Learning Periorbital Soft Tissue Motion

by

Anurag Ranjan

B. Tech., National Institute of Technology Karnataka, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

September 2015

Abstract

Human observers tend to pay a lot of attention to the eyes and the surrounding soft tissues. These periorbital soft tissues are associated with subtle and fast motions that convey emotions during facial expressions. Modeling the complex movements of these soft tissues is essential for capturing and reproducing realism in facial animations.

In this work, we present a data driven model that can efficiently learn and reproduce the complex motion of the periorbital soft tissues. We develop a system to capture the motion of the eye region using a high frame rate monocular camera. We estimate the high resolution texture of the surrounding eye regions using a Bayesian framework. Our learned model performs well in reproducing various animations of the eyes. We further improve realism by introducing methods to model facial wrinkles.

Preface

The methods described in Chapter 3, Chapter 5 and Chapter 6 formed part of the following poster, and will be part of a following full paper:

> Debanga Raj Neog, Anurag Ranjan, João L. Cardoso, and Dinesh K. Pai. "Gaze driven animation of eyes." In Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 198-198. ACM, 2015.

The work in Chapter 3 and Chapter 6 was done jointly with Debanga Raj Neog and will eventually form a part of his doctoral thesis. The rendering framework was developed by João L. Cardoso.

Except as noted above, all algorithms, experiments and analysis in this thesis were developed by me, with guidance from my supervisor, Dinesh Pai.

Table of Contents

List of Tables

List of Figures

Glossary

AAM	Active Appearance Models
ANN	Artificial Neural Network
AWGN	Additive White Gaussian Noise
CG	Conjugate Gradient
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FEM	Finite Element Methods
LDOF	Large Displacement Optical Flow
MAP	Maximum a Posterior
MCML	Maximally Collapsing Metric Learning
ML	Maximum Likelihood
MLR	Multivariate Linear Regression
MLS	Moving Least Squares
NCA	Neighborhood Component Analysis
NN	Neural Network
OFC	Optical Flow Constraint

PCA Principal Component Analysis

PPCA Probabilistic PCA

RBF Radial Basis Functions

SAG Stochastic Average Gradient

Acknowledgments

I am grateful to my research supervisor Prof. Dinesh K. Pai for supporting me throughout my graduate program. I am thankful to Debanga Raj Neog for his useful collaboration during the projects that became a part of this thesis. I extend my thanks to João L. Cardoso for helping with the rendering. I acknowledge MITACS and NSERC for funding my graduate program.

I will miss the espressos at Bean which always kept me awake, and their wonderful, welcoming people.

Finally, I express my deepest gratitude to Radhe Uncle for his love and guidance throughout my life. I am very thankful to my parents and family for their kindest support.

Dedication

to Mom

Chapter 1

Introduction

Eyes have always been the important subjects of human perception and facial expressions. In 1960, Yarbus showed that human observers spend a surprisingly large fraction of time fixated on the eyes in a picture [82]. The region around the eye forms the most important feature of Viola and Jones's face detector [74]. In Google's Inceptionism [1], we see that the neural network for classification of humans or animals generally looks for the eye region. This results in *dreamy* images corresponding to features learned by the classifiers to have a lot of eyes.

But what is it about eyes that conveys this important information to observers? To discuss this, we need to introduce some terms. The colloquial term "eye" is not sufficiently precise. It includes the *globe*, the approximately spherical optical apparatus of the eye that includes the colorful *iris*. The globe sits in a bony socket in the skull called the *orbit*. The term "eye" also usually includes the upper and lower *eyelids*, *eyelashes* and *periorbital soft tissues* that surround the orbit, including the margins of the *eyebrows*. When we refer to the eye we mean all of these *tissues*, and we will use the more specific term where appropriate.

The eyes convey subtle information about a person's mental state (e.g., attention, intention, emotion) and physical state (e.g., age, health, fatigue). Thus, creating realistic computer generated animations of eyes is a very important problem in computer graphics. Realistic facial performance capture has been widely researched [10, 29] but attention to details around the eye region has been paid

[1] http://googleresearch.blogspot.ca/2015/06/inceptionism-going-deeper-into-neural.html

only in handful of methods. Recently, there has been an emergence of methods [4, 5, 14, 25, 46] that focus on reconstruction of details of the eyes in facial animations. However, these methods require a complex setup for detailed capture and reconstruction of eye region. Our work is motivated by learning important aspects of the motion around the eyes using a simple video capture setup.

In order to learn about the motion of the eyes, we need a robust framework to track the detailed movements of the soft tissues around the eyes. The motion of the eye region is very subtle and fast. The blink of an eye takes around 100 to 400 milliseconds [56]. The eye saccades in humans can reach up to 900° per second [21]. This results in subtle motion of soft tissues surrounding the eyes, which generally goes unnoticed. Tracking of such detailed and fast motion of the eye thus requires a robust tracking method.

In addition, the motion can also help in estimating the detailed texture of the eye regions using superresolution methods. The various regions of the eye such as *the globe*, *eyelashes* and *skin texture* may occlude each other as a result of the motion of the eye region. Therefore, a framework is required to deal with these regions of occlusion to robustly recover textures of the eye region. A large number of tracking methods use the *Optical Flow* framework [32] for computing the dense motion field across the images and recent developments have made it more robust to occlusions and discontinuities.

The subtle motions of the eye are very complex, and it would be very useful to simplify them for understanding. A simplified model is essential for generating these complex motions using a small number of parameters. Most of the facial performance capture methods use *Blend Shapes* which can not capture all the detailed and non-linear motion that is usually produced around eye region. Hence, a non-linear model is required to capture the detailed motion of the eye region which can be controlled using small number of parameters such as *Gaze* and other affect parameters.

The realism in facial animation is highly influenced by the quality of different wrinkles that can be produced. The simple wrinkles on the forehead as a result of expressions like eyebrow raising are easier to capture. However, the wrinkles around the eye region are not very sharp and distinguishable. There have been a number of methods for modeling the wrinkles using *shape from shading* approach.

In this thesis, we design a robust method for dense tracking of the region around the eyes. We recover the texture of the eye region using an optical flow based superresolution method. We design a neural network framework to learn and control the motion of the eyes using a small number of parameters. Finally, we use a *shape from shading* approach to simulate wrinkles and add realism.

Our main **contributions** are as follows:

1. We implement a robust scene flow method [70] and improve its baseline performance using a stochastic optimization technique [57]. We develop *skin flow* method in a monocular setup to track the movements around the eye using "reduced coordinates" and estimate the dense 3D motion of the eye region.

2. We develop a robust method to estimate the texture of skin around the eye region and eyelashes in the presence of occlusion. Our framework relies on optical flow based superresolution methods and deals with occlusions by introducing material mask functions. We recover high resolution textures using Bayesian estimation.

3. We parametrize the complex motion of the eye in terms of the *Gaze* of the eye. We design two neural network models based on Radial Basis Functions (RBF) to learn the motion of the eye region. The first network learns the shape of the eyelid from the Gaze parameters. The second network parametrizes the motion of the eye in terms of eyelid shape.

4. We develop two approaches to deal with wrinkles that are produced as a result facial expressions. The first model generates *Appearance Textures* based on the strain of the skin that can simulate wrinkles. The second method, based on a *shape from shading* approach, parametrizes wrinkles in terms of geometric shape based on the observed intensities of their pixels.

Chapter 2

Background and Related Work

The goal of this thesis is to design a framework for learning the motion of the eye region. This objective is addressed by developing methods for motion estimation and texture recovery. Using the data from motion estimation, we train neural network models to produce realistic motion around the eye.

2.1 Motion Estimation

The work on motion estimation has followed from methods for estimation of *optical flow* [8, 11, 12, 32, 41]. The optical flow methods are used for estimating dense 2D motion between frames of an image sequence. Subsequently, the methods for computing 3D motion using multiple cameras emerged [34, 70, 73, 77]. The estimation of dense 3D motion in a scene using multiple cameras is referred to as *Scene Flow*.

2.1.1 Optical Flow

Tracking motion in video sequences, known as *Optical Flow* is a well studied problem in computer vision. It is defined as the estimation of motion, given by $u(x,y)$ and $v(x,y)$ in x and y directions respectively, of all the pixels between two image frames given by the sequence $I(x,y,t) \in \mathbb{R}$. There have been various developments to obtain state of the art performance in determining optical flow. Horn and Schunck's seminal work [32] on optical flow described it as a constrained opti-

mization problem introducing Optical Flow Constraint (OFC)

$$I_x u + I_y v + I_t = 0, \tag{2.1}$$

where, I_x, I_y and I_t are the partial derivatives of image. u, v are the flow fields between images $I(x, y, t_1)$ and $I(x, y, t_2)$. However a single OFC is insufficient to estimate u, v, hence Horn and Schunck introduced spatial regularization constraints that enforced flow smoothness across pixels. The spatial regularization constraints are given by

$$\nabla^2 u = 0 \text{ and } \nabla^2 v = 0, \tag{2.2}$$

where, ∇ is the gradient operator. This was followed by several approaches based on variational methods [60, 84, 85] including Brox et al.'s work [12] that incorporates the theory of warping. The variational methods provide a way of tracking precise displacement of scene objects to a sub-pixel level. However, these methods use spatial regularization over image pixels that leads to errors over edge discontinuities. Although the regularizer is very effective in providing a unique solution to the optical flow problem, it causes over-smoothness of resulting flow fields across edge discontinuities. A framework to deal with the discontinuities was developed by Black and Anandan [8] using graduated non-convexity in a variational approach. The emergence of new variational methods [60, 85] have provided better ways of dealing with the regularization, improving the estimation of optical flow at edge discontinuities.

We have also seen the emergence of fast methods which are based on tracking sparse points as opposed to dense optical flow methods. Such methods [37, 49, 76] track features across multiple image frames that follow the approach of the classic KLT tracker [41, 58, 66]. The KLT tracker uses a coarse-to-fine approach for estimation of optical flow using image pyramids, making it more robust. Several optical flow methods have since followed a coarse-to-fine approach to get more robust estimation.

Deformations in the upper head region surrounding the eye as a result of skin movements are very complex. The tracking of these deformations is affected by

occlusion due to buckling and folds. Some methods were specifically targeted towards robust tracking of facial movements. The model based facial tracking introduced by Pighin et al. [48] was successful only in tracking coarse points on face. This followed other approaches in active-mesh based methods [18, 55, 63, 72, 76] modeling the optical flow problem as a physical mass-spring system. Some work on estimation of deformation in various surfaces has been done by [55] and [49]. These works perform well only on linear deformation models and are based on sparse optical flow tracking.

The Active Appearance Models (AAM) introduced by Xiao et al. [80] were a more robust version of face tracking that learned a model by manually annotating positions of facial features in the images. Other AAM based methods [38] also became popular for facial tracking. As manually annotating all the pixels of the face for generating training data becomes extremely tedious, they could never be used for computing dense optical flow in facial deformations. Most of the work in robust facial tracking is based on sparse trackers like KLT or use learning approaches like AAM. As such, they do not perform very well in capturing the detailed and complex motion of skin especially around the eyes.

A major baseline improvement resulted in combining feature based tracking with a dense variation framework to compute large and fast motions. Brox and Malik's Large Displacement Optical Flow (LDOF) [11] performed significantly better than previous methods, especially on large and fast motions. A fast version of this method has been implemented by [62] which uses parallelizations on GPU. Improvements to the KLT-tracker by Kalal et al. [37] also performed better by detecting tracking failures using a *Median Flow* tracker. Another baseline improvement was made by Sun et al. [60] which provided various implementation strategies including the use of a *weighted median filter*. The work by Zimmer et al. [84, 85] provided a theoretical interpretation of improvements in variational methods in terms of diffusion images. This provided a much needed understanding of optical flow problem and the approaches that guarantee a better solution. A detailed survey of current practices in estimation of optical flow is provided by Sun et al. [61] and Baker et al. [1].

2.1.2 Scene Flow

Scene flow is defined as the motion of 3D points in a scene. As such, it requires depth information and can not be computed by monocular set ups. Thus, scene flow setups include binocular or multi-view stereo capture systems. Unlike *structure from motion* methods that estimate the 3D representation of static scenes, scene flow estimates the 3D motion of moving scenes. Scene flow estimation was first introduced by Vedula et al. in [73]. The proposed solution was based on solving an overdetermined linear system of equations.

Most of the algorithms for computing scene flow treat structure and motion of the scene separately. The approach often follows sequential computation of shape and motion parameters. This can be seen in Vedula et al.'s seminal work followed by Wedel et al.'s [77]. However, the joint estimation of shape and motion parameters in a scene flow framework has established much better improvements [34, 70]. These methods follow the variational optical flow setup with additional epipolar constraint to estimate the 3D scene flow.

Multi-camera scene flow has also seen improvements from the use of temporal information [35]. Some approaches have also used a second order variational framework in a joint estimation of scene flow to compute Lagrangian stress tensor of a deformable body [30]. The new methods have shown that joint estimation of shape and motion components results in significant increase of performance in computing scene flow.

Our approach follows from Valgaerts et al.'s work [70] that describes the joint estimation of scene flow along with camera intrinsics. The work shows a good improvement over previous methods without using any external measurements for camera calibration. We also look into various optimization methods for solving the variational setup. Our method uses Schmidt et al.'s Stochastic Average Gradient (SAG) [57] for improving the numerical solution obtained from the variational methods.

2.2 Realistic Performance Capture

In the direction of achieving realism in animations, most of the research is based on realistic facial animation [10, 26, 29]. We have seen a handful of methods which

focus on the details of the eye [4, 5, 46].

2.2.1 Facial Performance Capture

An estimation of detailed facial textures has been done using the light stage [47], a multiple array of polarized lights and cameras. Some systems also include an array of multiple cameras in controlled lighting [10] for a high resolution facial scan.

There has also been significant progress in achieving a high quality performance capture of human faces using cheap measurement setups. The work in [36] uses a hand-held monocular video input to create dynamic 3D avatars and their textures. Using structure from motion techniques, a texture is estimated by taking multiple images of the face from different viewpoints. These images are then stitched together using Poisson's blending to get the final texture. While the resolution of these textures is quite low, they can be used in the animation pipeline for creating dynamic avatars.

The use of a monocular camera in reconstruction of dynamic facial geometry has been done in [26] without paying any attention to textures. The work of [71] uses a binocular facial performance capture system to model the geometry while the textures are captured using projective texturing. While both of above methods have been impressive in capturing facial geometry, they usually reconstruct low resolution textures using primitive algorithms. There has not been enough contribution in reconstructing high resolution textures from a simple and inexpensive set up.

2.2.2 Dynamic Capture of the Eyes

Recently, the computer graphics community has started looking at detailed capture and reconstruction of subtle details of the eye [4, 5, 46]. The work on capturing the geometric structure of the globe of the eye and its texture has been discussed in [4]. While the authors have a detailed reconstruction system for the shape of the eyelids using multiple cameras, their texture estimation algorithm uses a simple Poisson's blending. The detailed spatio-temporal reconstruction of eyelids has been done by Bermano et al. [5]. However, their method is limited to reconstructing the blink sequences. A similar Poisson's blending approach has been used in our work [46]

where the authors develop a model of periorbital soft tissue deformation based on the gaze of the eyes. These works pay detailed attention to the eye and the region of skin around it which is important in capturing various subtle movements. A lot of these movements are a result of simple expressions like 'eye blink', 'look around' and 'Crow's Feet' sequences.

A detailed survey of eye and gaze animation has been discussed in [54]. It discusses various biologically motivated models to simulate eyeball movements such as saccades, smooth pursuits and vestibulo-ocular reflex. The animation pipelines are mainly concerned with low level characters. There has not been any contribution to capture highly detailed texture and motion of the eyes.

2.3 Texture Estimation

In computer graphics, textures are an important aspect of rendering appealing animations in movies and games. In the recent years, computer graphics research has made significant contributions in introducing better realism in computer generated imagery [10, 25]. The developments in generating realistic performance in character animation have also led to improved capture methods.

2.3.1 Multi-view Stereo Methods

Most of the works in realistic performance capture [4, 10, 25, 46] follow multi-view stereo techniques which capture the image of the object using multiple cameras and stitch them together to form the complete texture. These methods are very sophisticated and require a lot of precision setup, including setting up a light stage [47], camera arrays and measuring the physics of skin [29]. Although they are effective in capturing detailed texture, these set ups are very expensive and require a lot of time to construct.

2.3.2 Bayesian Methods

In recent years, people have started looking into reconstructing textures using superresolution methods [28, 68]. The work of [28] uses the approach of superresolution methods in a variational framework. Their work was mainly focused on capturing high quality textures of static 3D objects using multi-view stereo, and

did not account for dynamic elastic objects such as real human faces.

A Bayesian approach using superresolution to recover textures is discussed in [68] using multiple videos. The method combines the best ideas of superresolution and multi-view stereo in a unified framework. The approach is generalized for reconstructing 3D texture of static objects under Lambertian conditions. The set up requires calibration of multiple cameras to achieve good performance.

Superresolution

Various methods of estimating a superresolution image from multiple low resolution images have existed in computer vision literature. A detailed survey of superresolution methods is presented in [65]. The early superresolution methods were based on combining frequency domain coefficients [51, 67]. The work of [67] uses a Discrete Fourier Transform (DFT) whereas [51] use a Discrete Cosine Transform (DCT) to approach superresolution problem. The superresolution problem has also been approached using interpolation methods [9, 69]. A Moving Least Squares (MLS) interpolation has been used in [9].

In recent years, we have seen more use of Bayesian methods in estimation of superresolution [24, 53]. They are mostly based on Maximum Likelihood (ML) estimates [24] or Maximum a Posterior (MAP) estimates [64]. There has also been some research in exploring the models of image priors that can be used in Bayesian approach. The work of [53] uses a Gaussian Markov Random Field as the image prior. The use of optical flow from several video frames to reconstruct superresolution image using ML estimation has been discussed in [24]. There have also been developments in machine learning methods [75, 81] that learn to predict high resolution patches based on several low resolution patches. The work of [33] uses high resolution features from a single image to reconstruct low resolution parts of the same image.

2.4 Learning Skin Motion

The animation of character meshes is a well studied problem in computer graphics. In the context of realistic facial performances, the two most widely used approaches are *physical models* and *blend shapes*. The physical models [39, 59] rely

on accurately modeling the physical characteristics such as elasticity, strains etc. in a deformable body. In contrast, blend shapes [10, 25] rely on data driven methods. They use a set of pre-processed meshes and obtain animations by blending them linearly or non-linearly.

The physical models can reproduce all possible set of poses in an animation. However, constructing a physical model requires a lot of detail in estimating the physical properties of the system. Blend shapes are however easier to construct from data driven methods. The performance of a blend shape model depends on the number of blend shapes that are used for animation. As such, it leads to storage constraints on large number of blend shapes to cover all possible expressions.

In order to minimize the storage restrictions of blend shapes, and recover all the expressions as in physical models, we use machine learning approaches to model our data driven approach. We use Artificial Neural Network (ANN) models to learn the parameters from our dataset. As a result, we learn a non-linear model to produce large number of expressions without any storage restrictions.

2.5 Wrinkles

Detecting and modeling wrinkles is a widely researched topic in facial rejuvenation, computer vision, and computer graphics. The characteristics of wrinkles and buckles formed in various materials depend on their material properties and on the underlying substrate. Physical models of wrinkles have been described in [15, 27]. This work characterizes amplitude and frequency of wrinkles by the elastic properties of the material using an experimentally motivated mathematical model. There has been significant research in wrinkle modeling of clothes [50, 52], including the use of compressive strain model [16].

Tracking facial wrinkles in videos to generate useful mathematical representation has been a focus of research in graphics and vision. Automatic detection and quantification of facial wrinkles has been discussed in [17] and uses Hong et al.'s algorithm [31] for wrinkle extraction and enhancement. This work however does not assess the complex wrinkles formed around the eye. Bando et al. uses Bezier curves [2] for modeling wrinkles on human skin. This method employs dynamic modulation of fine-scale and large-scale wrinkles according to deformation

of skin for dynamic animation. Bickel et al. proposed a pose-space deformation technique [7] that learns the dependence of wrinkle formation on skin strain based on a small set of prior poses. The work uses scattered data interpolation of learned RBF [40, 79] for the transfer of facial details such as wrinkles.

Various Finite Element Methods (FEM) [22, 44] attempt to model wrinkles as volumetric layers, where the interaction among facial tissues determine the shape and frequency of the wrinkles. A Markov Point Process has been used to model wrinkles in [3]. The wrinkles are modeled as stochastic arrangements of line segments which localize the wrinkles in a Bayesian framework.

