

博士論文

Head Pose and Gaze Estimation
for Inferring Focus of Attention

(注目領域獲得のための頭部姿勢および注視点推定)



東京大学大学院
情報理工学系研究科
電子情報学専攻

48-077405

菅野 裕介

指導教員

佐藤 洋一 准教授

© Copyright by Yusuke Sugano 2010.
All rights reserved.

Abstract

Head pose and gaze direction play significant roles in inferring the focus of attention, and they also help us to design more human-centered computer systems. The use of camera-based techniques for remote sensing of head pose and gaze should lead to a wide range of applications. However, although many methods have been proposed, there are technical limitations in the estimation techniques. Accurate estimation using only a monocular camera is still a difficult task, and existing methods often require calibration prior to estimation. The goal of this thesis is developing a head pose and gaze estimation system with minimal requirements; the methods used in the developed system do not need active calibration or equipment other than a camera.

The first part of this thesis describes a monocular method of tracking 3D head poses and facial actions. Using a multilinear face model that treats interpersonal and intrapersonal shape variations separately, this method estimates the parameters in real time by integrating two frameworks: particle filter-based tracking for time-dependent poses and facial action estimation and incremental bundle adjustment for person-dependent shape estimation. The use of this unique combination in conjunction with the multilinear face model enables tracking of faces and facial actions in real time without having to use pre-learned individual face models.

In the second part of this thesis, an unconstrained gaze estimation method is presented that uses an online learning algorithm and that allows free head movement by the user in a casual desktop environment. The key assumption is that the user gazes at the cursor position whenever s/he presses the mouse button. The user's eye images and 3D head poses are continuously captured on the basis of the head pose estimation method described in the first part of the thesis. By using clicked positions as exemplars of gaze positions, our system collects learning samples for estimating gazes while the user uses the PC, un-

aware of the system's activities. The samples are adaptively clustered in accordance with the head pose, and the estimation parameters are incrementally updated. In this way, our method avoids the lengthy calibration stage prior to using the gaze estimator.

One of the drawbacks of our method is that it cannot be applied to passive displays without user interaction. To solve this problem, we developed a calibration-free gaze sensing framework, and it is presented in the last part of this thesis. It uses visual saliency maps of video frames that are computed in a bottom-up manner. By relating the saliency maps with the appearances of the eyes of the person watching the video frames, our method automatically constructs a gaze estimator. In order to identify gaze points efficiently from saliency maps, saliency maps are aggregated to generate probability distributions of gaze points. Mapping between the eye images and gaze points is then established by Gaussian process regression. This results in a gaze estimator that frees users from active calibration and that can be applied to any type of display device.

Using the methods proposed in this thesis will make head pose and gaze estimation substantially more practical by reducing installation and setup costs. They can be used with commonly available cameras, and estimation procedures without manual initialization can be seamlessly integrated into daily computer interactions. This will enhance the potential for future investigation of attention-based application systems that will enrich our daily lives with ubiquitous computing devices.

Acknowledgements

First and foremost, I would like to thank my advisor, Yoichi Sato, for his guidance and advice over the years. His clear insight and bright personality continually encouraged me to complete this thesis. I would also like to thank Hideki Koike, from the University of Electro-Communications, for his valuable comments and suggestions from the viewpoint of human-computer interaction. Besides my advisors, I would like to thank the members of my thesis committee, Katsushi Ikeuchi, Kiyoharu Aizawa, Takeshi Naemura, Shunsuke Kamijo and Toshihiko Yamasaki, for their helpful feedback on my research.

I obtained valuable and exciting experience by working as an intern at Microsoft Research Asia. I am grateful to everyone I met during my internship. I would like to express my appreciation especially to Yasuyuki Matsushita, my mentor at Microsoft Research Asia and my collaborator during this thesis. His positive and active attitude strongly influenced my views on my research and my career.

Imari Sato and Takahiro Okabe always gave me very helpful advice, not only on my research but also on my life. My work experience with Kenji Oka provided me with insights into how to fulfill the requirements of this thesis. I greatly appreciate the many insightful discussions I had with Yoshinori Kobayashi, Kris Kitani, Shiro Kumano, Yuyu Liu, Mihoko Shimano, Daisuke Sugimura, Michihiro Kobayashi and all of my laboratory colleagues. I will

never forget the rewarding time I spent with them. A special thanks goes to Sakie Suzuki, Mariko Araki, Yoko Imagawa and Chio Usui for helping me with my study and my life in many ways.

I am also grateful to all my friends in Tokyo, Beijing and all over the world outside of academia. This thesis would not have been possible without the great times I spent with them.

Lastly, my deepest gratitude goes to my family, my sister, my mother and my deceased father, for their love and support.

Contents

List of Figures	vii
List of Tables	viii
1 Introduction	1
1.1 Background	1
1.2 Related works	3
1.3 Overview	4
2 Monocular Head Pose Estimation	7
2.1 Introduction	8
2.2 Preliminary construction of multilinear face models	13
2.3 Modeling step	17
2.3.1 Incremental construction of the adjustment frame set	17
2.3.2 Error minimization with parameter constraints	19
2.4 Estimation step	20
2.4.1 Head pose estimation using particle filter	21
2.4.2 Finding true feature positions in images	22
2.5 Experimental results	24
2.6 Conclusions	25
3 Incremental Learning for Gaze Estimation	29
3.1 Introduction	30

3.1.1	Model-based methods	31
3.1.2	Appearance-based methods	32
3.1.3	Our approach	34
3.2	Overview	35
3.3	Algorithm	36
3.3.1	Prediction	38
3.3.2	Learning	41
3.4	Implementation	45
3.4.1	Head tracking and eye capturing	45
3.4.2	Blink detection	48
3.5	Experiments	49
3.5.1	Simulation experiments	50
3.5.2	Evaluation in real environments	54
3.6	Conclusions	60
4	Gaze Sensing using Saliency Maps	62
4.1	Introduction	63
4.2	Gaze estimation from saliency maps	65
4.2.1	Saliency extraction	67
4.2.2	Saliency aggregation	70
4.2.3	Estimator construction	75
4.2.4	Gaze estimation	78
4.3	Experimental results	78
4.3.1	Gaze estimation results	80
4.4	Conclusions	86
5	Conclusions	88
5.1	Summary	88
5.2	Contributions	89
5.3	Future Directions	90

<i>CONTENTS</i>	v
Bibliography	93
Publications	109

List of Figures

2.1	System overview.	11
2.2	Data tensor.	14
2.3	Example of facial deformation.	15
2.4	Flow of incremental bundle adjustment.	18
2.5	Finding true feature points.	23
2.6	Estimation results.	26
2.7	Result images.	27
3.1	Learning and prediction flow for proposed framework.	33
3.2	Example of gaze triangulation shown in screen coordinates.	38
3.3	Angular error against prediction reliability.	42
3.4	Process to refine gaze labels.	43
3.5	Results for head tracking and eye capturing.	46
3.6	Blink detection.	49
3.7	System settings for simulation experiments.	50
3.8	Cumulative average angular error in simulation experiments.	53
3.9	Comparison of error in simulation experiments.	54
3.10	Cumulative average angular error in real environments.	56
3.11	Distribution of points clicked by User A, during experiments in real environment.	57
3.12	Comparison between click distribution and average error.	58

3.13	Comparison between click distribution and average error. . . .	59
4.1	Illustration of our method.	64
4.2	Proposed framework.	66
4.3	Process flow of saliency extraction.	68
4.4	Examples of computed saliency maps.	71
4.5	Examples of gaze probability maps and corresponding average eye images.	73
4.6	ROC curves of raw saliency maps and gaze probability maps. .	74
4.7	Experimental setup.	79
4.8	Estimation results of subjects $s_1 \dots s_3$	82
4.9	Estimation results of subjects $s_4 \dots s_6$	83
4.10	Spatial distribution of estimation errors in the display coordi- nate.	85
4.11	Average saliency map and spatial histogram of gaze points. . .	86

List of Tables

2.1	Comparison of estimation errors.	28
3.1	Results of simulation experiments.	51
3.2	Results of experiments in real environment.	55
3.3	Results of experiments in real environment with ground-truth gaze points.	59
4.1	Combinations of video clips A to D and test subjects s_1 to s_6	79
4.2	Average error for each video clip.	84
4.3	Average error for each subject person.	84

Chapter 1

Introduction

1.1 Background

A major goal of many researchers over the last several decades has been to develop a flexible computer system in which the states of the user are taken into account. Based on the physical and mental states of the user, computers can provide richer information adapted for the user, and system developers can design better human-computer interactions. This will enable a user to interact naturally with the computer without having to express his/her states explicitly. The need for such human-centric systems is becoming more and more urgent in this age of ubiquitous computing with the increasing number of applications and devices [PPNH06].

One of the most important factors in sensing a person's states is the information that can be obtained from the person's face. Head pose and gaze direction are particularly important cues for inferring the focus of attention. By determining the object or area at which the user is looking, a computer system can help the user deal with the large amount of information that surrounds him or her. It can be argued that there are two kinds of benefits of inferring the focus of attention, *diagnostic* and *interactive* [Duc02].

The focus of attention gives useful insight into a person's behavior in various types of environment. For example, many studies on estimating a driver's head pose and gaze direction have been conducted [JY02, SSdVL03] as part of efforts to develop safe and intelligent driving systems. The intention is to monitor the driver's focus of attention, level of concentration, and degree of fatigue. Gaze tracking studies in the area of marketing and advertising [Loh97, RRS⁺01], which have been conducted over a long period of time, have focused on investigating information layouts that attract more attention. Designers and advertisers then use the findings to present information efficiently. Similar diagnostic studies have more recently been conducted for computer-related applications, including graphical user interfaces [BADM99], websites [JH02], and Internet search engines [GJG04, LHB⁺08].

Another possible application of sensing the focus of attention is as input for interactive systems. The proposed interactive applications include text input [MR02, WM02], zooming/scrolling [SB90, KSK01], and user interface enhancement [AOS05, KPW07]. In these applications, head pose and gaze direction are used as gesture input for personal computers. As an advanced concept, Attentive User Interface (AUI) [HKPH03, VSCM06] that uses knowledge about user's attention as an indirect input have been advocated in recent years. Although there is a lot of information with which a user can interact, the number of objects to which a user can pay attention is limited. It is thus important to design interfaces that do not interrupt the user's attention and that augment the user's limited resources.

These ideas can be applied to a wide variety of environments, from real world to cyberspace, in a small display. Hence, low-cost and convenient methods for sensing head pose and gaze point are needed to maximize the application of these ideas.

1.2 Related works

One way to measure head pose and gaze point is using contact sensors. Magnetic [RBSJ79] and inertial [Fox96] sensors can be used to track the head pose – signal sources or sensor devices are attached directly to the user’s head. Contact lenses, scleral coils, and electrooculography (EOG) sensors are often used for gaze tracking [Duc07], and camera-based head-mounted systems [BP04, LBP06] are a potent solution in some cases. Though such intrusive sensing methods have advantages like higher accuracy, they force the user to wear special equipment, and they can be used only in specific environments. Compared to these methods, camera-based remote sensing of head pose and gaze point has a great advantage – it can easily be used in casual environments without restricting the user’s freedom of movement. However, although many such methods have been proposed [MCT08a, HJ09], remote sensing techniques still have technical limitations.

For real-time head pose estimation, it is still not easy to estimate non-rigid head motion accurately with only a monocular camera. With a stereo camera [MZ00, YZ02, MRCD02, OS05, ZCSC07], 3D head position can be estimated directly from depth cues. In contrast, there is ambiguity between shape and motion in the monocular cases, so some sort of prior knowledge is needed to resolve the ambiguity. Three-dimensional deformable facial shape models are often used to achieve monocular head pose estimation [GBG01, MBB05, XBMK04, ZJ06]. When this approach is applied to a generic, person-independent case, the models need to be built so that they can describe any facial shape [GMB05, ZHL06, ZWT08]. Since there is inevitably ambiguous deformations like scale change, 3D pose estimation is negatively affected. Most previous work was unable to handle this ambiguity without preliminary customization of the model.

Building a convenient gaze estimator that uses only a monocular cam-

era is even more challenging. The most common approach to gaze estimation is using explicit geometric 3D models of the eyeball. Pupil-glint vectors [HWJM⁺89, Jac90] are often used in this approach to estimate the gaze direction. With a monocular camera [IBMK04, YUYA08], however, such detailed features are hard to extract without additional light sources. Another possible approach to gaze estimation is using eye images as direct features [BP94, XMS98, TKA02, WBC06]. Eye images are easier to capture with only a monocular camera, and gaze points can be estimated using non-linear regression methods. However, a lot of calibration data is needed to learn the mapping between the eye images and gaze points. Especially when there are changes in the head pose, this results in an unrealistically long calibration time.

The most important advantage of a monocular sensing system is that it can be used with existing cameras, like desktop webcams. Cameras to capture the user's face are being attached to more and more devices to meet the increasing demand for video chat systems, for example. Using such commonly available devices will help avoid creating a new digital divide caused by the use of expensive hardware. More importantly, it will become easier to provide attention-based application systems via the Internet. As summarized under the term *cloud computing* [Hay08], the dominant software platform is beginning to shift from local PCs to the Internet, and hardware dependency is a severe disadvantage under this condition.

1.3 Overview

In this thesis, we focused on developing a head pose and gaze estimation system that has minimal requirements for use. More specifically, our goal was an accurate estimation system that requires the use of only a monocular camera and that does not require customized setup or calibration.

In Chapter 2, we describe a method for estimating head pose in real time. A monocular method is proposed for tracking the user’s face and facial actions using a multilinear face model that treats interpersonal and intrapersonal shape variations separately. This method integrates two frameworks: particle filter-based tracking for time-dependent facial action and pose estimation and incremental bundle adjustment for person-dependent shape estimation. Use of this unique combination in conjunction with the multilinear face model is the key to tracking the face and facial actions of an arbitrary person in real time without having to use a pre-learned individual face model.

Chapter 3 describes a method for estimating gaze in real time using the head pose estimator described in Chapter 2. Unconstrained gaze estimation is done using an online learning algorithm that allows free head movement in a casual desktop environment. The key assumption is that a user gazes at the cursor position whenever s/he presses the mouse button. The user’s eye images and 3D head poses are continuously captured with a monocular camera. By using the clicked positions as exemplars of gaze positions, the system collects learning samples for estimating gazes while the user uses the PC, unaware of the system’s activities. The samples are adaptively clustered in accordance with the head pose, and the estimated parameters are incrementally updated. In this way, our method avoids lengthy calibration prior to use of the gaze estimator.

One of the drawbacks of the above approach is that it cannot be applied to cases without user interaction. In Chapter 4, a gaze estimation approach that overcomes this drawback is described. We use a calibration-free gaze sensing method that uses visual saliency maps of video frames computed in a bottom-up manner. By relating the saliency maps to the appearances of the eyes of a person watching the video frames, our method automatically constructs a gaze estimator. For efficient identification of the gaze point, we aggregate the maps to build a probability distribution of the points. A

mapping is established between the eye images and gaze points by Gaussian process regression.

Using these methods will make head pose and gaze estimation substantially more practical by reducing installation and setup costs. Chapter 5 concludes the thesis and describes possible future directions.

Chapter 2

Person-Independent Monocular Head Pose Estimation

Direction of the human head tells us a rough, but meaningful information about what and where the person is looking at. Also, accurate 3D head position is usually required for gaze point estimation. Since there often happen changes in facial shapes, it is required to deal with facial actions for robust 3D head pose tracking.

This chapter presents a monocular method of tracking faces and facial actions using a multilinear face model that treats two components of facial shape variations separately: shape variation between people and variation caused by different facial actions such as facial expressions. The proposed method is created by integrating two different frameworks: particle filter-based tracking for time-dependent facial action and pose estimation and incremental bundle adjustment for person-dependent shape estimation. This unique combination together with multilinear face models is the key to tracking faces and facial actions of arbitrary people in real time with no pre-learned individual face models.

2.1 Introduction

Real-time face and facial action tracking is a key component of applications in various fields including human-computer interactions, video surveillance, and intelligent transport systems. Techniques suited to such applications must be able to estimate 3D face poses and facial actions correctly using a single camera even when large facial shape deformations due to different facial expressions are present. To be used practically, the techniques must be able to work with arbitrary people without preliminary preparations, e.g., building a face model for each person. The aim of this study is to develop a novel tracking technique that satisfies these two requirements. Therefore, we have developed a person-independent monocular tracking technique for face and facial actions.

Head pose estimation methods can be roughly categorized into two categories: appearance-based and model-based. AAM (Active Appearance Model [CET01]) is a typical example of appearance-based methods that use facial textures as features. In AAMs, 2D facial shapes and textures are described as linear combinations of basis vectors. Matthews *et al.* [MB04] employed AAMs for real-time estimation of facial orientations. For 3D head pose estimation, there have been proposed some appearance-based methods combined with simple shape models such as ellipsoid [BEP96], 2D planar [HB98] and cylinder [LCS99]. Although appearance-based methods have the advantage that they are relatively more robust to image noise, accurate 3D head poses and facial shapes cannot be achieved only with simple appearance models.

In contrast, model-based methods that use geometric facial shape model have a potential for accurate 3D estimation. Some methods simply use rigid facial shape models [GC96, VLF04, THT06, MCT08b, Gri09], however, rigid models cannot handle changes in facial shapes. Hence, many deformable model-based methods have been proposed for face and facial

action tracking. Using a linear face model typically obtained by principal component analysis (PCA), these methods estimate the pose and coefficients of deformation bases of a face. However, most previous methods [GBG01, MBB05, XBMK04, ZJ06] used face models that were specially created for each person before estimation. Requiring preparation of person-specific face models is often too restrictive for practical applications. In order to use person-specific models without preliminary model preparation, Oka *et al.* proposed a multi-view method for simultaneously modeling faces and estimating motion [OS05]. However, their method was still too costly in terms of system installation, using multiple cameras that need to be accurately calibrated beforehand.

Meanwhile, another approach can be taken using a generic face model that represents facial shape deformation across multiple people with one parameter set [GMB05, ZHL06, ZWT08]. Zhu *et al.* [ZHL06] used an AAM-based generic face model to estimate 3D head pose and facial actions in real time. Zhang *et al.* [ZWT08] also used a 3D generic face model built from 3D face database and synthetic face samples. However, in their work, experimental evaluation were done only on fitting accuracy, i.e., how well the models fit the faces in image sequences on 2D image coordinates. No quantitative evaluation was performed on their 3D head pose estimation results. Especially in the case of 3D models, generic models inescapably contain a deformation factor that normally does not happen for a single person, such as scaling. These factors are hard to distinguish from the head pose, thus decreasing the tracking accuracy. Gross *et al.* presented an interesting empirical study on performance comparison between generic and person-specific models that were not 3D models but 2D active appearance models [GMB05]. It was reported that the use of generic models often resulted in a much worse rate of convergence in model parameter estimation.

To cope with this problem, some methods used 3D face models with two

separate sets of parameters (a set of *shape parameters* for interpersonal deformation and a set of *action parameters* for intrapersonal deformation). The use of such models limits the required number of parameters for each set without degrading the expressiveness of the model. In addition, these two sets of parameters with different behavior can be treated separately. Dornaika *et al.* used a model with separate sets of parameters in real-time face tracking [DD06]. Their method estimates time-dependent action parameters sequentially. However, shape parameters for person-dependent facial shape variations are set manually, and their method does not adjust shape parameters during the tracking process. Using a similar linear model, DeCarlo *et al.* [DM02] used tracking residuals from model-based optical flow to adjust all of the parameters, including shape parameters. Their method was computationally too costly to be executed in real time and, moreover, their method does not have global optimization scheme for shape parameters. Wang *et al.* [WL08] used a similar model whose action parameters are modeled by locally linear embedding. Since it was meant to be applied to a single image, their method cannot be executed in real time and also does not have the perspective of using multiple frames. Vlastic *et al.* [VBPP05] used a multilinear face model that describes interpersonal and intrapersonal deformations separately. They estimate shape parameters in a global way using multiple frames. However, the purpose of their method was to capture facial expression from a full video segment, so it is not clear how their method can be incrementalized and extended to real-time estimation.

As stated above, there is no method which is capable of estimating both shape and action parameters in real-time. In contrast, our method executes shape adjustment simultaneously with real-time non-rigid head pose tracking, by using a model-based bundle adjustment with a multilinear face model.

As shown in Figure 2.1, our method consists of two steps. The first step, called the *Estimation Step*, estimates action parameters, *i.e.*, intrapersonal

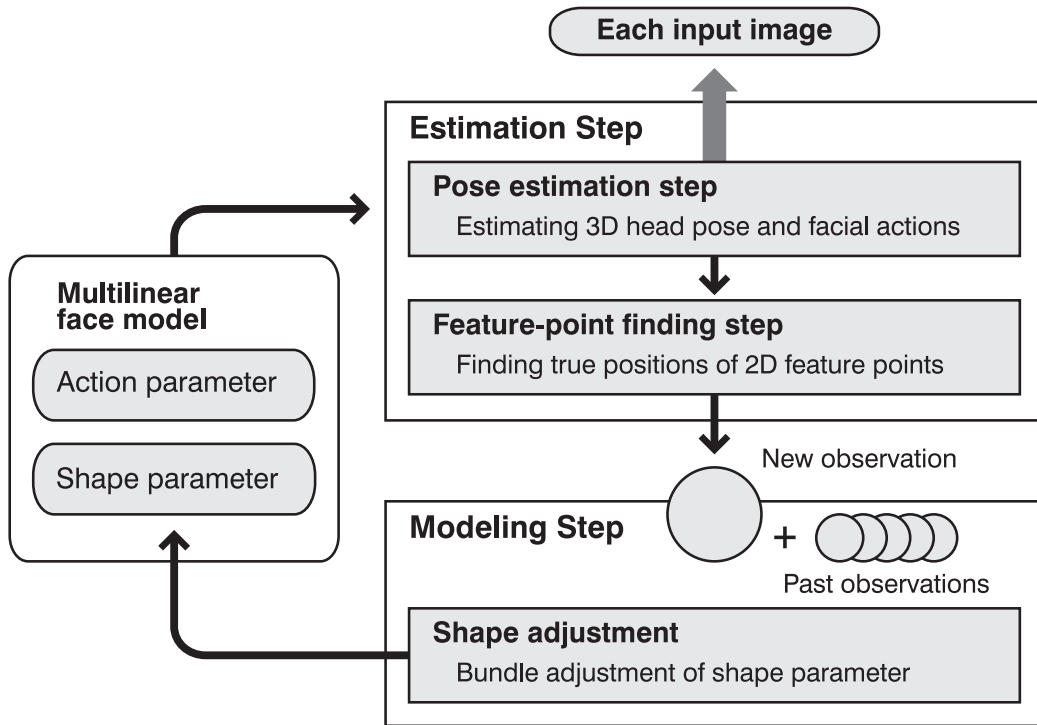


Figure 2.1: System overview. The *Estimation Step* estimates action parameters, *i.e.*, intrapersonal deformation, as well as the person’s 3D head pose for each input frame by using a particle filter. The *Modeling Step* incrementally refines shape parameters, *i.e.*, interpersonal deformation, by model-based bundle adjustment based on 2D facial feature positions obtained from the *Estimation Step*.

deformation, as well as the person’s 3D head pose for each input frame by using a particle filter. It also finds correct 2D positions of facial feature points in the image. This step enables a head pose and facial action tracking that is robust to partial occlusion or depth-directional movement.

The second step, called the *Modeling Step*, incrementally refines shape parameters, *i.e.*, interpersonal deformation, by model-based bundle adjustment based on 2D facial feature positions obtained from the *Estimation Step*. This step enables a stable adjustment of shape parameters that includes factors indistinguishable from head pose. Updated shape parameters are then used in the succeeding *Estimation Step*. In this way, our method enables progressive refinement of the estimation accuracy and personal customization of the face model.

This unique combination of particle filter-based tracking and incremental bundle adjustment enables monocular estimation of non-rigid 3D facial motion without preliminary learning of face models tailored for each person. As far as we know, this is the first research to propose a method using this approach.

The rest of this chapter is organized as follows. In Section 2.2, we begin by describing how multi-linear facial models with separate parameter sets are constructed prior to tracking. Then we describe the two steps in our method; the *Modeling Step* in Section 3 and the *Estimation Step* in Section 4. We present our experimental results in Section 2.5. Finally, we present concluding remarks in Section 2.6.

2.2 Preliminary construction of multilinear face models

In this section, we describe how a multilinear face model with shape and action parameters is prepared by using N-mode singular value decomposition (SVD) [VBPP05] prior to tracking.

A person’s face is represented in terms of its shape and appearance. More specifically, the face’s shape is represented as a $3K$ -dimensional shape vector \mathbf{M} composed of 3D coordinates of K feature points¹. These are defined in the local coordinate system fixed to the person’s head. The appearance of the face is modeled as appearances of the feature points, which are registered as image templates automatically at the beginning of each tracking.

A multilinear face model that represents facial shapes is built from a data tensor \mathcal{T} that varies with people’s identity and facial expressions (Figure 2.2). The first mode (noted as feature points in the figure) corresponds to each shape vector \mathbf{M} , while the second (shape) and the third (action) modes correspond to identity and facial expression, respectively. The data is arranged so that shape vectors of the same person making different facial expressions are aligned in a slice along the second mode, and shape vectors of different persons making the same expressions are aligned in a slice along the third mode.

Based on N-mode SVD [DLDMV00], the data tensor \mathcal{T} can be expressed as a mode product of an orthonormal matrix \mathbf{U}_i of the i th mode and a core tensor \mathcal{C} :

$$\begin{aligned}\mathcal{T} &= \mathcal{C} \times_{\text{feature}} \mathbf{U}_{\text{feature}} \times_{\text{shape}} \mathbf{U}_{\text{shape}} \times_{\text{action}} \mathbf{U}_{\text{action}} \\ &= \mathcal{M} \times_{\text{shape}} \mathbf{U}_{\text{shape}} \times_{\text{action}} \mathbf{U}_{\text{action}}.\end{aligned}\tag{2.1}$$

¹In this study, K is set to 10. Those feature points are the inner and outer corners of both eyes, both corners of the mouth, both nostrils, and the inner corner of both brows (indicated with plus signs in Figure 2.3).

