004. 5

[Regenbrecht 2003] H. Regenbrecht, C. Ott, M. Wagner, T. Lum, P. Kohler, W. Wilke and E. Mueller. *An augmented virtuality approach to 3D videoconferencing.* IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 290–291, 2003. 77

[R.Lienhart 2002] R.Lienhart, A.Kuranov and V.Pisarevsky. *Empirical analysis of detection cascades of boosted classifiers for rapid object detection.* MRL Tech. Report, Intel Labs, 2002. 89

[Ruddarraju 2003] Ravikrishna Ruddarraju, Antonio Haro and Irfan A. Essa. *Fast Multiple Camera Head Pose Tracking.* In Vision Interface'03, 2003. 6

[S. 2003] Romdhani S. and Vetter T. *Efficient, robust and accurate fitting of a 3D morphable model.* IEEE International Conference on Computer Vision, pages 59–66, October 2003. 4

[Schaefer 2006] Scott Schaefer, Travis McPhail and Joe Warren. *Image deformation using moving least squares.* In ACM SIGGRAPH'06, volume 25, pages 533–540, Boston, USA, 2006. 27, 28, 30

[Schreer 2005] O. Schreer, R. Tanger, P. Eisert, P. Kauff, B. Kaspar and R. Englert. *Real-Time Avatar Animation Steered by Live Body Motion.* In International Conference on Image Analysis and Processing, pages 147–154, 2005. 79

[Schroeder 2002] Ralph Schroeder. *Social interaction in virtual enviroments: key issues, common themes, and a framework for research.* The social life of avatars: presence and interaction in shared virtual environments, pages 1–18, 2002. 78

[Seemann 2004] E. Seemann, K. Nickel and R. Stiefelhagen. *Head pose estimation using stereo vision for human-robot interaction.* In IEEE International Conference on Automatic Face and Gesture Recognition, pages 626–631, 2004. 6

[Sherrah 2001] J. Sherrah and S.G. Gong. *Fusion of perceptual cues for robust tracking of head pose and position.* Pattern Recognition, vol. 34, no. 8, pages 1565–1572, August 2001. 6

[Sorkine 2007] Olga Sorkine and Marc Alexa. *As-rigid-as-possible surface modeling.* In SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing, pages 109–116, Barcelona, Spain, 2007. 31

[Strom 2002] Jacob Strom. *Model-based real-time head tracking.* EURASIP Journal on Applied Signal Processing, vol. 2002, no. 1, pages 1039–1052, 2002. 5

[Sung 2008] J.W. Sung, T. Kanade and D.J. Kim. *Pose Robust Face Tracking by Combining Active Appearance Models and Cylinder Head Models.* International Journal of Computer Vision, vol. 80, no. 2, pages 260–274, November 2008. 6

[Takahiro Ishikawa 2004] Iain Matthews Takahiro Ishikawa Simon Baker and Takeo Kanade. *Passive Driver Gaze Tracking with Active Appearance Models.* Rapport technique CMU-RI-TR-04-08, Robotics Institute, Pittsburgh, PA, February 2004. 84

[Takemura 2002] Masayuki Takemura and Yuichi Ohta. *Diminishing Head-Mounted Display for Shared Mixed Reality.* In IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 149–156, 2002. 77

[Terzopoulos 2004] D. Terzopoulos and Y. Lee. *Behavoiral animation of faces: Parallel, distributed, and real-time.* ACM SIGGRAPH 2004 Course Notes, 2004. 29

[Tuzel 2006] O. Tuzel, F. Porikli and P. Meer. *Region covariance: a fast descriptor for detection and classification.* In European Conference on Computer Vision, pages 589–600, 2006. 68

[Vacchetti 2004] L. Vacchetti, V. Lepetit and P. Fua. *Stable real-time 3D tracking using online and offline information.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 10, pages 1385–1391, October 2004. 59, 65, 96

[van de Sande 2010] K. E. A. van de Sande, T. Gevers and C. G. M. Snoek. *Evaluating Color Descriptors for Object and Scene Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010. 70, 72

[Vogler 2007] C. Vogler, Z.G. Li, A. Kanaujia, S.K. Goldenstein and D.N. Metaxas. *The Best of Both Worlds: Combining 3D Deformable Models with Active Shape Models.* In IEEE International Conference on Computer Vision and Pattern Recognition, pages 1–7, 2007. 6

[Voit 2006] Michael Voit, Kai Nickel and Rainer Stiefelhagen. *Neural Network-based Head Pose Estimation and Multi-view Fusion.* In Proc. CLEAR Workshop, LNCS, pages 299–304, 2006. 5

[Wang 2001] J.G. Wang and E. Sung. *Pose determination of human faces by using vanishing points.* Pattern Recognition, vol. 34, no. 12, pages 2427–2445, December 2001. 5

[Wang 2003] Jian-Gang Wang, Eric Sung and Ronda Venkateswarlu. *Eye gaze estimation from a single image of one eye.* IEEE International Conference on Computer Vision, vol. 1, pages 136–143, October 2003. 84, 89