Medium-scale facial wrinkles and buckles are modeled using Bezier curves in [6], which improves the appearance of forehead wrinkles. However, the method requires painting the forehead to make it *Lambertian* for wrinkle modeling. Some recent research also includes blending normal maps to produce wrinkles during facial animation [19]. This technique requires keeping only a few reference poses and corresponding normal maps, making it viable for real-time rendering.

Chapter 3

Motion Estimation

In this chapter, we discuss the methods to estimate the complex motion of the skin in the upper head region. In Section 3.1, we describe the scene flow algorithm to compute the dense 3D motion in images obtained from a binocular camera set up. We show improvements to baseline methods using Stochastic Optimization techniques. In Section 3.2, we briefly describe the *skin flow* technique in a monocular set up to track the skin of the upper head region using a high frame rate camera. We develop reduced coordinate spaces to interpret the 3D sliding skin motion using monocular videos.

3.1 Scene Flow

Scene flow is the estimation of the 3D flow field using a binocular camera set up. We follow the work of [70] to model the scene flow as an optimization problem. We summarize their work introducing similar notations in this section. The model uses consecutive images from two cameras and formulates the problem in a four-frame setup as shown in Figure 3.1. We estimate the flow fields between images of consecutive frames as well as the flow fields between each of the stereo pairs. We infer the 3D motion of points in the scene using the above estimation of different flow fields.

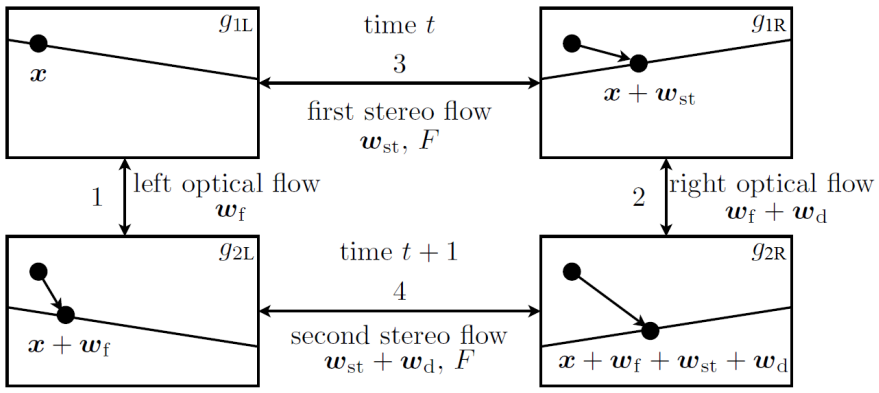

Figure 3.1: Frames of a stereo setup, reproduced from [70]

3.1.1 Definitions

Our setup includes two cameras labeled as left and right cameras. The images from two consecutive frames are used at time instants t and $t+1$. The two consecutive frames of the left camera are denoted by g_{1L} and g_{2L}. Similarly, the consecutive frames of right camera are denoted by g_{1R} and g_{2R}. This makes a four-frame set up for scene flow estimation across frames $(g_{1L}, g_{2L}, g_{1R}, g_{2R})$. We define position vector, $\mathbf{x} = (x,y)^T$ that describes the coordinates of the image plane. We denote the Fundamental matrix of the stereo set up by F. The optical flow of the left camera is given by $\mathbf{w_f} = (u_f, v_f)^T$. We represent the stereo flow between g_{1L} and g_{1R} by $\mathbf{w_{st}} = (u_{st}, v_{st})^T$. There is also a difference flow $\mathbf{w_d} = (u_d, v_d)^T$, which measures the change in stereo flow between two cameras from t to $t+1$. It can also be seen as a change in optical flow in the right camera images. Here, u and v are the flow components along x and y directions in the image. By estimating these six parameters, $\{\mathbf{w_f}, \mathbf{w_{st}}, \mathbf{w_d}\}$; we can determine the three dimensional flow of a scene.

3.1.2 Scene Flow Model

Scene flow estimation is modeled as minimizing the energy functional given by

$$E = \int_{\Omega} (E_D + E_E + E_S) d\mathbf{x}, \tag{3.1}$$

where, E_D is the data term that enforces the constraint that features present in a scene remain same across the four frames. E_E refers to the epipolar constraint which enforces that image pixels of stereo pairs should lie along the epipolar lines. E_S is the smoothness constraint that provides a good regularization for scene flow computation. The integral is over all the image pixels $\mathbf{x}$ in the domain of the image plane, Ω. In our work, we use an efficient optimizer, SAG [57] to estimate the numerical solution.

Data Term

The data term E_D models four constrains between the input frames given by

$$E_{D1} = \Psi(|g_{2L}(\mathbf{x}+\mathbf{w_f}) - g_{1L}(\mathbf{x})|^2), \tag{3.2}$$

$$E_{D2} = \Psi(|g_{2R}(\mathbf{x}+\mathbf{w_f}+\mathbf{w_{st}}+\mathbf{w_d}) - g_{1R}(\mathbf{x}+\mathbf{w_{st}})|^2), \tag{3.3}$$

$$E_{D3} = \Psi(|g_{1R}(\mathbf{x}+\mathbf{w_{st}}) - g_{1L}(\mathbf{x})|^2), \tag{3.4}$$

$$E_{D4} = \Psi(|g_{2R}(\mathbf{x}+\mathbf{w_f}+\mathbf{w_{st}}+\mathbf{w_d}) - g_{2L}(\mathbf{x}+\mathbf{w_f})|^2), \tag{3.5}$$

where, E_{D1} and E_{D2} define the optical flow constraint between images from the left and right images respectively. E_{D3} and E_{D4} model the stereo flow constraint between the stereo pairs of each of the cameras. We use a robust convex penalty function $\Psi(s^2) = \sqrt{s^2+\varepsilon^2}$ and set $\varepsilon = 0.001$. The sub-quadratic penalization function, $\Psi(s^2)$, was introduced by Black and Anandan in [8] and can be seen as an $L1$ approximation. It provides robustness across outliers in the data terms caused by noise and occlusion. We also assume gradient constancy and include RGB channels of the images in our data term. Hence our data term in Equation 3.2 becomes

$$E_{D1} = \Psi\left(\sum_{i\in\{r,g,b\}} \left(|g^i_{2L}(\mathbf{x}+\mathbf{w_f}) - g^i_{1L}(\mathbf{x})|^2 + \gamma|\nabla g^i_{2L}(\mathbf{x}+\mathbf{w_f}) - \nabla g^i_{1L}(\mathbf{x})|^2\right)\right), \tag{3.6}$$

where, $\gamma \geq 0$ is a constant that denotes the influence of gradient constancy assumption. $i \in \{r,g,b\}$ denotes RGB color channels. ∇ is the spatial gradient operator. We modify the other data terms accordingly to include RGB channels and gradient

constancy. We finally combine the data terms and get

$$E_D = E_{D1} + E_{D2} + E_{D3} + E_{D4}. \tag{3.7}$$

Epipolar Term

The epipolar energy term models the geometric relationship between the stereo image pairs and applies the constraint along the epipolar lines of the two cameras. The fundamental matrix F models the geometric transform between the stereo cameras and defines the epipolar constraint. The epipolar terms are given by

$$E_{E1} = \Psi\left(\left((\mathbf{x}+\mathbf{w_{st}})_h^T F(\mathbf{x})_h\right)^2\right), \tag{3.8}$$

$$\begin{aligned} E_{E2} = \Psi\Big(&\frac{1}{2}\left((\mathbf{x}+\mathbf{w_f}+\mathbf{w_{st}}+\mathbf{w_d})_h^T F(\mathbf{x}+\mathbf{w_a})_h\right)^2 \\ &+ \frac{1}{2}\left((\mathbf{x}+\mathbf{w_a}+\mathbf{w_{st}}+\mathbf{w_d})_h^T F(\mathbf{x}+\mathbf{w_f})_h\right)^2\Big) + \mu(|\mathbf{w_f}-\mathbf{w_a}|^2), \end{aligned} \tag{3.9}$$

where, $(\mathbf{x})_h = (x, y, 1)^T$ and denotes the homogeneous transformation. The epipolar constraints enforces the pixels to lie along the same epipolar line. E_{E1} is linear in $\mathbf{w_{st}}$, whereas E_{E2} is linear in $\mathbf{w_{st}}$ and $\mathbf{w_d}$, but it is quadratic in $\mathbf{w_f}$. Hence as in [70] , we introduce auxiliary variable $\mathbf{w_a}$ to make E_{E2} linear in $\mathbf{w_f}, \mathbf{w_{st}}$ and $\mathbf{w_d}$. μ is the coupling term between $\mathbf{w_f}$ and $\mathbf{w_a}$. Finally, we combine the epipolar terms to get

$$E_E = \beta_1 E_{E1} + \beta_2 E_{E_2}, \tag{3.10}$$

where β_1 and β_2 are constants.

Smoothness Term

In order to avoid flow discontinuities, regularization terms are added in the energy functional causing spatial smoothness. The smoothness regularization terms are

given by

$$E_{S1} = \Psi(|\nabla \mathbf{w_f}|^2), \tag{3.11}$$

$$E_{S2} = \Psi(|\nabla \mathbf{w_{st}}|^2), \tag{3.12}$$

$$E_{S3} = \Psi(|\nabla \mathbf{w_d}|^2), \tag{3.13}$$

$$E_S = \alpha_1 E_{S1} + \alpha_2 E_{S2} + \alpha_3 E_{S3}, \tag{3.14}$$

where, $\alpha_1, \alpha_2, \alpha_3$ are constants.

3.1.3 Numerical Solution

In order to obtain the solution to the energy function, we discretize the energy functional (Equation 3.1). We use a coarse-to-fine approach using image pyramids to solve for incremental flows. Let $\mathbf{d} = (du_f, dv_f, du_{st}, dv_{st}, du_d, dv_d, 1)^T$. Then, the data term from Equation 3.2 can be factorized in terms of $\mathbf{d}$ as $E_{D1} = \Psi(\mathbf{d}^T \hat{J}_1 \mathbf{d})$, and a data tensor $\hat{J}_1$ is obtained as shown in [70]. Similarly, data tensors $\hat{J}_2, \hat{J}_3$ and $\hat{J}_4$ can be obtained by factorizing other data terms.

Similarly, we can obtain epipolar tensors by defining

$$\mathbf{d}_1 = (du_{st}, dv_{st}, 1)^T, \tag{3.15}$$

$$\mathbf{d}_2 = (du_f + du_{st} + du_d, dv_f + dv_{st} + dv_d, 1)^T, \tag{3.16}$$

$$\mathbf{d}_3 = (du_a, dv_a, 1)^T, \tag{3.17}$$

$$\mathbf{d}_4 = (du_a + du_{st} + du_d, dv_a + dv_{st} + dv_d, 1)^T, \tag{3.18}$$

$$\mathbf{d}_5 = (du_f, dv_f, 1). \tag{3.19}$$

The epipolar tensors $\{\hat{E}_i | i \in \{1,2,3,4,5\}\}$ can be factorized from epipolar terms of Section 3.1.2 as

$$E_{E1} = \Psi(\mathbf{d}_1^T \hat{E}_1 \mathbf{d}_1), \tag{3.20}$$

$$E_{E2} = \Psi\left(\frac{1}{4}\mathbf{d}_2^T \hat{E}_2 \mathbf{d}_2 + \frac{1}{4}\mathbf{d}_3^T \hat{E}_3 \mathbf{d}_3 + \frac{1}{4}\mathbf{d}_4^T \hat{E}_4 \mathbf{d}_4 + \frac{1}{4}\mathbf{d}_5^T \hat{E}_5 \mathbf{d}_5\right) + \mu(|\mathbf{w}_f + d\mathbf{w}_f - \mathbf{w}_a - d\mathbf{w}_a|)^2. \tag{3.21}$$

Thus, the energy function can be represented in terms of incremental flows as

$$
\begin{aligned}
&E(d\mathbf{w_f}, d\mathbf{w_{st}}, d\mathbf{w_d}, d\mathbf{w_a}) = \\
&\int_\Omega \Big(\Psi(\mathbf{d}^T \hat{J}_1 \mathbf{d}) + \Psi(\mathbf{d}^T \hat{J}_2 \mathbf{d}) + \Psi(\mathbf{d}^T \hat{J}_3 \mathbf{d}) + \Psi(\mathbf{d}^T \hat{J}_4 \mathbf{d}) \Big) \\
&+ \beta_1 \Psi(\mathbf{d}_1^T \hat{E}_1 \mathbf{d}_1) + \beta_2 \Psi(\frac{1}{4}\mathbf{d}_2^T \hat{E}_2 \mathbf{d}_2 + \frac{1}{4}\mathbf{d}_3^T \hat{E}_3 \mathbf{d}_3 + \frac{1}{4}\mathbf{d}_4^T \hat{E}_4 \mathbf{d}_4 + \frac{1}{4}\mathbf{d}_5^T \hat{E}_5 \mathbf{d}_5) \\
&+ \alpha_1 \Psi(|\nabla(\mathbf{w_f} + d\mathbf{w_f})|^2) + \alpha_2 \Psi(|\nabla(\mathbf{w_{st}} + d\mathbf{w_{st}})|^2) \\
&+ \alpha_3 \Psi(|\nabla(\mathbf{w_d} + d\mathbf{w_d})|^2) + \beta_2 \mu(\mathbf{w_f} + d\mathbf{w_f} - \mathbf{w_a} - d\mathbf{w_a}) \Big) d\mathbf{x}.
\end{aligned}
\tag{3.22}
$$

In order to solve for the stepwise flow increments, we write the Euler-Lagrange forms [20] of Equation 3.22 and factor out incremental flow terms as

$$\partial_{\mathbf{u}} E = \mathbf{A_u}\mathbf{u} - \mathbf{b_u}, \tag{3.23}$$

$$\partial_{\mathbf{v}} E = \mathbf{A_v}\mathbf{v} - \mathbf{b_v}, \tag{3.24}$$

where $\mathbf{u} = (du_f, du_{st}, du_d, du_a)$ and $\mathbf{v} = (dv_f, dv_{st}, dv_d, dv_a)$. We now combine the Euler-Lagrange forms as

$$\begin{bmatrix} \partial_{\mathbf{u}} E \\ \partial_{\mathbf{v}} E \end{bmatrix} = \begin{bmatrix} \mathbf{A_u} & 0 \\ 0 & \mathbf{A_v} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} - \begin{bmatrix} \mathbf{b_u} \\ \mathbf{b_v} \end{bmatrix} \tag{3.25}$$

in a matrix representation. The right hand side can be represented as $\mathbf{A}d\mathbf{w} - \mathbf{b}$, where $d\mathbf{w}$ is a $8n$-dimensional vector. $d\mathbf{w}$ represents the incremental flows $(\mathbf{u}^T, \mathbf{v}^T)^T$ of all the n pixels in the image by a single column vector following a lexical order. The linear system is solved by logistic regression iteratively using SAG [57] and is represented as

$$f(d\mathbf{w}) = \sum_{i=1}^{8n} log(1 + exp(-b_i d\mathbf{w}^T \mathbf{a}_i)) + \frac{\lambda}{2}||d\mathbf{w}||^2 \tag{3.26}$$

and minimizing $f(d\mathbf{w})$, where $\mathbf{a_i}$ are the rows of the matrix $\mathbf{A}$ and b_i are the rows of the matrix $\mathbf{b}$. We apply an L2-regularization on the flow increments as shown in Equation 3.26.

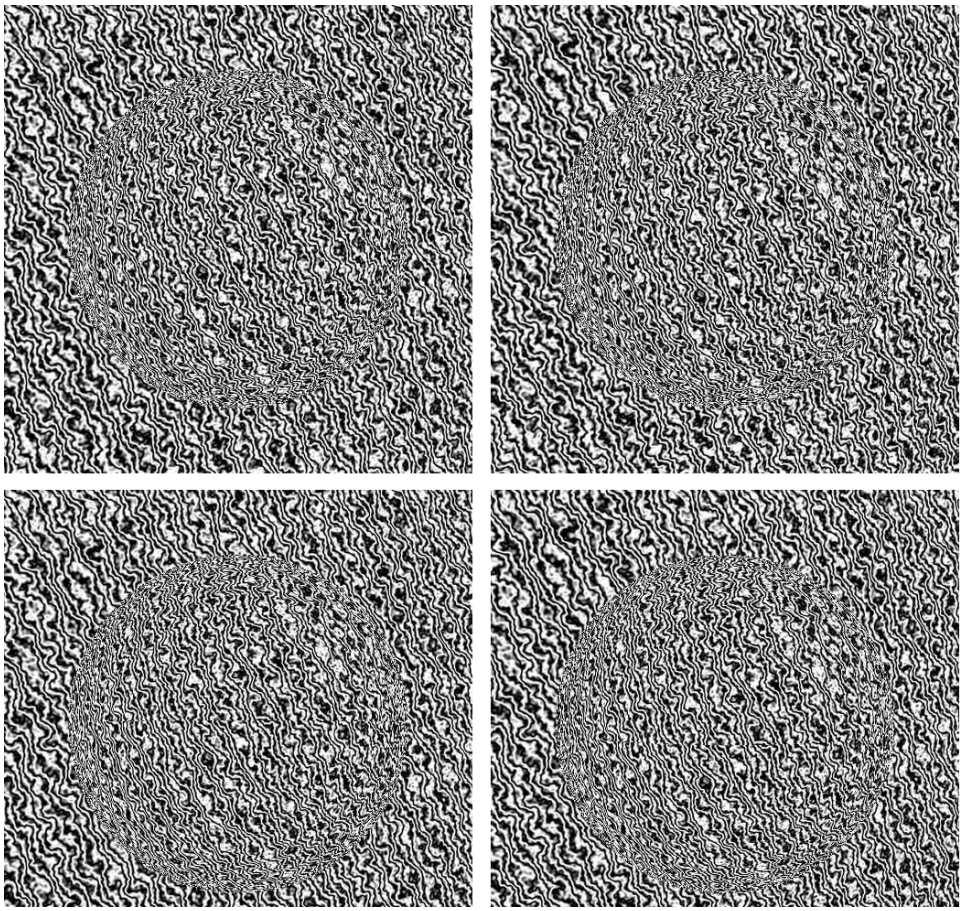

Figure 3.2: Frames of General Sphere dataset from [70]: Sequence from the left camera (left) and corresponding sequence from the right camera (right).

3.1.4 Experiments and Results

We obtain the stereo image sequence $g_{1L}, g_{2L}, g_{1R}, g_{2R}$ from the dataset provided by Valgaerts et al. [70]. It consists of a synthetic sequence of a rotating textured sphere as shown in Figure 3.2. The Fundamental Matrix, F is also provided in the dataset. In a real scenario, the sequence can be obtained from a stereo camera setup and the Fundamental Matrix can be obtained from camera calibration techniques.

We initialize the flow fields $(u_f, v_f, u_{st}, v_{st}, u_d, v_d)^T$ using Brox and Malik's LDOF [11]. We then follow a coarse-to-fine approach using image pyramids with

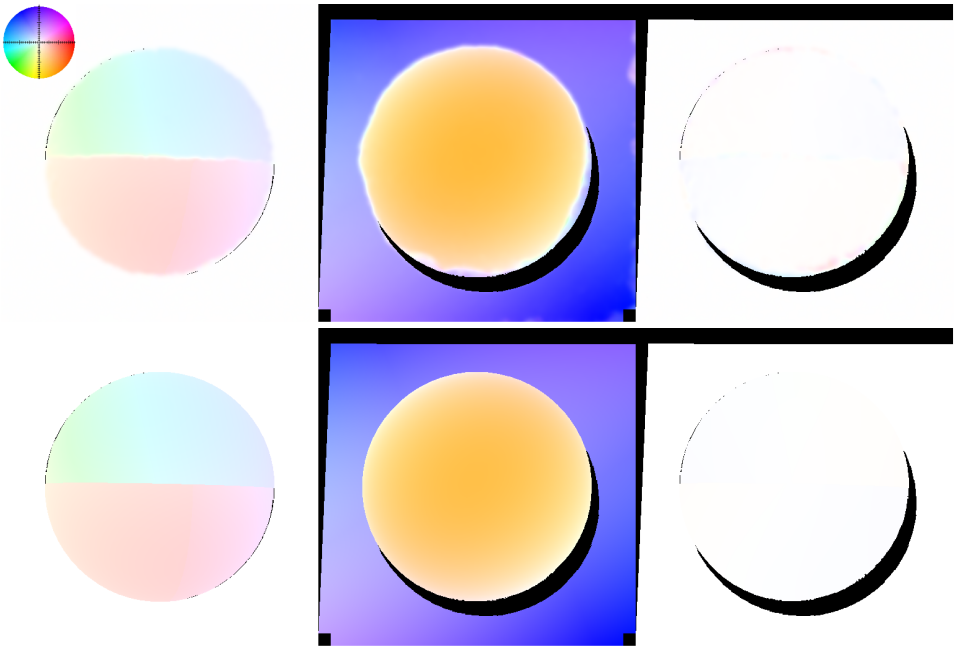

Figure 3.3: Results of Scene Flow estimation (top) with Ground truths (bottom). From left to right: forward flow ($\mathbf{w_f}$), stereo flow ($\mathbf{w_{st}}$) and difference flow ($\mathbf{w_d}$). The color coding is shown on top-left. (Regions under occlusion are manually removed)

Method	RMS Errors		
	(u_f, v_f, u_d, v_d)	(u_f, v_f)	(u_{st}, v_{st})
Our method initialized with [11]	0.49	0.54	1.60
Valgaerts et al. [70] (N + SR)	0.61	0.59	1.61
Valgaerts et al. [70] (N + JR)	0.63	0.59	1.86
Valgaerts et al. [70] (JR)	0.67	0.64	2.08
Wedel et al. [77] with Ground Truth	2.40	0.65	-
Wedel et al. [77] (87%)	2.45	0.66	2.9
Huguet and Devernay [34]	2.51	0.69	3.8
Wedel et al. [77] (100%)	2.55	0.77	10.9

Table 3.1: Comparison of RMS errors for Scene Flow. N: Normalization, SR: Separate Regularization, JR: Joint Regularization.