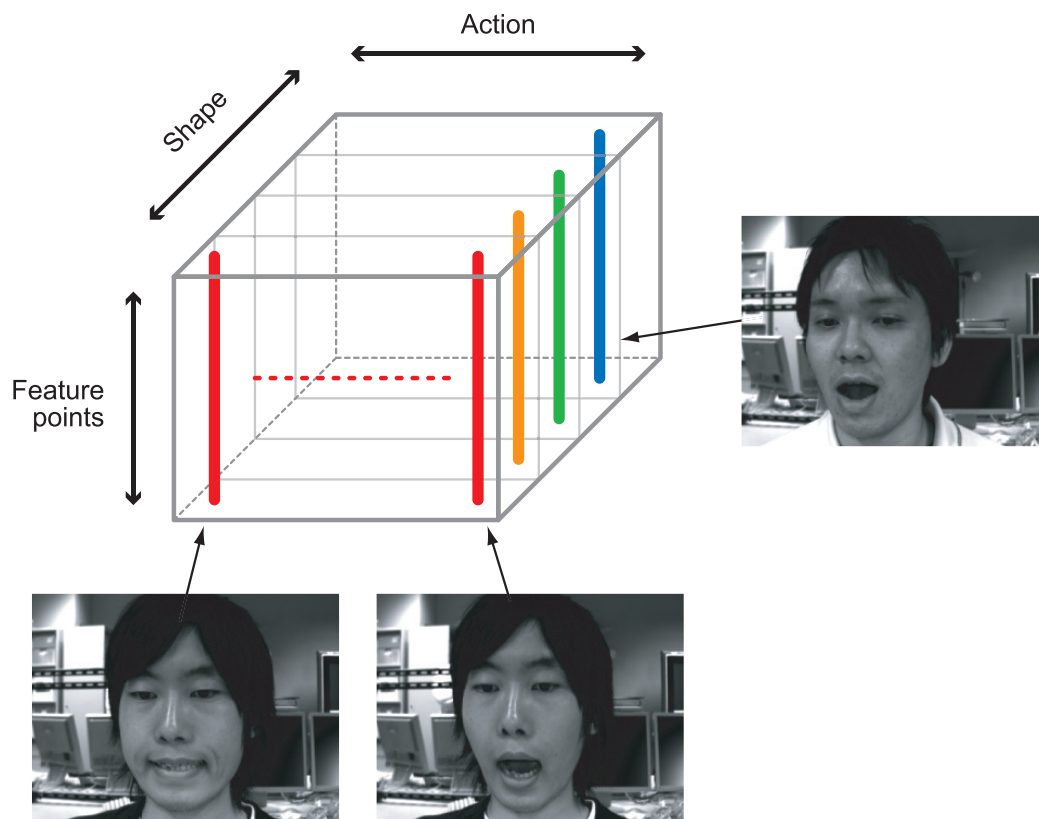


Figure 2.2: Data tensor that varies with people’s identity and facial expressions. The first mode (noted as feature points in the figure) corresponds to each shape vector \mathbf{M} , while the second (shape) and the third (action) modes correspond to identity and facial expression, respectively. The data is arranged so that shape vectors of the same person making different facial expressions are aligned in a slice along the second mode, and shape vectors of different persons making the same expressions are aligned in a slice along the third mode.

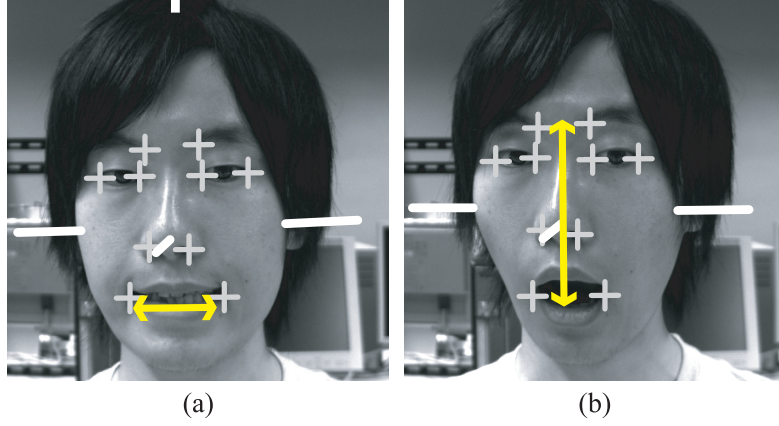


Figure 2.3: Example of facial deformation. To construct the data tensor \mathcal{T} , S people were asked to move their faces in 2 different ways: horizontally move the corners of their mouth, and vertically move their mouths and eyebrows. Plus signs indicate feature points used in our facial shape model.

where the notation \times_i indicates a mode- i product between a tensor and a matrix, which multiplies every mode- i vector of the tensor by the matrix. Columns of each \mathbf{U}_i correspond to orthonormal basis of the mode- i space, and thus the model tensor \mathcal{M} contains basis vectors of the $3K$ -dimensional face vector space.

Moreover, an approximated representation of \mathcal{T} is obtained with the truncated basis of action and shape spaces:

$$\mathcal{T} \approx \check{\mathcal{M}} \times_{\text{shape}} \check{\mathbf{U}}_{\text{shape}} \times_{\text{action}} \check{\mathbf{U}}_{\text{action}}. \quad (2.2)$$

Using this approximated model tensor, we can generate an arbitrary face vector \mathbf{M} using shape and action parameters defined as coefficient vectors of $\check{\mathcal{M}}$.

To construct the data tensor \mathcal{T} , we first need to prepare shape vectors for different persons moving their faces in different ways. In this study, we used a multiview-based face and facial action tracking technique [OS05] to obtain

shape vectors. While K facial features were being automatically tracked, S people were asked to move their faces in 2 different ways: horizontally move the corners of their mouth, and vertically move their mouths and eyebrows (Figure 2.3). Then, 5 intermediate facial shapes were chosen for each facial action (from beginning to completion of the action) for a total of $A = 10$ shape vectors for each person.

This gives us $S \times A$ samples of face shape. After calculating and subtracting mean shape $\bar{\mathbf{M}}$, we construct a data tensor $\mathcal{T} \in \mathbb{R}^{3K \times S \times A}$. By calculating the model tensor $\check{\mathcal{M}}$ with approximated shape ($S \rightarrow S'$) and action ($A \rightarrow A'$) spaces as Eq. (2.2), we can describe an arbitrary face vector \mathbf{M} using a shape parameter $\mathbf{s} \in \mathbb{R}^{S'}$, an action parameter $\mathbf{a} \in \mathbb{R}^{A'}$ and the mean shape $\bar{\mathbf{M}}$:

$$\mathbf{M} = \bar{\mathbf{M}} + \check{\mathcal{M}} \times_{\text{shape}} \mathbf{s}^{\text{T}} \times_{\text{action}} \mathbf{a}^{\text{T}}. \quad (2.3)$$

Here, each row of $\check{\mathbf{U}}_{\text{shape}} = (\check{\mathbf{s}}_1, \dots, \check{\mathbf{s}}_S)^{\text{T}}$ and $\check{\mathbf{U}}_{\text{action}} = (\check{\mathbf{a}}_1, \dots, \check{\mathbf{a}}_A)^{\text{T}}$ in Eq. (2.2) is a parameter vector corresponding to each of the $A \times S$ data. We calculate the mean vector $\bar{\mathbf{s}}$ and the vector $\boldsymbol{\sigma}_{\mathbf{s}}$ composed of standard deviations of elements of $\{\check{\mathbf{s}}_i\}$, and the mean vector $\bar{\mathbf{a}}$ and the vector $\boldsymbol{\sigma}_{\mathbf{a}}$ composed of standard deviations of elements of $\{\check{\mathbf{a}}_i\}$. These four vectors are later used to determine the constraint of a bundle adjustment (Section 2.3), and the diffusion and weighting process of a particle filter (Section 2.4).

This model enables us to describe any facial state of any person with a person-dependent shape vector \mathbf{s} , a time-dependent action vector \mathbf{a} and a head pose vector \mathbf{p} defined as a translation and a rotation from the world coordinate system to the model coordinate system. In the following sections, we explain the details of our real-time face and facial action tracking method using this face model.

2.3 Modeling step: estimation of interpersonal shape variations

In this section, we describe the *Modeling Step* of our method of incrementally adjusting shape parameter vector \mathbf{s} , which represents interpersonal facial shape variation, using model-based bundle adjustment.

Bundle adjustment is a maximum likelihood estimation method that optimizes parameters in 3D space by minimizing the 2D reprojection error in multiple images. In the context of facial shape estimation, it is used to model rigid faces [XWT⁺05], estimate rigid head motions in real time [VLF04], and adjust the shapes of deformable face models acquired from a non-rigid factorization method [DBSAM04].

In this research, we used model-based bundle adjustment to incrementally adjust the shape parameter vector \mathbf{s} of a multilinear face model. We introduce two modifications to stabilize estimation of shape parameters. One is an incremental construction of an adjustment frame set based on the result from the *Estimation Step* with a particle filter. The other is the use of parameter constraints determined on the basis of the distribution of the shape parameter and estimated pose and action parameters. We first explain how to choose a set of observation frames and then explain model-based bundle adjustment with parameter constraints.

2.3.1 Incremental construction of the adjustment frame set

Using the face model presented in Section 2.2, the bundle adjustment problem is formulated as follows. First, we calculate the face shape vector \mathbf{M}_t from Eq. (2.3). Then K feature points in shape vector \mathbf{M}_t are projected onto the

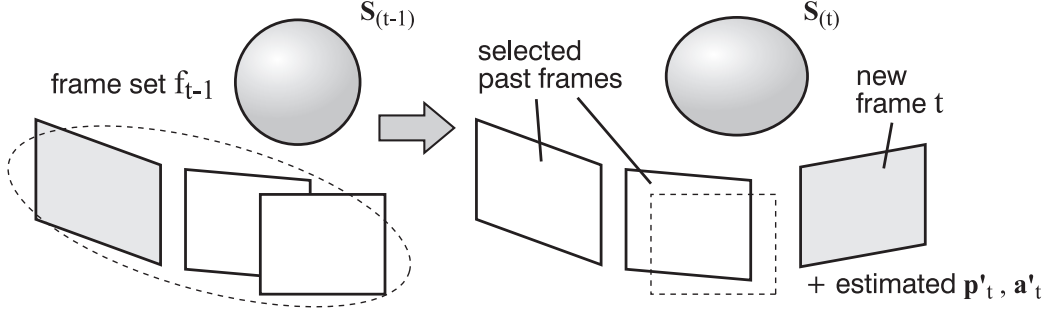


Figure 2.4: Flow of incremental bundle adjustment. Our method generates the frames of this frame set f_t one at a time, by replacing one frame of the previous set f_{t-1} with a new frame.

image plane as:

$$\mathbf{m}_t = \mathcal{P}(\mathbf{p}_t, \mathbf{M}_t(\mathbf{a}_t, \mathbf{s})), \quad (2.4)$$

where \mathcal{P} is a projection function given by camera parameters that are obtained prior to tracking, and \mathbf{m}_t is a $2K$ -dimensional vector that consists of 2D coordinates of K projected feature points.

Let $\hat{\mathbf{m}}_t$ be a vector that represents the true 2D coordinates of K feature points. This $2K$ -dimensional vector $\hat{\mathbf{m}}_t$ is obtained in the *Estimation Step* as explained later in Section 2.4.2. Finally, we can define an error function for the sum of the reprojection errors over a set of observation frames as:

$$F_t = \sum_{i \in f_t} D(\hat{\mathbf{m}}_i, \mathbf{m}_i(\mathbf{p}_i, \mathbf{a}_i, \mathbf{s}))^2, \quad (2.5)$$

where f_t means a set of n observation frames used in the bundle adjustment at time t , as illustrated in Figure 2.4.

Our method generates the frames of this frame set f_t one at a time, by replacing one frame of the previous set f_{t-1} with a new frame. For the new frame t , pose \mathbf{p}'_t and action \mathbf{a}'_t estimated in the *Estimation Step* are assigned as initial values for the minimization of F_t . Meanwhile, selected $n - 1$ frames

are initialized from previous minimization results of F_{t-1} and adjusted on an ongoing basis.

Zhang *et al.* [ZS03] used a similar approach of updating a set of observation frames by replacing the oldest frame with a new incoming frame. However, it is often the case in real-time tracking that object appearances do not change much between consecutive frames, and, as a result, depth ambiguities cannot be resolved reliably with bundle adjustment. This problem is avoided in our method by maximizing the variation of poses in the adjustment frame set. More specifically, we choose the frame set with the widest pose variance at the initial state of the minimization, from among all n frame combinations possible at the time. By repeating this selection scheme, the pose variation in the frame set increases as the tracking proceeds.

2.3.2 Error minimization with parameter constraints

Next, we describe in detail the minimization procedure of F_t (Eq. (2.5)) with parameter constraints, which is meant to stabilize the adjustment process. F_t is minimized using a Levenberg-Marquardt method under the parameter constraints [Lou]:

$$\min_{\{\mathbf{p}_i\}, \{\mathbf{a}_i\}, \mathbf{s}} F_t, \quad \mathbf{p}_i \in \mathbf{C}_{p_i}, \quad \mathbf{a}_i \in \mathbf{C}_{a_i}, \quad \mathbf{s} \in \mathbf{C}_s, \quad (2.6)$$

where \mathbf{C}_{p_i} , \mathbf{C}_{a_i} and \mathbf{C}_s denotes the constraints on each parameter.

As mentioned above, initial pose $\hat{\mathbf{p}}_t$ and action parameter $\hat{\mathbf{a}}_t$ for the minimization are estimated almost exactly, based on the value obtained in the *Estimation Step*. Accordingly, tight constraints \mathbf{C}_{p_i} and \mathbf{C}_{a_i} are imposed such that only small changes are allowed in each iteration:

$$\mathbf{C}_{p_i} = \{\mathbf{p}_i \mid \hat{\mathbf{p}}_i - \boldsymbol{\lambda}_p \leq \mathbf{p}_i \leq \hat{\mathbf{p}}_i + \boldsymbol{\lambda}_p\}, \quad (2.7)$$

where $\boldsymbol{\lambda}_p$ is a constant vector that denotes the adjustment range. The action constraint, \mathbf{C}_{a_i} , is set in the same way. Currently, $\boldsymbol{\lambda}_a$ and $\boldsymbol{\lambda}_p$ are determined

empirically.

In contrast, a relatively weak constraint is imposed on shape parameter \mathbf{s} based on the vector of standard deviations σ_s from Section 2.2:

$$\mathbf{C}_s = \{\mathbf{s} \mid \bar{\mathbf{s}} - 2\boldsymbol{\sigma}_s \leq \mathbf{s} \leq \bar{\mathbf{s}} + 2\boldsymbol{\sigma}_s\}. \quad (2.8)$$

This allows shape parameters to be adjusted to the shape of the person's face smoothly while excessive shape deformations are prohibited.

Finally, the shape parameter $\mathbf{s}_{(t)}$ for the next *Estimation Step* is calculated as the mean of the estimation results up to the present time:

$$\mathbf{s}_{(t)} = \frac{t-1}{t}\mathbf{s}_{(t-1)} + \frac{1}{t}\mathbf{s}', \quad (2.9)$$

where \mathbf{s}' denotes the result of estimation at time t , calculated from the process mentioned above. Eq. (2.9) reduces the influence of short-term fluctuation in the adjustment.

2.4 Estimation step: estimation of head pose and facial actions

In this section we describe the *Estimation Step* (Figure 2.1). It is important to note that time-varying action and pose parameters cannot be estimated properly with the model-based bundle adjustment process of the *Modeling Step* for several reasons. First, the estimation result tends to jitter, especially in the depth direction. Second, 2D positions of feature points required for the bundle adjustment cannot be obtained stably with simple 2D tracking or detection. Last, if some of the feature points are not observed, the pose and action parameters cannot be estimated correctly. To solve these problems, we use a particle filter to estimate pose and action parameters based on a 3D model-based motion prediction.

As shown in Figure 2.1, the *Estimation Step* consists of two components: the *Pose estimation step*, which estimates pose \mathbf{p}_t and action \mathbf{a}_t , and the *Feature-point finding step*, which calculates the true 2D positions of feature points $\hat{\mathbf{m}}_t$ which are used as the observation vector in Eq. (2.5) in the *Modeling Step*. In the following sections, we first explain the *Pose estimation step*, and then explain the *Feature-point finding step*.

2.4.1 Head pose estimation using particle filter

To estimate facial action, the multilinear model in Eq. (2.3) is rewritten as a linear deformation model with the shape parameter $\mathbf{s}_{(t-1)}$ calculated in the previous frame:

$$\mathbf{M}_t = \bar{\mathbf{M}} + \mathcal{M}_t \mathbf{a}_t \quad (\mathcal{M}_t = \check{\mathcal{M}} \times_{\text{shape}} \mathbf{s}_{(t-1)}^T). \quad (2.10)$$

Using this model, we estimate a $(6 + A')$ dimensional state vector $\mathbf{x}_t = (\mathbf{p}_t^T, \mathbf{a}_t^T)^T$ at frame t . The sample set $\{(\mathbf{u}_t^{(i)}; \pi_t^{(i)})\}$ for the particle filter in our method consists of N discrete samples $\mathbf{u}_t^{(i)}$ in the $(6 + A')$ dimensional state space and of associated weights $\pi_t^{(i)}$.

To generate N new samples at each time t , we define a uniform linear motion model as follows:

$$\mathbf{u}_t^{(i)} = \mathbf{u}'_{t-1} + \tau \mathbf{v}_{t-1} + \boldsymbol{\omega}, \quad (2.11)$$

where \mathbf{u}'_{t-1} is a chosen sample from the previous sample set, τ is the interval between frames, and \mathbf{v}_{t-1} is the velocity of the state vector \mathbf{x} calculated at the previous frame $t - 1$. Note that the elements of \mathbf{v}_{t-1} corresponding to the action parameter \mathbf{a}_t are set to 0, because \mathbf{a}_t does not always match the assumption of uniform linear motion.

$\boldsymbol{\omega}$ is a system noise that affects the diffusion property, and each element of $\boldsymbol{\omega}$ is a Gaussian noise with a zero mean and a uniquely defined variance. The elements corresponding to the head pose are adaptively controlled depending

on velocity [OS05]. Meanwhile, the standard deviation of the Gaussian noise for the other elements corresponding to the action parameter is set to $\kappa\sigma_a$ based on the parameter distribution calculated in Section 2.2. Here, κ is empirically set to 0.2.

Weight $\pi_t^{(i)}$ of each sample $\mathbf{u}_t^{(i)}$ is calculated as:

$$\pi_t^{(i)} \propto \exp\left(-\frac{\left(K - \mathcal{N}(\mathbf{u}_t^{(i)})\right)^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{1}{2} \sum_{b=1}^{A'} \left(\frac{a_{t,b}^{(i)} - \bar{a}_b}{\varsigma_b}\right)^2\right) \quad (2.12)$$

where $\mathcal{N}(\mathbf{u}_t^{(i)})$ is a sum of the normalized correlation score for all K feature points based on template image T , which has a value between $-K$ and K . The first term of Eq. (2.12) is a Gaussian function evaluating $\mathcal{N}(\mathbf{u}_t^{(i)})$, and the standard deviation σ is set to 1.0. The second term is an evaluation function for the action parameter $\mathbf{a}_t^{(i)}$, which prevents excessive face deformation. Here, $a_{t,b}^{(i)}$, \bar{a}_b and ς_b is the b -th element of $\mathbf{a}_t^{(i)}$, $\bar{\mathbf{a}}$ and σ_a , respectively.

After the calculation, each weight $\pi_t^{(i)}$ is normalized so that the sum is equal to 1. Eventually, the current state vector \mathbf{x}_t is computed as a weighted average of all samples.

Note that the initial state vector \mathbf{x}_0 is calculated from the bundle adjustment. After a person's face and K feature points are automatically detected over n frames (using OKAO Vision library developed by OMRON Corporation), all parameters are initialized by minimizing Eq. (2.6). In this case, we use predefined values as the start point of the iteration: a head pose facing the center of the camera and mean parameters $\bar{\mathbf{a}}$ and $\bar{\mathbf{s}}$.

2.4.2 Finding true feature positions in images

Next, we describe the *Feature-point finding step* in detail. The 2D positions \mathbf{m}'_t of the estimated feature points can be calculated from the estimated state vector \mathbf{x}_t and the projection function \mathcal{P} (Eq. 2.4). However, if the adjustment of the shape parameter is not done properly, the estimated positions do not

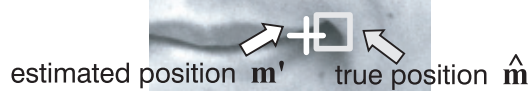


Figure 2.5: Finding true feature points. We find the true 2D positions $\hat{\mathbf{m}}_t$ around the estimated positions \mathbf{m}'_t .

always correspond with the true positions (as shown in Figure 2.5). In this step, we find the true 2D positions $\hat{\mathbf{m}}_t$ around the estimated positions \mathbf{m}'_t .

We define the following energy function E_t similar to the one used in Gokturk *et al.* [GBG01], and calculate the difference $d\hat{\mathbf{m}} = \hat{\mathbf{m}}_t - \hat{\mathbf{m}}_{t-1}$ by successively minimizing it.

$$E_t = \sum_{\text{ROI}} \left\{ \rho \left\| \mathbf{I}_t(\hat{\mathbf{m}}_t) - \mathbf{I}_{t-1}(\hat{\mathbf{m}}_{t-1}) \right\|^2 + \left\| \mathbf{I}_t(\hat{\mathbf{m}}_t) - \mathbf{I}_1(\hat{\mathbf{m}}_1) \right\|^2 \right\} \quad (2.13)$$

$$+ \epsilon \left\| \hat{\mathbf{m}}_t - \mathbf{m}'_t \right\|^2.$$

The first term of Eq. (2.14) denotes the difference between the appearances of regions of interest (ROIs) around the feature points. $\mathbf{I}_t(\hat{\mathbf{m}}_t) \in \mathbb{R}^K$ is an intensity vector corresponding to $\hat{\mathbf{m}}_t$, whose k th element is the intensity of the input image at the k th 2D position of $\hat{\mathbf{m}}_t$. We use both the difference from the previous image and the difference from the first image, which [GBG01] also uses. This avoids the problem of drift of the calculated feature points. ρ is empirically set to 4, and the size of ROI is 16×16 . In contrast, the second term denotes the geometric difference between \mathbf{m}'_t and $\hat{\mathbf{m}}_t$. Using this term, we find the true positions $\hat{\mathbf{m}}_t$ in the neighboring region of estimated positions \mathbf{m}'_t . ϵ is empirically set to 4000.

By approximating $\hat{\mathbf{I}}_t$ using Taylor expansion, Eq. (2.14) can be written as a function of $d\hat{\mathbf{m}}$ as

$$E_t = \sum_{\text{ROI}} \left\{ \rho \left\| \hat{\mathbf{K}}_t d\hat{\mathbf{m}} + \Delta \mathbf{I} \right\|^2 + \left\| \hat{\mathbf{K}}_t d\hat{\mathbf{m}} + \Delta \mathbf{I}_0 \right\|^2 \right\} \quad (2.14)$$

$$+ \epsilon \left\| d\hat{\mathbf{m}} + \hat{\mathbf{m}}_{t-1} - \mathbf{m}'_t \right\|^2,$$

where

$$\hat{\mathbf{K}}_t = \left. \frac{\partial \mathbf{I}_t}{\partial \hat{\mathbf{m}}_t} \right|_{\hat{\mathbf{m}}_{t-1}}, \quad (2.15)$$

$$\Delta \mathbf{I} = \mathbf{I}_t(\hat{\mathbf{m}}_{t-1}) - \mathbf{I}_{t-1}(\hat{\mathbf{m}}_{t-1}) \quad (2.16)$$

$$\Delta \mathbf{I}_0 = \mathbf{I}_t(\hat{\mathbf{m}}_{t-1}) - \mathbf{I}_1(\hat{\mathbf{m}}_1) \quad (2.17)$$

$\mathbf{I}_t(\hat{\mathbf{m}}_{t-1})$ is an intensity vector corresponding to $\hat{\mathbf{m}}_{t-1}$ in \mathbf{I}_t . By setting $\frac{\partial E_t}{\partial d\hat{\mathbf{m}}} = 0$, $d\hat{\mathbf{m}}$ that minimizes E_t can be derived as

$$d\hat{\mathbf{m}} = -\mathbf{D}^{-1}\mathbf{d}, \quad (2.18)$$

where

$$\mathbf{D} = \sum_{\text{ROI}} \left\{ 2(\rho + 1) \hat{\mathbf{K}}_t^T \hat{\mathbf{K}}_t \right\} + \epsilon \mathbf{E}_{2K \times 2K}, \quad (2.19)$$

$$\mathbf{d} = \sum_{\text{ROI}} \left\{ 2 \hat{\mathbf{K}}_t^T (\rho \Delta \mathbf{I} + \Delta \mathbf{I}_0) \right\} + \epsilon (\hat{\mathbf{m}}_{t-1} - \mathbf{m}'_t). \quad (2.20)$$

$\hat{\mathbf{m}}_t$ is successively updated until $d\hat{\mathbf{m}}$ given by Eq. (2.18) converges.

2.5 Experimental results

We have conducted a number of experiments to evaluate the performance of our method. First, we compared our method with the multiview-based tracking method [OS05]. In addition, to evaluate the effect of the use of the multilinear model and the bundle adjustment, we made another comparison with the particle filter-based estimation result using a generic PCA model with one parameter set.

The face model was built from $S = 26$ persons \times $A = 10$ actions, and the resulting model had $S' = 15$ shape parameters and $A' = 5$ action parameters. The generic model was also built from the same data set using PCA, and had 20 deformation parameters. Note that the target person in the experiment was not included among the 26 persons.

Our tracking system consisted of a Windows-based PC with Intel Core 2 Duo E6700. We captured 60-second long (1800 frames) video sequences from two calibrated BW cameras via IEEE-1394. The image resolution was 640×480 , the size of image templates T was set to 16×16 . A set of 1000 samples was used for particle filtering. $n = 7$ frames were used for the bundle adjustment. The initialization step, with 10 iterations of LM minimization, took approximately 90 [ms], and the overall tracking process, with 5 iterations per frame, took approximately 32 [ms/frame].