[Wang 2006] Qiang Wang, Wei Zhang, Xiaoou Tang and Heung-Yeung Shum. *Real-Time Bayesian 3-D Pose Tracking.* IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 12, pages 1533–1541, 2006. 65, 96

[Wendland 1995] Holger Wendland. *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree.* Advances in Computational Mathematics, vol. 4, pages 389–396, 1995. 32

[Whittaker 1991] Steve Whittaker, Susan E. Brennan and Herbert H. Clark. *Co-ordinating activity: an analysis of interaction in computer-supported co-operative work*. In ACM SIGCHI conference on Human factors in computing systems, pages 361–367, New York, NY, USA, 1991. ACM. 78

[Wu 2000] Ying Wu and Kentaro Toyama. *Wide-Range, Person- and Illumination-Insensitive Head Orientation Estimation*. In IEEE International Conference on Automatic Face and Gesture Recognition, pages 183–188, 2000. 4

[Wu 2008] Junwen Wu and Mohan M. Trivedi. *A two-stage head pose estimation framework and evaluation*. Pattern Recognition, vol. 41, no. 3, pages 1138–1158, 2008. 5

[Xiao 2003] Jing Xiao, Tsuyoshi Moriyama, Takeo Kanade and Jeffrey Cohn. *Robust Full-Motion Recovery of Head by Dynamic Templates and Re-registration Techniques*. International Journal of Imaging Systems and Technology, vol. 13, pages 85 – 94, September 2003. 4

[Xiao 2004] Jing Xiao, Simon Baker, Iain Matthews and Takeo Kanade. *Real-Time Combined 2D+3D Active Appearance Models*. In IEEE International Conference on Computer Vision and Pattern Recognition, volume 2, pages 535–542, Washington, DC, June 2004. 4

[Xin 2005] Le Xin, Qiang Wang, Jianhua Tao, Xiaoou Tang, Tieniu Tan and Harry Shum. *Automatic 3D Face Modeling from Video*. In IEEE International Conference on Computer Vision, pages 1193–1199, Washington, DC, USA, 2005. 29

[Yamazoe 2008] Hirotake Yamazoe, Akira Utsumi, Tomoko Yonezawa and Shinji Abe. *Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions*. Symposium on Eye Tracking Research and Applications, pages 245–250, 2008. 84

[Yang 2002a] Ming-Hsuan Yang, David J. Kriegman and Narendra Ahuja. *Detecting faces in images: A survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pages 34–58, 2002. 66

[Yang 2002b] Ruigang Yang. *Model-based head pose tracking with stereovision*. In IEEE International Conference on Automatic Face and Gesture Recognition, pages 255–260, 2002. 6

[Yang 2002c] Ruigang Yang and Zhengyou Zhang. *Eye Gaze Correction with Stereovision for Video-Teleconferencing*. In European Conference on Computer Vision, pages 479–494, London, UK, 2002. Springer-Verlag. 81

[Yao 2010] Bangpeng Yao and Li Fei-Fei. *Modeling Mutual Context of Object and Human Pose in Human-Object Interaction Activities*. In IEEE International

Conference on Computer Vision and Pattern Recognition, San Francisco, CA, June 2010. 96

[Yin 2009] Zhaozheng Yin and R.T. Collins. *Shape constrained figure-ground segmentation and tracking.* IEEE International Conference on Computer Vision and Pattern Recognition, vol. 0, pages 731–738, 2009. 66

[Yoo 2002] Dong Hyun Yoo, Bang Rae Lee and Myoung Jin Chung. *Non-Contact Eye Gaze Tracking System by Mapping of Corneal Reflections.* IEEE International Conference on Automatic Face and Gesture Recognition, page 101, 2002. 84

[Zhang 2001] Zhengyou Zhang, Zicheng Liu, Chuck Jacobs and Michael Cohen. *Rapid Modeling of Animated Faces From Video.* Journal of Visualization and Compute Animation, vol. 12, no. 4, pages 227–240, September 2001. 29

[Zhang 2002] Ye Zhang, Chandra Kambhamettu and Ra Kambhamettu. *3D head-tracking under partial occlusion.* Pattern Recognition, vol. 35, pages 176–182, 2002. 3

[Zhang 2008] Wei Zhang, Qiang Wang and Xiaoou Tang. *Real Time Feature Based 3-D Deformable Face Tracking.* In European Conference on Computer Vision, pages 720–732, Berlin, Heidelberg, 2008. Springer-Verlag. 5, 59

[Zhou 2010] Mingcai Zhou, Lin Liang, Jian Sun and Yangsheng Wang. *AAM based Face Tracking with Temporal Matching and Face Segmentation.* In IEEE International Conference on Computer Vision and Pattern Recognition, 2010. 66

[Zhu 2002] Jie Zhu and Jie Yang. *Subpixel eye gaze tracking.* IEEE International Conference on Automatic Face and Gesture Recognition, pages 124–129, May 2002. 84, 89

[Zhu 2005] Zhiwei Zhu and Qiang Ji. *Eye Gaze Tracking Under Natural Head Movements.* IEEE International Conference on Computer Vision and Pattern Recognition, pages 918–923, 2005. 84, 85, 86