Algorithm 1: Scene Flow Estimation

Data: Sequence : $g_{1L}, g_{2L}, g_{1R}, g_{2R}$, Fundamental Matrix, F
Result: Scene Flow: $(u_f, v_f, u_{st}, v_{st}, u_d, v_d)^T$
Initialize (u_f, v_f) from g_{1L}, g_{2L}, using [11] ;
Initialize (u_{st}, v_{st}) from g_{1L}, g_{1R}, using [11];
Initialize $(u_a, v_a) \leftarrow (u_f, v_f)$;
Compute $(\tilde{u_f}, \tilde{v_f})$ from g_{1R}, g_{2R}, using [11] ;
Compute $(\hat{u_f}, \hat{v_f})$ by warping $(\tilde{u_f}, \tilde{v_f})$ using (u_{st}, v_{st}) ;
Initialize $(u_d, v_d) \leftarrow (\hat{u_f}, \hat{v_f})$;
Compute pyramid images with $\eta = 0.9$;
for *Each level of Pyramid* **do**
    Fetch the downsampled pyramid frames $g^i_{1L}, g^i_{2L}, g^i_{1R}, g^i_{2R}$;
    Resample flows $\mathbf{w} := (u_f, v_f, u_{st}, v_{st}, u_d, v_d, u_a, v_a)^T$;
    Compute Data Tensors $\hat{J}_1, \hat{J}_2, \hat{J}_3, \hat{J}_4$;
    Compute Epipolar Tensors $\hat{E_1}, \hat{E_2}, \hat{E_3}, \hat{E_4}, \hat{E_5}$;
    Initialize flow increments $d\mathbf{w} \leftarrow 0$;
    Arrange all tensors in the matrix form $\mathbf{A}d\mathbf{w} - \mathbf{b}$;
    while $||\mathbf{A}d\mathbf{w} - \mathbf{b}|| > tol$ **do**
        Compute d$\mathbf{w}$ by solving Equation 3.26 using [57];
        Update $\mathbf{A}(\mathbf{w}) \leftarrow \mathbf{A}(\mathbf{w} + d\mathbf{w})$;
    end
    Update Flows, $\mathbf{w} \leftarrow \mathbf{w} + d\mathbf{w}$;
end

a downsampling factor of $\eta = 0.9$. At each level of the pyramid, we compute the data and epipolar tensors by factoring out the incremental flows from the corresponding terms (Section 3.1.3). We then rearrange the terms in a linear system $\mathbf{A}d\mathbf{w} - \mathbf{b}$ and solve it using SAG [57] with $\lambda = 0.1$. The framework is described in algorithm 1.

We compare our scene flow algorithm with the pre-existing algorithms and show the results in Table 3.1. We see that our scene flow performs better than pre-existing methods with the introduction of the stochastic optimizer and the *L2* regularization term for flow increments. The qualitative results of scene flow estimation are shown in Figure 3.3.

3.2 Skin Flow

We now develop a system to compute 3D flow of the skin using a high frame rate monocular camera and a pre-computed 3D mesh. We accomplish it by representing the skin using reduced spaces. To represent the motion of skin on the face, and its measurement by video cameras, we use the reduced coordinate representation of skin introduced by [39]. In Section 3.2.1, we summarize our method and describe the details in [45].

3.2.1 Skin Tracking Model

We represent skin by a 2D map denoted as the *skin atlas* in Figure 3.4. The skin slides on a fixed 3D "body" corresponding to the shape of the head around the eyes. It is mapped to 3D skin in skin space using parameterization π. This can undergo rigid transformation ϕ_0 to reach an initial state $\boldsymbol{x}$ at $t = 0$ in physical space. The skin undergoes deformations ϕ_t with time to reach $\boldsymbol{x_t}$. The projection of these states can be observed by tracking the corresponding points $\boldsymbol{u_t}$ using a single camera. P denotes the projection transform.

The parameterization π is obtained using Faceshift [78] and ϕ_0 is estimated using the Orthogonal Procrustes algorithm. The projected points $\boldsymbol{u_t}$ are tracked using LDOF [11]. The states $\boldsymbol{x_t}$ are then inferred using projection P. The flow corrections are applied by observing the skin atlas using the estimated values of π and ϕ_0. Flows are corrected such that the tracked point across video remains same as the corresponding skin point in the skin atlas.

3.2.2 Experiments and Results

To measure motion around the eye region, we used a single Grasshopper3[1] camera that can capture up to 120 fps with image resolution of 1960×1200 pixels. The setup is shown in Figure 3.5. The actor sits on a chair and faces the camera with the head rested on a chin rest. The scene is lit by a DC powered LED source[2] to overcome the flickering due to aliasing effects of an AC light source on a high frame rate capture. We use polarizing filters with the cameras to reduce specularity.

[1] Point Grey Research, Vancouver, Canada

[2] https://www.superbrightleds.com

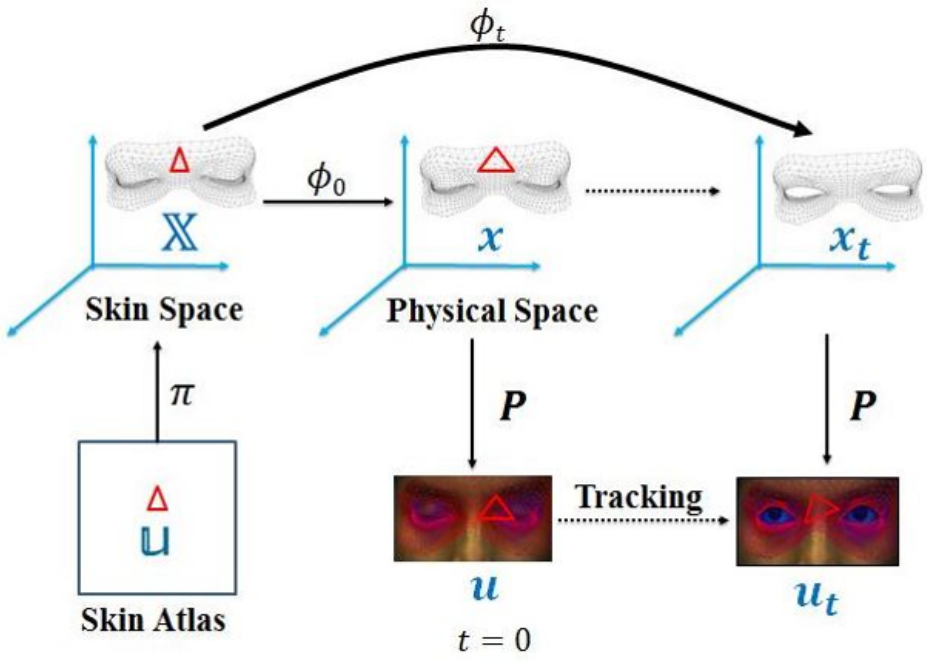

Figure 3.4: Overview of the spaces related to skin tracking.

Camera Calibration

We perform camera calibration using Matlab's Computer Vision Toolbox. The camera used for capturing video data is calibrated using a standard checkerboard grid. We use a 7x9 grid of black and white squares as shown in the Figure 3.6 and take its snapshots at different viewing angles. The width of each square is 7.25 mm. The checkerboard is placed in the same position as the face of the subject during video capture. After obtaining the checkerboard patterns at different orientations, we reconstruct the position of checkerboard planes in 3D (Figure 3.6) and compute the projection matrix, P using [83].

Mesh Reconstructions

Our skin tracking algorithm requires a subject specific 3D mesh and the parametrization π. To acquire this, we use FaceShift [78] technology with a Kinect RGB/D camera. This process takes less than 15 minutes per subject.

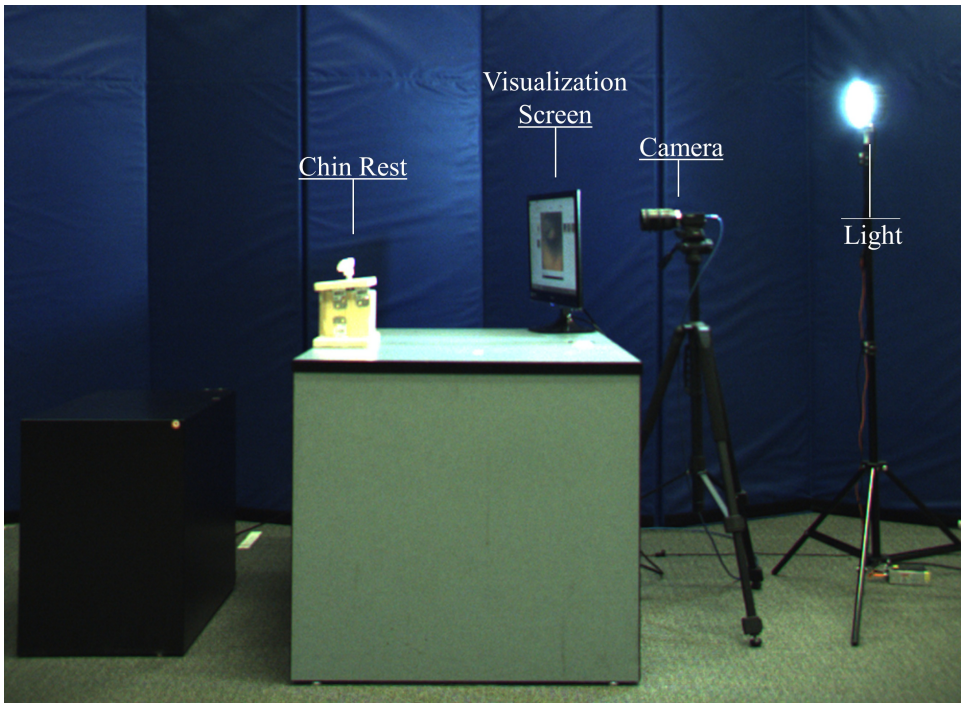

Figure 3.5: The capture setup used in our experiments.

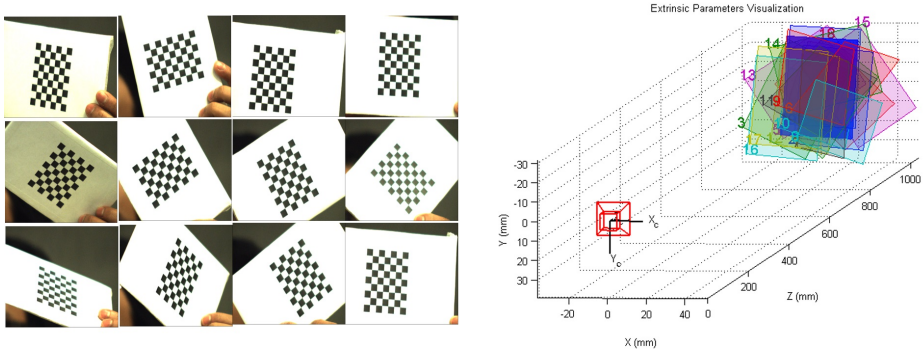

Figure 3.6: Checkerboard snapshots for Calibration (Left) and Reconstruction (right)

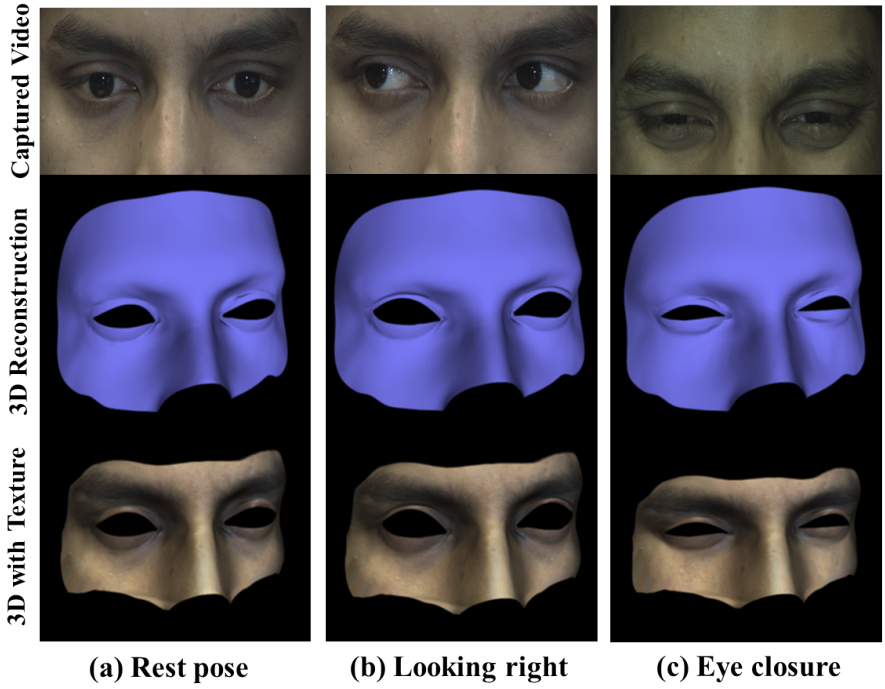

Figure 3.7: Tracking results for monocular videos. From top to bottom: Images from videos for three examples, corresponding 3D reconstructed mesh, and 3D meshes with reconstructed texture.

We show the reconstructions using our skin tracking approach in Figure 3.7. We can see robust reconstructions of our 3D mesh using monocular skin tracking. The tracking of skin around the eyes and eyelid in the reconstruction is compared with the original video. Thus, we obtain realistic reconstruction of skin movement in the 3D meshes.

Chapter 4

Texture Estimation

In this chapter, we will discuss the framework for recovering the skin texture around the eye and the texture of the eyelashes. We will start by formulating our problem as a superresolution estimation problem based on optical flow. We follow an approach of texture estimation in a Bayesian framework. We then introduce material mask functions to recover eyelashes or skin selectively under mutual occlusion. We show the recovered textures and the improvements.

4.1 Overview

Developments in dynamic performance capture of faces have led to their high quality and realistic rendering [25]. However, there has not been significant development to capture the details of the eyes. Eyes have extremely fast and subtle motions, like blink, saccades etc., which are difficult to capture. This also influences the tracking of fast motions, structure and the texture of the eyes. Capturing texture of the details of the eye, such as the upper and lower eyelids and the eyelashes is affected not only by the eye's fast motion but also by occlusion in natural expressions.

This chapter is motivated by detailed capture of the texture around the eye in natural expressions. We use the motion of eyes from multiple frames of a high frame rate camera to capture the detailed texture. Using optical flow methods, we track the details of the eyes across various frames and reconstruct the texture using

a Bayesian approach. We introduce material mask functions in order to address various occlusions that arise as a result of eyelash and skin motion. We show that, using an optical flow based superresolution approach in the video data of natural expressions, we can reconstruct high resolution texture of the eyes.

4.2 Theory

We formulate the estimation of texture of the eyes using an optical-flow based superresolution approach motivated by [24]. Our formulation uses Bayesian estimation in a generative imaging model. We will consider the estimation of texture of the eyelashes and the skin under mutual occlusion of each other. Under the small and fast motions of the eye, parts of the skin are often occluded by the eyelashes and are visible only for a small duration. As such, both eyelashes and the eyelid skin are difficult to track as their corresponding pixel-features change dynamically. Moreover, videos of the eyes at a high frame rate have a lot of noise which we model as a Gaussian random noise. In our framework, we jointly estimate the texture using optical flow and the noise present in the input data.

In Figure 4.1, we see the motion of the eyelashes which are 13 frames apart in a 200 fps video sequence. A closer look at such a video sequence highlights the horizontal motion of the skin underneath as a result of forces from the orbicularis muscle of the eye.

4.2.1 Definitions

We consider a set of $2n+1$ images, $[I_{-n}(x)...I_0(x)...I_n(x)]$ obtained from our video sequence. Each image obtained using a high frame rate camera has a resolution of $l \times l$. Each of the values of x represent a *pixel* in the image. In computer graphics, an object is associated with its structure, the mesh geometry and the texture associated with it. We would like to construct a high resolution texture image $J_0(\mathbf{x})$ with a resolution $h \times h$ which corresponds to the reference image $I_0(x)$. Each of the values of $\mathbf{x}$ in J_0 represent a *texel* in the texture image. Our approach utilizes the neighboring images in the video sequence $I_i(x)$ to obtain the high resolution texture image.

We also define a transformation T_i that maps a point $\mathbf{x}$ in texture space of $J_0(\mathbf{x})$

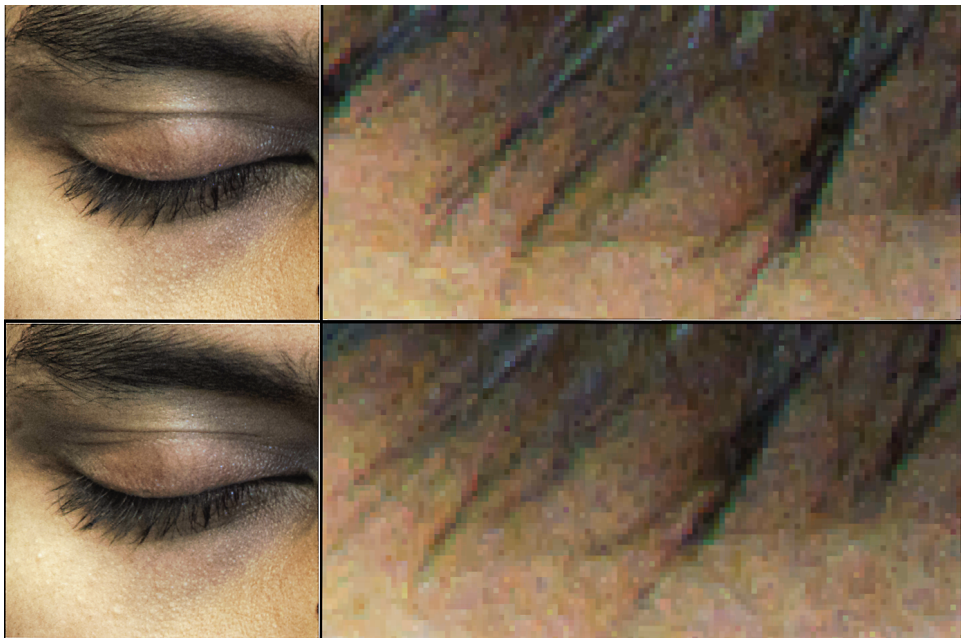

Figure 4.1: The first and last frames of the eyelashes in motion. As the eyelid moves up, the lower lid skin has a subtle sideways motion revealing occluded skin texture.

to a corresponding point $T_i(\mathbf{x})$ in $I_i(x)$. The transformation, T_i can be decomposed into an optical flow field, $\mathbf{w}_i(\mathbf{x})$ and a barycentric mapping $D(\mathbf{x})$ from texture image to video image. Therefore, a texel in the texture space $J_0(\mathbf{x})$ is warped using the flow $\mathbf{w}_i(\mathbf{x})$ and then mapped to the image $I_i(x)$ using the barycentric transform $D(\mathbf{x})$. It is noted that the flow fields $\mathbf{w}_i(\mathbf{x})$ are defined on the texture space corresponding to high resolution texture image $J_0(\mathbf{x})$. The mapping can hence be written as

$$T_i(\mathbf{x}) = D(\mathbf{x} + \mathbf{w}_i(\mathbf{x})).$$

In a similar manner, we can define the inverse transformation T_i^{-1} that maps a point x in $I_i(x)$ to a corresponding point $T_i^{-1}(x)$ in the texture image $J_0(\mathbf{x})$. It is given by,

$$T_i^{-1}(x) = D^{-1}(x) - \mathbf{w}_i(T_i^{-1}(x)).$$

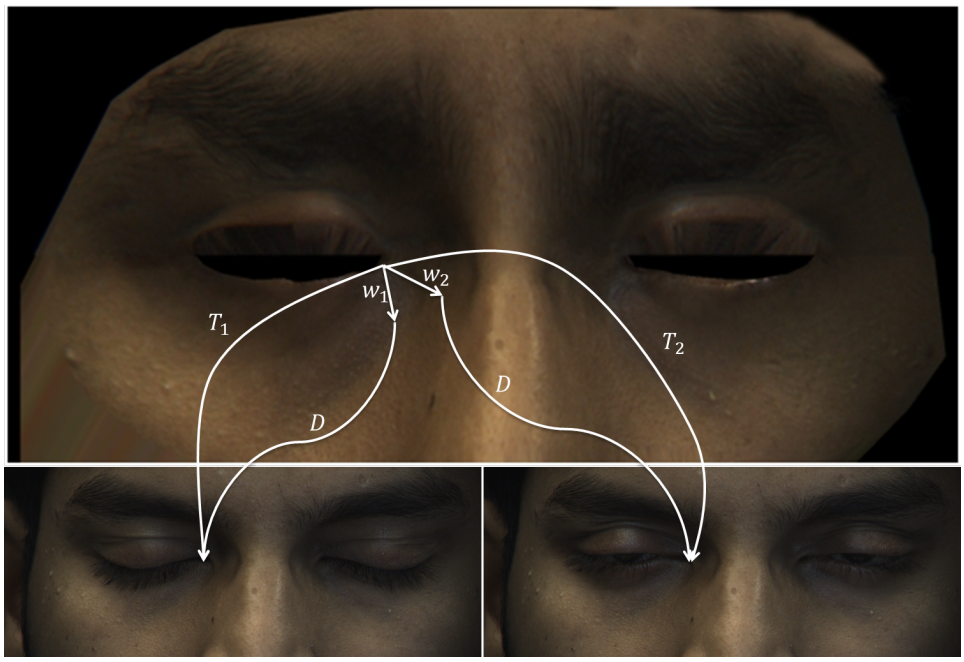

Figure 4.2: The transform T_i maps texture space(top) and the image frames (bottom). The transform T_i can be represented as a combination of barycentric transform, D and flow fields $\mathbf{w}_i$.