Table 2.1 shows the estimation error of our method and the generic model-based method. x , y , and z are the horizontal, vertical, and depth-directional translation, and $roll$, $pitch$, and yaw are the rotation around the z , y , and x axes, respectively. Additionally, Figure 2.6 shows the detailed estimation results and the facial shape estimation error in the model coordinate system. The difference between two monocular estimation methods is evident here. In Figure 2.7, the right and center columns show actual images of the estimation results, and the left column shows these results rendered from a different viewpoint. The whole sequences can be seen on our website ². These results demonstrate that our method is more accurate than the method using the generic PCA model, and favorably compares with stereo estimation.

2.6 Conclusions

In this work, we presented a person-independent monocular method for real-time 3D face and facial action tracking. The key idea of our method is a unique combination of i) particle filter-based tracking for time-dependent pose and facial action estimation and ii) incremental model-based bundle adjustment for person-dependent shape estimation, together with multilinear face models. To our knowledge, this is the first work to achieve fully auto-

²<http://www.hci.iis.u-tokyo.ac.jp/~sugano/research/3d-face-tracking/>

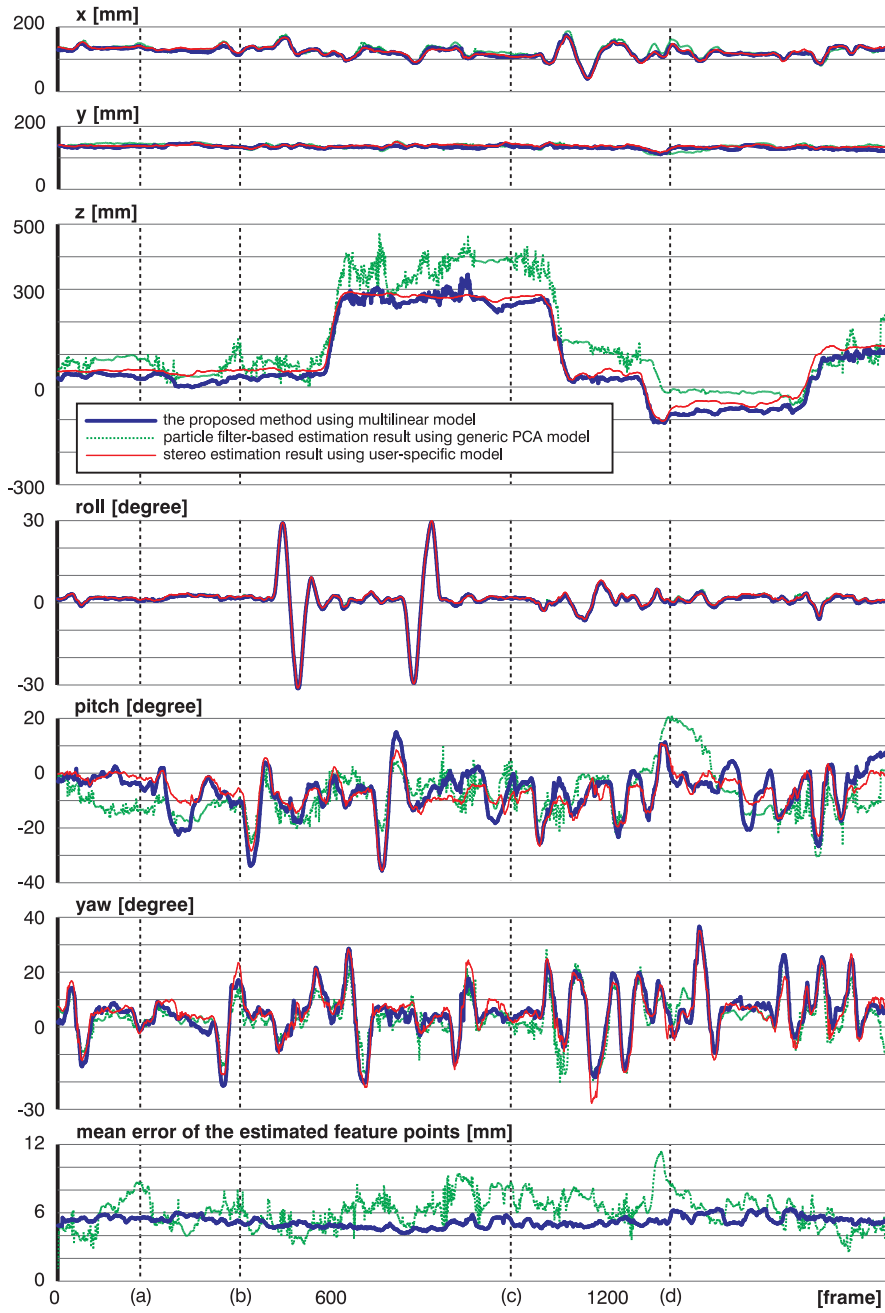


Figure 2.6: Estimation results: x , y , and z are the horizontal, vertical, and depth-directional translation, and $roll$, $pitch$, and yaw are the rotation around the z , y , and x axes, respectively. The bottom graph shows the facial shape estimation error in the model coordinate system.

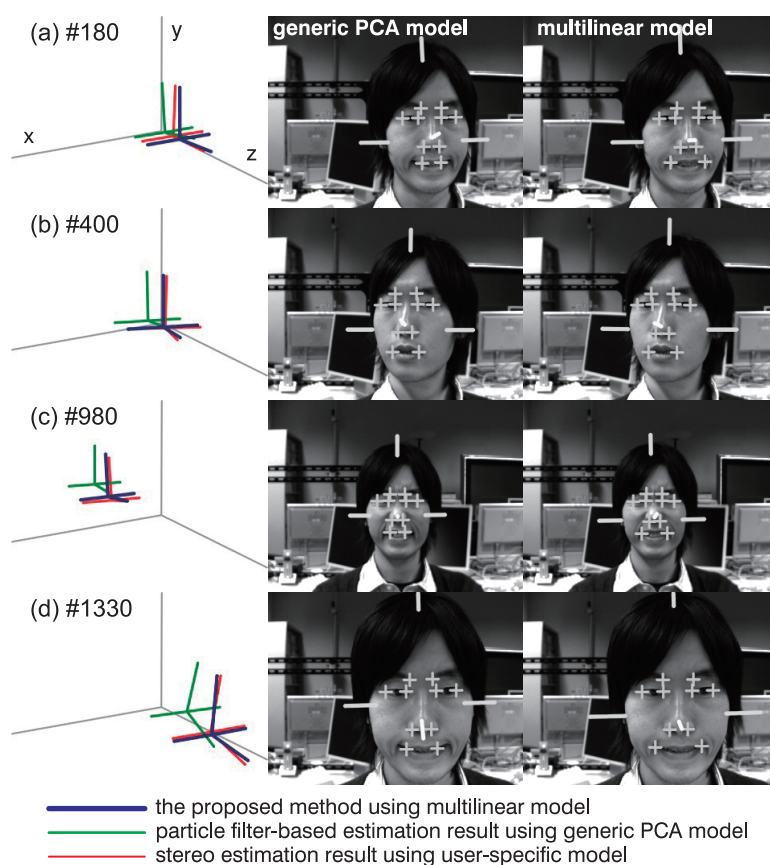


Figure 2.7: Result images: the right column shows actual estimation results of our method using the multilinear model, and the center column shows results of the generic model-based method. The left column shows these results rendered from a different viewpoint.

Table 2.1: Comparison of estimation errors. x , y , and z are the horizontal, vertical, and depth-directional translation, and $roll$, $pitch$, and yaw are the rotation around the z , y , and x axes, respectively.

[mm]	x	y	z	[deg.]	roll	pitch	yaw
Particle filter-based estimation using the generic PCA model							
Mean	6.14	4.71	51.32	Mean	0.34	6.54	3.34
Std. Dev.	4.88	4.09	38.29	Std. Dev.	0.29	4.71	2.73
Our method using the multilinear model							
Mean	3.26	4.37	20.18	Mean	0.41	3.12	2.33
Std. Dev.	2.62	2.83	11.18	Std. Dev.	0.27	2.49	1.98

matic 3D tracking of face and facial actions without preliminary training of person-specific face models. Our experimental results demonstrate that our method performs significantly better than monocular tracking with a generic face model, confirming the effectiveness of our real-time tracking method based on a multilinear face models. In our future work, we are planning to use our tracking method for real-time facial expression analysis.

Chapter 3

Incremental Learning for Gaze Estimation

Although head pose can be a cue for inferring human attention, it is not enough to specify an actual point at which the person is looking. Obviously, information about gaze point is needed to determine focus of attention in a small region.

This chapter presents an unconstrained gaze estimation method using an online learning algorithm. Based on the head pose estimation method described in the previous chapter, it allows free head movement in a casual desktop environment. The key assumption is that a user gazes at a cursor position when the user clicks. The user's eye images and 3D head poses are continuously captured by a monocular camera. By using clicked positions as exemplars of gaze positions, our system collects learning samples for estimating gazes while a user is unconscious of the system while using a PC. The samples are adaptively clustered according to the head pose and estimation parameters are incrementally updated. In this way, our method avoids the lengthy calibration stage prior to using the gaze estimator.

3.1 Introduction

Gaze estimation is a process of detecting what position the eyes are looking at. Because gaze is a key factor in estimating a person's attention, techniques of estimating gazes have been an active research topic in computer vision. Also, a wide range of applications has been proposed in various fields (see [Duc07] for a recent survey). For example, gaze trackers are used to measure people's eye movements in fields such as neuroscience and psychology. In terms of marketing, it can also play a significant role in capturing user's interests on Websites and in advertising media. More importantly, many applications have been proposed in the field of human-computer interactions, including gaze-controlled or gaze-assisted interfaces and information-presentation techniques that take user attention into consideration.

However, despite considerable advances in recent research, current techniques of estimating gazes still suffer from many limitations. Creating an accurate gaze estimator that uses simple and low-cost equipment while allowing users to move their heads freely still remains an open challenge.

Our goal is to make a completely passive, non-contact, single-camera system for estimating gazes that has no calibration stage yet still allows changes in head poses. To achieve this goal, we develop a new appearance-based system of estimating gazes based on an online-learning approach. Our system incorporates recent advances in robust, single-camera, three-dimensional (3D) estimates of head poses to continuously capture users' head poses and eye images. We assume a desktop environment with a personal-computer (PC) camera mounted on a monitor, and that the user is looking at the mouse cursor when he or she is clicking. By using the clicked coordinates as gaze labels, the system automatically acquires learning samples while users are operating the PC, and it can learn mapping between the eye and gaze adaptively during operation, without the need for long preliminary calibra-

tions.

Prior methods have either been model-based or appearance-based. First, we will describe the advantages and drawbacks of these two approaches.

3.1.1 Model-based methods

Model-based approaches use an explicit geometric model of an eye and estimate the eye's gaze direction using geometric eye features. One of the most well-known features is the pupil-glint vector [HWJM⁺89, Jac90], which encodes the offset between the pupil's center and the specular reflection from a light source. While model-based approaches can be precise, they typically require accurate localization of geometric features in high-resolution images of the eye. Moreover, they often require additional hardware such as light sources. This often results in large systems with special equipment that are difficult to implement with only an ordinary camera.

Of the model-based methods, one popular approach that handles head movements involves using multiple light sources and camera(s) to accurately locate 3D eye features. Shih and Liu use both multiple cameras and multiple lights for 3D estimates of gaze [SL04]. Zhu *et al.* use a stereo-camera setup with one light source to locate the 3D position of the eye. They estimate 2D gaze points by learning a mapping function of the pupil-glint vector and the eye position [ZJ05, ZJB06]. Morimoto *et al.* propose a method of using a single camera with at least two lights, but they only present their simulation results [MAF02]. Hennessey *et al.* develop a similar system with multiple light sources to locate the 3D center of the cornea by triangulation. Their method computes the gaze point as a 3D intersection of the monitor's surface and the optical axis of the eye [HNL06]. Yoo and Chung use a structured rectangular light pattern and estimate the gaze point from the pupil's position relative to the light pattern [YC05]. Coutinho and Morimoto later

extend this method with a more precise eye model [CM06].

In addition to eye features, there are methods that also use information from 3D head poses. Beymer and Flickner, for example, use a pair of stereo systems [BF03]. Their first stereo system computes the 3D head pose, which guides the second, high-resolution stereo system that tracks the eye region. Matsumoto *et al.*'s method uses a single stereo system to compute the 3D head pose and estimate the 3D position of the eyeball [MOZ00]. A similar approach is also taken by Wang and Sung [WS02] by using iris edges as features. While successful, these approaches all require special equipment, preventing their use with a monocular camera.

Some methods in recent years try to remove such restrictions on the model-based methods. Ishikawa *et al.*'s method [IBMK04] uses an active appearance model [CET01] to extract eye features and head poses and it enables monocular gaze estimation. Yamazoe *et al.*'s method [YUYA08] estimates gaze direction by fitting a 3D eye model to 2D eye images. An ordinary low-resolution camera is used in these methods. For this reason, their methods are limited to only computing coarse features, such as the edge of the iris and the corners of the eyes. This results in poor accuracy and reduces the advantages of model-based methods.

3.1.2 Appearance-based methods

Appearance-based approaches directly compute features from eye images, and estimate gaze points by learning mapping from image features. Compared to model-based methods, they can make systems less restrictive, more robust, and accurate, even when used with relatively low-resolution cameras.

Baluja and Pomerleau use a neural network to learn the mapping function between eye images and gaze points in display coordinates using 2,000 training samples [BP94]. Xu *et al.* propose a similar neural-network-based

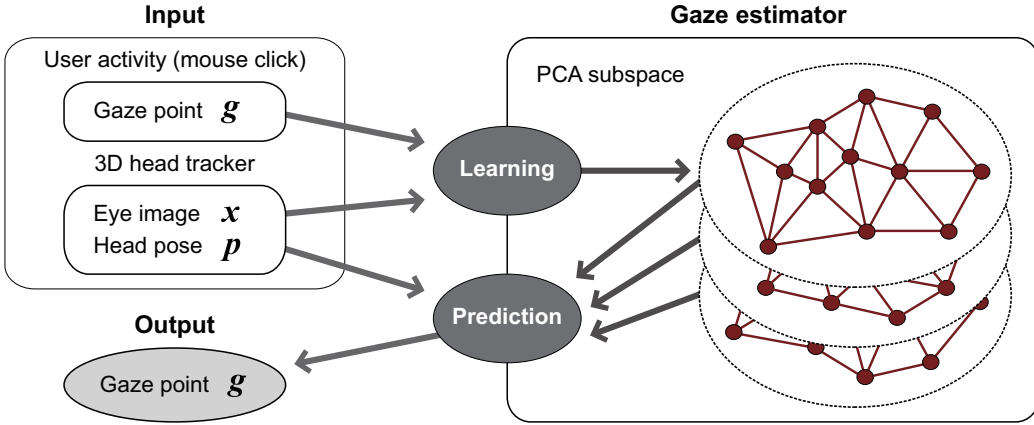


Figure 3.1: Learning and prediction flow for proposed framework.

approach [XMS98] that uses more training samples, *i.e.*, about 3,000. Tan *et al.* use a local interpolation approach to estimate unknown gaze points from relatively sparse (252) samples [TKA02]. More recently, Williams *et al.* propose a novel regression method called S³GP (Sparse, Semi-Supervised Gaussian Process), and apply it to a gaze estimation task with partially labeled (16 out of 80) training samples [WBC06].

Of the appearance-based approaches, few studies have been dedicated to dealing with changes in head pose. Baluja *et al.*'s method [BP94] allows some head movements using training samples from different head poses, but the range of movements is limited. They describe two major difficulties. The first is that the appearance of an eye gazing at the same point drastically varies with head motion. Therefore, additional information about the head pose is needed to solve this problem. The second difficulty is that the training samples has to be collected across the pose space to account for head movements. This results in a large number of training samples and an unrealistically long calibration stage.

3.1.3 Our approach

As described above, both model-based and appearance-based methods have some drawbacks. Model-based methods require more than a normal camera to achieve sufficient accuracy. Appearance-based methods need longer periods of calibration to handle head-pose variations.

We use an appearance-based approach to build a casual gaze estimator using an ordinary desktop camera. To overcome the drawbacks with appearance-based methods, we propose an incremental framework to collect learning samples from typical PC operations. Learning samples are collected under the assumption that the user is looking at a mouse cursor when the mouse is being clicked. In this way, our method continuously accumulates more learning samples as the user continues to click. As the learning and estimates can be implemented as a background process, the user can avoid a long calibration stage.

We use an approach of locally interpolating the learnt gaze points for the estimation. To efficiently create the appearance manifold in a sample space with varying poses, we propose a method using sample clusters associated with similar head poses. A local appearance manifold is independently built in each cluster, and is used to appropriately select the samples for interpolation. In addition, we avoid the effect of human errors included in the clicked coordinates by using a framework to refine the learning labels.

This work makes two main contributions.

- **Incremental learning framework:** To eliminate the lengthy calibration stages required for appearance-based gaze estimators with free head movements, we employ an incremental learning framework that only requires normal operations by a user on a PC.
- **Adaptive clustering:** We extended the appearance-based framework for gaze estimation by adaptively clustering learning samples to handle

large variations in head poses. In addition, we used a framework to refine the learning labels to handle human errors.

The rest of the chapter is organized as follows. Section 3.2 describes the architecture of our system. Section 3.3 explains our incremental-learning algorithm. The details on implementing the head tracker and our eye-image cropping are subsequently described in Section 3.4. We explain how we tested the proposed method to verify its effectiveness in Section 3.5. Section 3.6 closes with a discussion on potential applications of our method and future research directions.

3.2 Overview

This section first describes an overview of our method. The process flow for the system is summarized in Figure 3.1.

The input to the system is a continuous video stream from the camera. The 3D model-based head tracker [SS07] keeps running during the entire process to estimate the head pose, \mathbf{p} , and to crop the eye image, \mathbf{x} .

We assume that the user’s gaze is directed at the mouse cursor on the monitor when he or she is clicking the mouse. Using this assumption, we collect learning samples by capturing eye images and cursor positions for all mouse clicks. We create a training sample at each mouse click using the mouse position in the screen coordinates as the gaze label, \mathbf{g} , associated with the features (head-pose \mathbf{p} and eye-image \mathbf{x}). Our system incrementally updates the mapping function between appearance-based features and the gaze by using this labeled sample.

Incremental learning is done in a reduced PCA subspace to decrease the computational cost of dealing with multi-dimensional image features. The samples are adaptively clustered according to their head poses, and the local appearance manifold is updated in each cluster.

When the new training samples are not given to the system, the system runs in a prediction loop, and the gaze is estimated using the updated state of clusters and local manifolds.

3.3 Algorithm

The goal of our gaze estimator is to learn the mapping between features $\{\mathbf{x}, \mathbf{p}\}$ and the gaze label, \mathbf{g} . We use a local linear interpolation method that is similar to [RS00, TKA02]. Given unlabeled features $\{\mathbf{x}, \mathbf{p}\}$, we predict the unknown label, $\check{\mathbf{g}}$, by choosing k nearest neighbors from the labeled samples and interpolating their labels using distance-based weights.

It is critical to choose neighbors from a manifold that models changes in appearance for different gaze directions in the appearance space for our application. Tan *et al.* [TKA02] use 2D topological information about the coordinates of the gaze labels as a constraint. Two points are assumed to be neighbors on the manifold in their method if they also have similar gaze directions, not only similar appearances. However, this assumption does not always hold if the head pose changes; two different gaze directions lead to a similar appearance, or conversely similar gaze directions lead to very different appearances.

To overcome this problem, we compute sample clusters with similar head poses and create a local manifold for each sample cluster. This model is inspired by the locally weighted projection regression (LWPR) algorithm [VDS05]. Local linear regressors are adaptively created and learned in LWPR according to the input feature distance itself. We employ a similar adaptive architecture to create pose-dependent clusters of eye images.

The similarity measure of the cluster, *i.e.*, how close the head pose and the cluster are, is defined as a product of the Gaussian functions of head translation and rotation. Given a pose, \mathbf{p}_i , specified by translation \mathbf{t}_i and

rotation \mathbf{r}_i in 3D, the similarity of the pose to a certain cluster is computed as

$$s_k(\mathbf{p}_i) = \frac{1}{\sqrt{2\pi\kappa_t\sigma_t^2}} \exp\left(-\frac{\|\mathbf{t}_i - \bar{\mathbf{t}}\|^2}{2\kappa_t\sigma_t^2}\right) \frac{1}{\sqrt{2\pi\kappa_r\sigma_r^2}} \exp\left(-\frac{\|\mathbf{r}_i - \bar{\mathbf{r}}\|^2}{2\kappa_r\sigma_r^2}\right), \quad (3.1)$$

where $\bar{\mathbf{t}}$ and σ_t^2 are the average and variance of head translation calculated from the samples contained in the cluster. Likewise, $\bar{\mathbf{r}}$ and σ_r^2 are the average and variance in head rotation. The constant weights, κ_t and κ_r , are empirically set.

The Euclidean distance measure is used in Eq. (3.1) for both translation and rotation vectors. Of these, the rotation vector can be described with a quaternion. Strictly speaking, the distance between two quaternions has to be measured with an angular distance, ω_d , *i.e.*, the angle of rotation from one quaternion to the other. However, we employed the Euclidean distance for the following reasons. In our incremental case, calculating the average, $\bar{\mathbf{r}}$, was computationally expensive in the angular-distance measure. In contrast, the average orientation in the Euclidean-distance measure could easily be obtained as an arithmetic average of the quaternions [HGME96]. The Euclidean distance, $\|\mathbf{r} - \bar{\mathbf{r}}\|^2 = \|\mathbf{I} - \bar{\mathbf{r}}\mathbf{r}^{-1}\|^2 = 4\sin^2(\omega_d/4)$, can also be a good approximation of the angular distance when the two rotations are close.

Given a labeled sample, the image, \mathbf{x}_t , is first used to update the PCA subspace of the eye images. After updating the subspace, the sample is added to all clusters whose similarity $s(\mathbf{p}_t)$ is higher than the predefined threshold, τ_x . The learning algorithm is outlined in Algorithm 1, where K is the total number of clusters created by the time $t - 1$. If no suitable clusters are found, a new cluster is created to only contain the new sample. Given an unlabeled feature, the output gaze, $\hat{\mathbf{g}}$, is computed as a weighted sum of

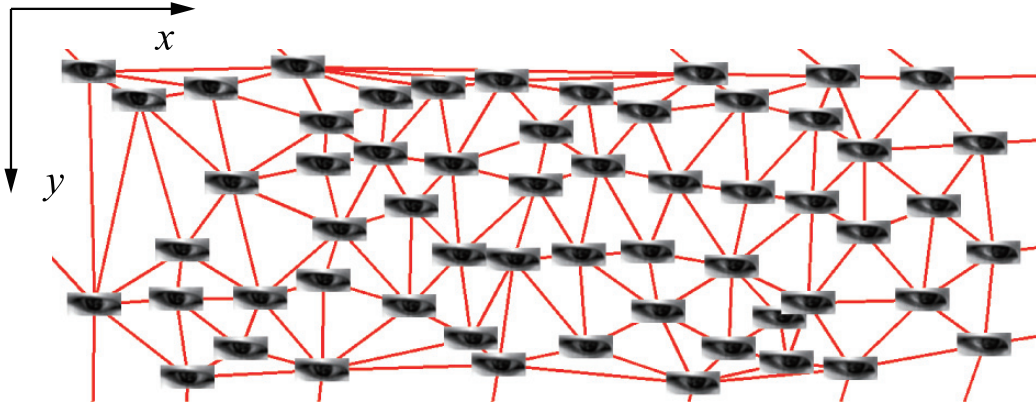


Figure 3.2: Example of gaze triangulation shown in screen coordinates. Each eye image (flipped horizontally to clarify presentation) is located at the corresponding gaze point, and the lines indicate Delaunay edges between these gaze points.

candidate predictions from different clusters in the prediction stage. The following sections, 3.3.1 and 3.3.2, describe further details on prediction and learning methods.

3.3.1 Prediction

When unlabeled data, $\{\mathbf{x}, \mathbf{p}\}$, are given, the system predicts the estimated gaze, $\check{\mathbf{g}}$, from the learnt data. First, the eye image, \mathbf{x} , is projected into the current PCA subspace computed from all i learning samples as

$$\mathbf{x} \approx \bar{\mathbf{x}}^{(i)} + \mathbf{U}^{(i)} \mathbf{a}, \quad (3.2)$$

using the mean eye image, $\bar{\mathbf{x}}^{(i)}$, and the matrix, $\mathbf{U}^{(i)}$, whose columns are composed of the first N eigenvectors. \mathbf{a} is an N -dimensional vector of PCA coefficients.

An intermediate gaze estimate is first computed in each cluster from the input PCA coefficients, \mathbf{a} , by local interpolation of neighbors. The neighboring samples of \mathbf{a} are selected based on the manifold, and the gaze labels of

Algorithm 1 Adaptive clustering framework

Prediction: Given unlabeled features $\{\mathbf{x}, \mathbf{p}\}$ Project image \mathbf{x} into the current subspace: $\mathbf{a} = \mathbf{U}^{(t)T}(\mathbf{x} - \bar{\mathbf{x}}^{(t)})$ **for** $k = 1$ to K **do**Calculate the interpolated gaze $\check{\mathbf{g}}_k$ and prediction confidence c_k (Section 3.3.1).**end for**Compute final prediction as a weighted sum: $\check{\mathbf{g}} = \sum_k c_k \check{\mathbf{g}}_k / \sum_k c_k$.**Learning:** Given the i -th learning sample $\{\mathbf{x}_i, \mathbf{p}_i\}$ associated with the gaze label \mathbf{g}_i Update image subspace using incremental PCA: mean $\bar{\mathbf{x}}^{(i)}$, eigenvectors $\mathbf{U}^{(i)}$, eigenvalues $\boldsymbol{\lambda}^{(i)}$, coefficients $\{\mathbf{a}_1 \dots \mathbf{a}_i\}$. \mathbf{x}_i can be approximated as $\mathbf{x}_i \approx \bar{\mathbf{x}}^{(i)} + \mathbf{U}^{(i)} \mathbf{a}_i$.**for** $k = 1$ to K (with respect to each of all K clusters) **do****if** $g_k(\mathbf{p}_i) > \tau_x$ **then**

Add sample to the cluster and update its local manifold (Section 3.3.2).

end if**end for****if** No $g_k(\mathbf{p}_i)$ is above the threshold τ_x **then**Create new $(K + 1)$ -th cluster and add the sample. $K \leftarrow K + 1$.**end if**

the neighbors are interpolated to determine the intermediate gaze estimate in the cluster.