4.2.2 Generative Model

We are now in a position to describe our imaging model. In order to generate the video image $I_i(x)$, we warp the texture image $J_0(\mathbf{x})$ by the flow field $\mathbf{w}_i(\mathbf{x})$ and then project it on to image space using the barycentric transform, D. This process can be represented by a single transformation $T_i(\mathbf{x})$ as described in Section 4.2.1. We apply an optical blur using a Gaussian filter g after the transformation $T_i(x)$. We also model random noise present in the system using an Additive White Gaussian Noise (AWGN) term, ε. The generative model is formulated as

$$I_i(x) = g * J_0(T_i^{-1}(x)) + \varepsilon, \tag{4.1}$$

where $\varepsilon \sim \mathcal{N}(\mathbf{0}, \Sigma)$, and g simulates a Gaussian optical blur using the convolution operator, $*$.

4.2.3 Bayesian Estimation

The generative model produces images $I_i(\mathbf{x})$ from the texture $J_0(\mathbf{x})$ depending on the flow field, $\mathbf{w}_i$, and the noise parameter, Σ that models its variance. Thus, we formulate our problem as the estimation of $\theta = \{J_0, \mathbf{w}_i, \Sigma\}$ that maximizes the posterior,

$$p(\theta|I) \propto p(I|\theta)p(\theta). \tag{4.2}$$

In order to obtain the conditional data likelihood term, we make i.i.d assumptions on image noise for all pixels in the image. The conditional data likelihood term, $p(I|\theta)$ then becomes the product of probabilities of individual observed pixels over all the frames,

$$p(I|\theta) = \prod_{i=-n}^{n} \prod_{\mathbf{x}} p(I_i(T_i(\mathbf{x}))|\theta), \tag{4.3}$$

where, $\mathbf{x} \in [1,h] \times [1,h]$ are the texels in the high resolution texture image. The conditional probability of an observed texel value can be modeled by a normal distribution given the previous estimates of texture, J_0, and noise variance, Σ, as

$$p(I|\theta) = \prod_{i=-n}^{n} \prod_{\mathbf{x}} \left[\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} exp\left(-\frac{1}{2}\mathbf{m}_i^T \Sigma^{-1} \mathbf{m}_i \right) \right], \tag{4.4}$$

where, $\mathbf{m}_i = I_i(T_i(\mathbf{x})) - g * J_0(\mathbf{x})$, d is the dimensionality of $\mathbf{m}_i$ and is equal to number of channels in the image frame. $\mathbf{m}_i$ models the error between the observed pixel values in the image I_i and the previous estimate of high resolution texture J_0 using our generative model.

Now, we turn to the formulation of our prior term. Several different prior terms for Bayesian estimation have been proposed before. In the event of unavailable data, these prior terms generally model smoothness over texels to regularize the optimization term. We factorize the prior term into texture dependent and flow dependent parts motivated by [24]. The prior term can be written as

$$p(\theta) \propto p(J_0) \prod_{i=-n}^{n} p(\mathbf{w}_i), \tag{4.5}$$

where, $p(J_0)$ is the texture dependent term, and $p(\mathbf{w}_i)$ is the flow dependent term over individual frames. The flow dependent prior is modeled as an exponential distribution parametrized by width λ_f. This prior applies a smoothness constraint over the optical flow field $\mathbf{w}_i(\mathbf{x})$ and is given by

$$p(\mathbf{w}_i) \propto exp\left(-\frac{1}{\lambda_f}\sum_{\mathbf{x}}||\nabla \mathbf{w}_i(\mathbf{x})||^2\right). \tag{4.6}$$

This prior is generally used with computing the flow fields on a coarse-to-fine grid involving optimization over multiple layers [1]. We introduce a second prior for optical flow field that can be modeled by precomputing flow fields using [62]. The model uses an exponential distribution that expresses the deviation of new flow fields $\mathbf{w}_i$ from the precomputed flow fields $\mathbf{w}_i^*$. Here, $\mathbf{w}_i^*$ is the optical flow between $J_0(T_0(x))$ and $J_0(T_i(x))$. This can be written as

$$p(\mathbf{w}_i) \propto exp\left(-\frac{1}{\lambda_f}\sum_{\mathbf{x}}||\mathbf{w}_i(\mathbf{x}) - \mathbf{w}_i^*(\mathbf{x})||^2\right). \tag{4.7}$$

Similarly, we can introduce a prior on the high resolution texture J_0 which introduces a smoothness constraint. The prior can be modeled as an exponential distribution with a width λ_j,

$$p(J_0) \propto exp\left(-\frac{1}{\lambda_j}\sum_{\mathbf{x}}||\nabla J_0(\mathbf{x})||^2\right). \tag{4.8}$$

However, in the presence of observations from the image frames, we would like to model our prior based on available data. To obtain this prior, we use the average of all the image frames, transformed on to texture image coordinates which is given by

$$J_0^*(\mathbf{x}) = \frac{1}{2n+1}\sum_i I_i(T(\mathbf{x})). \tag{4.9}$$

We express the prior on the texture as an exponential distribution of the error over $J_0^*(\mathbf{x})$. The prior over J_0 becomes

$$p(J_0) \propto exp\left(-\frac{1}{\lambda_j}\sum_{\mathbf{x}}||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2\right). \tag{4.10}$$

However, the above prior results in artifacts over the computed texture image. The artifacts are generally due to discontinuities over the texture. Hence, we combine the best features of (Equation 4.8) and (Equation 4.10) to obtain a prior which removes the discontinuities by introducing a smoothness term. Hence, we obtain the prior as

$$p(J_0) \propto exp\left(-\frac{1}{\lambda_j}\sum_{\mathbf{x}}\nabla||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2\right). \tag{4.11}$$

After obtaining the likelihood and prior terms of the Bayesian estimation, we proceed to compute the posterior and optimize it to obtain the texture, flow fields and the noise, $\theta = \{J_0, \mathbf{w}_i, \Sigma\}$ in our system.

4.2.4 Optimization

After modeling the posterior by setting up the likelihood (Equation 4.4) and prior (Equation 4.5) terms, we aim to maximize the posterior to estimate $\theta = \{J_0, \mathbf{w}_i, \Sigma\}$. We simplify our formulation by taking the negative logarithm of the posterior (Equation 4.2) that can be minimized in an optimization framework. From Equation 4.2, we have

$$\hat{\theta} = arg\min_{\theta}\left(-\log\{p(I|\theta)p(\theta)\}\right), \tag{4.12}$$

$$= arg\min_{\theta}\left(-\log p(I|\theta) - \log p(\theta)\right). \tag{4.13}$$

We now evaluate the negative logarithm of the likelihood and the prior terms separately. From (Equation 4.4), we evaluate the likelihood term as

$$-\log p(I|\theta) = -\log\prod_{i=-n}^{n}\prod_{\mathbf{x}}\left[\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}exp\left(-\frac{1}{2}\mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_i\right)\right], \tag{4.14}$$

$$= -\sum_{i=-n}^{n}\sum_{\mathbf{x}}\left[\log\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} - \left(\frac{1}{2}\mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_i\right)\right], \tag{4.15}$$

$$= \sum_{i=-n}^{n}\sum_{\mathbf{x}}\log\left\{(2\pi)^{d/2}|\Sigma|^{1/2}\right\} + \sum_{i=-n}^{n}\sum_{\mathbf{x}}\frac{1}{2}\mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_i, \tag{4.16}$$

where, $\mathbf{m}_i = I_i(T_i(\mathbf{x})) - g * J_0(\mathbf{x})$.

Similarly, we evaluate the negative logarithm of the prior terms. From (Equation 4.5), using priors (Equation 4.6) and (Equation 4.8), we have

$$-\log p(\theta) = \frac{1}{\lambda_j}\sum_{\mathbf{x}} ||\nabla J_0(\mathbf{x})||^2 + \frac{1}{\lambda_f}\sum_{i=-n}^{n}\sum_{\mathbf{x}} ||\nabla \mathbf{w}_i(\mathbf{x})||^2. \tag{4.17}$$

The above prior term introduces only a smoothness constraint on the flow fields $\mathbf{w}_i$ and the high resolution texture J_0. We modify the prior terms using the priors (Equation 4.7) and (Equation 4.11) to take into account the observations made from input data as well. Hence we recompute the prior terms as

$$-\log p(\theta) = \frac{1}{\lambda_j}\sum_{\mathbf{x}} \nabla||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2 + \frac{1}{\lambda_f}\sum_{i=-n}^{n}\sum_{\mathbf{x}} ||\mathbf{w}_i(\mathbf{x}) - \mathbf{w}_i^*(\mathbf{x})||^2. \tag{4.18}$$

Finally, we express the negative logarithm of the posterior term as an energy function. We seek to minimize this energy function to obtain the estimates of $\theta = \{J_0, \mathbf{w}_i, \Sigma\}$. From (Equation 4.14) and (Equation 4.18), we have

$$\begin{aligned} E(\theta) = &\sum_{i=-n}^{n}\sum_{\mathbf{x}} \log\left\{(2\pi)^{d/2}|\Sigma|^{1/2}\right\} + \sum_{i=-n}^{n}\sum_{\mathbf{x}} \frac{1}{2}\mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_i \\ &+ \frac{1}{\lambda_j}\sum_{\mathbf{x}} \nabla||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2 + \frac{1}{\lambda_f}\sum_{i=-n}^{n}\sum_{\mathbf{x}} ||\mathbf{w}_i(\mathbf{x}) - \mathbf{w}_i^*(\mathbf{x})||^2. \end{aligned} \tag{4.19}$$

Material Mask Functions

The optical flow fields $\mathbf{w}_i(\mathbf{x})$ suffer from errors in our framework due to various reasons. The eye movements are extremely fast, and are difficult to track. There is constant occlusion from the eyelashes and the eyelid skin. Moreover, the eyelashes are difficult to resolve due to their size and resolution constraints. As such, we propose a material mask function $\nu(\mathbf{x})$ over each of the texels in the high resolution texture.

In order to formulate our material mask function appropriately, we look at the distribution of color for a particular texel tracked across several frames. This is given by $\{I_i(T_i(\mathbf{x}))|i \in [-n,n]\}$ for a constant $\mathbf{x}$. The distribution usually forms a cluster containing one or two modes. A single mode results from distributions

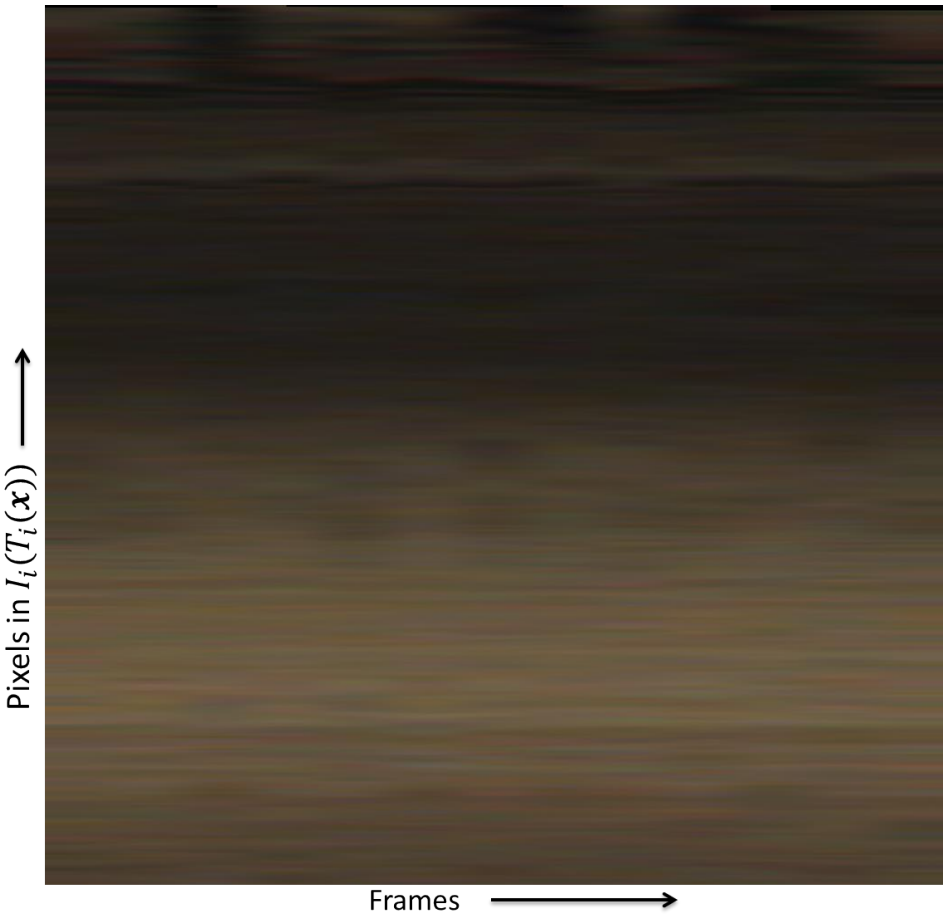

Figure 4.3: Color of texels in texture space tracked across frames: Some texels under occlusion are likely to have two color values corresponding to the eyelash and the skin texture in a horizontal line. Histograms of some horizontal scan lines are shown in Figure 4.4.

where the texel does not undergo any occlusion. A double mode results where a texel undergoes mutual occlusion from eyelash and the eyelid skin. One mode corresponds to color values of the eyelash, and the other mode corresponds to the eyelid skin under the eyelash. Since both are under mutual occlusion, the tracker picks up both while tracking across multiple frames.

The tracked pixels across frames are shown in Figure 4.3. The corresponding

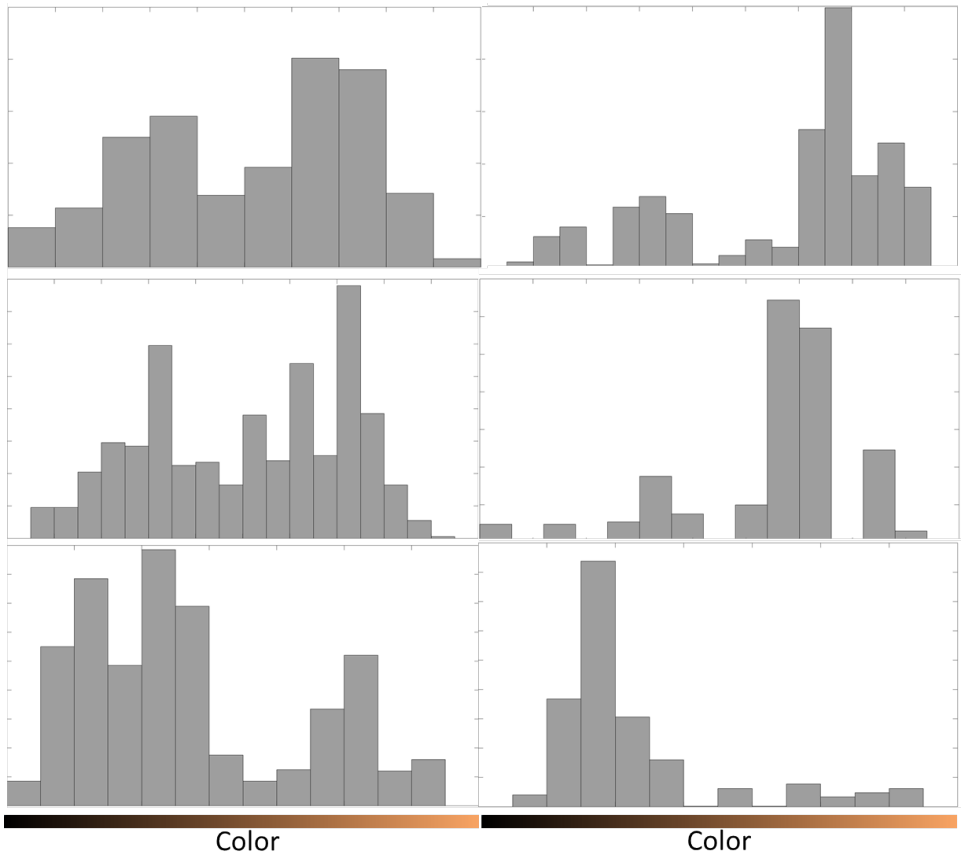

Figure 4.4: Texel Distribution across of a tracked point across frames. The texels under occlusion (left) show two modes corresponding to the eyelash color and the skin color. The texels not under occlusion (right) show a single mode.

texels under occlusion are likely to have two color values corresponding to the eyelash and the skin texture. The distribution for a few tracked pixels corresponding to texels, $\mathbf{x}$ can be seen in Figure 4.4. The modes of the distribution correspond to the intensity values of the eyelash and the skin given by μ_e and μ_s respectively. These modes are different for different texels in the high resolution texture and can be represented as $\mu_e(\mathbf{x})$ and $\mu_s(\mathbf{x})$. Hence, in order to observe the eyelashes in our texture image, we formulate the material mask function for each of the frames $i \in [-n, n]$ as

$$\nu_i(\mathbf{x}) = exp\left(-\frac{1}{\lambda_\nu}\left(I_i(T_i(\mathbf{x})) - \mu_e(\mathbf{x})\right)^2\right), \tag{4.20}$$

where, λ_ν is a parameter that controls the amount of penalty as a result of deviation from μ_e.

Similarly, for observing the skin in the high resolution texture which undergoes occlusion from the eyelash, we construct the material mask function centered around $\mu_s(\mathbf{x})$ given by

$$\nu_i(\mathbf{x}) = exp\left(-\frac{1}{\lambda_\nu}\left(I_i(T_i(\mathbf{x})) - \mu_s(\mathbf{x})\right)^2\right). \tag{4.21}$$

The material mask terms are normalized such that, $\sum_i \nu_i(\mathbf{x}) = 1$, $\forall\ \mathbf{x}$. Finally, we incorporate the material mask term in the energy function (Equation 4.19) that we seek to minimize as

$$\begin{aligned} E(\theta) = &\sum_{i=-n}^{n}\sum_{\mathbf{x}}\log\left\{(2\pi)^{d/2}|\Sigma|^{1/2}\right\} + \sum_{i=-n}^{n}\sum_{\mathbf{x}}\frac{1}{2}\nu_i(\mathbf{x})\mathbf{m}_i^T\Sigma^{-1}\mathbf{m}_i \\ &+ \frac{1}{\lambda_j}\sum_{\mathbf{x}}\nabla||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2 + \frac{1}{\lambda_f}\sum_{i=-n}^{n}\sum_{\mathbf{x}}||\mathbf{w}_i(\mathbf{x}) - \mathbf{w}_i^*(\mathbf{x})||^2. \end{aligned} \tag{4.22}$$

We can explore several methods of optimization for obtaining this solution. In order to do so, we proceed to evaluate the gradient of our function.

4.3 Numerical Solution

We proceed to compute a numerical solution of the energy function (Equation 4.22) to get estimates of $\theta = \{J_0, \mathbf{w}_i, \Sigma\}$. We derive the updates for each of the $\{J_0, \mathbf{w}_i, \Sigma\}$ which will be computed in an iterative manner. We also account for the material mask function $\nu_i(\mathbf{x})$ while deriving the updates for each of the parameters.

4.3.1 Update for Noise Variance, Σ

To estimate the update on the noise variance Σ, in the energy function (Equation 4.22), we set the derivative of $E(\theta)$ w.r.t Σ^{-1} to zero similar to [24]. To

simplify the notation, we represent $m_i(\mathbf{x}), \nu_i(\mathbf{x})$ as m_i, ν_i respectively. We have

$$\frac{\partial E(\theta)}{\partial \Sigma^{-1}} = \frac{1}{2}\frac{\partial}{\partial \Sigma^{-1}} \sum_i \sum_{\mathbf{x}} \nu_i \mathbf{m}_i^T \Sigma^{-1} \mathbf{m}_i + \log((2\pi)^d |\Sigma|), \tag{4.23}$$

$$0 = \frac{\partial}{\partial \Sigma^{-1}} \sum_i \sum_{\mathbf{x}} tr(\Sigma^{-1} \nu_i \mathbf{m}_i^T \mathbf{m}_i) - \log(|\Sigma^{-1}) + \log((2\pi)^d), \tag{4.24}$$

$$0 = \sum_i \sum_{\mathbf{x}} \left[2(\nu_i \mathbf{m}_i \mathbf{m}_i^T - \Sigma) - diag(\nu_i \mathbf{m}_i \mathbf{m}_i^T - \Sigma)\right], \tag{4.25}$$

$$0 = \sum_i \sum_{\mathbf{x}} (\nu_i \mathbf{m}_i \mathbf{m}_i^T - \Sigma), \tag{4.26}$$

$$\Sigma = \frac{\sum_i \sum_{\mathbf{x}} \nu_i \mathbf{m}_i \mathbf{m}_i^T}{\sum_i \sum_{\mathbf{x}} \nu_i}. \tag{4.27}$$

4.3.2 Update for Texture Image, J_0

In order to derive an update for the high resolution texture, J_0, we introduce some vector notations. Let $\mathbf{J_0}$ be the set of all $J_0(\mathbf{x})$ in the texture represented by a column vector. Let $\mathbf{I}_i$ be the set of all $I_i(T_i(\mathbf{x}))$ represented by a column vector. Let $\sum_{\mathbf{x}} g * J_0 = \mathbf{G}\mathbf{J_0}$, where $\mathbf{G}$ is a transform representing convolution. Let $\mathbf{S}$ be a block diagonal matrix of Σ^{-1}. Let $\mathbf{V_i}$ be all the material mask function terms for a particular frame i represented by a diagonal matrix. We note that $\sum_i \mathbf{V_i} = \mathbf{1}$, an identity matrix, since ν_i is normalized. Let Q be a matrix that computes discrete approximation of spatial derivatives, ∇. Then, taking the derivative of the energy function (Equation 4.22), we have

$$\frac{\partial E(\theta)}{\partial J_0} = \frac{\partial}{\partial J_0} \sum_{i=-n}^{n} \sum_{\mathbf{x}} \frac{1}{2} \nu_i \mathbf{m}_i^T \Sigma^{-1} \mathbf{m}_i + \frac{1}{\lambda_j} \sum_{\mathbf{x}} \nabla ||J_0(\mathbf{x}) - J_0^*(\mathbf{x})||^2. \tag{4.28}$$

We rewrite the above equation in a vector form using the notation introduced above. Thus, taking the derivative w.r.t $\mathbf{J_0}$, we have

$$\frac{\partial E(\theta)}{\partial \mathbf{J_0}} = \frac{\partial}{\partial \mathbf{J_0}} \frac{1}{2} \sum_i \mathbf{V_i} (\mathbf{I}_i - \mathbf{G}\mathbf{J_0})^T \mathbf{S} (\mathbf{I}_i - \mathbf{G}\mathbf{J_0}) + \frac{1}{\lambda_j} (\mathbf{J_0} - \mathbf{J}^*)^T Q (\mathbf{J_0} - \mathbf{J}^*) \quad (4.29)$$

$$0 = -\frac{1}{2} \sum_i \mathbf{V_i} \mathbf{G} \mathbf{S} \mathbf{I}_i + \frac{1}{2} \left(\sum_i \mathbf{V_i} \mathbf{G} \mathbf{S} \mathbf{G} \right) \mathbf{J_0} + \frac{2}{\lambda_j} Q (\mathbf{J_0} - \mathbf{J}^*). \quad (4.30)$$

Hence, setting the value of $\sum_i \mathbf{V_i} = \mathbf{1}$ on the R.H.S terms, we estimate the value of $\mathbf{J_0}$ by solving the following linear system:

$$\frac{1}{2} \sum_i \mathbf{V_i} \mathbf{G} \mathbf{S} \mathbf{I}_i + \frac{2}{\lambda_j} Q \mathbf{J}^* = \left(\frac{1}{2} \mathbf{G} \mathbf{S} \mathbf{G} + \frac{2}{\lambda_j} Q \right) \mathbf{J_0}. \quad (4.31)$$

The above equation can be solved for $\mathbf{J_0}$ using direct methods. However, the size of the matrices become as large as $10^6 \times 10^6$ even at smaller texture images of 1 megapixels. Hence, we use the an iterative approach to solve it using the Conjugate Gradient (CG) method.

4.3.3 Update for Optical Flow Fields, $\mathbf{w}_i$

We now derive the updates for optical flow fields. Let $\mathbf{W}_i$ be set of all $\mathbf{w}_i(\mathbf{x})$ represented by a column vector. Then, from Equation 4.19, we have

$$E(\mathbf{W}_i) = \frac{1}{2} \mathbf{V_i} (\mathbf{I}_i - \mathbf{G}\mathbf{J_0})^T \mathbf{S} (\mathbf{I}_i - \mathbf{G}\mathbf{J_0}) + \frac{1}{\lambda_f} (\mathbf{W}_i - \mathbf{W}_i^*)^T (\mathbf{W}_i - \mathbf{W}_i^*). \quad (4.32)$$

The above energy function computes the optical flow field between $\mathbf{I}_i$ and $\mathbf{G}\mathbf{J_0}$ with a regularization over $\mathbf{W}_i$. It can be solved using Euler-Lagrange equations as described in [62].

We iterate over the updates as described above until the convergence is reached. The overall framework is described in algorithm 2.

Algorithm 2: Texture Estimation

Input: $I_i, i \in [-n,n]$, $niters$
Output: $\Sigma, J_0, \mathbf{w}_i, i \in [-n,n]$
Initialize D using 3D mesh obtained from Faceshift[78] ;
Compute $\mathbf{w}_i^*(\mathbf{x})$ using [62];
Compute $J_0^*(x)$ using Eq. (4.9) ;
$\mathbf{w}_i(\mathbf{x}) \leftarrow \mathbf{w}_i^*(\mathbf{x})$;
$J_0(\mathbf{x}) \leftarrow J_0^*(\mathbf{x})$;
for $k \leftarrow 1$ *to* $niters$ **do**
    Compute $v_i(\mathbf{x})$ using §4.2.4 ;
    Compute Σ using §4.3.1 ;
    Update J_0 using §4.3.2 ;
    Update $\mathbf{w}_i$ using §4.3.3 ;
end

4.4 Results

We used the images of a closing blink sequence of the eye recorded from a high frame rate camera at the rate of 200fps. We used image sequences of the eyes at a resolution of 500×500 pixels and obtained high resolution texture maps of 1000×1000 pixels. The first and last frames of the sequence are shown in Figure 4.1. The parameters were set to $\lambda_j = 1000, niters = 10, \lambda_v = 10$. For the experiments, we selected the modes of the material mask function with the higher frequency, according to the following condition,

$$Mode(\mathbf{x}) = \begin{cases} \mu_s(\mathbf{x}), & \text{if } \boldsymbol{f}(\mu_s(\mathbf{x})) > \boldsymbol{f}(\mu_e(\mathbf{x})) \\ \mu_e(\mathbf{x}), & \text{otherwise.} \end{cases} \tag{4.33}$$

where, $\boldsymbol{f}(\mu_s(\mathbf{x}))$ and $\boldsymbol{f}(\mu_e(\mathbf{x}))$ are frequency of pixels with color values $\mu_s(\mathbf{x})$ and $\mu_e(\mathbf{x})$ respectively.

The eyelash from the video sequence is shown in Figure 4.5. The reconstruction of the eyelash texture is shown in Figure 4.6. It took about 60 seconds for each iteration of algorithm 2 to run using MATLAB on a 2.67 GHz Intel Core i5 processor. We ran 10 iterations for about 600 seconds. Our algorithm successfully recovered the detailed eyelash texture from low resolution videos of blinking

eyes. We used 13 frames of a blink sequence for this reconstruction. We also reconstructed the skin texture from the videos. The reconstruction is shown in Figure 4.8, and the corresponding image from the video is shown in Figure 4.7.

We recovered the texture of the full face shown in Figure 4.10. We used 13 frames of the video sequence containing the open eyes captured at a resolution of 750×750 with a frame rate of 200fps. We selected the material mask functions according to Equation 4.33. The recovered full face texture in Figure 4.10 has a resolution of 1500 × 1500.

We also recovered the skin under the eyelashes by using 13 frames of a sequence where the skin is occluded by the eyelashes in all the frames. The first and last frames of the sequence are shown in Figure 4.1. All the frames that are used are of closed eye where the skin under the eyelash is densely occluded. The reconstruction was done by selecting the modes of the material mask function, $\mu_s(\mathbf{x})$ corresponding to skin, irrespective of conditions in Equation 4.33. The reconstruction is shown in Figure 4.11.

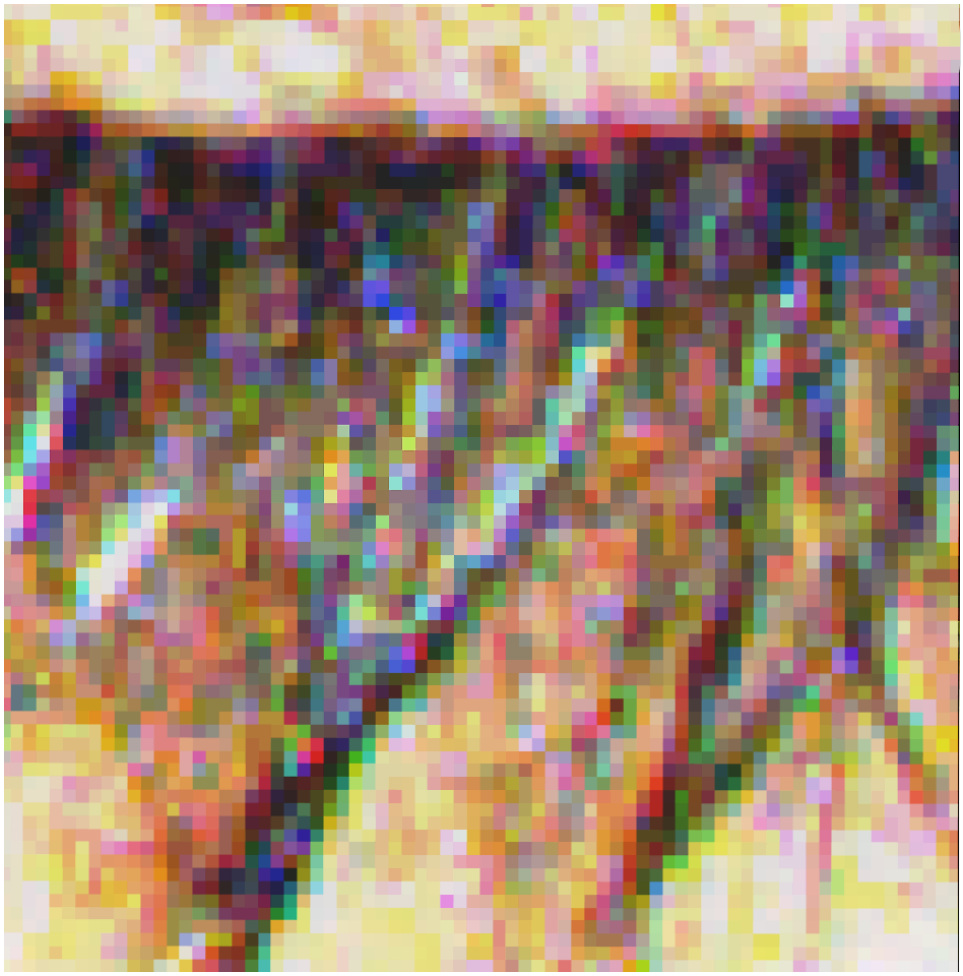

Figure 4.5: Eyelashes from the Video Sequence.

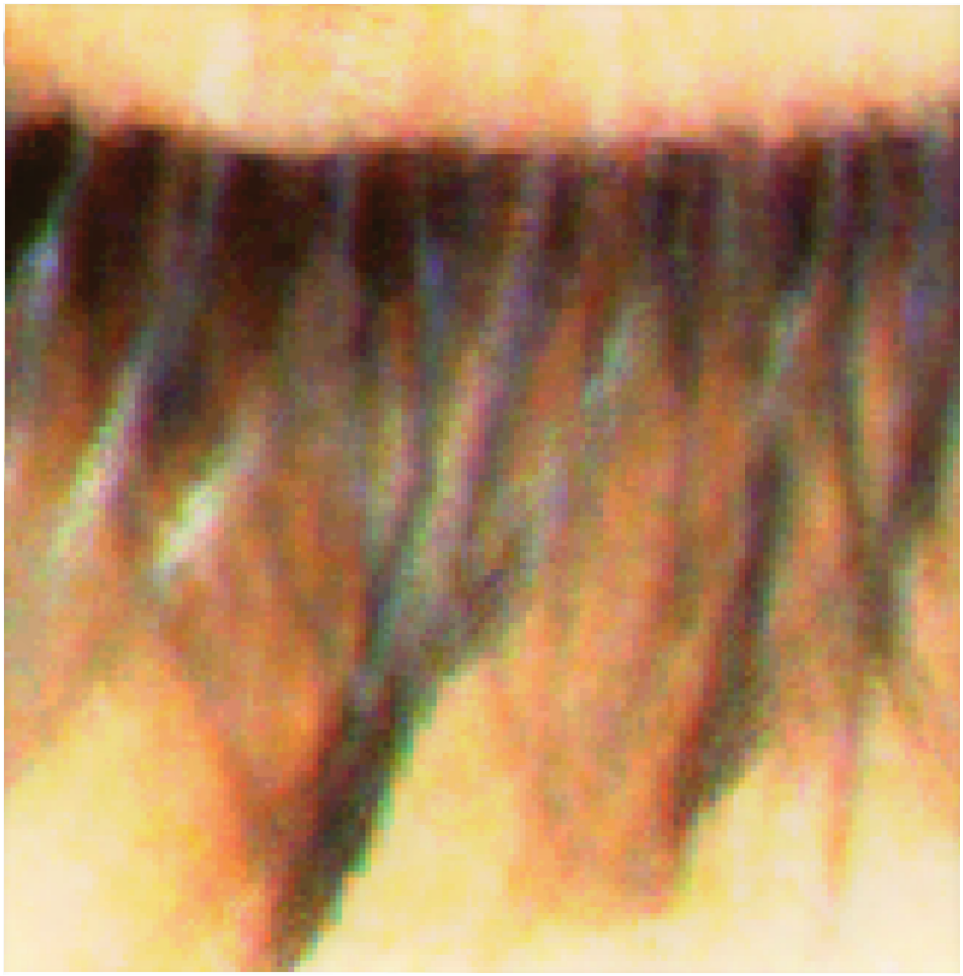

Figure 4.6: Reconstructed Eyelashes with 13 frames of the closing blink sequence by masking using Equation 4.33.

Figure 4.7: Skin texture from the Video Sequence.

Figure 4.8: Reconstructed Skin texture with 13 frames of the closing blink sequence by masking using Equation 4.33.

Figure 4.9: Reconstructed Skin texture of the Eyelid: The blood vessels and wrinkles are more prominent in the reconstruction (bottom) as compared to original image (top).

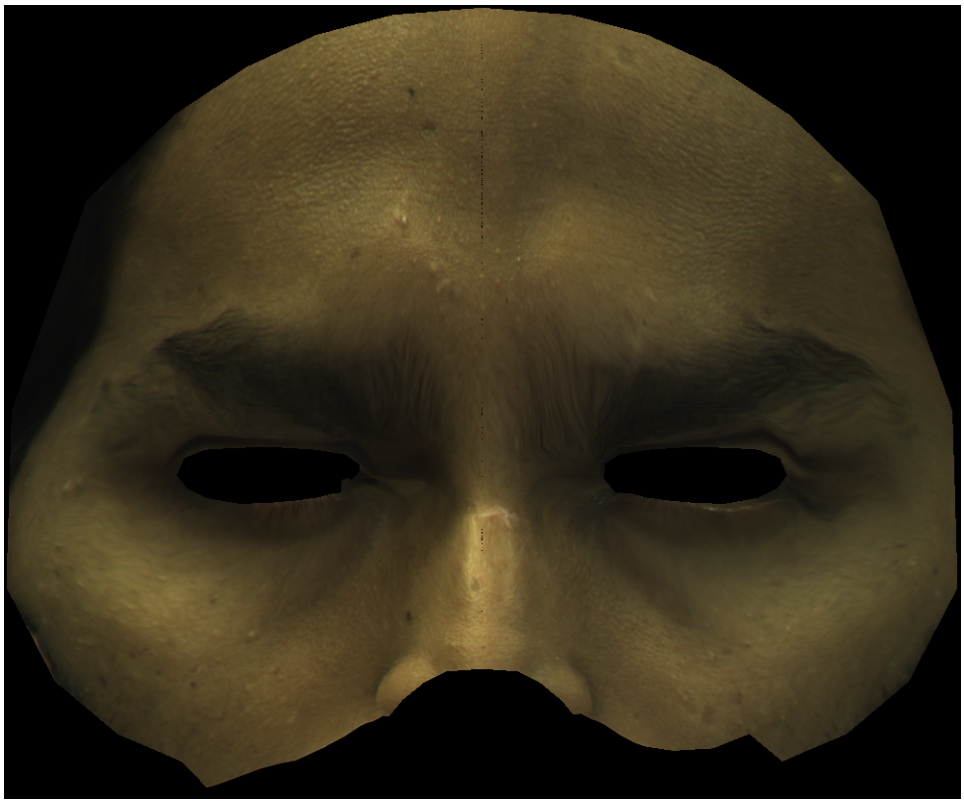

Figure 4.10: Reconstructed Skin texture of the Full Face using 13 frame sequence with eyes open. Masking using Equation 4.33.

Figure 4.11: Reconstructed Skin under the eyelash (left) by using frames from a sequence under occlusion (Figure 4.1) by masking the mode corresponding to $\mu_e(\mathbf{x})$. One frame from the sequence of similar looking frames (right).

Chapter 5

Learning Skin Motion

In this chapter, we will discuss the generation of skin motion of the upper head region from a small number of parameters. In Chapter 3, we developed a framework for effective estimation of the skin motion. We will show that these measurements can be used to create a model for generating the motion of the skin around the eyes. We use artificial neural networks for our learning model.

5.1 Overview

The soft tissues around the eye move primarily due to the activation of *orbicularis oculi* and *levator palpebrae* muscles that control the eyelids. Therefore, the motion of the eyelids and the skin around the eyes can be easily parameterized in terms of activation of *orbicularis* and *levator* muscles. This parameterization requires the activations of different muscles during natural expressions. The muscle activations can be measured by recording their stress or strain characteristics during natural expressions. However, there does not exist a non-invasive way of measuring these muscle activations when the subject performs natural expressions.