As done by Tan *et al.* [TKA02], we use Delaunay triangulation of the gaze label as the manifold model. Figure 3.2 shows an example of the computed triangulation. By selecting neighboring samples that are located along the triangle, the sample set for interpolation are constrained to have a limited range of gaze variations. Our algorithm finds a set of neighboring triangles by measuring the average distance from \mathbf{a} to the samples (vertices) of each triangle. The sample points adjacent to this triangle are also selected as neighbors. To ensure computational efficiency, the process for selecting triangles is performed using the N_s closest samples in the cluster. If triangles are not found by parameter N_s , size N_s is increased by n_s , and the selection process is iterated until a triangle set is found.

Using selected-set \mathcal{N}_p , interpolation-weights,

$$\mathbf{w} = (w_1, w_2, \dots, w_{|\mathcal{N}_p|}), \quad (3.3)$$

are computed by minimizing the reconstruction error as

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{a} - \sum_{i \in \mathcal{N}_p} w_i \mathbf{a}_i)^2, \quad (3.4)$$

subject to

$$\sum_{i \in \mathcal{N}_p} w_i = 1, \quad (3.5)$$

where w_i denotes the weight of the i -th neighbor's appearance, \mathbf{a}_i . Finally, assuming local linearity, the intermediate gaze estimate, \mathbf{g}_k , from the k -th cluster is computed as

$$\check{\mathbf{g}}_k = \sum_{i \in \mathcal{N}_p} w_i \mathbf{g}_i. \quad (3.6)$$

To reject outliers from the clusters that do not contain a sufficient number of samples, we define an interpolation-reliability measure that represents how

well the input, \mathbf{a} , can be described by the selected neighbors as

$$r_k(\mathbf{a}) = \exp\left(-\frac{(\mathbf{a} - \sum_{i \in \mathcal{N}_p} w_i \mathbf{a}_i)^2}{2\varsigma_r^2}\right). \quad (3.7)$$

Eq. (3.7) evaluates the reconstruction error in the appearance, \mathbf{a} . The factor, ς_r , is empirically set. We define the prediction confidence, c_k , as a product of the reliability, $r(\mathbf{a})$, and the pose similarity, $s(\mathbf{p})$, as

$$c_k = s_k(\mathbf{p})r_k(\mathbf{a}). \quad (3.8)$$

The final prediction result, $\check{\mathbf{g}}$, is computed as a weighted sum of $\check{\mathbf{g}}_k$ based on c_k by deriving

$$\check{\mathbf{g}} = \frac{\sum_k c_k \check{\mathbf{g}}_k}{\sum_k c_k}. \quad (3.9)$$

We also compute a weighted average of $r(\mathbf{a})$ between all clusters as

$$\bar{r}(\mathbf{p}, \mathbf{a}) = \frac{\sum_k s_k(\mathbf{p})r_k(\mathbf{a})}{\sum_k s_k(\mathbf{p})}, \quad (3.10)$$

to assess the overall reliability of the gaze estimate. Figure 3.3 shows the angular error plotted against reliability \bar{r} . The larger rectangles indicate windowed averages of angular error with a window width of 0.1 in reliability. The plot shows that the accuracy of estimation increases as the reliability measure increases. Therefore, gaze-estimate $\check{\mathbf{g}}$ can be stabilized by taking a temporal weighted average based on \bar{r} . The effect of averaging is discussed in more detail in Section 3.5.

3.3.2 Learning

Given the i -th learning sample, $\{\mathbf{x}_i, \mathbf{p}_i, \mathbf{g}_i\}$, we first update the appearance subspace using Skocaj *et al.* [SL03]'s incremental PCA. $\bar{\mathbf{x}}^{(i)}$ and $\mathbf{U}^{(i)}$ in Eq. (3.2), and all stored coefficients $\{\mathbf{a}_1 \dots \mathbf{a}_{i-1}\}$ are updated at a time.

After that, a reduced learning sample, $\{\mathbf{a}_i, \mathbf{p}_i, \mathbf{g}_i\}$, is added to a pose cluster only when its pose \mathbf{p}_i is sufficiently close to the cluster's center. With

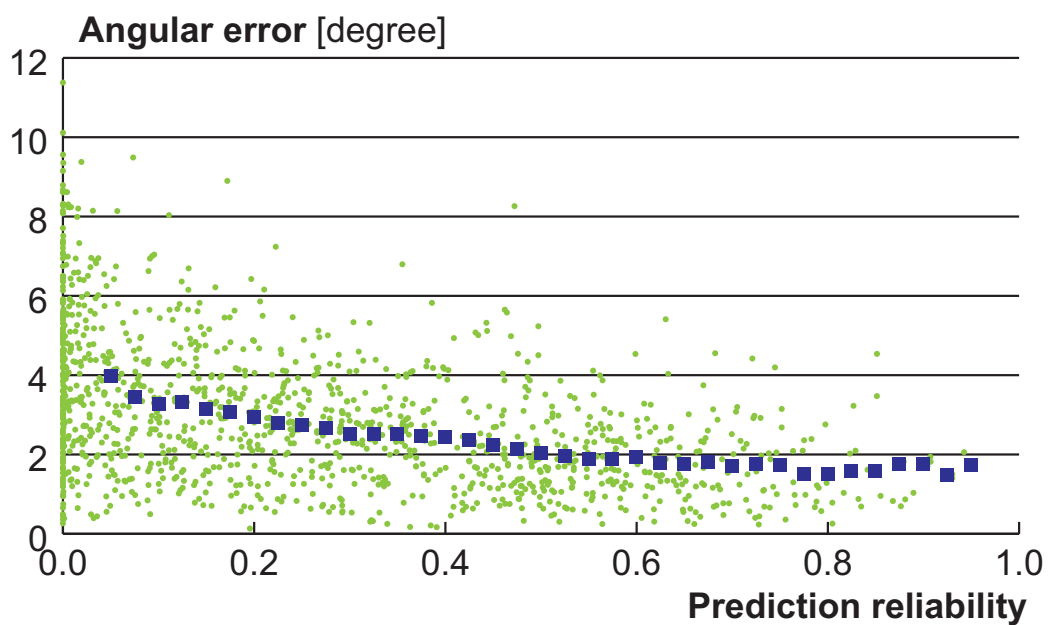


Figure 3.3: Angular error against prediction reliability. The graph has a scatter plot of the estimation error versus the reliability we defined in Eq. (3.7) and Eq. (3.10). Larger rectangles indicate partial averages with a window width of 0.1 in reliability.

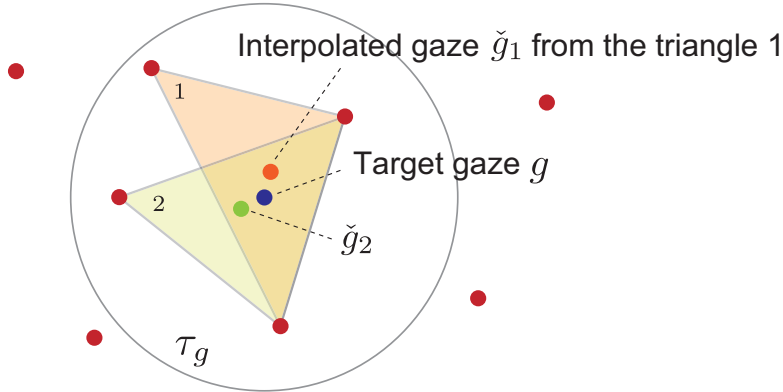


Figure 3.4: Process to refine gaze labels. The gaze label of target-sample \mathbf{g} is refined as a weighted sum of interpolated-labels $\check{\mathbf{g}}$ from enclosing triangles within distance-threshold τ_g .

this approach, every cluster is guaranteed to contain samples with similar head poses. Tan *et al.* [TKA02] use a topological manifold model in similar settings. However, the gaze label, \mathbf{g} , in their method is treated as a static quantity without any error. In reality, humans cannot gaze at a point with pixel-level accuracy. Even when a user is looking at a target carefully, a certain level of fixational eye movement occurs. It has been reported that about 1 degree of microsaccades occur during fixation [MCMH04]. In our case, since the user is not forced to look carefully at the mouse cursor when clicking, larger errors can be included in the gaze label, \mathbf{g} . Moreover, there could be meaningless samples due to random mouse clicks without due attention.

For these reasons, we avoid directly using the clicked coordinates, \mathbf{g}_c , as a gaze label, \mathbf{g} . Instead, we estimate the probable gaze label, \mathbf{g} , constrained by \mathbf{g}_c using a refining approach starting with \mathbf{g}_c as an initial value for \mathbf{g} . To refine the gaze label, \mathbf{g} , of a sample, we first select all existing samples whose distance to the incoming sample's click point \mathbf{g}_c are under a threshold, τ_g , in the gaze space. Using these existing samples and incoming-sample \mathbf{g} , the set of all combinations of three samples that enclose incoming-sample \mathbf{g} can

be computed (see Figure 3.4). Using the method described in the previous section, interpolated gaze-label $\check{\mathbf{g}}$ can be calculated from each triangle. All interpolated labels are aggregated as a Gaussian-weighted sum around \mathbf{g}_c as

$$\mathbf{g} = \frac{\mathbf{g}_c + \sum_i r_i q_i \check{\mathbf{g}}_i}{1 + \sum_i r_i q_i}, \quad (3.11)$$

where

$$q_i = \exp\left(-\frac{\|\mathbf{g}_i - \mathbf{g}_c\|^2}{2\varsigma_q^2}\right). \quad (3.12)$$

Here, i is the triangle index, and r_i is the reliability measure calculated in the same way as Eq. (3.7). The factor, ς_q , is empirically set. The clicked point, \mathbf{g}_c , is added with full-weight 1.

Treatment of learning data As mentioned earlier, there are incoming samples that are not useful as learning samples, *e.g.*, clicks without due attention. These samples do not capture the correlation between the appearance and gaze label (clicked point). To avoid such outliers, we assess the data through cross validation. In parallel with computing the interpolated gaze label, \mathbf{g} , we can compute pure interpolation without the constraint of \mathbf{g}_c as

$$\dot{\mathbf{g}} = \frac{\sum_i r_i \check{\mathbf{g}}_i}{\sum_i r_i}. \quad (3.13)$$

Compared to Eq. (3.11), Eq. (3.13) indicates a weighted average of interpolated gaze labels regardless of the clicked point, \mathbf{g}_c . If the distance, $d_g = \|\dot{\mathbf{g}} - \mathbf{g}_c\|^2$, is too long, *i.e.*, interpolated-gaze $\dot{\mathbf{g}}$ is too far from the clicked point, \mathbf{g}_c , the sample can be considered as an outlier. In that case, we delete the sample from the cluster instead of refining its gaze label.

In addition, since the sample distribution in the learnt gaze space does not have to be too dense, we introduce another method of pruning learning samples. If there is more than one sample around the new position of gaze-label \mathbf{g} , we keep the sample with the lowest d_g and delete the others. We use a threshold, τ_r , to control the sample density. The threshold value should be set with respect to both the size of the display area and memory capacity.

For example, Tan *et al.* [TKA02] used one sample per 2.76 [cm²]. Whenever a new incoming sample is provided, the data treatment processes described above are executed for every sample in the clusters.

Once a sample is added to the cluster, we update cluster-mean $\bar{\mathbf{t}}_k$ and the variance, $\sigma_{t,k}^2$, of Eq. (3.1) incrementally as

$$\begin{aligned}\bar{\mathbf{t}}_{\text{new}} &= \frac{n\bar{\mathbf{t}}_{\text{old}} + \mathbf{t}_i}{n + 1}, \\ \sigma_{t,\text{new}}^2 &= \frac{n\sigma_{t,\text{old}}^2 + (\mathbf{t}_i - \bar{\mathbf{t}}_{\text{new}})(\mathbf{t}_i - \bar{\mathbf{t}}_{\text{old}})}{n + 1},\end{aligned}\tag{3.14}$$

where n denotes the total number of samples in the cluster before updating, and \mathbf{t}_i represents the translation vector of the new incoming sample. The same goes for updating $\bar{\mathbf{r}}$ and σ_r^2 . Delaunay triangulation in the gaze-label coordinates is consequently recomputed.

3.4 Implementation

This section describes the details on implementing our system. We particularly describe methods of obtaining input features (head-pose \mathbf{p} and eye-image \mathbf{x}). We also introduce a method of detecting blinks to improve the accuracy of gaze estimates.

3.4.1 Head tracking and eye capturing

The system captures head-pose \mathbf{p} and eye-image \mathbf{x} from a sequence of gray-scale input images for both learning and prediction.

Our method uses the head-tracking method [SS07] based on a multilinear model, which separately represents facial-shape variations due to two factors: different people and different facial expressions. Since facial deformations did not need to be captured in our case, a linear model that only

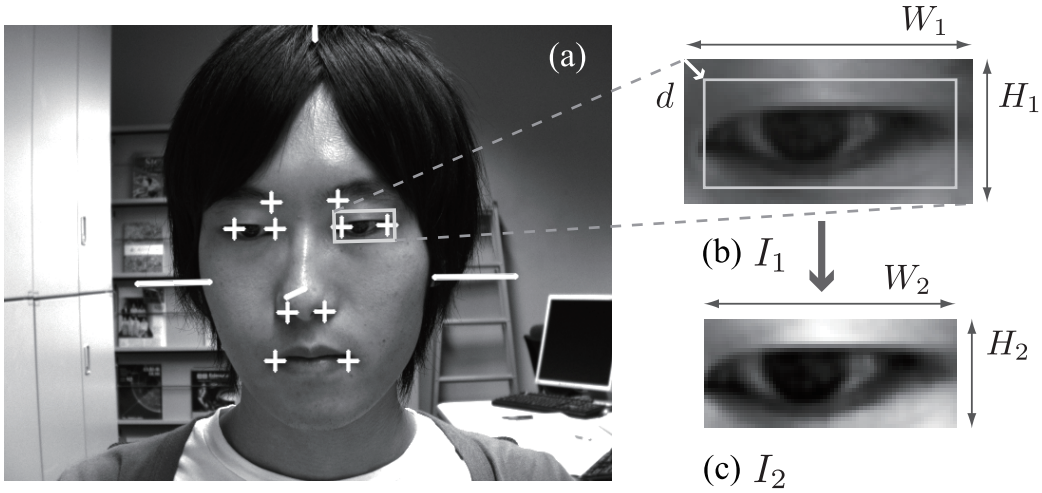


Figure 3.5: Results for head tracking and eye capturing. (a) Results for estimating head poses. (b) Results for cropping around predefined eye region on facial mesh (rectangle in (a)). (c) Eye-alignment and image-preprocessing results (image feature used in our gaze estimator.)

describes shape variations for different people is used. The face is represented as the appearance and 3D positions of 10 feature points defined in a local-coordinate system fixed to the user’s head (Figure 3.5 (a)). In this work, a 30-dimensional facial-shape vector is described as a weighted sum of eigenshapes that are precomputed by PCA. Details on the computation of the facial shape vector can be found in [SS07]. Using the model, our system simultaneously tracks the 3D head pose using a particle filter [IB98] and estimates the facial shape based on bundle adjustment [TMHF99]. As a result, the tracker outputs the user’s 3D head pose, $\mathbf{p} = \{\mathbf{t}, \mathbf{r}\}$, where $\mathbf{t} = {}^t(x, y, z)$ is a 3D translation and $\mathbf{r} = {}^t(q_1, q_2, q_3, q_4)$ is a 4D rotation vector defined by four quaternions. Figure 3.5 (a) shows an example of head-pose tracking. The crosses indicate the positions of the feature points, and the long lines indicate the head pose.

Once the head pose is estimated, the system crops the eye image. Us-

ing the estimated head position, it first extracts a rough eye region from the input image. Based on the distance between eye-corners in the image coordinates, a rectangle region with a fixed aspect ratio (the rectangle in Figure 3.5 (a)) is cropped. The rectangle is then transformed to a $W_1 \times H_1$ image \mathbf{I}_1 (Figure 3.5 (b)). We further apply histogram equalization to the image, \mathbf{I}_1 , to normalize its brightness.

While head-pose tracking is robust, there is a small amount of error when cropping eye images. This appears as a small amount of jitter in eye images when viewed in sequence. Because our method evaluates the distance between images, the accuracy of estimated gazes is dependent on the distance measure. For that reason, we applied a method of subspace alignment, which is described below, to avoid letting jitter error from adversely affecting estimates.

Subimage \mathbf{I}_2 of size $W_2 \times H_2$ (Figure 3.5 (c)) is first cropped from larger image \mathbf{I}_1 of size $W_1 \times H_1$ with a top-left margin, $\mathbf{d} = {}^t(x, y)$, in our eye-cropping method. As described in Section 3.3, the PCA subspace used in the learning algorithm is updated incrementally using labeled samples. Basically, we chose \mathbf{I}_2 that maximizes correlation with a reconstructed image, $\hat{\mathbf{I}}_2$. It is calculated in the form of a raster-scanned vector, $\hat{\mathbf{x}}_2 = \bar{\mathbf{x}} + \mathbf{U}^t \mathbf{U}(\mathbf{x}_2 - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ and \mathbf{U} are mean and eigenvectors of the PCA subspace. We calculate $(W_1 - W_2 + 1) \times (H_1 - H_2 + 1)$ correlation-map \mathbf{C} first. The value at (x, y) corresponds to the correlation between \mathbf{I}_2 and $\hat{\mathbf{I}}_2$ with an offset, $\mathbf{d} = {}^t(x, y)$. The offset at the pixel level is acquired as a maximal point, (d_x, d_y) , of the map, \mathbf{C} . To ensure sub-pixel accuracy, we furthermore calculate the sub-pixel differences, (δ_x, δ_y) , by using simple 2D parabola fitting described as

$$\begin{bmatrix} \frac{\partial \mathbf{C}}{\partial x}(d_x + \delta_x, d_y + \delta_y) \\ \frac{\partial \mathbf{C}}{\partial y}(d_x + \delta_x, d_y + \delta_y) \end{bmatrix} = \mathbf{0}. \quad (3.15)$$

Eq. (3.15) can be expanded as a Taylor expansion around (d_x, d_y) as

$$\begin{bmatrix} \mathbf{C}'_x \\ \mathbf{C}'_y \end{bmatrix} + \begin{bmatrix} \mathbf{C}''_{xx} & \mathbf{C}''_{xy} \\ \mathbf{C}''_{xy} & \mathbf{C}''_{yy} \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \mathbf{0}, \quad (3.16)$$

and the difference, (δ_x, δ_y) , can be calculated as

$$\begin{cases} \delta_x = \frac{\mathbf{C}'_y \mathbf{C}''_{xy} - \mathbf{C}'_x \mathbf{C}''_{yy}}{\mathbf{C}''_{xx} \mathbf{C}''_{yy} - (\mathbf{C}''_{xy})^2} \\ \delta_y = \frac{\mathbf{C}'_x \mathbf{C}''_{xy} - \mathbf{C}'_y \mathbf{C}''_{xx}}{\mathbf{C}''_{xx} \mathbf{C}''_{yy} - (\mathbf{C}''_{xy})^2} \end{cases}. \quad (3.17)$$

Here, \mathbf{C}' and \mathbf{C}'' correspond to the 1st and 2nd order derivatives at (d_x, d_y) .

Finally, \mathbf{I}_2 is cropped with the offset, $(d_x + \delta_x, d_y + \delta_y)$, and raster-scanned to create an image vector, \mathbf{x} . In our experiment, the size of the final image was set to $W_2 = 70 \times H_2 = 30$ pixels, so \mathbf{x} was 2100-dimensional. As a result, we acquire eye-image \mathbf{x} and head-pose \mathbf{p} .

3.4.2 Blink detection

If a user blinks when clicking, the data becomes an inappropriate sample for the learning process. To reject such samples, we detect blinks by using the maximum correlation. As described earlier, the eye images were cropped in a way where the correlation with the reconstructed image was maximized. However, if the input eye image was dissimilar to all samples that span the subspace, the maximum correlation becomes relatively small. For this reason, the eye images with blinking can be found by evaluating the correlation, as shown in Figure 3.6.

The system ignores the input if the correlation is lower than threshold τ_b . Blinks usually last for about $150ms$, which is long enough to appear in multiple video frames as illustrated in Figure 3.6. Therefore, we discard neighboring frames of obvious blinks that are observed within a certain time range.

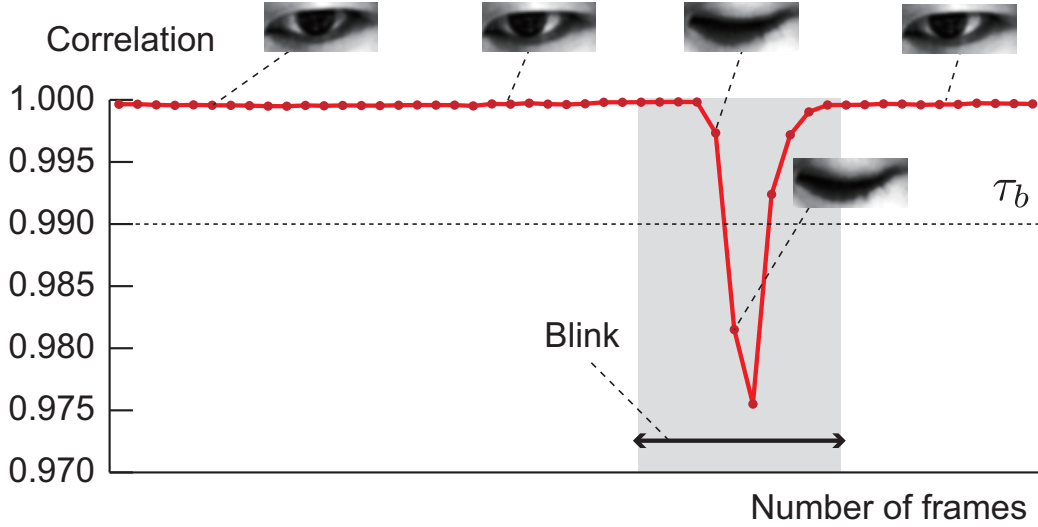


Figure 3.6: Blink detection. The graph shows the correlation between the resulting image for eye-cropping I_2 and reconstructed-image \hat{I}_2 . Each image in the figure corresponds to I_2 . As correlation drops to a lower value when the user blinks, input can be rejected by giving the correlation a threshold.

3.5 Experiments

We evaluated the proposed method with two different setups. The first was with a simulation experiment (Section 3.5.1) where a click target was explicitly shown to a user. The second evaluation was carried out in a real scenario where a user operated a PC for Internet Web browsing (Section 3.5.2). Throughout the experiments, we used the following parameters: $N = 30$, $\kappa_t = \kappa_r = 2.0$, $\tau_x = 0.001$, $N_s = 30$, $n_s = 10$, $\zeta_r^2 = 25000$, $\zeta_q^2 = 2500$, $\tau_g = 100$ [px], $\tau_r = 30$ [px], and $\tau_b = 0.99$.

Our system consisted of a VGA resolution camera (PointGrey Flea) and a Windows PC with a 2.67 GHz dual core CPU and 3 GB of RAM. The whole process including display rendering ran at about 20 fps in our research implementation without any optimization.

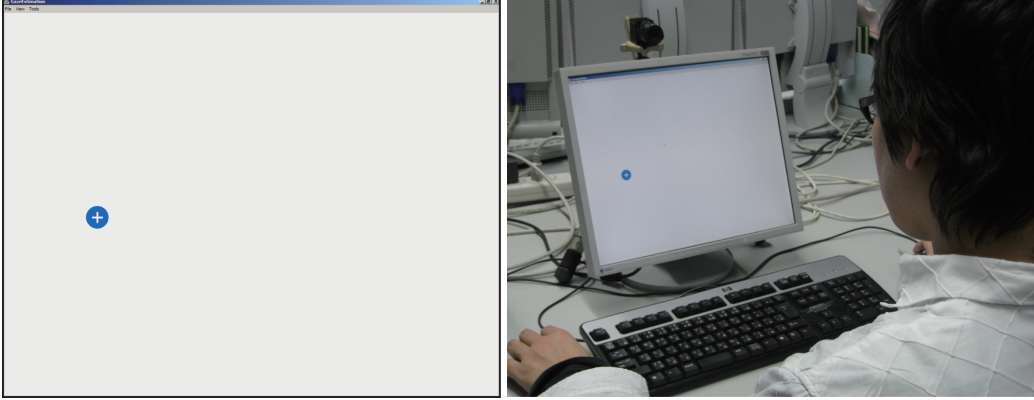


Figure 3.7: System settings for simulation experiments. The figure at left is a screen-shot of the full-screen window shown to the user. That at right is a photograph of the actual experimental setup.

3.5.1 Simulation experiments











We will first present the results obtained in a simulation environment to evaluate the performance of our method. We used a full-screen window in this experiment and randomly displayed a clicking target to a user. To simulate a typical target like a button or an icon, the target was rendered as a circle 64 pixels in diameter (Figure 3.7). Ten users were asked to click the target about 1200 times (about a 20-minute operation). During this operation, the users were allowed to freely move their heads. The target for clicking was only used to prompt the users to click it, and our method acquired the actual click position as a learning label.

Whenever a new labeled sample was given, the prediction was performed prior to the learning. The estimation error was evaluated as the distance between the clicked position, \mathbf{g}_t , and the estimated gaze position, $\check{\mathbf{g}}_t$. The angular error, θ , is computed as

$$\theta_t = \tan^{-1} \left(\frac{D_m(\mathbf{g}_t, \check{\mathbf{g}}_t)}{z_t - d_{\text{cam}}} \right), \quad (3.18)$$

where D_m measures the distance between two points in a metric system, z_t

Table 3.1: Results of simulation experiments. Starting from the left, the table shows the angular and pixel errors (notated as *average* \pm *standard deviation*), numbers of clicks, numbers of clusters, and ranges of head-pose movements in the camera coordinate system. “Normal” error corresponds to the raw output, $\tilde{\mathbf{g}}$, of the system, and “weighted” error corresponds to the temporal weighted average based on the weight, \bar{r} , in Eq. (3.10). x , y , and z correspond to horizontal, vertical, and depth-directional translation, and ϕ , θ , and ψ correspond to the rotation around the z , x , and y axes.

Person	Angular error [deg]		Pixel error [px]		Num. clicks Used/All	Num. clusters	Trans. [cm]			Rot. [deg]		
	Weighted	Normal	Weighted	Normal			x	y	z	ϕ	θ	ψ
 A	2.5 \pm 1.5	2.9 \pm 1.8	89 \pm 58	105 \pm 67	1313/1313	13	16	7	31	5	18	17
 B	3.0 \pm 2.2	3.7 \pm 2.8	135 \pm 101	165 \pm 130	1293/1302	11	27	10	36	9	32	19
 C	2.4 \pm 1.5	3.0 \pm 1.9	102 \pm 65	126 \pm 82	1305/1308	7	23	3	37	6	32	12
 D	3.1 \pm 1.5	3.7 \pm 2.7	107 \pm 73	129 \pm 92	1301/1302	7	22	8	31	7	29	21
 E	3.0 \pm 2.3	3.3 \pm 2.4	126 \pm 92	140 \pm 101	1226/1248	21	27	6	52	16	33	29
 F	3.2 \pm 2.0	3.6 \pm 2.3	150 \pm 95	170 \pm 112	1318/1319	17	34	7	37	18	36	25
 G	2.8 \pm 2.0	3.5 \pm 3.0	120 \pm 85	148 \pm 130	1308/1312	11	19	6	34	17	23	28
 H	3.1 \pm 2.2	3.6 \pm 2.6	150 \pm 105	174 \pm 125	1305/1308	7	23	8	31	8	29	16
 I	2.8 \pm 2.2	3.1 \pm 2.4	110 \pm 89	122 \pm 99	1278/1309	13	21	10	34	41	43	29
 J	3.3 \pm 2.6	3.7 \pm 2.9	145 \pm 117	164 \pm 132	1267/1315	17	17	8	23	22	42	43
Average	2.9 \pm 2.1	3.4 \pm 2.5	123 \pm 88	144 \pm 107			23	7	35	15	32	24

is the depth of the estimated head pose at time t in a camera-coordinate system, and d_{cam} is the pre-defined distance between the camera and the display. The d_{cam} was 100 [mm] in the experiments. We used a 17-inch display with a resolution of 1280×1024 pixels (96 dpi) for this experiment.

Table 3.1 summarizes the results obtained from the experiment. From left to right in the table, we can see the angular and pixel errors (denoted as *average \pm standard deviation*), numbers of clicks, numbers of clusters, and ranges of head-pose movements. For the head-pose movements, x , y , and z correspond to horizontal, vertical, and depth-directional translation, and ϕ , θ , ψ correspond to rotations around the z , x , and y axes. “Normal” error corresponds to the raw output, $\check{\mathbf{g}}$, of the system, and “weighted” error indicates the results with the temporal weighted average (taking the past 5 frames in the experiments) based on the weight, \bar{r} , in Eq. (3.10). “Used” clicks denote the number of clicks that were not rejected as blinking by the rejection process described in Section 3.4.2.

The angular error is consistently low (around 3 degrees), and it demonstrated better performance when the temporal weighted average was used. The range of head-pose movements in the experiment was $23 \times 7 \times 35$ cm and $15 \times 32 \times 24$ degrees on average. This is not the theoretical limit for permissible head-pose movements, and it is dependent on the angle of view and the resolution of the camera.

Figure 3.8 shows the cumulative average of weighted angular error against the number of clicks. While there are some variations across users, the errors consistently converge to certain ranges after 600 clicks. In the early stages of learning, *e.g.*, less than 400 clicks, the learning samples are sparsely distributed. The accuracy of prediction then has large variance. When there are enough samples for interpolation, it yields good estimates. However, if there are not enough samples, it may yield large errors. Due to this instability, the cumulative average occasionally increases in the early stages.

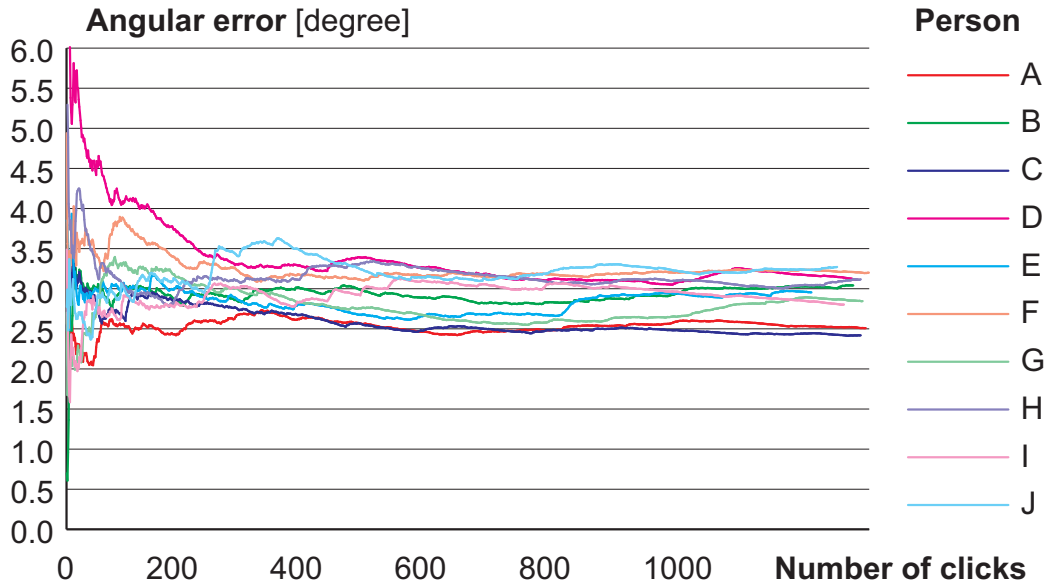


Figure 3.8: Cumulative average angular error in simulation experiments. The graphs indicate the cumulative average of weighted angular error in the simulation experiments against the number of clicks. Each line corresponds to each user in Table 3.1.

To assess the effectiveness of our clustering-based approach, we compared the results with and without the clustering method. Figure 3.9 shows one of the results from comparison. The plots indicate cumulative average errors for the gaze estimates for User A. The middle and the bottom lines correspond to the normal and weighted output, and the top line corresponds to a normal output without clustering, *i.e.*, all samples are added to a single cluster. By comparing the middle and bottom lines, we can see that the error in estimation was greatly reduced by taking the temporal weighted average. The percentage of reduced error was about 85% on average for all users. Additionally, our clustering method consistently performed better (bottom two lines with clustering and top one without). We observed that, without clustering, the error gradually increased after a while and did not converge to a certain error.

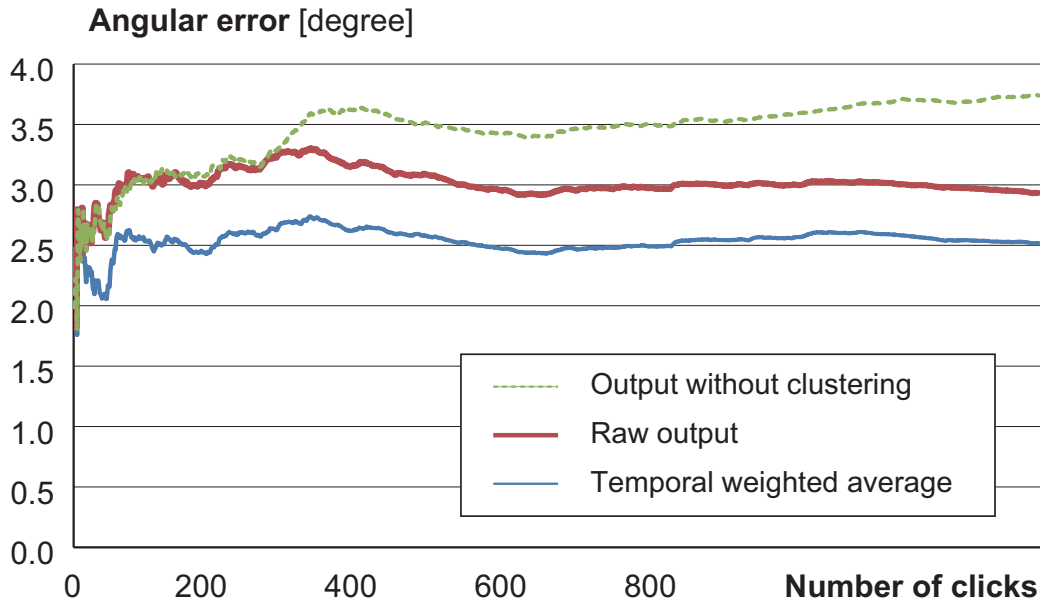







Figure 3.9: Comparison of error in simulation experiments. The graph plots the cumulative average errors of User A. The red and blue lines correspond to normal and weighted output, and the green line corresponds to normal output without clustering.

3.5.2 Evaluation in real environments

Next, we will present the results in a more realistic scenario. Unlike the previous simulation setting, we used users' natural operations on a PC in this experiment. We asked five users to operate a PC to browse the Web for about 30 minutes, so as to make the total number of clicks about 600. To capture mouse-click events and positions, we implemented a global system hook that ran as a background process. Also, the system captured the user's head poses and eye images in the background. We created a natural desktop environment with these implementations. We evaluated how accurately it performed in the same way as in the simulation experiment.

Table 3.2 lists the results of the experiment. Just as in Table 3.1, we can see angular and pixel errors, the numbers of clicks, numbers of clusters, and

Table 3.2: Results of experiments in real environment. Shown are the angular and pixel errors, numbers of clicks, numbers of clusters and ranges of head-pose movements as in Table 3.1.

Person	Angular error [deg]		Pixel error [px]		Num. clicks Used/All	Num. clusters	Trans. [cm]			Rot. [deg]		
	Weighted	Normal	Weighted	Normal			x	y	z	ϕ	θ	ψ
 A	3.0 ± 2.8	3.4 ± 3.2	111 ± 108	127 ± 124	717/728	4	14	6	28	9	23	17
 B	2.7 ± 2.1	3.0 ± 2.4	119 ± 95	136 ± 110	706/718	6	8	5	31	26	47	20
 C	2.3 ± 1.7	2.8 ± 2.0	121 ± 90	148 ± 104	700/700	3	18	2	28	4	22	10
 D	2.7 ± 1.7	3.3 ± 2.1	112 ± 73	134 ± 90	618/619	1	3	4	13	7	9	11
 E	2.3 ± 1.9	2.7 ± 2.2	92 ± 78	105 ± 88	679/692	3	4	4	10	8	21	16
Average	2.6 ± 2.1	3.0 ± 2.4	111 ± 89	130 ± 103			10	4	22	11	25	14

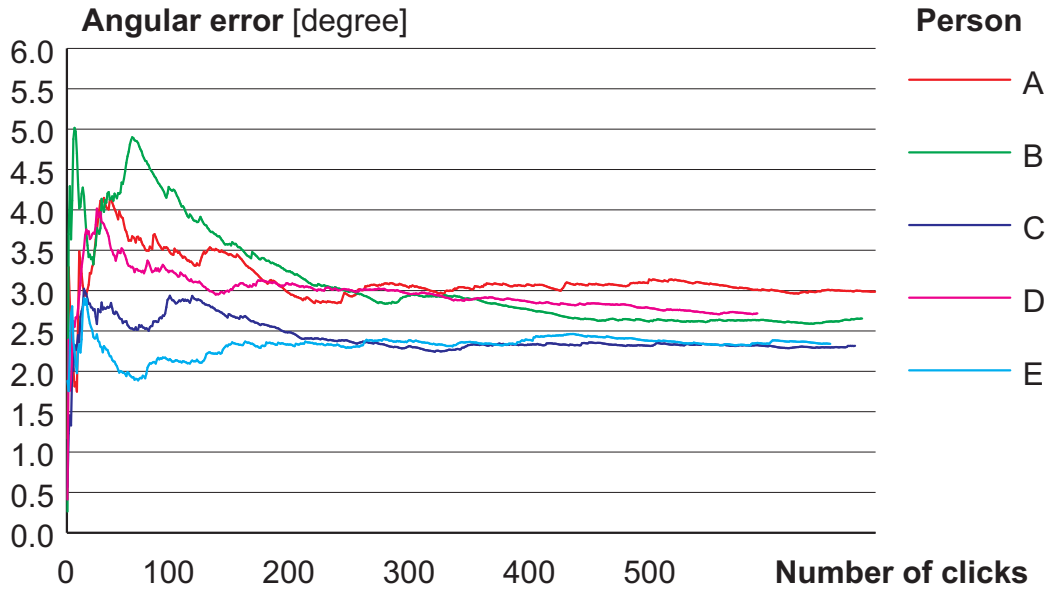


Figure 3.10: Cumulative average angular error in real environments as in Figure 3.8. Each line corresponds to each user in Table 3.2.

ranges of head-pose movements. The user IDs (A-E) correspond to the IDs in the simulation experiment. The accuracy of estimating gazes was as good as in the simulation experiment, *i.e.*, 2.6 degrees on average. Figure 3.10 shows the cumulative average of weighted angular error, similar to Figure 3.8. We observed that error converged similarly to that in the simulation results.

One of the most important factors in this experimental setting was the biased distribution of the click positions. Figure 3.11 shows the actual distribution of the positions clicked by User A. As in this example, there is a certain bias in the distribution of clicked positions. For example, we can see more clicks on menu buttons and at the top of the desktop. The distribution of click points in the real-world scenario is always biased as in this example.

Figure 3.12 shows a comparison between distributions of clicked points and average errors. Left image shows the spatial histogram of the clicked points in the experiments. Higher intensity corresponds to larger counts of

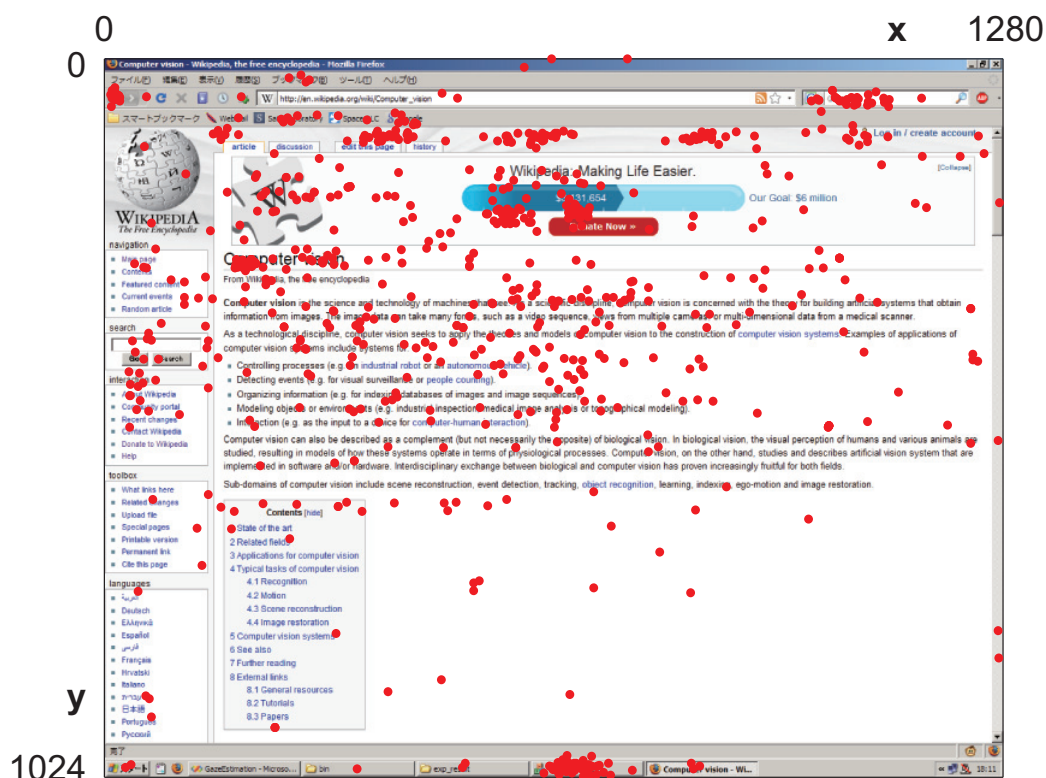


Figure 3.11: Distribution of points clicked by User A, during experiments in real environment. Each point indicates the coordinates of the clicked points in the 1280×1024 desktop region.

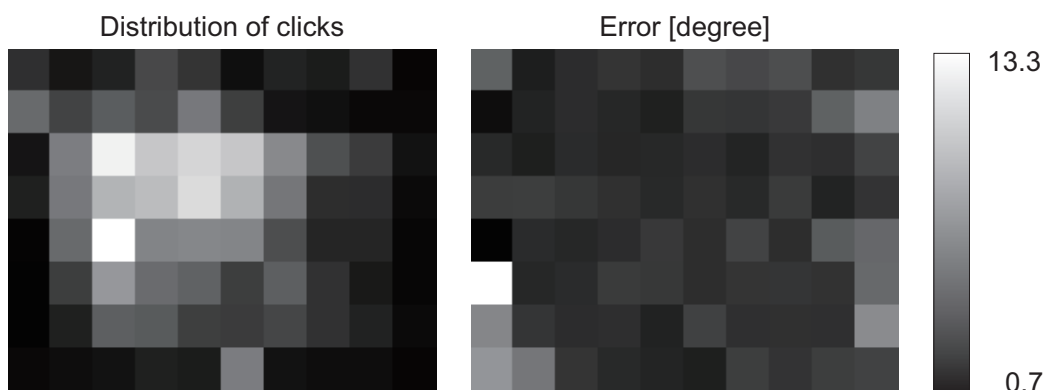


Figure 3.12: Comparison between click distribution and average error. Left image shows the spatial histogram of the clicked points in the experiments. Higher intensity corresponds to larger counts of clicks. Right image shows the spatial distribution of estimation errors in the display coordinate. Lower intensity corresponds to the lower estimation error as illustrated in the right bar.

clicks. Right image shows the spatial distribution of estimation errors in the display coordinate. Lower intensity corresponds to the lower estimation error as illustrated in the right bar. As can be seen, it is certainly hard to achieve good estimates at a screen point where only a few samples are available.

This can be considered as a prior for gaze estimates, as this describes the statistics of clicked points. However, at the same time, we did not expect users to click such locations based on the prior. One interesting observation is that the distribution varies with the tasks and application scenarios. Our method can naturally stay updated with changes in the distribution because of the incremental learning scheme.

We also present additional results to validate the above results that used clicked positions as ground truths. It can be argued that clicked positions do not exactly correspond to true gaze positions. Since it is almost impossible to acquire ground-truth gaze points across every head pose, we have conducted

Table 3.3: Results of experiments in real environment with ground-truth gaze points. Shown are the angular and pixel errors, numbers of clicks.

Person	Angular error [deg]	Pixel error [px]	Num. used clicks
a	2.3 ± 1.4	90 ± 56	181
b	3.1 ± 1.9	124 ± 76	162
c	4.7 ± 2.6	187 ± 104	152
Average	3.4 ± 2.0	133 ± 79	

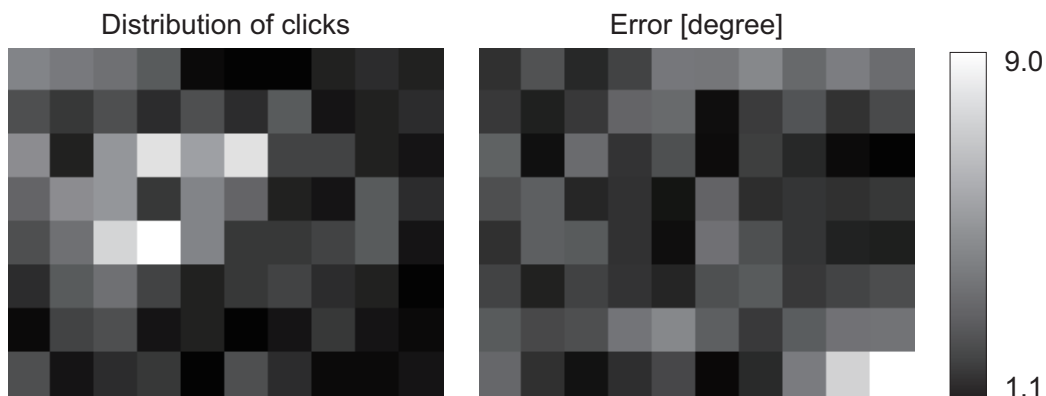


Figure 3.13: Comparison between click distribution and average error. Right image shows average estimation errors of reference data. Each grid corresponds to displayed positions of the reference points. Left image shows the spatial histogram of the clicked points as in Figure 3.12.

experiments under a fixed head pose. Three persons were asked to browse the Web for about 10 minutes with a chin rest to fix their head. Afterward, equally spaced 10×8 reference points (32 pixels in diameter) were shown to them. When they click the reference points, actual positions of the displayed points were recorded instead of clicked positions. Learned gaze estimators were evaluated using these 80 input data with ground-truth gaze points.

Table 3.3 lists the results of the experiment. From left to right in the table, angular and pixel errors of the raw output without temporal averaging, and numbers of used clicks are shown. In Figure 3.13, right image shows average estimation errors at each position of reference points. Lower intensity corresponds to the lower estimation error as illustrated in the right bar. Left image shows the spatial histogram of the clicked points during experiments as in Figure 3.12. Compared to Table 3.2 and Figure 3.12, errors in these cases are not significantly different from the previous results. Although these are simpler cases with a fixed head pose, it ensures the evaluation with clicked positions as ground truths.

3.6 Conclusions

We proposed an appearance-based gaze estimation method based on an incremental-learning approach. To avoid the lengthy calibration stage, we couple the learning and prediction stages to continuously refine the mapping function that is related to appearance and the gaze direction. The proposed approach is implemented in a desktop scenario, where a user clicks a mouse in casual PC operations. The clicked position is used as a gaze label, and the head pose and appearance of the eye are recorded with a PC camera. To handle free-head movement, we use 3D head-pose tracking and proposed a clustering-based method of interpolating appearance. The interpolation is done by clustering learning samples with similar head poses and creating a

local manifold in a cluster. We further introduced a method of refining labels to impose local smoothness on the manifold.

We use a PCA-based distance measure in our implementation to achieve computational efficiency and simplify implementation. However, it can become sensitive to variations in appearance that are unrelated to gaze, *e.g.*, small shifts and rotations in image cropping. To avoid this, we carry out subspace-based eye alignment after cropping as described in Section 3.4.1.

The effectiveness of the method we propose was demonstrated through experiments conducted in both simulation and real environments. Our method achieved an estimation accuracy of 2.6 degrees. It was less accurate than state-of-the-art products for estimating gazes that achieve an accuracy of less than 1 degree (*e.g.*, Tobii eye trackers [Tob]). However, the new method has immense advantages in that, unlike these approaches, it works with a minimal amount of equipment, *i.e.*, a single PC camera without any special hardware. Moreover, Our system achieved higher accuracy than the previously proposed monocular methods [IBMK04, YUYA08]. We believe that the proposed approach has considerable potential for developing an practical gaze estimator without a laborious calibration stage. With more advanced algorithms like [WBC06] to solve the regression problem in each cluster, we expect that the accuracy of estimation with the system will be further improved.

Chapter 4

Calibration-free Gaze Sensing using Saliency Maps

The previous chapter described a gaze estimation method that employs mouse operations on PCs as a key for the calibration. However, the method has a significant drawback that it cannot be applied to cases without user interaction. Finding more universal information about where a person is looking at, will extend the idea of unconscious calibration from human behavior to passive environments without user interactions.

In this chapter, we propose a novel, calibration-free gaze sensing method using visual saliency maps. Our method uses visual saliency maps of video frames that are computed in a bottom-up manner. By relating the saliency maps with appearances of eyes of a person watching video frames, our method automatically constructs a gaze estimator. To efficiently identify gaze points from saliency maps, we aggregate saliency maps to build a probability distribution of gaze points. We establish mapping between eye images to gaze points by Gaussian process regression.

4.1 Introduction

Gaze estimation is important for predicting human attention, and therefore can be used for various interactive systems. There are a wide range of applications of gaze estimation including marketing analysis of online content and digital signage, gaze-driven interactive displays, and many other human-machine interfaces.

In general, gaze estimation is achieved by analyzing a person's eyes with an image sensor. Exact gaze points can be determined by directly analyzing gaze directions from observations of eyes. Many implementations of camera-based gaze estimator have been proposed including commercial products (see [HJ09] for a recent survey). One of the limitations of camera-based gaze estimators is explicit calibration for learning person-dependent parameters. Although the number of reference points for calibration can be reduced using multiple light sources [VC08], or stereo cameras [NKIT08], it still requires a user to actively participate in the calibration task. In some practical scenarios, the active calibration is too restrictive because it interrupts natural interactions and makes the unnoticeable gaze estimation impossible.

To avoid active calibration, Yamazoe *et al.* used a simple eyeball model for gaze estimation and performed automatic calibration by fitting the model to appearance of a user's eye [YUYA08]. Sugano *et al.* proposed a method using input from a user's mouse as exemplar data for calibration [SMSK08]. However, these approaches are restricted to specific scenarios. Yamazoe *et al.*'s approach relies on a specific geometric model, and Sugano *et al.*'s approach can only be applied to interactive environments with user inputs.