Gaze of the eye is defined as the direction in which the eye is looking. We learn the skin motion of the upper head region parameterized by gaze of the eye which can be measured easily using tracking methods [45]. We learn two models to parametrize the skin motion. The first model learns the shape of the eyelid margin (which we will refer to as "eyelid" for short) from gaze parameters since

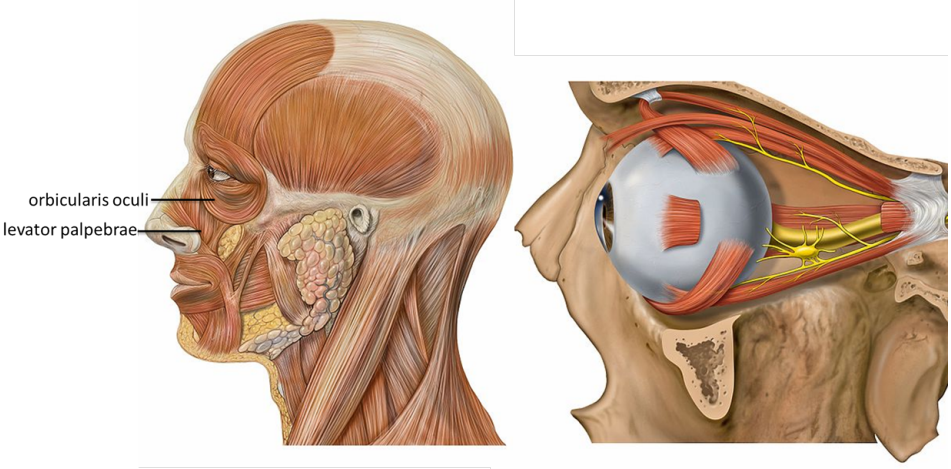

Figure 5.1: *orbicularis oculi* and *levator palpebrae* control the eyelids (left). Extraocular muscles control gaze (right). Reproduced from [42]

eyelid shape is highly dependent on gaze parameters. The second model learns skin motion depending on eyelid shape. This makes sense since the soft tissues around the eye move primarily due to the activation of *orbicularis oculi* and *levator palpebrae* muscles that control the eyelids. We learn two models also to make the implementation more robust. Learning a skin motion model directly from gaze parameters results in overfitting and does not perform well for wide ranges of gaze parameters. Our model performs well for wide range of gaze parameters with a very small reconstruction error.

As described above, we factor the generative model into two models which are learned separately. The schematic of our implementation is shown in Fig. 5.2. The eyelid model predicts the shape of the eyelid from the gaze parameters. Finally, the skin motion is learned as a function of eyelid shape.

5.2 Eyelid Shape Model

We construct the eyelid shape model using neural networks. In the rest of the section, we will formally define the eyelid shape and its control parameters. We will proceed with construction of the model and validate the model after training it.

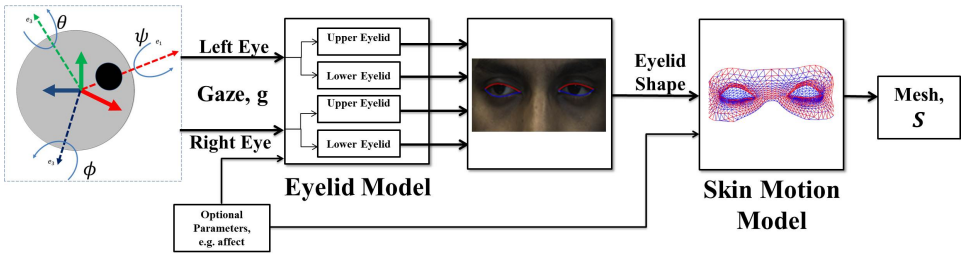

Figure 5.2: Gaze parameterized skin motion: skin motion is estimated from eyelid shape which is recovered using our lid model from gaze parameters. Other affect parameters can also be included in the model.

5.2.1 Definitions

In order to construct a generative eyelid shape model based on the gaze parameters, we start by defining gaze. Gaze is the direction in which the eyes are looking, modeled as the optical axis or as the visual axis. Estimating such a model of gaze is very well studied and widely used, and available in all off-the-shelf eye trackers. Since the gaze represents a ray, it has two degrees of freedom; in the eye tracking literature this is referred to as *2D Gaze*. However, it has been known since the time of Helmholtz that the globe also rotates within the orbit about the visual axis, a phenomenon known as *ocular torsion*. In other words, it is necessary to estimate the full 3D rotation of the globe; this is referred to as *3D Gaze*. Torsion can be high, particularly during head tilt and in extreme gaze, and therefore we also estimate torsion to accurately register the globe between frames. Thus, we can represent the 3D rotation of the eye as $\{\theta, \phi, \psi\}$ corresponding to x, y and z axis of the globe.

However, the shape of the eyelids are also influenced by the aperture of the eye. The aperture of the eye can be defined as the measure of the eye opening. It can be estimated by tracking the eyelid positions. We define the aperture of each eye by a 2D vector given by $\{a_U, a_D\}$. a_U and a_D for an eye are defined as the position of centers of upper and lower eyelid respectively.

Thus, we define the gaze vector for an eye as a 5-dimensional vector which is a combination of 3D globe rotations and 2D apertures and is given by

$$\mathbf{g} = (\theta, \phi, \psi, a_U, a_D). \tag{5.1}$$

We define the eyelid shape for the left eye as a combination of the shape of its upper and lower lids, $\mathbf{e_L} = [\mathbf{e_{LU}}^T, \mathbf{e_{LD}}^T]^T$. Similarly, the eyelid shape for the right eye can be defined as $\mathbf{e_R} = [\mathbf{e_{RU}}^T, \mathbf{e_{RD}}^T]^T$. Each of the eyelid shapes, $\{\mathbf{e_{LU}}, \mathbf{e_{LD}}, \mathbf{e_{RU}}, \mathbf{e_{RD}}\}$ are modeled as spline curves. Each of the spline curves given by $\{\mathbf{e_{LU}}, \mathbf{e_{LD}}, \mathbf{e_{RU}}, \mathbf{e_{RD}}\}$ is a $d \times 2$ matrix representing the vertices in the mesh corresponding to each eyelid. In our model d varies from 17 to 22 depending on the eyelid being modeled.

The five eyelid model parameters are the most basic ones that were able to reproduce the motions captured during measurement. Our system can be easily extended to include other parameters that characterize affect, mouth state, etc., as long as one captures data under those conditions.

5.2.2 Training Model

In order to train the model, we estimate the rotations of the globe (θ, ϕ, ψ) during a video sequence using [46]. We also estimate the eyelid shapes, $\{\mathbf{e_L}, \mathbf{e_R}\}$ using [46]. From the eyelid shape, we estimate the aperture (a_U, a_D) for each eye by tracking the center of each of the eyelids. For each eyelid shape, $\{\mathbf{e_{LU}}, \mathbf{e_{LD}}, \mathbf{e_{RU}}, \mathbf{e_{RD}}\}$ and the corresponding gaze parameters of the globe $\mathbf{g}$, we train a neural network using $n = 670$ data points. Our neural network framework implements 2 hidden layers using Matlab's Neural Network Toolbox. The first hidden layer of neurons are modeled using radial basis functions,

$$y_i = b_i^r \exp(-||\mathbf{w_i^r}.\mathbf{x}||^2), \tag{5.2}$$

where, $\mathbf{w_i^r}$ is the weight vector for each of the inputs represented by a vector $\mathbf{x}$. b_i^r is the bias vector. The second layer of neurons are modeled using pure linear functions given by

$$y_i = \mathbf{w_i^l}.\mathbf{x} + b_i^l, \tag{5.3}$$

with weight vectors $\mathbf{w_i^l}.\mathbf{x}$ and bias b_i^l. The linear layer is used as a weighting function for the outputs from the RBF layer of the neural network. Our learning model is shown in Figure 5.3. The model uses 25 neurons in each of the hidden layers for training. The weights $\mathbf{w_i^r}$ and $\mathbf{w_i^l}$ of the model are learned by minimizing least

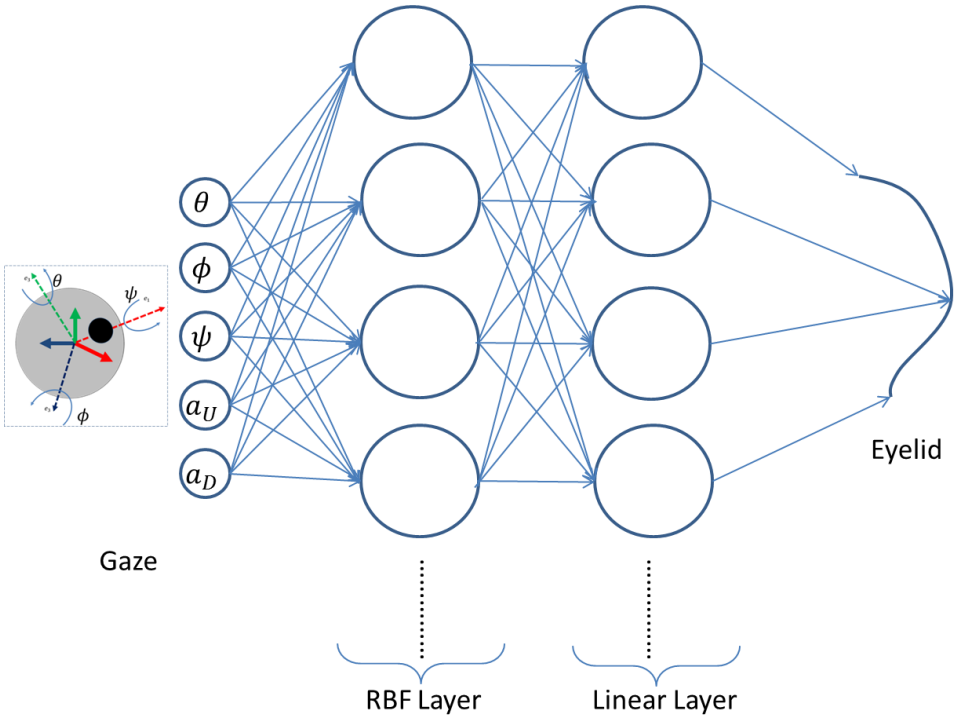

Figure 5.3: Model of the neural network for training Eyelid shape.

square loss between input and output using back-propagation. The loss function is minimized using stochastic gradient descent.

Our framework uses four of the neural network models described using Equation 5.2 and Equation 5.3. Each neural network learns an eyelid shape which is given by $\{\mathbf{e_{LU}}, \mathbf{e_{LD}}, \mathbf{e_{RU}}, \mathbf{e_{RD}}\}$ from the gaze parameter, $\mathbf{g}$ of the corresponding eye. Our neural network is capable of capturing the high non-linearity of the skin motion as shown in the results.

5.3 Skin Motion Model

The skin motion model can be learned as a function of the eyelid shape. In the rest of the section, we will describe the representation of skin motion and the learning model. We use a similar neural network model as discussed in Section 5.2 to parametrize the skin motion.

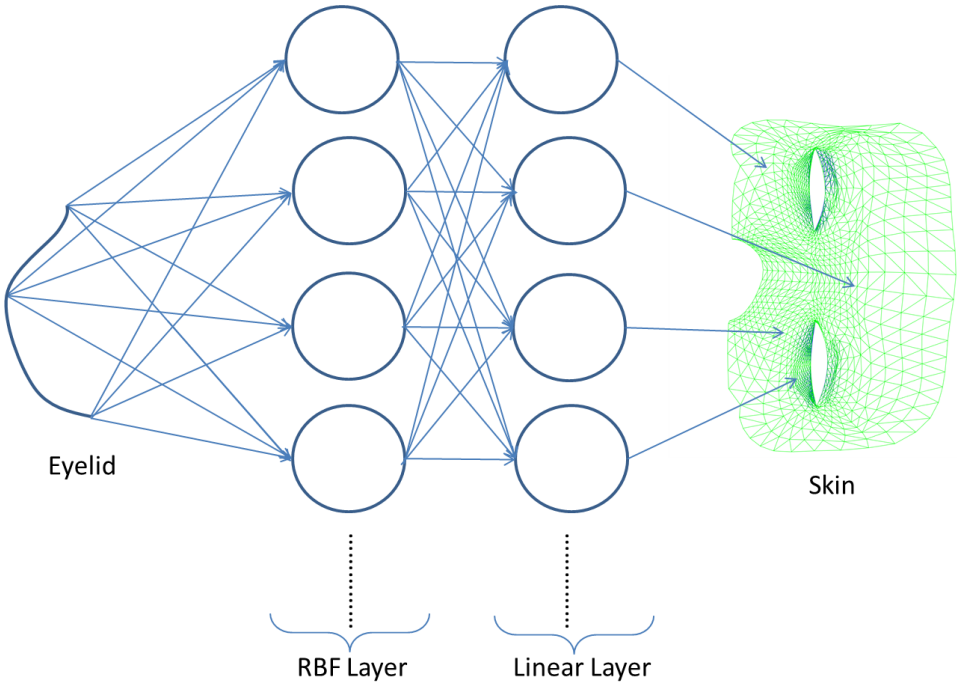

Figure 5.4: Model of the neural network for training Skin Motion.

5.3.1 Definitions

In order to construct a generative model of the skin motion based on the eyelid shape, we combine the eyelid shape vectors for both left and right eyes as, $[\mathbf{e_L}^T, \mathbf{e_R}^T]^T = [\mathbf{e_{LU}}^T, \mathbf{e_{RU}}^T, \mathbf{e_{LD}}^T, \mathbf{e_{RD}}^T]^T$. Each control point that belongs to the eyelid shape, $[\mathbf{e_L}^T, \mathbf{e_R}^T]^T$ forms a row in a $l \times 2$ matrix, where l is the total number of control points corresponding to all the eyelid shape vectors in $[\mathbf{e_L}^T, \mathbf{e_R}^T]^T$. Each control point of the eyelid is 2-dimensional representing its position (x, y).

We define skin mesh $S = (V, E)$ as a collection of vertices and edges. The vertices, V in S, form a $m \times 2$ matrix with each row denoting the (x, y) positions of a particular vertex.

5.3.2 Training Model

In order to train the skin motion model, we estimate skin motion as a flow of the subject mesh S over our video sequence as described in Section 3.2. We obtain the

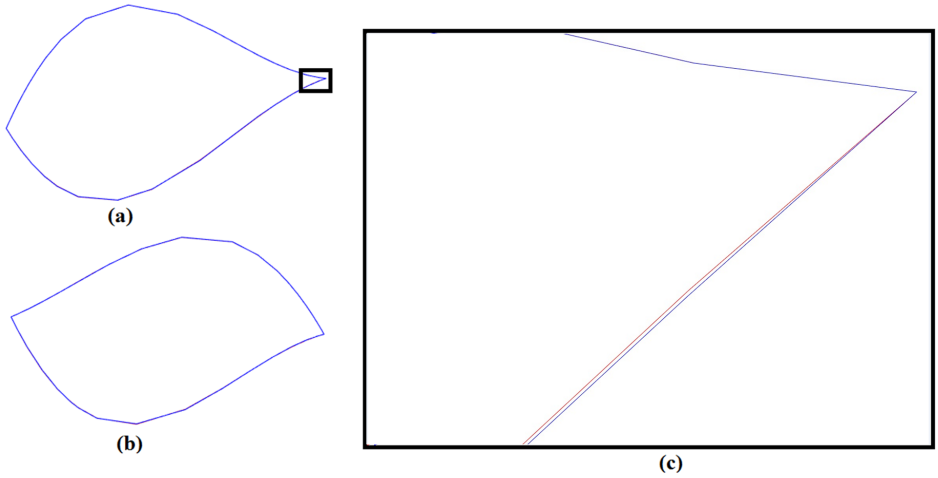

Figure 5.5: Reconstruction of the Eyelid. Ground truth (red) and overlayed reconstruction (blue). Left eyelid (a), right eyelid (b) and the corresponding magnified portion of left eyelid (c).

lid shapes by using tracking methods in [45] represented by splines. We train two neural network models as described in Section 5.2 to independently model x and y components of eyelid shapes. To predict x components of the skin vertices, V_x, we drive our neural network using the x components of the eyelid shape $[\mathbf{e_{L_x}}^T, \mathbf{e_{R_x}}^T]^T$. A similar neural network is learned for predicting the y component of skin motions, V_y. The network is shown in Figure 5.4.

Both neural network models use the same structure as described in Section 5.2, that uses two hidden layers. The first hidden layer contains radial basis functions followed by a second layer of pure linear function. The two models are trained using $n = 670$ data points. The number of vertices in a subject mesh is large, spanning $m = 1662$ dimensions. In order to deal with high dimensionality, we perform dimensional reduction using Principal Component Analysis (PCA) on high-dimensional $\{V_x, V_y\}$ and reduce them to 15 dimensions to achieve better speed ups while training.

5.4 Results

Once the generative model is learned, movements of the eyes are efficiently synthesized from a specification of gaze, and a small number of additional parameters such as the aperture of the eyelid opening. We show that the error rate for the reconstruction of eyelid shape and skin mesh on a validation dataset is very small. Therefore, our model can easily generate realistic motion of eyelids and the skin around the eyes. We also compare our approach with other learning methods and achieve better error rates.

5.4.1 Eyelid Shape Model

During the testing and implementation of the learned eyelid shape model, we provide gaze parameters to our model to obtain the corresponding shape of the eyelids. On a cross validation dataset of 158 points picked uniformly at random from the training dataset, we obtain the error in the ∞-norm, $||e||_\infty = 0.04372$ mm. This represents the reconstruction error of the eyelid control point which suffers maximum error on a real-sized face mesh. The reconstruction of the eyelid is shown in Figure 5.5.

5.4.2 Skin Motion Model

The implementation of the skin motion model consists of providing eyelid shapes obtained in Section 5.2 to estimate skin motions using this learned model. On a validation dataset of 158 points picked uniformly at random from training dataset, we obtain the error in the ∞-norm, $||e||_\infty = 0.265$ mm. This represents the reconstruction error of the vertex V in S which suffers maximum error on a real-size face mesh. The reconstruction of the mesh is shown in Figure 5.6.

5.4.3 Comparison with Other Methods

We tested a variety of more sophisticated dimensionality reduction methods, including Probabilistic PCA (PPCA), Neighborhood Component Analysis (NCA), Maximally Collapsing Metric Learning (MCML) and others. These were evaluated with our Neural Network (NN) model and a Multivariate Linear Regression (MLR) model for their performance as shown in Table 5.1. Due to the nature of our high

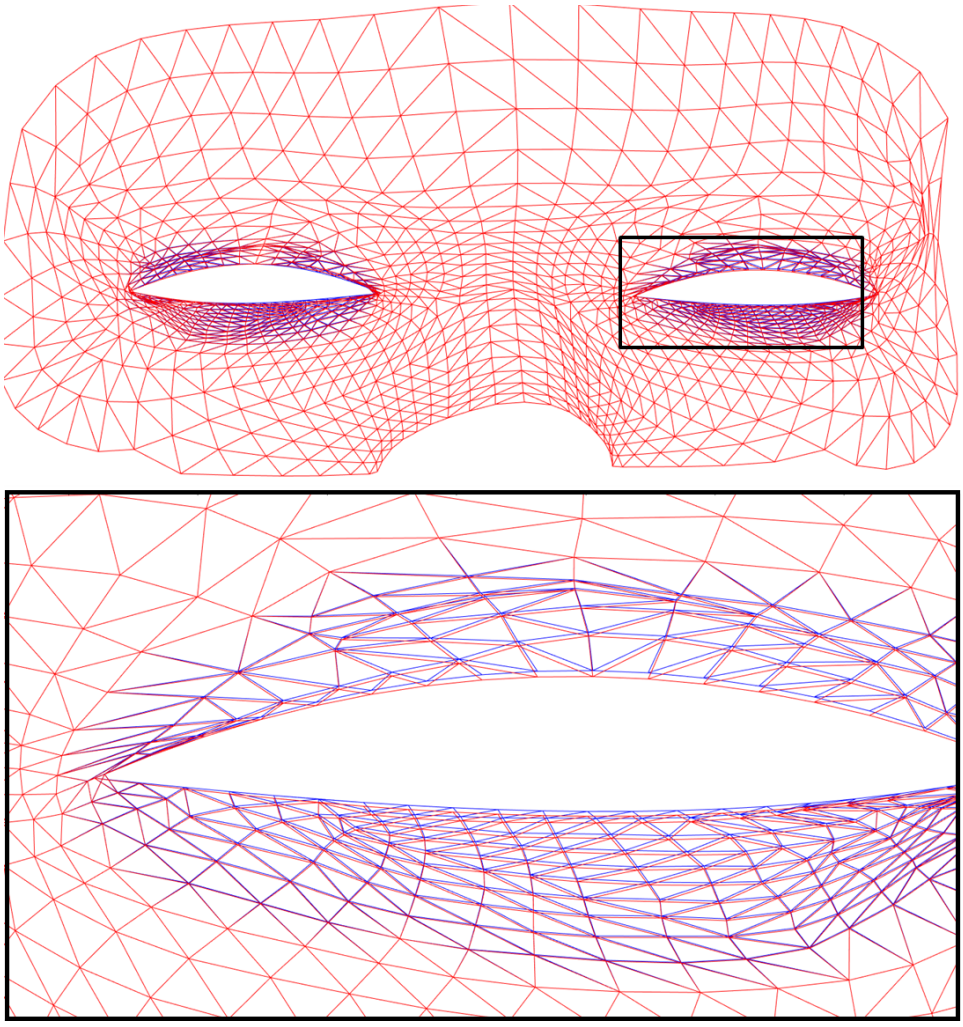

Figure 5.6: Reconstruction of the Skin(top). Ground truth (red) and reconstruction overlayed (blue). Magnified version of the left eye at the (bottom).