Apart from these gaze estimation studies, computational models of visual saliency have been studied to analyze visual attention on an image. While gaze estimation approaches aim at determining where people's eyes actually look at, the visual saliency give us information about which im-

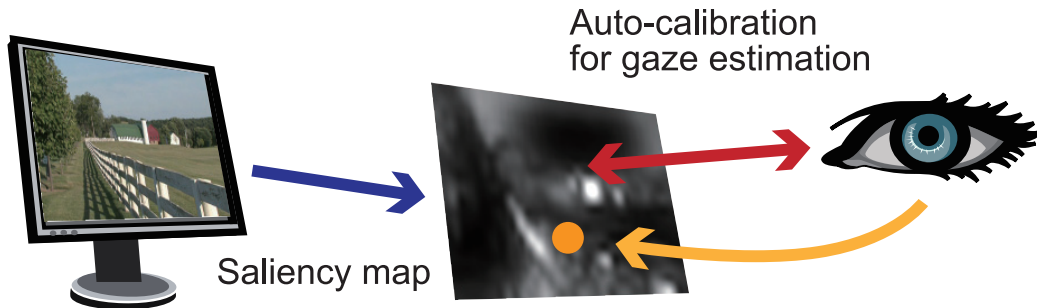


Figure 4.1: Illustration of our method. Our method uses saliency maps computed from video frames in bottom-up manner for automatically constructing a gaze estimator.

age region attracts more attention, as illustrated in Figure 4.1. Biologically, humans tend to gaze at an image region with high saliency, *i.e.*, a region containing more unique and distinctive visual features compared with the surrounding regions. Hence, by knowing the visual saliency map of an image, the gaze point of a person looking at an image can be predicted. After Koch and Ullman proposed the original concept [KU85] of visual saliency, many bottom-up computational models of visual saliency maps have been proposed [IKN98, PS00, IB06, HKP07, BT09]. It is experimentally shown that there indeed exists a correlation between bottom-up visual saliency and fixation locations [PLN02].

Gaze estimation and visual saliency models are closely related; nonetheless, not many previous studies relate these two. Kienzle *et al.* [KWSF06, KSWF07] proposed a method for learning computational models of bottom-up visual saliency using gaze estimation data. Judd *et al.* [JEDT09] followed the approach with more features and a larger database. These approaches learn accurate saliency models using gaze points. In contrast to these methods, our goal is to construct a gaze estimator from saliency maps. To our knowledge, this is the first work to use visual saliency as prior information for gaze estimation.

We propose a novel calibration-free gaze sensing method using computational visual saliency. The key idea of our method is generating a probability distribution of gaze points using saliency maps. From eye images of a user watching a video clip, we acquire learning datasets that consists of saliency maps and eye images under a fixed head position. Gaze probability maps are generated by aggregating the saliency maps based on the similarity of eye appearances. Once the gaze probability maps are obtained, our method learns the relationship between the gaze probability maps and eye images. As a result, this leads to a completely ambient gaze estimator that exempts users from active calibration.

4.2 Gaze estimation from saliency maps

Our goal is to construct a gaze estimator without any explicit calibration stages. The inputs for our system are N video frames $\{\mathbf{I}_1, \dots, \mathbf{I}_N\}$ and associated eye images $\{\mathbf{e}_1, \dots, \mathbf{e}_N\}$ of a person watching a video clip under a fixed head position. In our setting, eye images and video frames are synchronized; eye image \mathbf{e}_i is captured at the same time when frame \mathbf{I}_i is shown to the person. Using this dataset $\{(\mathbf{I}_1, \mathbf{e}_1), \dots, (\mathbf{I}_N, \mathbf{e}_N)\}$, our goal is to construct a gaze estimator for estimating an unknown gaze point \mathbf{g} from an eye image \mathbf{e} .

Our method consists of three steps; saliency extraction, saliency aggregation, and estimator construction as shown in Figure 4.2. Saliency extraction, is the step in which saliency maps from an input video are calculated. From the video clips, a visual saliency map that represents distinctive visual features is extracted from each frame. Saliency aggregation combines all saliency maps to obtain a gaze probability map that has a peak around the true gaze point. This step produces pairs of the average eye image and gaze probability map. The third step is the estimator construction. Using the gaze

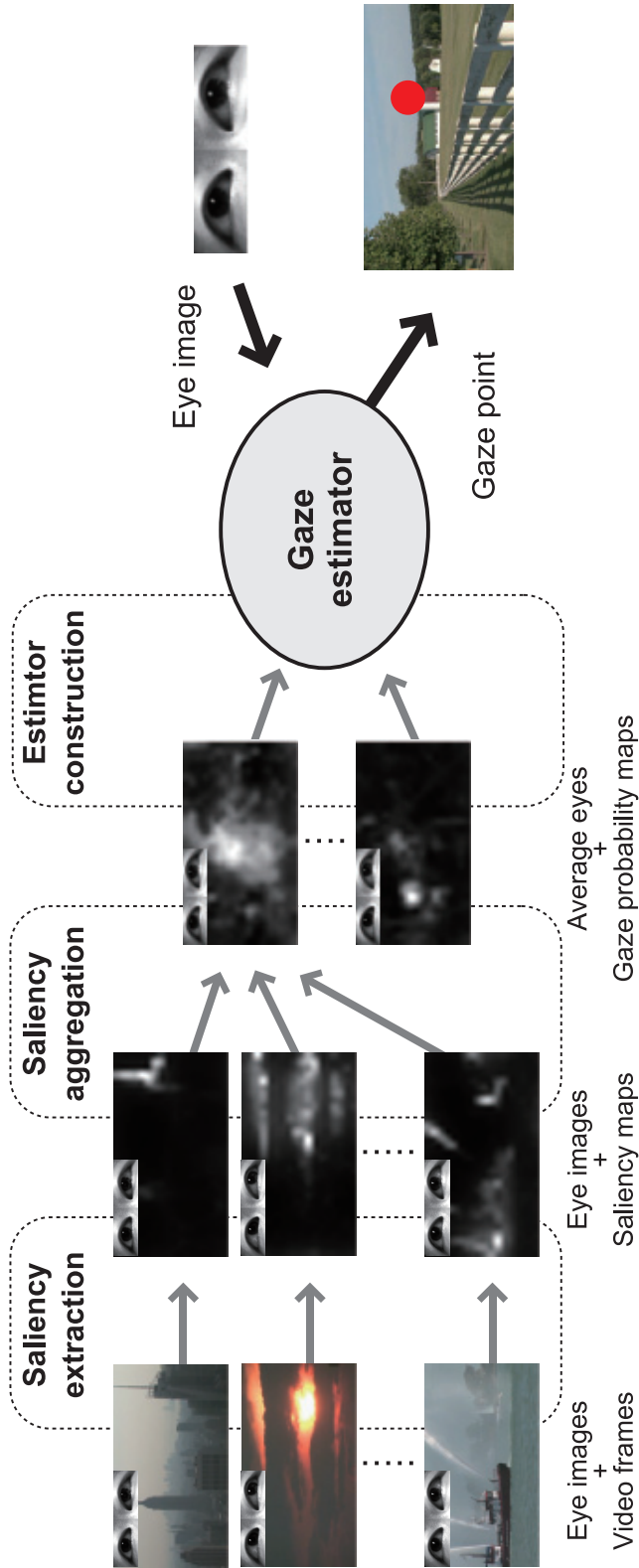


Figure 4.2: Proposed framework. Our method consists of three steps. Saliency extraction step computes saliency maps from the input video. Saliency aggregation step combines saliency maps to produce gaze probability maps. Using the gaze probability maps and associated eye images, the estimator construction step learns the mapping from an eye image to a gaze point.

probability maps and associated eye images, the estimator construction step learns the mapping from an eye image to a gaze point. The resulting gaze estimator outputs gaze points for any eye image from the person. Details of each step are described in the following sections.

4.2.1 Saliency extraction

This step extracts visual saliency maps $\{\mathbf{s}_1, \dots, \mathbf{s}_N\}$ from input video frames $\{\mathbf{I}_1, \dots, \mathbf{I}_N\}$. The flow of the extraction scheme is summarized in Figure 4.3. The basic concept of bottom-up saliency map is evaluating unusual regions in these feature maps. If some locations in the feature maps have unique values compared to its neighbors, they are thought to be more salient for humans. We use graph-based visual saliency (GBVS) [HKP07] as a base saliency model and add a higher level saliency model based on face detection.

First, low-level feature maps are extracted from the input video frame. We employ commonly used feature channels, *i.e.*, color, intensity and orientation as static features, and flicker and motion are used as dynamic features [IDP03]. Color indicates red/green and blue/yellow differences, intensity indicates a grayscale luminance, and orientation indicates responses of 2D Gabor filters with orientation at 0° , 45° , 90° and 135° , flicker indicates an absolute intensity difference from the previous frame and motion indicates spatially-shifted differences between Gabor responses. Each feature channel is extracted at different 2 levels of a image pyramid with $1/2$ and $1/4$ scaling. Hence a total of 24 maps are extracted.

Next, Activation map \mathbf{A} is computed from each feature map \mathbf{F} . In the GBVS algorithm, this process is done in a form of steady-state analysis of a Markov chain G_A . Each state of G_A corresponds to a pixel in \mathbf{A} and a transition probability ω_a between states (i, j) and (p, q) is defined based on

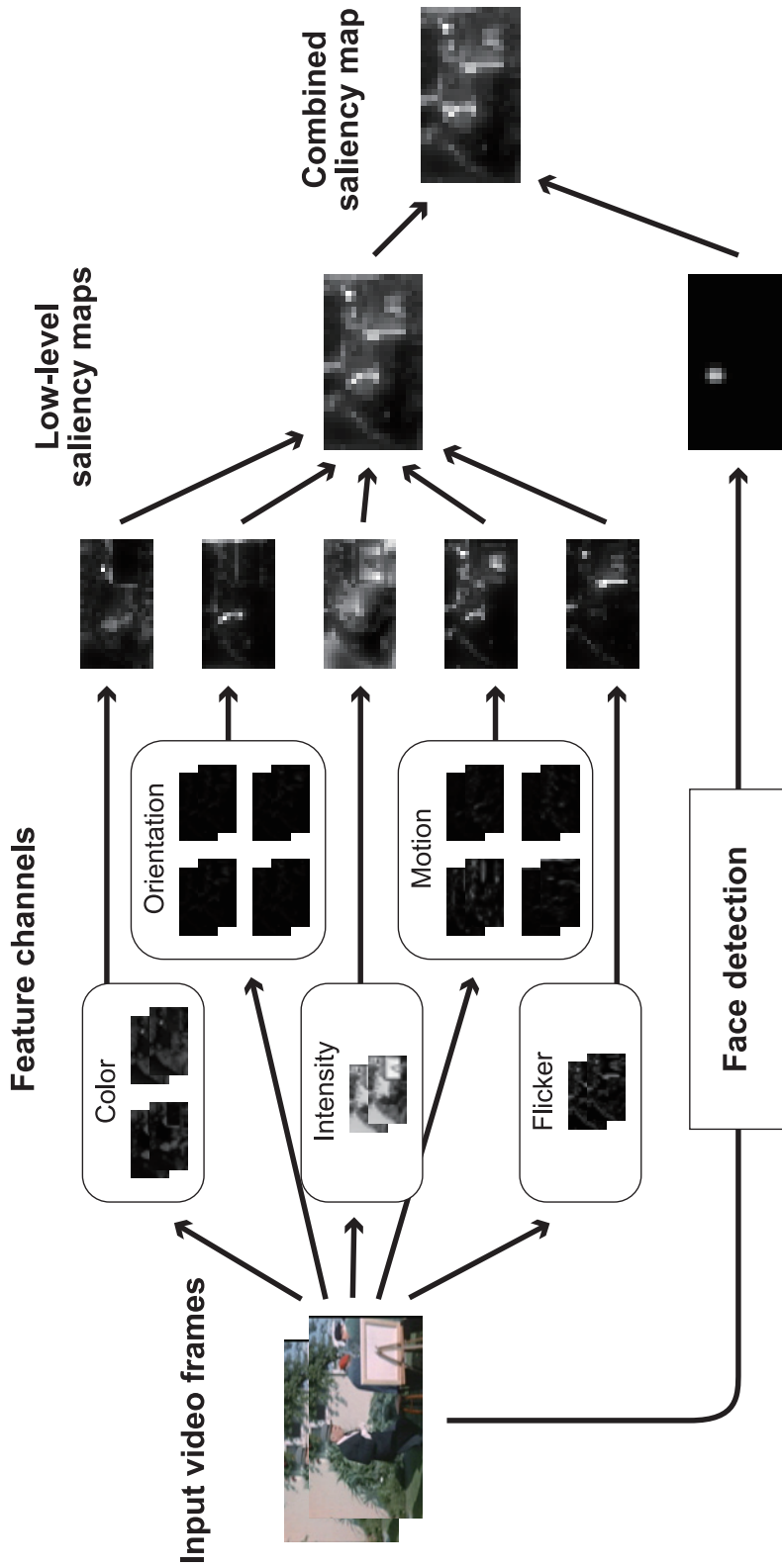


Figure 4.3: Process flow of saliency extraction. We use graph-based visual saliency [HKP07] as a base saliency model and add a higher level saliency model based on face detection.

a dissimilarity between two corresponding pixels in \mathbf{F} as

$$\omega_a((i, j), (p, q)) \triangleq \omega_d |\mathbf{F}(i, j) - \mathbf{F}(p, q)|, \quad (4.1)$$

where ω_d indicates a Gaussian weight evaluating Euclidean distance between (i, j) and (p, q) . In this way, pixels with higher dissimilarity with their surroundings have higher value in an equilibrium distribution of G_A .

Resulting activation maps are then normalized. A Markov chain G_N is defined in a similar way with a transition probability:

$$\omega_n((i, j), (p, q)) \triangleq \omega_d |\mathbf{A}(p, q)|. \quad (4.2)$$

In an equilibrium distribution of G_N , maps are concentrated so that they have fewer important peaks. Normalized maps are averaged within each channel and all channel maps are finally combined with a same weight to form a low-level saliency map \mathbf{s}^l .

On top of the low-level saliency, we use a higher level saliency model. Humans tend to fixate on faces, especially the eyes, which are highly salient for humans. Cerf *et al.* [CHEK08] proposed a face channel-based saliency model using a face detector. We follow this approach to produce reliable saliency maps using a facial feature detector (OKAO Vision library developed by OMRON Corporation). The face channel saliency map \mathbf{s}^f is modeled as 2-D Gaussian circles with a fixed variance at the detected positions of the center between two eyes. When the detector detects only a face, *e.g.*, when the face region is small, the facial saliency is defined at the center of the facial region.

We combine the low-level saliency \mathbf{s}^l and face channel saliency \mathbf{s}^f after normalizing them to span in the same range. We also take the temporal average of the aggregated saliency maps to compute the final saliency map \mathbf{s} as

$$\mathbf{s}_i = \frac{1}{2(n_s + 1)} \sum_{j=i-n_s}^i (\mathbf{s}_j^l + \mathbf{s}_j^f), \quad (4.3)$$

where n_s is the number of frames used for temporal smoothing. Since humans cannot instantly follow rapid scene changes, only past frames are used for the smoothing to account for latency. As a result, synchronized pairs of saliency maps and eye images $\mathcal{D}_s = \{(\mathbf{s}_1, \mathbf{e}_1), \dots, (\mathbf{s}_N, \mathbf{e}_N)\}$ are produced.

Figure 4.4 shows examples of the computed saliency maps. From top to bottom are input images \mathbf{I} , low-level saliency \mathbf{s}^l , face channel saliency \mathbf{s}^f , and combined saliency maps \mathbf{s} .

4.2.2 Saliency aggregation

Although saliency maps extracted in the previous step accurately predict gaze points, the accuracy is still not good enough to determine exact locations of gaze points.

The saliency maps \mathbf{s} encode distinctive visual features in a video frame. While a saliency map does not provide exact gaze points, highly salient regions in a saliency map are likely to coincide with the true gaze point. Suppose we have a set of saliency maps that statistically have high saliency scores around the true gaze point, with random saliency scores at other regions. By aggregating the saliency maps, it is expected that the image region around the true gaze point has a vivid peak of saliency. The map can be used as the probability distribution of the gaze point. This step aims at producing such probability maps using the associated eye images.

Our basic idea is to use a similarity of eye images for the aggregation. The similarity measure w_s is defined as

$$w_s(\mathbf{e}_i, \mathbf{e}_j) = \exp(-\kappa_s \|\mathbf{e}_i - \mathbf{e}_j\|^2). \quad (4.4)$$

When the gaze points of eye images \mathbf{e}_i and \mathbf{e}_j are close, the appearances are similar and w_s becomes high.

In this step, we first eliminate unreliable eye images, *e.g.*, images during blinking, from the learning set. Eye images recorded during fixation are

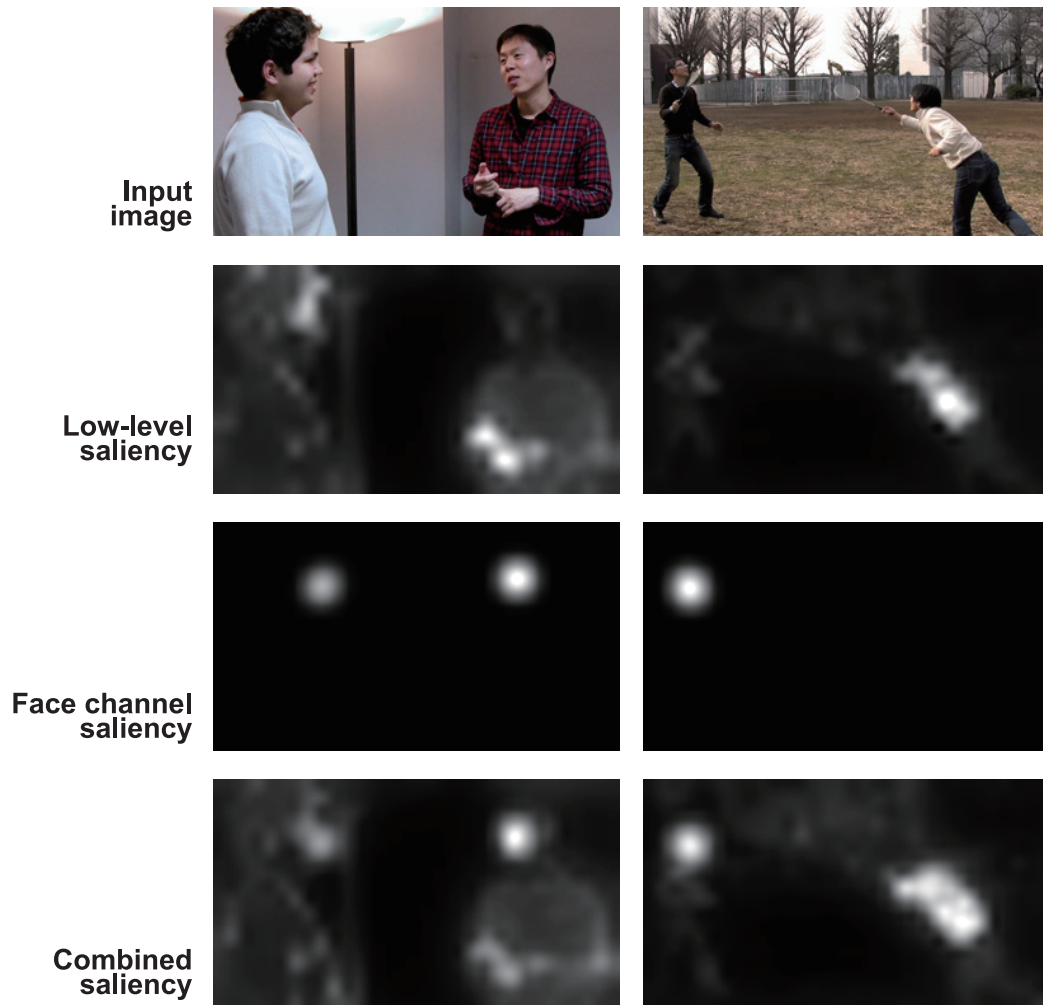


Figure 4.4: Examples of computed saliency maps. First row shows input images I , second row shows corresponding low-level saliency s^l , third row corresponds to face channel saliency s^f , and the bottom row shows combined saliency maps s .

useful as a learning data. To identify such eye images, we use a fixation measure of an eye image \mathbf{e} defined as

$$w_f(\mathbf{e}_i) = \exp(-\kappa_f \text{Var}(\mathbf{e}_i)), \quad (4.5)$$

where $\text{Var}(\mathbf{e}_i)$ denotes the variance of eye images $\{\mathbf{e}_{i-n_f}, \dots, \mathbf{e}_{i+n_f}\}$ averaged over pixels. Since appearances of the eye images change rapidly during fast movement, w_f becomes lower when \mathbf{e}_i is captured during eye movement or blinking. A subset $\mathcal{D}_{s'} = \{(\mathbf{s}_1, \mathbf{e}_1), \dots, (\mathbf{s}_{N'}, \mathbf{e}_{N'})\}$ is created from \mathcal{D}_s by removing eye images where w_f scores are lower than a predefined threshold τ_f .

Since variation in the gaze points is limited in $\mathcal{D}_{s'}$, and there can be many samples that share almost the same gaze point, eye images are clustered according to similarity w_f to reduce redundancy and computational cost. Each eye image \mathbf{e} is sequentially added to the cluster whose average eye image $\bar{\mathbf{e}}$ is the most similar to \mathbf{e} . A new cluster is adaptively created if the highest similarity among all existing clusters is lower than the threshold τ_s . Finally, M clusters and their average eye images $\{\bar{\mathbf{e}}_1, \dots, \bar{\mathbf{e}}_M\}$ are calculated.

After these steps, we compute the gaze probability map $\bar{\mathbf{e}}_i$ as

$$\bar{\mathbf{p}}_i = \frac{\sum_j^{N'} w_s(\bar{\mathbf{e}}_i, \mathbf{e}_j)(\mathbf{s}_j - \bar{\mathbf{s}}_{\text{all}})}{\sum_j^{N'} w_s(\bar{\mathbf{e}}_i, \mathbf{e}_j)}, \quad (4.6)$$

where $\bar{\mathbf{s}}_{\text{all}}$ is the average of saliency maps in $\mathcal{D}_{s'}$. Man-made pictures usually have higher saliency at the center of the image, Hence, without normalization, the gaze probability map $\bar{\mathbf{p}}_i$ tends to have higher value at the center regardless of $\bar{\mathbf{e}}_i$. The average saliency map $\bar{\mathbf{s}}_{\text{all}}$ is used to eliminate this centering bias in the gaze probability map. Each gaze probability map $\bar{\mathbf{p}}_i$ is normalized to a fixed range. Finally, we obtain a dataset $\mathcal{D}_p = \{(\bar{\mathbf{p}}_1, \bar{\mathbf{e}}_1), \dots, (\bar{\mathbf{p}}_M, \bar{\mathbf{e}}_M)\}$.

Figure 4.5 shows examples of the obtained gaze probability maps. The eye images shown at the top-left indicate corresponding average eye images $\bar{\mathbf{e}}$. The six images are some examples taken from six different people. The

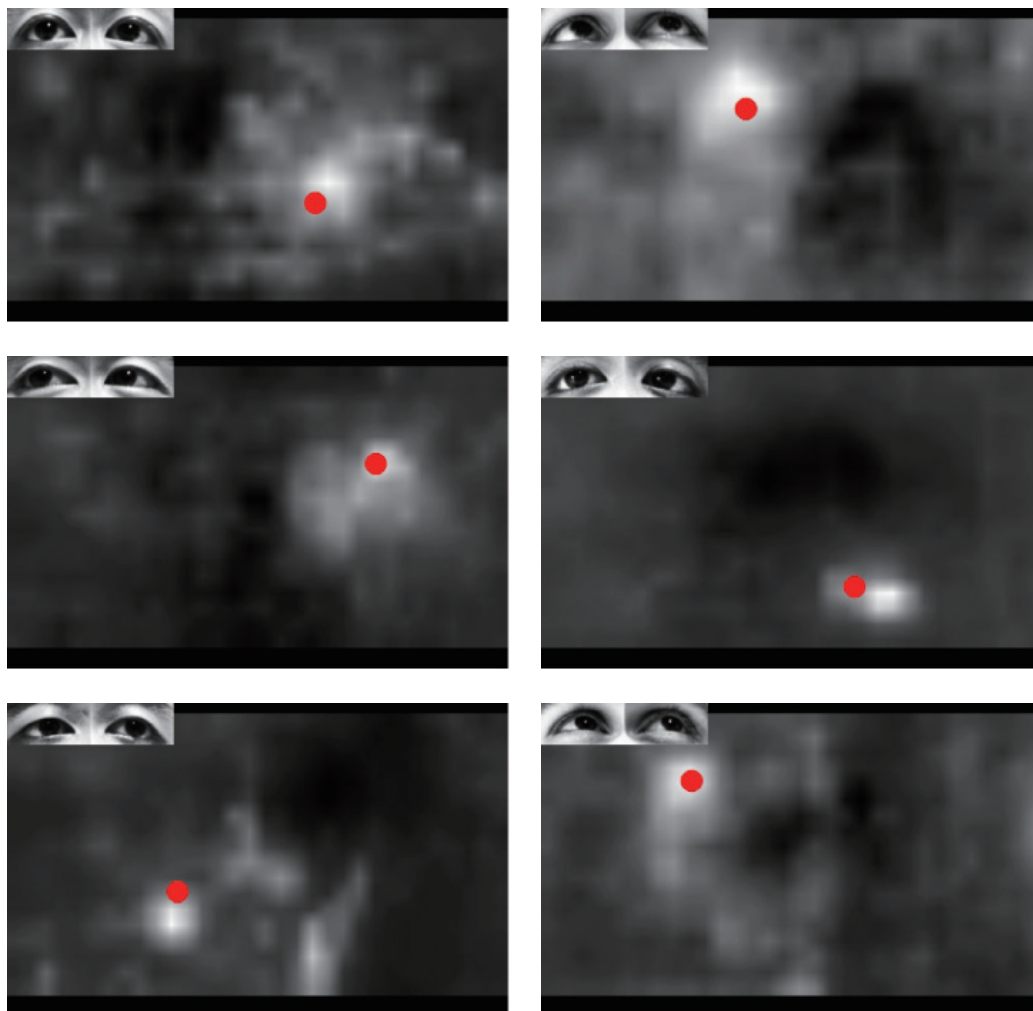


Figure 4.5: Examples of gaze probability maps \bar{p} and corresponding average eye images \bar{e} . Overlaid circles depict true gaze points of \bar{e} to illustrate the correspondence between a gaze point and the peak in the gaze probability. The true gaze points are obtained using a calibration-based gaze estimator, and our method does not know the true gaze points.

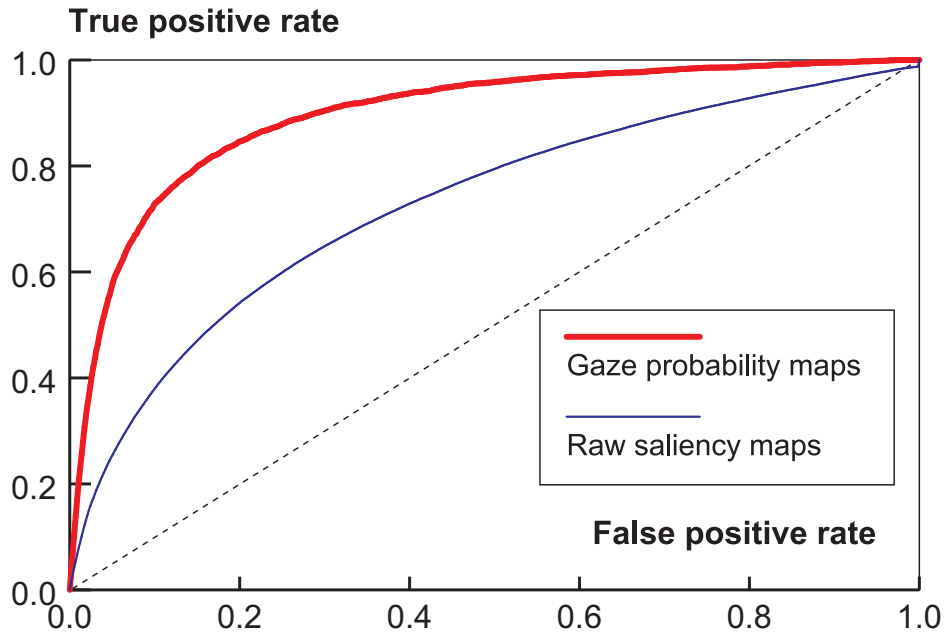


Figure 4.6: ROC curves of raw saliency maps and gaze probability maps. Horizontal axis indicates the false positive rate, *i.e.*, pixel rate above a threshold. Vertical axis indicates the true positive rate, *i.e.*, rate of frames which have a higher saliency value than a threshold at the true gaze point. Thin line (AUC = 0.73) indicates raw saliency maps extracted through process described in Section 4.2.1. Bold line (AUC = 0.90) corresponds to the gaze probability maps described in Section 4.2.2.

overlaid circles indicate true gaze points of \bar{e} . The true gaze points are unknown in our method, and these are obtained using a calibration-based gaze estimator and placed as a reference. Although the gaze probability maps \bar{p}_i are generated without knowing true gaze points, these highly correspond to the true gaze points.

Figure 4.6 shows the improvement of the correlation between the true gaze point and saliency maps by this aggregation step. The curves are drawn by changing saliency threshold values from minimum to maximum. The horizontal axis indicates a false positive rate, *i.e.*, rate of pixels in a map

above a threshold. The vertical axis indicates a true positive rate, *i.e.*, rate of frames whose saliency value at the true gaze point is higher than the threshold. This plot is obtained using all data used in our experiment. The thin line shows the average receiver operating characteristic (ROC) curve (area under the curve (AUC) = 0.73) of the extracted saliency maps before aggregation. After aggregation, the accuracy is improved as shown by the bold line in Figure 4.6, which shows the average ROC curve (AUC = 0.90) of all the gaze probability maps.

4.2.3 Estimator construction

In the previous step, the average eye images $\{\bar{\mathbf{e}}_1, \dots, \bar{\mathbf{e}}_M\}$ and corresponding gaze probability maps $\{\bar{\mathbf{p}}_1, \dots, \bar{\mathbf{p}}_M\}$ are produced.

In a standard Gaussian process regression, a model can be built to estimate the probability distribution $P(\mathbf{g}^* | \mathbf{e}^*, \mathcal{D}_g)$ of an unknown gaze point \mathbf{g}^* of an eye image \mathbf{e}^* , given labeled data points $\mathcal{D}_g = \{(\mathbf{g}_1, \bar{\mathbf{e}}_1), \dots, (\mathbf{g}_M, \bar{\mathbf{e}}_M)\}$. However in our case, we only know $\mathcal{D}_p = \{(\bar{\mathbf{p}}_1, \bar{\mathbf{e}}_1), \dots, (\bar{\mathbf{p}}_M, \bar{\mathbf{e}}_M)\}$ where $\bar{\mathbf{p}}_i$ can be treated as a probability map of \mathbf{g}_i .

Therefore, we re-formulate Gaussian process regression using the probability maps as follows. By normalizing the gaze probability maps, we define probability distributions as

$$P(\mathbf{g} | \bar{\mathbf{p}}) = \frac{\bar{\mathbf{p}}(\mathbf{g})}{\sum_x \sum_y \bar{\mathbf{p}}}, \quad (4.7)$$

where $\bar{\mathbf{p}}(\mathbf{g})$ indicates the value of $\bar{\mathbf{p}}$ at the gaze point \mathbf{g} , and $\sum_x \sum_y \bar{\mathbf{p}}$ indicates overall summation of $\bar{\mathbf{p}}$. Given Eq. (4.7), the target distribution $P(\mathbf{g}^* | \mathbf{e}^*, \mathcal{D}_p)$ can be obtained by marginalizing over all possible gaze points

$\{g_1, \dots, g_M\}$ ¹ as

$$P(g^*|\mathbf{e}^*, \mathcal{D}_p) = \sum_{g_1} \cdots \sum_{g_M} P(g^*|\mathbf{e}^*, \mathcal{D}_g)P(\mathcal{D}_g|\mathcal{D}_p), \quad (4.8)$$

where

$$P(\mathcal{D}_g|\mathcal{D}_p) = \prod_i^M P(g_i|\bar{\mathbf{p}}_i). \quad (4.9)$$

In Eq. (4.8), g^* indicates the unknown gaze point of the eye image \mathbf{e}^* , and g_i is the gaze point corresponding to $\bar{\mathbf{p}}_i$.

Because the integral (summation) of Eq. (4.8) is computationally expensive, we solve Eq. (4.8) by Monte Carlo approximation. We randomly produce n_g sets of samples $\mathcal{D}_g^{(l)} = \{(g_1^{(l)}, \bar{\mathbf{e}}_1), \dots, (g_M^{(l)}, \bar{\mathbf{e}}_M)\}_{l=1}^{n_g}$ according to the probability distribution defined by Eq. (4.7). Namely, $g_i^{(l)}$ in the l -th set is generated according to the distribution $P(g_i|\bar{\mathbf{p}}_i)$ defined by the i -th probability map. Because the gaze probability maps accurately predict true gaze points as shown in Figure 4.6 (85% of the true gaze points were included within the top 20% of the aggregated saliency scores), we cut off the low saliency values from the gaze probability maps. To reduce the number of samples in the approximation, we use a threshold τ_s to set the probability to zero if $\bar{\mathbf{p}}(x, y)$ is lower than the threshold. Using these sets, Eq. (4.8) can be approximated as

$$P(g^*|\mathbf{e}^*, \mathcal{D}_p) = \frac{1}{n_g} \sum_{l=1}^{n_g} P(g^*|\mathbf{e}^*, \mathcal{D}_g^{(l)}). \quad (4.10)$$

Finally, each $P(g^*|\mathbf{e}^*, \mathcal{D}_g^{(l)})$ can be estimated based on a Gaussian process regression [RW06].

¹Here, we describe a one-dimensional case to simplify the notation; however, two regressors are independently built for each X- and Y-direction.

Gaussian process regression We assume a noisy observation model $g_i = f(\mathbf{e}_i) + \epsilon_i$, *i.e.*, a gaze point g_i is given as a function of \mathbf{e}_i with the data-dependent noise term $\epsilon_i = \mathcal{N}(0, \zeta_i^2)$. In standard methods, the noise variance ζ^2 is treated as an unknown parameter that takes a constant value across all data. This assumption does not always hold in practice, and there have been proposed some methods [GWB98, KPPB07] to learn an input-dependent noise variance function. In our case, because the sample distribution is known, the noise variance ζ_i^2 can be set to an actual variance of generated samples $\{g_i^{(1)}, \dots, g_i^{(n_g)}\}$. It explicitly assigns a higher noise variance for samples from ambiguous saliency maps with several peaks. $f(\mathbf{e}_i)$ is assumed to be a zero-mean Gaussian process with a covariance function k :

$$k(\mathbf{e}_i, \mathbf{e}_j) = \alpha \exp(-\beta \|\mathbf{e}_i - \mathbf{e}_j\|^2), \quad (4.11)$$

with parameters α and β . With this assumption, $P(g^* | \mathbf{e}^*, \mathcal{D}_g^{(l)})$ is calculated as a Gaussian distribution $\mathcal{N}(\mu_l, \sigma_l^2)$ with

$$\mu_l = \mathbf{K}^* (\mathbf{K} + \mathbf{S})^{-1} \mathbf{G}^{(l)}, \quad (4.12)$$

and

$$\sigma_l^2 = k(\mathbf{e}^*, \mathbf{e}^*) - \mathbf{K}^* (\mathbf{K} + \mathbf{S})^{-1} \mathbf{K}^*, \quad (4.13)$$

where $\mathbf{K}_{ij} = k(\bar{\mathbf{e}}_i, \bar{\mathbf{e}}_j)$, $\mathbf{K}_i^* = k(\bar{\mathbf{e}}_i, \mathbf{e}^*)$, $\mathbf{S}_{ij} = \zeta_i^2 \delta_{ij}$ and $\mathbf{G}_i^{(l)} = g_i^{(l)}$ ² [Bis06]. As a result, the distribution $P(g^* | \mathbf{e}^*, \mathcal{D}_p)$ can be derived as a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with

$$\mu = \frac{1}{n_g} \sum_{l=1}^{n_g} \mu_l, \quad \sigma^2 = \frac{1}{n_g} \sum_{l=1}^{n_g} \sigma_l^2 = \sigma_1^2. \quad (4.14)$$

The variance σ^2 equals to σ_1^2 , because σ_l^2 of Eq. (4.13) is independent of the index l . Therefore, σ^2 can be calculated by taking σ_1^2 .

² $\mathbf{K} \in \mathbb{R}^{M \times M}$, $\mathbf{K}^* \in \mathbb{R}^{1 \times M}$, $\mathbf{S} \in \mathbb{R}^{M \times M}$ and $\mathbf{G}^{(l)} \in \mathbb{R}^{1 \times M}$

4.2.4 Gaze estimation

Once we have matrices \mathbf{K} , \mathbf{S} and $\{\mathbf{G}^{(1)}, \dots, \mathbf{G}^{(n_g)}\}$ in Eqs. (4.12) and (4.13), a gaze point can be estimated by taking any eye image \mathbf{e} as input. The estimated distributions for each X- and Y-direction, $\mathcal{N}(\mu_x, \sigma_x^2)$ and $\mathcal{N}(\mu_y, \sigma_y^2)$, are converted to the display coordinates $\mathcal{N}(\hat{\mu}_x, \hat{\sigma}_x^2)$ and $\mathcal{N}(\hat{\mu}_y, \hat{\sigma}_y^2)$ as

$$\hat{\mu}_x = x_o + \frac{W_I}{W_s} \mu_x, \quad \hat{\mu}_y = y_o + \frac{H_I}{H_s} \mu_y, \quad (4.15)$$

and

$$\hat{\sigma}_x^2 = \frac{W_I}{W_s} \sigma_x^2, \quad \hat{\sigma}_y^2 = \frac{H_I}{H_s} \sigma_y^2, \quad (4.16)$$

where W_s , H_s indicates the width and height of the saliency maps, W_I , H_I indicates the actual width and height of the displayed images $\{\mathbf{I}_1, \dots, \mathbf{I}_N\}$, and (x_o, y_o) indicates the display origin of the images. The average $(\hat{\mu}_x, \hat{\mu}_y)$ corresponds to the estimated gaze point \mathbf{g} .

4.3 Experimental results

In this section, we show experimental results to evaluate our method. In the experiments, we used four video clips from four films: A) *2001: A Space Odyssey*, Stanley Kubrick, 1968, B) *Dreams*, Akira Kurosawa, 1990, C) *Nuovo Cinema Paradiso*, Giuseppe Tornatore, 1988 and D) *Forrest Gump*, Robert Zemeckis, 1994. It is known that human gaze control is also strongly influenced by contexts and plots of films, however, such high-level attentions are not modeled by the bottom-up saliency model we employed. Hence, each film was shortened to a 10-minute video clip without audio signal by extracting 2-second sequences at regular intervals to remove these effects. The video clips were resized to a fixed dimension of 720×405 , and the display resolution was set to $W_I = 1920$ and $H_I = 1080$. The video clips were shown



Figure 4.7: Experimental setup. A chin rest is used to fix their head positions, and a 22.0-inches WUXGA (473.8×296.1 mm) display was placed 400 mm in front of the subject when video clips were shown. The red circle indicates the position of the camera.

Table 4.1: Combinations of video clips A to D and test subjects s_1 to s_6 ; *e.g.*, person s_1 watched clips A and B.

Source	Test			
	A	B	C	D
A		s_1	s_2	s_3
B	s_1		s_4	s_5
C	s_2	s_4		s_6
D	s_3	s_5	s_6	

at 15 fps; therefore, $N = 9000$ in the experiments. The saliency maps were calculated at a smaller resolution, $W_s = 32$ and $H_s = 18$.

Six novice test subjects $s_1 \dots s_6$ were asked to watch two video clips. The combinations of video clips and test subjects are defined so that every clip was tested as learning data against three different subject persons as listed in Table 4.1. A chin rest is used to fix their head positions, and a 22.0-inches WUXGA (473.8×296.1 mm) display was placed 400 mm in front of the subject when video clips were shown. Figure 4.7 shows the actual setup. The red circle indicates the position of the camera. While the subjects were watching the clips, their eyes were automatically detected and captured using OMRON OKAO Vision library.

The ground truth calibration data were collected for each user by showing reference points in a separate stage. For this, 16×9 points were shown at 120×120 -pixel intervals and eye images were captured in the same way. The ground truth was used to quantitatively assess our method in comparison with the gaze estimation method that involves an explicit calibration stage.

Throughout the experiment, the parameters were set as follows; $n_s = 5$, $\kappa_s = 7.8 \times 10^{-7}$, $\tau_s = 0.4$, $n_f = 5$, $\kappa_f = 0.02$, $n_g = 50$, $\alpha = 50$, $\beta = 5.0 \times 10^{-9}$, and τ_s was adaptively set to keep the top 20% of pixels and set remaining 80% to zero in each map. These parameter settings are empirically obtained from our experiment.

4.3.1 Gaze estimation results

Using the two clips \times six subject people, we tested our method in two scenarios. In Scenario 1, we assessed our method using the learning dataset as a test dataset. Because the true gaze points are not known in the learning dataset, this experiment was designed to verify the performance of the algorithm. In Scenario 2, evaluations were performed using another dataset from

the user as a test dataset to confirm the applicability of the trained gaze estimator to other datasets.

The ground truth gaze points of the datasets were obtained using a calibration-based gaze estimator. It was achieved by a standard Gaussian process regression method with a labeled data set. Namely, pairs of the ground-truth gaze points and eye images were explicitly given to learn the relationship between gaze points and eye images. The same covariance function (Eq. (4.11)) was used. α and β were set to be the same values as our estimator, and the noise variance ζ^2 was set to zero.

Figure 4.8 and Figure 4.9 shows examples of the estimation results. Outputs of the estimators are rendered as 2-D Gaussian circles centered at $(\hat{\mu}_x, \hat{\mu}_y)$ with variance $(\hat{\sigma}_x^2, \hat{\sigma}_y^2)$ given by Eq. (4.14). The center coordinate $(\hat{\mu}_x, \hat{\mu}_y)$ corresponds to the estimated gaze point. The eye images shown at the top-left corner show input eye images for estimation, and the overlaid circles represent true gaze points obtained from the calibration-based estimator.

Table 4.2 summarizes the estimation results for each video clips. Each row corresponds to the average of three subjects' results where the corresponding video clip is used as the training dataset (see Table 4.1). First two columns indicate AUCs of average ROC curves of the raw saliency maps \mathbf{s} and gaze probability maps $\bar{\mathbf{p}}$. The rest of the columns indicate estimation errors in distance and angle represented as *average \pm standard deviation*. Distance errors are evaluated as the Euclidean distance between the estimated and ground-truth gaze points, and angular errors are calculated using the distance between eyes and the display.

From these results, it is observed that the gaze estimation accuracy depends on the accuracy of the gaze probability maps. When the AUC of the gaze probability maps $\bar{\mathbf{p}}$ is lower, the estimation error tends to become larger.

Table 4.3 lists the estimation error of each subject person. Each row

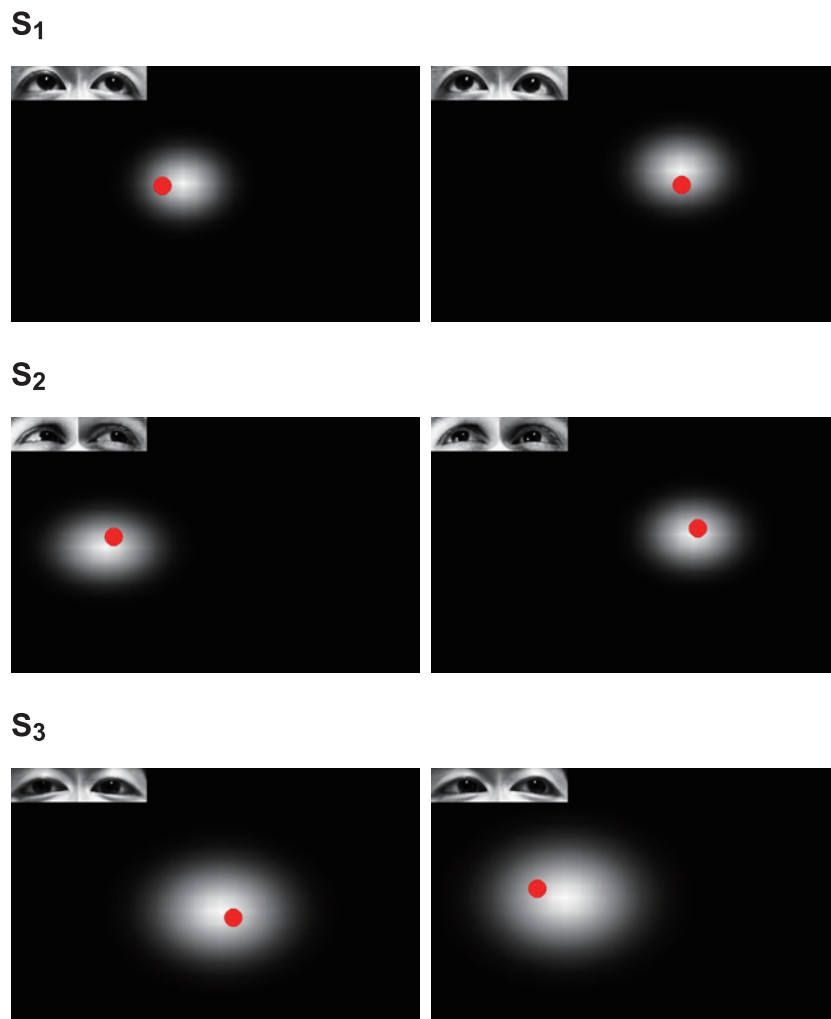


Figure 4.8: Estimation results of subjects $s_1 \dots s_3$. The estimation results are rendered as 2-D Gaussian circles. The corresponding input eye images are shown at the top-left corner. Overlaid circles are the ground truth gaze points obtained from a calibration-based gaze estimator.

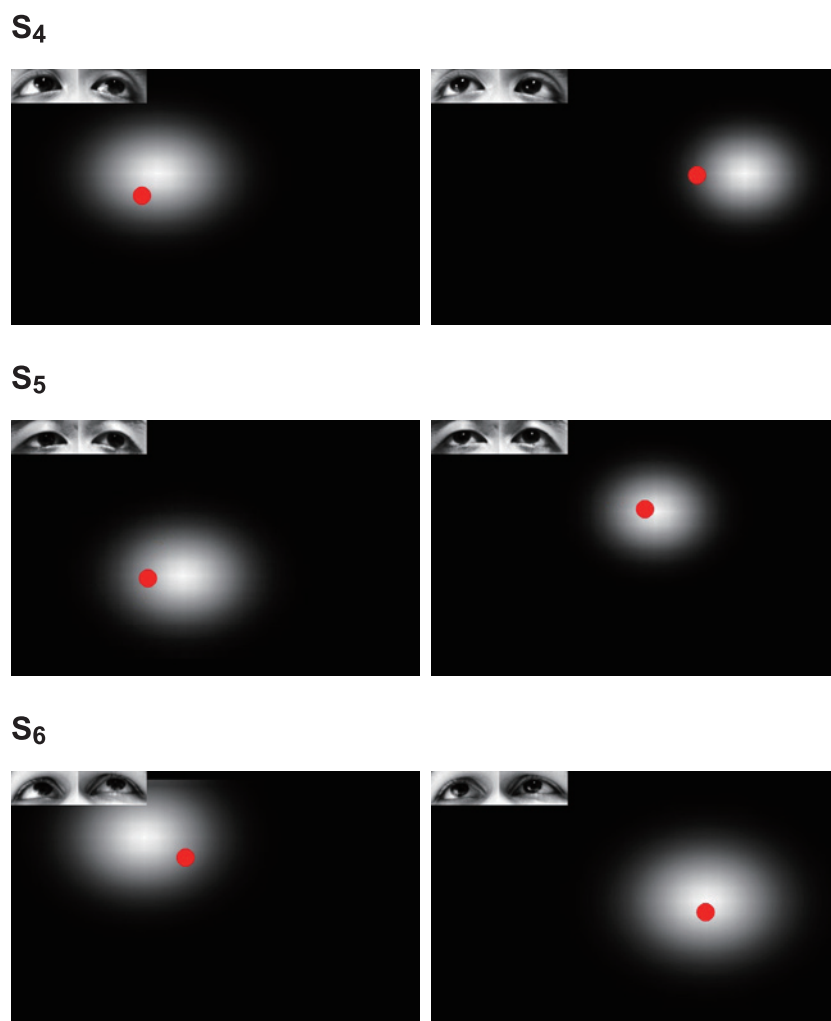


Figure 4.9: Estimation results of subjects $s_4 \dots s_6$. The estimation results are rendered as 2-D Gaussian circles as in Figure 4.8.

Table 4.2: Average error for each video clip. Two AUC columns indicates AUCs of the average ROC curves of raw saliency maps \mathbf{s} and gaze probability maps $\bar{\mathbf{p}}$. The rest of columns indicates distance and angular estimation errors (*average \pm standard deviation*) in two estimation scenarios.

Clip	\mathbf{s} AUC	$\bar{\mathbf{p}}$ AUC	Scenario 1		Scenario 2	
			error [mm]	error [deg.]	error [mm]	error [deg.]
A	0.75	0.90	38 ± 26	5.3 ± 3.6	45 ± 28	6.3 ± 3.8
B	0.71	0.87	60 ± 32	8.3 ± 4.4	56 ± 32	7.9 ± 4.3
C	0.74	0.92	31 ± 20	4.3 ± 2.7	36 ± 19	5.0 ± 2.5
D	0.70	0.89	36 ± 23	5.0 ± 3.1	42 ± 25	5.9 ± 3.4
Average	0.73	0.90	41 ± 25	5.7 ± 3.5	45 ± 26	6.3 ± 3.5

Table 4.3: Average error for each subject person. Columns indicate AUCs of the average ROC curves and estimation errors as in Table 4.2.

Subject	\mathbf{s} AUC	$\bar{\mathbf{p}}$ AUC	Scenario 1		Scenario 2	
			error [mm]	error [deg.]	error [mm]	error [deg.]
s_1	0.74	0.90	48 ± 35	6.8 ± 4.8	48 ± 36	6.7 ± 5.0
s_2	0.75	0.93	30 ± 20	4.1 ± 2.7	30 ± 19	4.2 ± 2.6
s_3	0.72	0.87	42 ± 27	5.9 ± 3.6	58 ± 27	8.1 ± 3.7
s_4	0.71	0.87	43 ± 26	6.0 ± 3.5	48 ± 27	6.7 ± 3.6
s_5	0.71	0.89	51 ± 26	7.1 ± 3.6	52 ± 28	7.2 ± 3.8
s_6	0.74	0.92	33 ± 18	4.6 ± 2.5	34 ± 18	4.8 ± 2.4

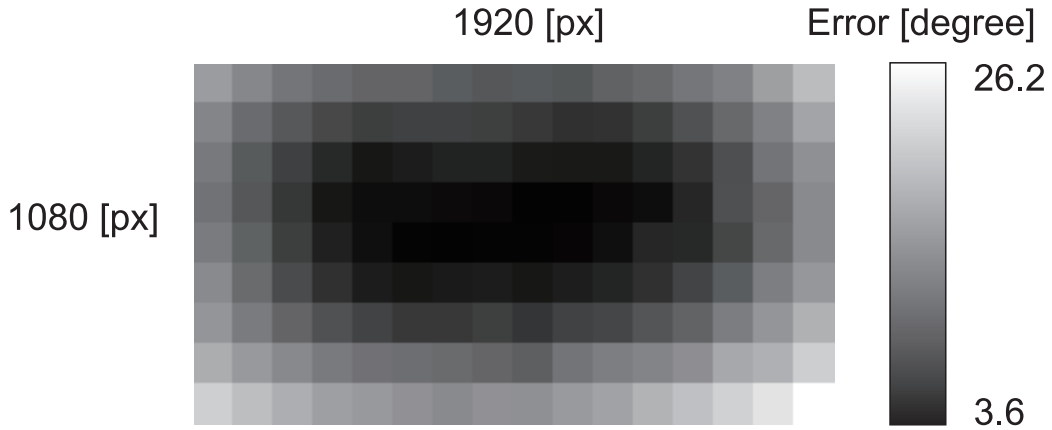


Figure 4.10: Spatial distribution of estimation errors in the display coordinate. Lower intensity corresponds to the lower estimation error as illustrated in the right bar.

corresponds to average of results of the corresponding test subject with two different training datasets. The columns show AUCs and estimation errors in the similar manner as in Table 4.2. In contrast to Table 4.2, subject dependency of our method is not clearly observed.