Models	$\|\|e\|\|_\infty$ (mm)	Training Time (s)
PPCA + MLR	6.187	240.14
NCA + MLR	4.279	61.62
MCML + MLR	5.288	778.54
PCA + MLR	1.156	19.55
PCA + NN	0.265	8.72

Table 5.1: Performance comparison of different learning algorithms on skin motion model.

Figure 5.7: Model transfer: Model trained on one subject (left) is used to deform the mesh of another subject from one pose to another (center). Deformed mesh overlayed on second subject (right).

dimensional output data (face mesh) and low-dimensional input data, we found that our Neural Network model performed best with PCA on our skin model. We used 670 data points to model a 1662 dimensional output mesh; more sophisticated methods could perform better on much larger input data, but capturing the data then becomes expensive.

5.4.4 Model Transfer

Once trained on a single user, the generative model can be used to produce skin movements in other characters based on different gaze or affect parameters. The output of the skin motion model is the location of the projected skin mesh, S, on the image plane. From this output and given reference mesh, we can compute

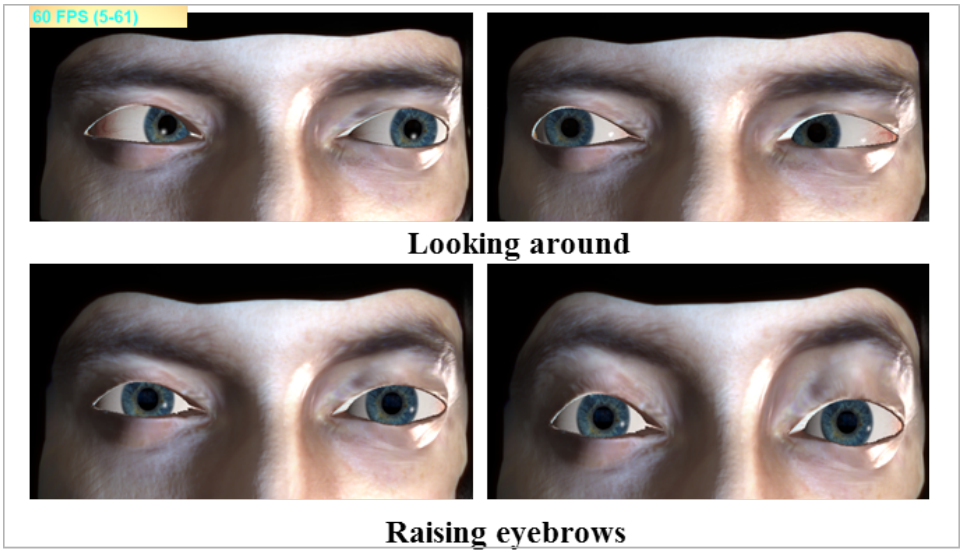

Figure 5.8: Interactive WebGL implementation in real-time.

a mapping Ψ from the reference mesh to the deformed mesh. This mapping remains constant across subjects in our framework, and can be stored. Now, all that is needed is a new mesh registered to the model mesh using software such as Faceshift. Given the new 3D reference mesh, we can use the same mapping Ψ, to generate the 3D deformed mesh configuration for any gaze input parameters. In Figure 5.7, we have shown an example of facial expression transfer.

5.4.5 Interactive Real Time Implementation

For interactive applications we use the PCA+MLR model of Section 5.3. Although the error is worse than the neural network model, it is faster to evaluate and implement on WebGL framework. Since all operations are linear, they can be converted to matrix multiplications and premultiplied, resulting in $p \times q$ matrices, one for each skin atlas coordinate axis, which correlate q gaze inputs with p learned weights. A WebGL application in [46] running using our model is shown in Figure 5.8.

Chapter 6

Wrinkles

In this chapter, we discuss different methods to simulate wrinkles that can be incorporated while rendering to achieve realism. We discuss two methods for creating normal maps that can be blended while rendering to simulate wrinkles. In Section 6.1, we describe an appearance model based on the strain of the skin which is rendered using normal maps. In Section 6.2, we follow the *shape from shading* approach and parametrize wrinkles as geometric models in terms of their depth and height.

6.1 Strain based Appearance Model

In computer graphics, facial features like wrinkles and buckles are used to enhance realism of a rendered human face in facial animations. Some of the facial wrinkles are shown in Figure 6.1. It is therefore crucial to establish an appearance model for these features, which can be used to generate them. We develop an appearance model based on the changing strains in the skin that can be used for generating wrinkles.

6.1.1 Definitions

Following the notations from Section 3.2.1, we define skin atlas in 2D as a parameterization, using the map π, of 3D skin mesh in skin space. The skin mesh and the parameterization are obtained using Faceshift [78]. The skin mesh can undergo

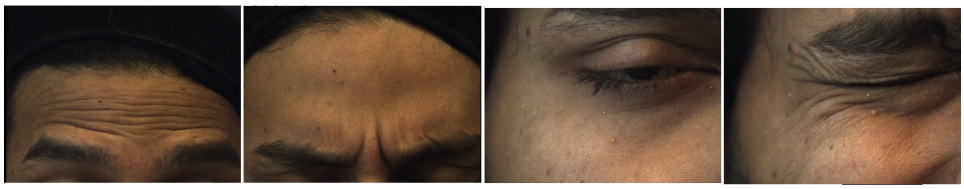

Figure 6.1: Facial expression and skin deformations: (Left to Right) Forehead wrinkles, frown, blink and Crow's feet.

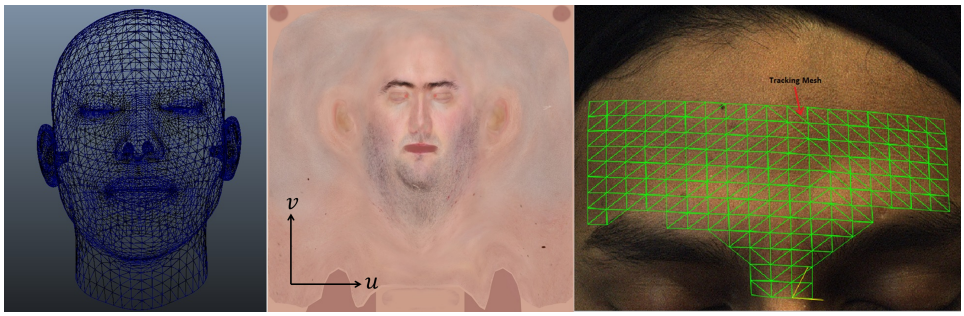

Figure 6.2: Skin Mesh (left), Skin Atlas (middle), and Tracking Mesh (right)

deformations given by ϕ_t as shown in Figure 3.4. The deformation ϕ_0 at t_0 is computed using the Orthogonal Procrustes algorithm [45]. The projections of these deformed meshes are observed by our monocular setup at time t. The projection matrix P is obtained by camera calibration as described in Section 3.2.2.

Given a mesh deformation, a strain measure may be computed in various ways. We use the strain measure from [39] to compute directional strains (ϵ_u, ϵ_v) in the orthogonal directions (u, v) of the skin atlas as shown in Figure 6.2.

In our monocular setup, we track the point $\boldsymbol{u_t}$ across time in the video frames as defined in Section 3.2.1. Each point $\boldsymbol{u_t}$ can be mapped back to the skin atlas while tracking given the estimates of π, P and ϕ_0. We represent the pixel intensity of a mapped point from image $I(\boldsymbol{u_t})$ in skin atlas as $J(\mathfrak{u}_t)$. We define *Appearance Texture*, J, as an image that discretizes the 2D skin atlas.

The skin mesh projected on the image plane using the estimates of π, P and ϕ_0 is cropped to contain the wrinkles. This cropped mesh is referred as the *Tracking Mesh*. The skin mesh, skin atlas and the tracking mesh are shown in Figure 6.2.

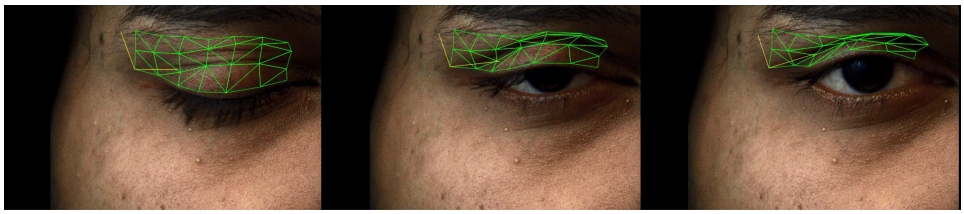

Figure 6.3: Tracking of Eyelid across frames of a video sequence

6.1.2 Tracking

We record a color video of a subject performing different facial expressions using a Basler Pilot series camera[1] with a frame size of 646x486 pixels. The frames are captured at 200Hz with a shutter duration of 5ms. We project skin mesh on image plane and crop it to obtain the Tracking Mesh.

We track the vertices of the tracking mesh across the video frames using the Kanade-Lucas-Tomasi (KLT) algorithm [41, 58, 66] as shown in Figure 6.3. We also detect tracking failures using [37]. The poorly tracked points are corrected using the neighborhood points. We use bi-cubic scattered data interpolation of the neighborhood points to estimate the motion of poorly tracked points.

Facial expressions result in skin deformations causing wrinkles. Skin deformation in a facial expression is a result of strains produced in the skin. We estimate the strain produced in the skin using [39] by tracking the mesh across the video sequence. We then map the points in the tracked image to the appearance texture given by J. The points in the appearance texture correspond to the same points on the skin in the video sequence irrespective of facial deformation. The appearance texture of the forehead corresponding to the eyebrow raising expression is shown in Figure 6.4 for two different time instants.

The motion of the skin and its features viz. wrinkles, texture changes etc., in the original video are difficult to localize. However, the wrinkles occur at the same position across all the frames in the appearance texture as shown in Figure 6.4. The intensity of points in this appearance texture is given by J.

[1] Basler Camera, http://www.baslerweb.com/Cameras-42173.html

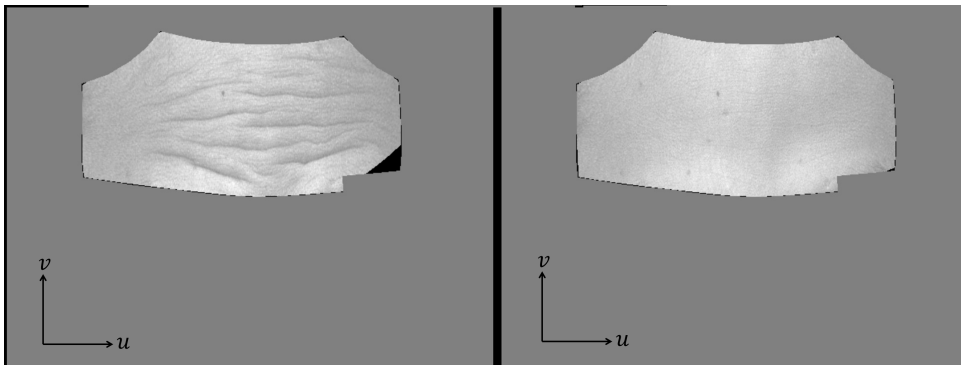

Figure 6.4: Forehead wrinkles projected to Skin Atlas as an appearance texture.

6.1.3 Learning Strain to Appearance Map

We now describe a model to generate the appearance textures based on the strains produced in the skin. Our model learns the weights of an RBF to generate appearance textures $J(\mathbb{u}_t)$ based on skin strains ϵ_u, ϵ_v.

Training

We train our model using the intensity values in the appearance texture, $J(\mathbb{u}_t)$ for given estimates of skin strains, ϵ_u, ϵ_v, obtained in Section 6.1.2. We use radial basis functions for our model as described in [40, 79].

For each face triangle of the Tracking Mesh F_i ($i : 1, 2, ..N$; N is number of face triangles), we obtain strain along orthogonal directions corresponding to the axes of the skin atlas, u and v, as shown in Figure 6.2. We first compute the strain of the i^{th} triangle w.r.t. its rest configuration, and then find its strain components, ϵ_u and ϵ_v, along the orthogonal directions at time t. These strains are computed for each triangle in all frames using [39]. The strain vectors are used to define a $2N$ dimensional feature space for the N faces. The $2N$ dimensional feature space is formed by the strain vectors ϵ_u and ϵ_v for each of the N face triangles.

For a particular facial expression, strains are produced in F_i. We learn a function, $\omega : \mathbb{R}^{2N} \rightarrow \mathbb{R}^M$ that maps these strains $\boldsymbol{\varepsilon}_t = [\epsilon_u^{(1)}, \epsilon_v^{(2)}, ..., \epsilon_u^{(N)}, \epsilon_v^{(N)}]$ to a target appearance texture $J(\mathbb{u}_t)$, containing M skin points. We learn this mapping us-

ing scattered data interpolation that employs Linear RBF kernels, Φ. We model each point in the appearance texture $J(\mathbb{u}_t)$ as a set of linear radial basis functions, $\Phi(r) = r$ [40, 79]. Our model is given by

$$J(\mathbb{u}_t) = \Sigma_{k=1}^{n} w_k^{\mathbb{u}} \Phi(\boldsymbol{\varepsilon}_t - \boldsymbol{\varepsilon}_k), \tag{6.1}$$

where, n is the total number of frames in the sequence of appearance textures. We determine the weights $w_k^{\mathbb{u}}, k = 1:n$ using our appearance textures and corresponding strains for each $\mathbb{u}_t \in [1, M]$. These weights represent our model and can be used to compute the appearance texture depending upon the skin strain.

Blending of Normal Maps

The computed appearance textures based on strains can not be used directly for rendering. Therefore, we use radiometry to estimate the normal maps of the expression wrinkles and buckles. The normal maps are obtained by illuminating the subject using a light source from different directions following [43]. The normal maps corresponding to neutral state and activated state are obtained (Figure 6.5).

The normal maps obtained using radiometry are blended together using weights that correspond to appearance texture obtained from strain appearance learning model in Section 6.1.3. The appearance texture provides the blending weights for normal maps during a facial expression. Let J_T be the appearance texture at time T for an activated state, and J_0 be the appearance texture for a neutral state. Let $\mathbb{N}_T$ and $\mathbb{N}_0$ be the normal maps at activated and neural states respectively. Then, a normal map, $\mathbb{N}_t$, at any time t for a facial expression can be computed from appearance texture, J_t, as

$$\mathbb{N}_t = \frac{J_T - J_t}{J_T - J_0}\mathbb{N}_0 + \frac{J_t - J_0}{J_T - J_0}\mathbb{N}_T. \tag{6.2}$$

6.1.4 Experiments and Results

We used a 2.67 GHz Intel Core i5 processor for video processing. The tracking and generation of appearance textures was done using MATLAB which gave us a processing speed of 3-4 frames per second. The radial basis weights were com-

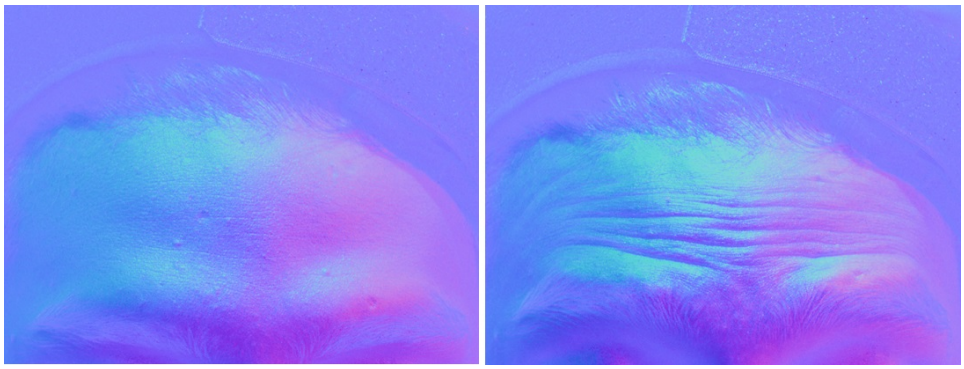

Figure 6.5: Normal map of neutral state(left) and activated state(right) for forehead wrinkles

puted using MATLAB which took about 15 seconds to learn the model using the captured data.

For obtaining the normal maps, we captured the facial expression of the subject in two states: neutral state and activated state. In the neutral state, the subject has neural expression, and no wrinkles or buckles are formed. In the activated state, the subject performs a specific expression resulting in wrinkles or buckles on the skin. Each of the two states are captured using four different scene illuminations. The subject is illuminated from top, bottom, left and right for capturing. These four images captured with different scene illuminations are used to compute the normal maps using xNormal[2] software.

The rendering was done using Maya which used normal maps and strain appearance model. We estimated skin strains based on the face deformations during animation. This strain estimation was subsequently used for computing the appearance texture from the learning model described in Section 6.1.3. The normal maps are then obtained by blending which are controlled by appearance textures. The normal maps in neutral and activated states are blended using Equation 6.2 to generate wrinkles in a facial expression. We show the result in Figure 6.6 for an eyebrow raising expression.

[2]http://www.xnormal.net/

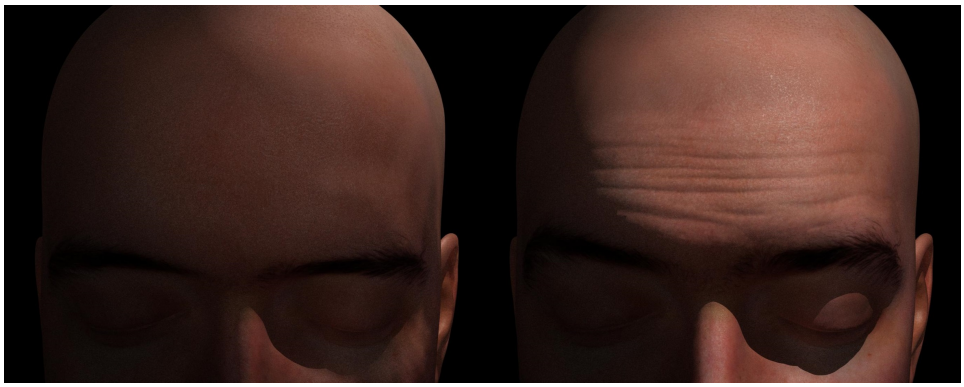

Figure 6.6: Rendered Upper Head animation. Steady state pose(Left) and Activated pose (Right). (Snapshots from the video)

6.2 Shape from Shading Model

In this approach, we describe the use of *shape from shading* to model wrinkles around the eye following Bickel et al. [6]. Bickel et al. use the intensity changes in a video capture during wrinkle formation to estimate the geometric properties of the wrinkles. They use a Lambertian set up by painting specific wrinkles on subject using a non-reflective paint. In our approach, we use polarized filters to remove specularity and achieve Lambertian conditions.

6.2.1 Geometric Model

We parametrize each of the observed wrinkle lines using a spline curve, p. We then model the cross-section of the wrinkles as an exponential function along the splines, similar to [6]:

$$W(p,d,w) = d.\left(\frac{p}{w} - 1\right)\exp(-p/w). \tag{6.3}$$

Each wrinkle line is modeled separately as a spline curve parametrized by p. The depth d and width w of the wrinkles are estimated using the illumination of the point p with respect to ambient illumination.

6.2.2 Implementation

We manually mark the position of wrinkles in the video frames corresponding to extreme poses, such as crow's feet. We need to do this only once per subject. This process can also be made automatic by detecting wrinkle lines using methods such as edge detection [13] or vessel/ridge like structure detection techniques [23]. However, these techniques are not as robust for detecting wrinkles in the eye region. Hence, we manually mark the splines of the wrinkles corresponding to parameter p as shown in Figure 6.7.

After setting the spline parameter, p, for the wrinkle line, we estimate the geometric parameters d and w using shape from shading approach. The parameters (d,w) are obtained by estimating the illumination during wrinkle formation with respect to ambient illumination as shown in [6]. These parameters result in a depth map corresponding to wrinkles for each of the poses.

We then obtain a normal map from the depth map for a particular pose from the estimated values of d and w. We transfer the normal maps to the skin atlas (Section 6.1.1) and store them as normal maps. The normal maps are then used to render wrinkles and folds around the eye.