The accuracy of our method has dependency on the distribution of learning samples. Figure 4.10 shows the spatial distribution of average estimation errors. Each grid corresponds to a reference point that is used to capture the calibration data when producing the ground truth data. Using eye images obtained from the ground truth dataset as input to our method, we compute the errors of our method. Lower intensity corresponds to the lower estimation error. From this, the larger errors can be observed at edges of the display. Figure 4.11 shows the average saliency map and spatial histogram of gaze points. The left image shows the average of all raw saliency maps extracted from the four video clips used in our experiment. The right image shows the spatial histogram of ground-truth gaze points obtained from the experiment dataset. Higher intensity corresponds to larger amount of gaze points given at the grid. Usually salient objects are located at the center of

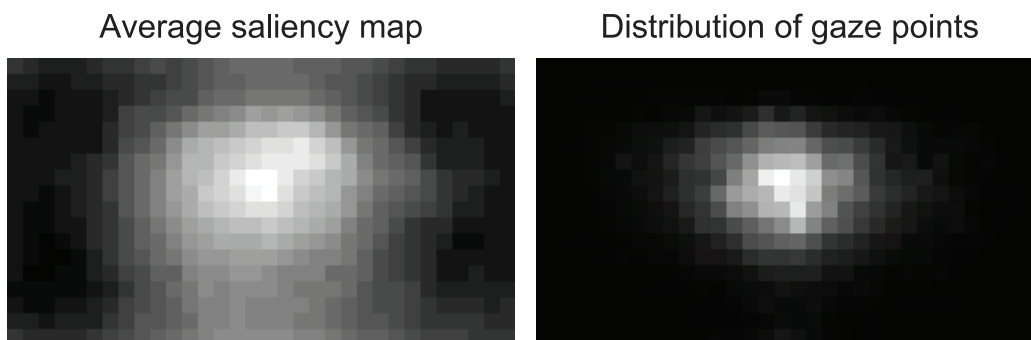


Figure 4.11: Average saliency map and spatial histogram of gaze points. Left image shows the average of all raw saliency maps extracted from four video clips used in the experiment. Right image shows the spatial histogram of the ground-truth gaze points of experimental dataset. Higher intensity corresponds to larger counts of gaze points.

video frames, and the gaze point also tends to concentrate at the center of the display. Because of these reasons, the number of learning samples at the display edges are limited, and these cause the bias of the estimation accuracy shown in Figure 4.10.

4.4 Conclusions

We proposed a novel calibration-free gaze estimation framework using saliency maps. By only using a synchronized set of eye images and video frames, a gaze estimator can be constructed by treating saliency maps as probabilistic distributions of gaze points. To the best of our knowledge, this is the first work to use saliency maps as the key for gaze estimation. Our method naturally avoids an explicit and noticeable gaze calibration step that is often demanding for users. In our experimental setting with fixed head positions, our method achieves the accuracy of about 6-degree error.

One the most important future tasks is dealing with changes in head

poses. Although this might be solved by an adaptive clustering like [SMSK08], it is still an open problem to find more sophisticated algorithms for handling head poses in appearance-based gaze estimation. Making the learning algorithm incremental and real-time is also an important task. Adoption of fast incremental regression algorithms like sparse on-line Gaussian processes [CO02] is thought to be a help to accomplish it.

The estimation accuracy of our method depends on the raw saliency maps extracted from input video clips. The mechanism of human gaze control has not been completely investigated, and there is a wide range of possibilities of more advanced saliency models for accurately predicting gaze. Our method can benefit from the further investigation of more accurate saliency models.

Chapter 5

Conclusions

5.1 Summary

Head pose and gaze estimation techniques play significant roles in inferring the focus of attention. Determining the object or area on which a user is focusing will enable the design of more efficient interactions and of computer systems that can adapt to the user's states. This thesis presented several methods that enable estimation of head pose and gaze with minimal requirements.

In Chapter 2, a monocular 3D head pose estimation method was presented. It uses a multilinear facial shape model with separate parameters for interpersonal and intrapersonal facial deformations and combines two estimation frameworks. Particle filter-based tracking is used to efficiently track head poses and facial actions, and model-based bundle adjustment is used to adjust the facial shape model to the target person. In this way, accurate estimations of head pose and facial action are achieved in real time without preliminary model customization for each person.

A real-time gaze estimation method based on the head tracker was presented in Chapter 3. The key is the assumption that the user gazes at the

cursor position whenever s/he clicks the mouse button. Calibration examples are automatically collected from the user's daily mouse operations, to learn mapping function between the eye images and gaze points. The examples are adaptively clustered in accordance with the estimated head poses, and local models of gaze estimation are incrementally built. This results in a monocular gaze estimator that requires no additional light sources and does not restrict the user's head movements.

In addition to the above method, a novel gaze estimation method was presented in Chapter 4. The inputs are video frames and associated eye images captured from a person watching the video frames. Bottom-up visual saliency maps are extracted from the video frames and used as probabilistic supervisors. Probability maps of the gaze points for the eye images are generated by aggregating the saliency maps and are used to construct a gaze estimator that uses a Gaussian process regression framework. Hence, completely calibration-free gaze estimation is enabled by using only data captured while the user watches video clips.

Using these methods will make head pose and gaze estimation noticeably more convenient by reducing installation and setup costs. This will enhance the potential for future investigation of techniques for sensing the focus of attention and for development of attention-based application systems.

5.2 Contributions

The main contributions of this thesis are summarized as follows.

Accurate monocular head pose estimation

The use of the unique combination of particle filter-based tracking and model-based bundle adjustment in conjunction with a multilinear facial shape model for monocular 3D head pose estimation is the first main contribution. The

generic multilinear face model is gradually adapted to the target person, while temporal actions are robustly tracked. This results in an accurate estimation of head pose and facial action without manual initialization of the shape model.

Automatic calibration of gaze estimators

The calibration of the gaze estimator on the basis of the user's natural behaviors is the second main contribution. Under the assumption that the user gazes at the cursor position whenever s/he presses the mouse button, we proposed a fully incremental learning algorithm for appearance-based gaze estimation. This enables background learning of a gaze estimator during daily PC usage. This is also the first work to extend an appearance-based algorithm to the case of various head poses. The estimation results are more accurate than those of previous monocular methods.

Saliency-driven learning of gaze estimators

The framework for gaze estimation using visual saliency maps is the third main contribution. Treating visual saliency maps as probability distributions of gaze points enables fully automatic learning of a gaze estimator without explicit calibration. To our knowledge, this is the first application of visual saliency as prior information to gaze estimation.

5.3 Future Directions

Practical head pose and gaze estimation methods

Monocular 3D head pose estimation techniques are becoming good enough for practical use. One of the remaining challenges is achieving more accurate facial action estimation for advanced tasks like facial expression recognition.

In contrast, monocular gaze estimation techniques still have problems to be solved, and constructing practical and accurate estimation systems is still a challenging task for researchers. For example, reducing the time needed for calibration is one of the keys to practical use. We assumed a scenario of personal customization in which a gaze estimator is learned during some period of observation, without forcing the user to engage in active calibration. Reducing the time needed for customization will enable the estimator to also be used quickly by various applications.

Another key is developing an advanced algorithm for compensating for the changes in head pose for appearance-based gaze estimation. The ability to use a calibration result for a certain head pose for gaze estimation under different head poses will drastically improve the capabilities of the estimation system.

Mechanisms of human attention

In Chapter 4, we described the use of a bottom-up computational model of visual attention. Despite the many studies in the field of cognitive science, the mechanism of human gaze control is still not completely understood. It is well-known that gaze control is affected by top-down knowledge [Hen03] as well as by low-level visual stimuli. Hence, if the viewing behavior is under the influence of a certain task, a bottom-up computational model cannot predict the gaze points particularly well [PI08]. Although some proposed computational models incorporate scene context [TOCH06], there are many open problems that computer systems cannot deal with.

Regarding the detection of the focus of attention, investigating the mechanisms of attention is also a significant research task. To better estimate the focus of attention, computer systems might have to consider the circumstances and environment in addition to the user's states.

Attention-based interactive systems

The availability of practical head pose and gaze estimation systems will enable the development of more attention-based application systems. It is also important for developers of sensing techniques to explore the possibility of head pose and gaze-based applications and future AUI directions.

In this thesis, we focused on the use of only computer-vision-based techniques to estimate the focus of the user's attention. However, the estimated focus can be used to analyze the viewed image. For instance, some proposed interactive algorithms use gaze estimation for image manipulation such as abstraction [DS02] and cropping [SAD⁺06].

We believe that more sophisticated methods, like the one described in Chapter 4, can be built by observing both viewers and images, instead of by using separate techniques for estimating the focus of attention and for analyzing images. That is, computer vision and related techniques will play a key role in designing rich interactions between humans and digital contents by analyzing both factors together.

Bibliography

- [AOS05] M. Ashdown, K. Oka, and Y. Sato. Combining head tracking and mouse input for a GUI on multiple monitors. In *CHI'05 extended abstracts on Human factors in computing systems*, page 1191, 2005.
- [BADM99] M. D. Byrne, J. R. Anderson, S. Douglass, and M. Matessa. Eye tracking the visual search of click-down menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 402–409, 1999.
- [BEP96] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head tracking. In *International Conference on Pattern Recognition*, volume 13, pages 611–616, 1996.
- [BF03] D. Beymer and M. Flickner. Eye gaze tracking using an active stereo head. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 2, pages 451–458, 2003.
- [Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

- [BP94] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 6:753–760, 1994.
- [BP04] J. S. Babcock and J. B. Pelz. Building a lightweight eyetracking headgear. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, pages 109–114, 2004.
- [BT09] N. D. B. Bruce and J. K. Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision*, 9(3):1–24, 2009.
- [CET01] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.
- [CHEK08] M. Cerf, J. Harel, W. Einhauser, and C. Koch. Predicting human gaze using low-level saliency combined with face detection. In *Proceedings of Advances in neural information processing systems (NIPS 2008)*, volume 20, pages 241–248, 2008.
- [CM06] F. L. Coutinho and C. H. Morimoto. Free head motion eye gaze tracking using a single camera and multiple light sources. In *Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing*, pages 171–178, 2006.
- [CO02] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.
- [DBSAM04] A. Del Bue, F. Smeraldi, L. Agapito, and Q. Mary. Non-rigid structure from motion using non-parametric tracking and non-linear optimization. In *Proc. IEEE Workshop on Articulated and Non-Rigid Motion*, volume 1, 2004.

- [DD06] F. Dornaika and F. Davoine. On appearance based face and facial action tracking. *IEEE transactions on circuits and systems for video technology*, 16(9):1107–1124, 2006.
- [DLDMV00] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [DM02] D. DeCarlo and D. Metaxas. Adjusting shape parameters using model-based optical flow residuals. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(6):814–823, 2002.
- [DS02] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 769–776, 2002.
- [Duc02] A. T. Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments and Computers*, 34(4):455–470, 2002.
- [Duc07] A. T. Duchowski. *Eye tracking methodology: Theory and practice*. Springer, 2007.
- [Fox96] E. Foxlin. Inertial head-tracker sensor fusion by a complimentary separate-bias kalman filter. *Virtual Reality Annual International Symposium*, 0:185, 1996.
- [GBG01] S. B. Gokturk, J. Y. Bouguet, and R. Grzeszczuk. A data-driven model for monocular face tracking. In *Proc. IEEE Int. Conf. Computer Vision*, volume 2, pages 701–708, 2001.
- [GC96] A. Gee and R. Cipolla. Fast visual tracking by temporal consensus. *Image and Vision Computing*, 14(2):105–114, 1996.

- [GJG04] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in WWW search. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 478–479, 2004.
- [GMB05] R. Gross, I. Matthews, and S. Baker. Generic vs. person specific active appearance models. *Image and Vision Computing*, 23(11):1080–1093, 2005.
- [Gri09] T. Gritti. Toward fully automated face pose estimation. In *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics (IMCE '09)*, pages 79–88, 2009.
- [GWB98] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: A gaussian process treatment. *Advances in neural information processing systems*, pages 493–499, 1998.
- [Hay08] B. Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.
- [HB98] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.
- [Hen03] J. M. Henderson. Human gaze control during real-world scene perception. *Trends in Cognitive Sciences*, 7(11):498–504, 2003.
- [HGME96] M. Humbert, N. Gey, J. Muller, and C. Esling. Determination of a mean orientation from a cloud of orientations. application to electron back-scattering pattern measurements. *Journal of Applied Crystallography*, 29:662–666, 1996.

- [HJ09] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on pattern analysis and machine intelligence*, 2009.
- [HKP07] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Proceedings of Advances in neural information processing systems (NIPS 2007)*, volume 19, pages 545–552, 2007.
- [HKPH03] E. Horvitz, C. Kadie, T. Paek, and D. Hovel. Models of attention in computing and communication: from principles to applications. *Commun. ACM*, 46(3):52–59, 2003.
- [HNL06] C. Hennessey, B. Nouredin, and P. Lawrence. A single camera eye-gaze tracking system with free head motion. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 87–94, 2006.
- [HWJM⁺89] T. E. Hutchinson, K. P. White Jr., W. N. Martin, K. C. Reichert, and L. A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1527–1534, 1989.
- [IB98] M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
- [IB06] L. Itti and P. F. Baldi. Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems, Vol. 19 (NIPS 2005)*, pages 547–554, 2006.
- [IBMK04] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade. Passive driver gaze tracking with active appearance models. In *Proceed-*

ings of the 11th World Congress on Intelligent Transportation Systems, 2004.

- [IDP03] L. Itti, N. Dhavale, and F. Pighin. Realistic avatar eye and head animation using a neurobiological model of visual attention. In *Proc. SPIE 48th Annual International Symposium on Optical Science and Technology*, volume 5200, pages 64–78, 2003.
- [IKN98] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.
- [Jac90] R. J. K. Jacob. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18, 1990.
- [JEDT09] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *Proceedings of the Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, 2009.
- [JH02] S. Josephson and M. E. Holmes. Visual attention to repeated internet images: testing the scanpath theory on the world wide web. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 43–49, 2002.
- [JY02] Q. Ji and X. Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

- [KPPB07] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *Proceedings of the 24th international conference on Machine learning (ICML 2007)*, pages 393–400, 2007.
- [KPW07] M. Kumar, A. Paepcke, and T. Winograd. EyePoint: practical pointing and selection using gaze and keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, 2007.
- [KSK01] K. Kitajima, Y. Sato, and H. Koike. Vision-based face tracking system for window interface: prototype application and empirical studies. In *CHI'01 extended abstracts on Human factors in computing systems*, page 360. ACM, 2001.
- [KSWF07] W. Kienzle, B. Scholkopf, F. A. Wichmann, and M. O. Franz. How to find interesting locations in video: a spatiotemporal interest point detector learned from human eye movements. In *Proceedings of the 29th Annual Symposium of the German Association for Pattern Recognition (DAGM 2007)*, pages 405–414, 2007.
- [KU85] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human neurobiology*, 4(4):219–227, 1985.
- [KWSF06] W. Kienzle, F. A. Wichmann, B. Scholkopf, and M. O. Franz. A nonparametric approach to bottom-up visual saliency. In *Proceedings of Advances in neural information processing systems (NIPS 2006)*, volume 19, pages 689–696, 2006.
- [LBP06] D. Li, J. S. Babcock, and D. J. Parkhurst. openEyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the*

2006 symposium on Eye tracking research & applications, pages 95–100, 2006.

- [LCS99] M. La Cascia and S. Sclaroff. Fast, reliable head tracking under varying illumination. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 604–610, 1999.
- [LHB⁺08] L. Lorigo, M. Haridasan, H. Brynjarsdottir, L. Xia, T. Joachims, G. Gay, L. Granka, F. Pellacini, and B. Pan. Eye tracking and online search: Lessons learned and challenges ahead. *Journal of the American Society for Information Science and Technology*, 59(7):1041–1052, 2008.
- [Loh97] G. L. Lohse. Consumer eye movement patterns on yellow pages advertising. *Journal of Advertising*, 26(1):61–73, 1997.
- [Lou] M. I. A. Lourakis. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. <http://www.ics.forth.gr/~lourakis/levmar/>.
- [MAF02] C. Morimoto, A. Amir, and M. Flickner. Detecting eye position and gaze from a single camera and 2 light sources. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)*, pages 314–317, 2002.
- [MB04] I. Matthews and S. Baker. Active appearance models revisited. *Int. J. Computer Vision*, 60(2):135–164, 2004.
- [MBB05] E. Munoz, J. M. Buenaposada, and L. Baumela. Efficient model-based 3d tracking of deformable objects. In *Proc. IEEE Int. Conf. Computer Vision*, pages 877–882, 2005.

- [MCMH04] S. Martinez-Conde, S.L. Macknik, and D.H. Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5:229–240, 2004.
- [MCT08a] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626, 2008.
- [MCT08b] E. Murphy-Chutorian and M. M. Trivedi. HyHOPE: Hybrid head orientation and position estimation for vision-based driver head tracking. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 512–517, 2008.
- [MOZ00] Y. Matsumoto, T. Ogasawara, and A. Zelinsky. Behavior recognition based on head pose and gaze direction measurement. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 3, pages 2127–2132, 2000.
- [MR02] P. Majaranta and K.-J. R  ih  . Twenty years of eye typing: systems and design issues. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 15–22, 2002.
- [MRCD02] L.-P. Morency, A. Rahimi, N. Checka, and T. Darrell. Fast stereo-based head tracking for interactive environments. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 2002.*, pages 390–395, 2002.
- [MZ00] Y. Matsumoto and A. Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction

- measurement. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 499, 2000.
- [NKIT08] T. Nagamatsu, J. Kamahara, T. Iko, and N. Tanaka. One-point calibration gaze tracking based on eyeball kinematics using stereo cameras. In *Proceedings of the 2008 symposium on Eye tracking research & applications (ETRA '08)*, pages 95–98, 2008.
- [OS05] K. Oka and Y. Sato. Real-time modeling of face deformation for 3-d head pose estimation. In *Proc. IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG 2005)*, pages 308–320, 2005.
- [PI08] R. J. Peters and L. Itti. Applying computational tools to predict gaze direction in interactive visual environments. *ACM Trans. Appl. Percept.*, 5(2):1–19, 2008.
- [PLN02] D. Parkhurst, K. Law, and E. Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision research*, 42(1):107–123, 2002.
- [PPNH06] M. Pantic, A. Pentland, A. Nijholt, and T. Huang. Human computing and machine understanding of human behavior: A survey. In *ACM SIGCHI Proceedings Eighth International Conference on Multimodal Interfaces*, pages 239–248, 2006.
- [PS00] C. M. Privitera and L. W. Stark. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Transactions on pattern analysis and machine intelligence*, 22(9):970–982, 2000.

- [RBSJ79] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. Magnetic position and orientation tracking system. *IEEE Transactions on Aerospace and Electronic Systems*, AES-15(5):709–718, 1979.
- [RRS⁺01] K. Rayner, C. M. Rotello, A. J. Stewart, J. Keir, and S. A. Duffy. Integrating text and pictorial information: Eye movements when looking at print advertisements. *Journal of Experimental Psychology Applied*, 7(3):219–226, 2001.
- [RS00] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.
- [RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.
- [SAD⁺06] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 771–780, 2006.
- [SB90] I. Starker and R. A. Bolt. A gaze-responsive self-disclosing display. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, page 10, 1990.
- [SL03] D. Skocaj and A. Leonardis. Weighted and robust incremental method for subspace learning. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, pages 1494–1501, 2003.

- [SL04] S. W. Shih and J. Liu. A novel approach to 3-d gaze tracking using stereo cameras. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(1):234–245, 2004.
- [SMSK08] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike. An incremental learning method for unconstrained gaze estimation. In *Proceedings of the 10th European Conference on Computer Vision (ECCV 2008)*, pages 656–667, 2008.
- [SS07] Y. Sugano and Y. Sato. Person-independent monocular tracking of face and facial actions with multilinear models. In *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG2007)*, pages 58–70, 2007.
- [SSdVL03] P. Smith, M. Shah, and N. da Vitoria Lobo. Determining driver visual attention with one camera. *IEEE transactions on intelligent transportation systems*, 4(4):205–218, 2003.
- [THT06] J. Tu, T. Huang, and H. Tao. Accurate head pose tracking in low resolution video. In *7th International Conference on Automatic Face and Gesture Recognition (FGR 2006)*, pages 573–578, 2006.
- [TKA02] K. H. Tan, D. J. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV 2002)*, pages 191–195, 2002.
- [TMHF99] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. *Lecture Notes in Computer Science*, pages 298–372, 1999.

- [Tob] Tobii Technologies. <http://www.tobii.com/>.
- [TOCH06] A. Torralba, A. Oliva, M. S. Castelhana, and J. M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychological Review*, 113(4):766–786, 2006.
- [VBPP05] D. Vlastic, M. Brand, H. Pfister, and J. Popovic. Face transfer with multilinear models. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH 2005)*, 24(3):426–433, 2005.
- [VC08] A. Villanueva and R. Cabeza. A novel gaze estimation system with one calibration point. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 38(4):1123–1138, 2008.
- [VDS05] S. Vijayakumar, A. D’Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634, 2005.
- [VLF04] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(10):1380–1384, 2004.
- [VSCM06] R. Vertegaal, J.S. Shell, D. Chen, and A. Mamuji. Designing for augmented attention: Towards a framework for attentive user interfaces. *Computers in Human Behavior*, 22(4):771–789, 2006.
- [WBC06] O. Williams, A. Blake, and R. Cipolla. Sparse and semi-supervised visual mapping with the S³GP. *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, pages 230–237, 2006.

- [WL08] S. F. Wang and S. H. Lai. Estimating 3d face model and facial deformation from a single image based on expression manifold optimization. In *Proc. 10th European Conf. Computer Vision*, pages 589–602, 2008.
- [WM02] D. J. Ward and D. J. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838, 2002.
- [WS02] J. G. Wang and E. Sung. Study on eye gaze estimation. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 32(3):332–350, 2002.
- [XBMK04] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2d+3d active appearance models. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 535–542, 2004.
- [XMS98] L. Q. Xu, D. Machin, and P. Sheppard. A novel approach to real-time non-intrusive gaze finding. In *Proceedings of the British Machine Vision Conference*, pages 428–437, 1998.
- [XWT⁺05] L. Xin, Q. Wang, J. Tao, X. Tang, T. Tan, and H. Shum. Automatic 3d face modeling from video. In *Proc. IEEE Int. Conf. Computer Vision*, volume 2, pages 1193–1199, 2005.
- [YC05] D. H. Yoo and M. J. Chung. A novel non-intrusive eye gaze estimation using cross-ratio under large head motion. *Computer Vision and Image Understanding*, 98(1):25–51, 2005.
- [YUYA08] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe. Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions. In *Proceedings of*

- the 2008 symposium on Eye tracking research & applications (ETRA '08)*, pages 245–250, 2008.
- [YZ02] R. Yang and Z. Zhang. Model-based head pose tracking with stereovision. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 0255, 2002.
- [ZCSC07] G. Zhao, L. Chen, J. Song, and G. Chen. Large head movement tracking using SIFT-based registration. In *Proceedings of the 15th international conference on Multimedia*, pages 807–810, 2007.
- [ZHL06] J. Zhu, S. C. H. Hoi, and M. R. Lyu. Real-time non-rigid shape recovery via active appearance models for augmented reality. In *Proc. 9th European Conf. Computer Vision*, pages 186–197, 2006.
- [ZJ05] Z. Zhu and Q. Ji. Eye gaze tracking under natural head movements. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 918–923, 2005.
- [ZJ06] Z. Zhu and Q. Ji. Robust real-time face pose and facial expression recovery. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pages 681–688, 2006.
- [ZJB06] Z. Zhu, Q. Ji, and K. P. Bennett. Nonlinear eye gaze mapping function estimation via support vector regression. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006)*, volume 1, pages 1132–1135, 2006.

- [ZS03] Z. Zhang and Y. Shan. Incremental motion estimation through modified bundle adjustment. In *Proc. IEEE Int. Conf. Image Processing*, volume 2, pages 343–346, 2003.
- [ZWT08] W. Zhang, Q. Wang, and X. Tang. Real time feature based 3-d deformable face tracking. In *Proc. 10th European Conf. Computer Vision*, pages 720–732, 2008.

Publications

Journals

1. 菅野裕介, 松下康之, 佐藤洋一, 小池英樹, “ マウス操作を利用した逐次学習による自由頭部姿勢下での視線推定 ”, 電子情報通信学会論文誌 (D) (条件付き採録).
2. 菅野裕介, 佐藤洋一, “ 顔変形をともなう 3次元頭部姿勢の単眼推定 ”, 情報処理学会論文誌 コンピュータビジョンとイメージメディア, Vol. 1, No. 2 (CVIM 22), pp. 41-49, July 2008.
3. 岡兼司, 菅野裕介, 佐藤洋一, “ 頭部変形モデルの自動構築をともなう実時間頭部姿勢推定 ”, 情報処理学会論文誌 コンピュータビジョンとイメージメディア, Vol. 47, No. SIG10 (CVIM 15), pp. 185-194, July 2006.

Refereed conferences

1. Yusuke Sugano, Yasuyuki Matsushita, Yoichi Sato and Hideki Koike, “ An Incremental Learning Method for Unconstrained Gaze Estimation ”, in Proc. European Conference on Computer Vision (ECCV2008), October 2008.
2. Yusuke Sugano and Yoichi Sato, “ Person-Independent Monocular Track-

- ing of Face and Facial Actions with Multilinear Models ”, in Proc. IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG2007), October 2007.
3. 菅野裕介, 松下康之, 佐藤洋一, 小池英樹, “ マウス操作情報を用いた逐次学習による視線推定 ”, 画像の認識・理解シンポジウム (MIRU 2009), July 2009.
 4. 菅野裕介, 佐藤洋一, “ 顔変形を伴う3次元頭部姿勢の単眼推定 ”, 画像の認識・理解シンポジウム (MIRU 2007), August 2007.

Others

1. 菅野裕介, 佐藤洋一, “ 表情変動を許容した実時間頭部姿勢推定のための個人間および個人内変動に対する顔形状推定 ”, 情報処理学会コンピュータビジョンとイメージメディア研究会, 2006-CVIM-156-21, pp. 179-186, November 2006.