We separate the eye region into six vertex groups (see Figure 6.7) based on anatomical location and motion of facial muscles around the eye. For example, upper and lower eyelid regions are separated into two separate groups, as opening of upper and lower eyelids are independently controlled by different muscles. Now we define several key poses which are the skin poses corresponding to a few extreme expressions, such as forceful eye closure. We compute the normal maps $\mathcal{N}_i$, $i \in 1,...,n$ corresponding to n gaze vectors $\mathbf{g_i}$, $i \in 1,...,n$ during key poses and store them to train a wrinkle model.

As in Chapter 5, for each skin group, $\mathcal{G}$, we train a linear wrinkle model, $\mathcal{W}_\mathcal{G}$: $\mathbf{g} \rightarrow \Gamma$ using gaze vectors, $\mathbf{g_i}$ and corresponding normal map blending weights, Γ_i. The blending weights of the normal maps are set as the average grayscale intensity in the image of the key poses along the wrinkle lines in that group, followed by a normalization across all the key poses. We can now use the wrinkle model $\mathcal{W}_\mathcal{G}$ to generate blending weights for a novel gaze.

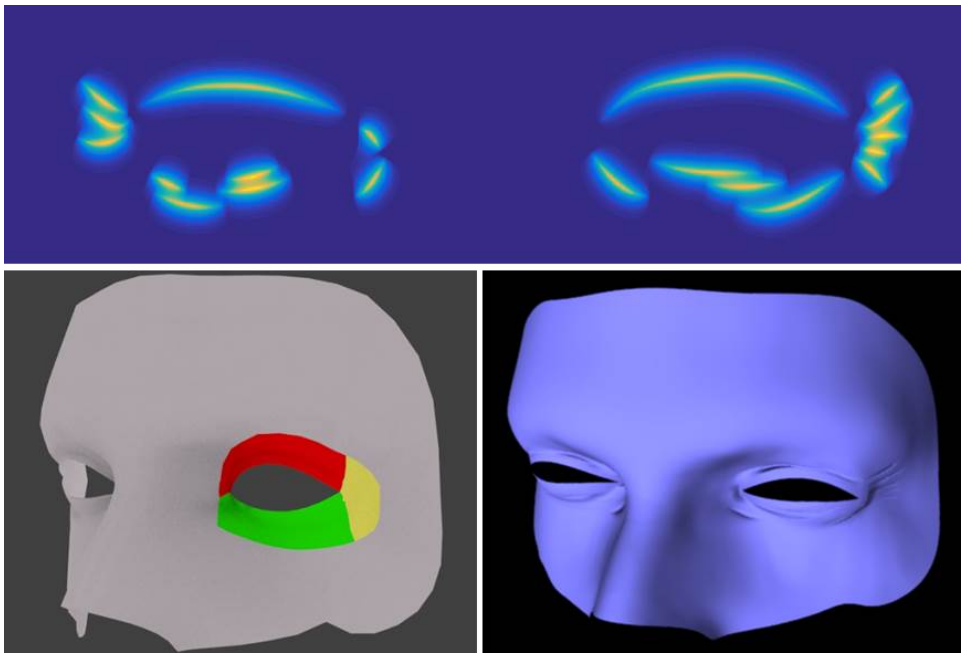

Figure 6.7: Top: Wrinkle markings (with shape shown as intensity), Bottom: Three skin groups in the left eye region (left), and wrinkle generation in the eyes (right).

6.2.3 Results

The strain appearance model in Section 6.1 can not produce realistic effect in modeling complex wrinkles around the eye. This is because computing a normal map for complex expressions around the eye using our basic system is prone to errors. Hence, we use shape from shading approach in this section to parametrize the geometric characteristics of these wrinkles. The reconstructions using blending weights learned from this wrinkle model are shown in Figure 6.8.

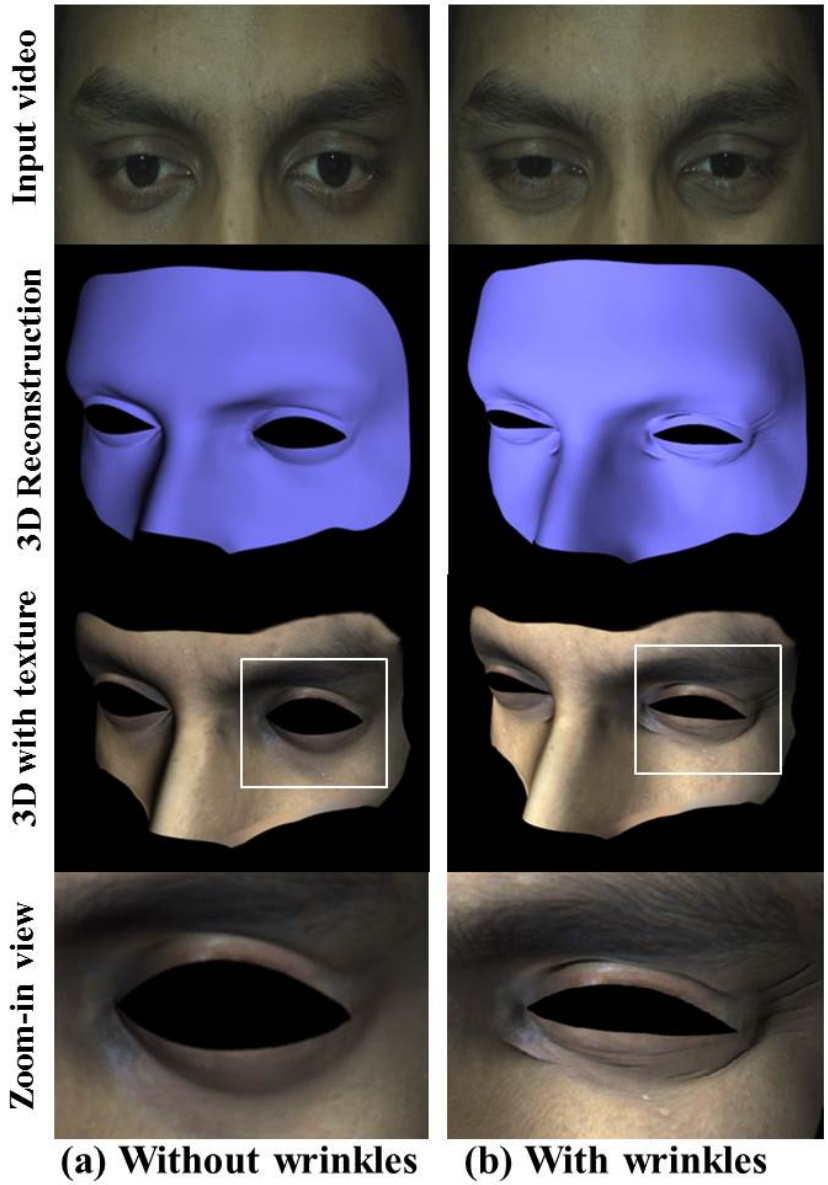

Figure 6.8: Wrinkle modeling: Without wrinkle (left) and with wrinkle (right) in a forceful eye closure example.

Chapter 7

Conclusion and Future Work

Eyes have always been one of the most difficult elements to capture and reconstruct in computer graphics. The complex motion of lid folding, the fast skin motion due to saccades and blink sequences require better algorithms for capturing and modeling them. The traditional motion trackers fail due to complex motions of the periorbital soft tissues. We improve the methods for capturing the motion of the periorbital soft tissues in Chapter 3 and also improve upon the existing scene flow techniques for motion estimation.

The textures of the soft tissues around the eye generally have a low resolution in natural videos, as they are a small part of the whole face. Moreover, the textures of the periorbital tissues are often under occlusion due to eyelashes. The eyelash occlusions are corrected mainly using inpainting. Moreover, the eyelashes are fine structures and therefore it is very difficult extract their shape and texture. The high resolution textures form an important part of computer generated imagery. We successfully recover the textures of the eyelashes using an optical flow based superresolution technique in Chapter 4. We also generate a complete high resolution skin texture for periorbital tissues that can be directly used for animation.

The complex motions of the periorbital tissues have always been difficult to reproduce. Most of the facial animations lack in good control of the periorbital soft tissues. We model the complex motions of the periorbital tissues using simple gaze parameters in Chapter 5. Hence, the realistic motions of the soft tissues can easily be produced and tuned using gaze and affect parameters.

A large part of realism around the eye comes from wrinkles that are formed during various expressions. We develop techniques inspired by shape from shading to produce realistic expression wrinkles around the eye in Chapter 6.

In this work, we improved the methods for capturing the motion and texture of the periorbital soft tissues. We developed a controller that can produce realistic eye movements using a small number of parameters. To add realism, we modeled wrinkles in the soft tissues.

Future Work

Tracking The estimation of scene flow can be improved by the introduction of additional constraints with respect to the mesh structure and elasticity. This could accomplish a detailed capture of the facial expressions with a finer mesh. Furthermore, the scene flow method is very expensive with respect to computation. A good area of exploration would be making the computations faster by sampling sparse points, or by learning priors.

Texture Recovery The texture estimation algorithm is highly dependent on flow estimation. Thus, increasing the speed of flow estimation would boost the performance of our method. The penalty functions introduced in Chapter 4 are based on the color values of eyelashes and skin. They can be made more efficient by learning a prior on the features. Furthermore, the high resolution textures can also be useful in extracting physical model of eyelashes in terms of a small number of parameters.

Learning Skin Motion The ANN models can be trained on larger datasets to capture all the poses that can be reproduced. Other affect parameters can be added to create expressions that involve the entire face. The model can be easily implemented for real-time interfaces, and can be ported to mobile devices and web.

Wrinkles One of the major elements of realism in facial performance capture systems are wrinkles. There have been only a handful of methods for reproducing and modeling wrinkles in a monocular set up. Our model could be extended to reproduce the finer complex wrinkles which are difficult to model.

Bibliography

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. → pages 6, 31

[2] Y. Bando, T. Kuratate, and T. Nishita. A simple method for modeling wrinkles on human skin. *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 166–175, 2002. → pages 11

[3] N. Batool and R. Chellappa. A Markov point process model for wrinkles in human faces. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1809–1812. IEEE, 2012. → pages 12

[4] P. Bérard, D. Bradley, M. Nitti, T. Beeler, and M. Gross. High-Quality capture of eyes. *ACM Transactions on Graphics*, 33(6):223, 2014. → pages 2, 8, 9

[5] A. Bermano, T. Beeler, Y. Kozlov, D. Bradley, B. Bickel, and M. Gross. Detailed spatio-temporal reconstruction of eyelids. *ACM Transactions on Graphics (TOG)*, 34(4):44, 2015. → pages 2, 8

[6] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross. Multi-scale capture of facial geometry and motion. In *ACM Transactions on Graphics (TOG)*, volume 26, page 33. ACM, 2007. → pages 12, 65, 66

[7] B. Bickel, M. Lang, M. Botsch, M. A. Otaduy, and M. Gross. Pose-space animation and transfer of facial details. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 57–66. Eurographics Association, 2008. → pages 12

[8] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 231–236. IEEE, 1993. → pages 4, 5, 15

[9] N. K. Bose, N. Ahuja, et al. Superresolution and noise filtering using moving least squares. *Image Processing, IEEE Transactions on*, 15(8): 2239–2248, 2006. → pages 10

[10] D. Bradley, W. Heidrich, T. Popa, and A. Sheffer. High resolution passive facial performance capture. *ACM Transactions on Graphics (TOG)*, 29(4): 41, 2010. → pages 1, 7, 8, 9, 11

[11] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):500–513, 2011. → pages 4, 6, 19, 20, 21, 22

[12] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004. → pages 4, 5

[13] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986. → pages 66

[14] C. Cao, D. Bradley, K. Zhou, and T. Beeler. Real-time high-fidelity facial performance capture. *ACM Transactions on Graphics (TOG)*, 34(4):46, 2015. → pages 2

[15] E. Cerda and L. Mahadevan. Geometry and physics of wrinkling. *Physical Review Letters*, 90(7):074302, 2003. → pages 11

[16] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics (TOG)*, 21(3):604–611, 2002. → pages 11

[17] G. O. Cula, P. R. Bargo, A. Nkengne, and N. Kollias. Assessing facial wrinkles: Automatic detection and quantification. *Skin Research and Technology*, 19(1):e243–e251, 2013. → pages 11

[18] D. Decarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, 2000. → pages 6

[19] L. Dutreve, A. Meyer, and S. Bouakaz. Easy acquisition and real-time animation of facial wrinkles. *Computer Animation and Virtual Worlds*, 22 (2-3):169–176, 2011. → pages 12

[20] L. Elsgolc. Calculus of variations. 1962. → pages 18

[21] B. Fischer and E. Ramsperger. Human express saccades: Extremely short reaction times of goal directed eye movements. *Experimental Brain Research*, 57(1):191–195, 1984. → pages 2

[22] C. Flynn and B. A. McCormack. Finite element modeling of forearm skin wrinkling. *Skin research and technology*, 14(3):261–269, 2008. → pages 12

[23] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. *Medical Image Computing and Computer-Assisted Interventation MICCAI98*, pages 130–137, 1998. → pages 66

[24] R. Fransens, C. Strecha, and L. Van Gool. Optical flow based super-resolution: A probabilistic approach. *Computer Vision and Image Understanding*, 106(1):106–115, 2007. → pages 10, 27, 30, 36

[25] G. Fyffe, A. Jones, O. Alexander, R. Ichikari, and P. Debevec. Driving High-Resolution Facial Scans with Video Performance Capture. *ACM Transactions on Graphics (TOG)*, 34(1):8, 2014. → pages 2, 9, 11, 26

[26] P. Garrido, L. Valgaerts, C. Wu, and C. Theobalt. Reconstructing detailed dynamic face geometry from monocular video. *ACM Transactions on Graphics*, 32(6):158, 2013. → pages 7, 8

[27] J. Genzer and J. Groenewold. Soft matter with hard skin: From skin wrinkles to templating and material characterization. *Soft Matter*, 2(4): 310–323, 2006. → pages 11

[28] B. Goldluecke and D. Cremers. Superresolution texture maps for multiview reconstruction. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1677–1684. IEEE, 2009. → pages 9

[29] P. Graham, B. Tunwattanapong, J. Busch, X. Yu, A. Jones, P. Debevec, and A. Ghosh. Measurement-Based Synthesis of Facial Microgeometry. In *Computer Graphics Forum*, volume 32, pages 335–344. Wiley Online Library, 2013. → pages 1, 7, 9

[30] A. Hewer, J. Weickert, H. Seibert, T. Scheffer, and S. Diebels. Lagrangian strain tensor computation with higher order variational models. In *Proc. British Machine Vision Conference. BMVA Press, Bristol (September 2013)*, 2013. → pages 7

[31] L. Hong, Y. Wan, and A. Jain. Fingerprint image enhancement: Algorithm and performance evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):777–789, 1998. → pages 11

[32] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. → pages 2, 4, 5

[33] J.-B. Huang, A. Singh, and N. Ahuja. Single Image Super-resolution from Transformed Self-Exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015. → pages 10

[34] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7. IEEE, 2007. → pages 4, 7, 20

[35] C. H. Hung, L. Xu, and J. Jia. Consistent binocular depth and scene flow with chained temporal profiles. *International journal of computer vision*, 102(1-3):271–292, 2013. → pages 7

[36] A. E. Ichim, S. Bouaziz, and M. Pauly. Dynamic 3D avatar creation from hand-held video input. *ACM Transactions on Graphics (TOG)*, 34(4):45, 2015. → pages 8

[37] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759, 2010. → pages 5, 6, 61

[38] S. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. Cohn, and T. Kanade. Multi-view aam fitting and camera calibration. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 511–518. IEEE, 2005. → pages 6

[39] D. Li, S. Sueda, D. R. Neog, and D. K. Pai. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG)*, 32(4):49, 2013. → pages 10, 22, 60, 61, 62

[40] G.-R. Liu. *Meshfree methods: Moving beyond the finite element method*. CRC press, 2010. → pages 12, 62, 63

[41] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pages 674–679. Morgan Kaufmann Publishers Inc., 1981. → pages 4, 5, 61

[42] P. J. Lynch and C. C. Jaffe. Head lateral anatomy. *Creative Commons Attribution 2.5 License 2006*. URL https://commons.wikimedia.org/wiki/File:Lateral_head_anatomy.jpg. → pages 49

[43] W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 183–194. Eurographics Association, 2007. → pages 63

[44] N. Magnenat-Thalmann, P. Kalra, J. L. Lévêque, R. Bazin, D. Batisse, and B. Querleux. A computational skin model: Fold and wrinkle formation. *IEEE transactions on information technology in biomedicine*, 6(4):317–323, 2002. → pages 12

[45] D. R. Neog, A. Ranjan, J. L. Cardoso, and D. K. Pai. EyeMove: Gaze driven animation of eyes. In *Preparation*. → pages 22, 48, 54, 60

[46] D. R. Neog, A. Ranjan, J. L. Cardoso, and D. K. Pai. Gaze driven animation of eyes. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 198–198. ACM, 2015. → pages 2, 8, 9, 51, 58

[47] P. Peers, T. Hawkins, and P. Debevec. A reflective light stage. *ICT Technical Report ICT-TR-04.2006*, 2006. → pages 8, 9

[48] F. Pighin, R. Szeliski, and D. H. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 143–150. IEEE, 1999. → pages 6

[49] D. Pizarro and A. Bartoli. Feature-based deformable surface detection with self-occlusion reasoning. *International Journal of Computer Vision*, 97(1): 54–70, 2012. → pages 5, 6

[50] T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich. Wrinkling captured garments using space-time data-driven deformation. In *Computer Graphics Forum*, volume 28, pages 427–435. Wiley Online Library, 2009. → pages 11

[51] S. Rhee and M. G. Kang. Discrete cosine transform based regularized high-resolution image reconstruction algorithm. *Optical Engineering*, 38(8): 1348–1356, 1999. → pages 10

[52] D. Rohmer, T. Popa, M.-P. Cani, S. Hahmann, and A. Sheffer. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph.*, 29(6):157:1–157:8, Dec. 2010. ISSN 0730-0301. doi:10.1145/1882261.1866183. URL http://doi.acm.org/10.1145/1882261.1866183. → pages 11

[53] H. Rue and L. Held. *Gaussian Markov random fields: Theory and Applications*. CRC Press, 2005. → pages 10

[54] K. Ruhland, S. Andrist, J. Badler, C. Peters, N. Badler, M. Gleicher, B. Mutlu, and R. Mcdonnell. Look me in the eyes: A survey of eye and gaze animation for virtual agents and artificial systems. In *Eurographics State-of-the-Art Report*, pages 69–91, 2014. → pages 9

[55] M. Salzmann and P. Fua. Linear local models for monocular reconstruction of deformable surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):931–944, 2011. → pages 6

[56] H. R. Schiffman. *Sensation and perception: An integrated approach* . John Wiley & Sons, 1990. → pages 2

[57] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013. → pages 3, 7, 15, 18, 21

[58] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings, IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994. → pages 5, 61

[59] E. Sifakis, I. Neverov, and R. Fedkiw. Automatic determination of facial muscle activations from sparse motion capture marker data. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 417–425. ACM, 2005. → pages 10

[60] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439, 2010. → pages 5, 6

[61] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. → pages 6

[62] N. Sundaram, T. Brox, and K. Keutzer. Dense Point Trajectories by GPU-accelerated Large Displacement Optical Flow. Technical Report UCB/EECS-2010-104, EECS Department, University of California, Berkeley, Jul 2010. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-104.html. → pages 6, 31, 38, 39

[63] D. Terzopoulos and K. Waters. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation*, 1(2): 73–80, Dec. 1990. ISSN 1049-8907. → pages 6

[64] J. Tian and K.-K. Ma. Stochastic super-resolution image reconstruction. *Journal of Visual Communication and Image Representation*, 21(3): 232–244, 2010. → pages 10

[65] J. Tian and K.-K. Ma. A survey on super-resolution imaging. *Signal, Image and Video Processing*, 5(3):329–342, 2011. → pages 10

[66] C. Tomasi and T. Kanade. *Detection and tracking of point features*. Pittsburgh: School of Computer Science, Carnegie Mellon Univ., 1991. → pages 5, 61

[67] R. Tsai and T. S. Huang. Multiframe image restoration and registration. *Advances in computer vision and Image Processing*, 1(2):317–339, 1984. → pages 10

[68] V. Tsiminaki, J.-S. Franco, and E. Boyer. High Resolution 3D Shape Texture from Multiple Videos. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1502–1509. IEEE, 2014. → pages 9, 10

[69] H. Ur and D. Gross. Improved resolution from subpixel shifted pictures. *CVGIP: Graphical Models and Image Processing*, 54(2):181–186, 1992. → pages 10

[70] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt. Joint estimation of motion, structure and geometry from stereo sequences. *Computer Vision–ECCV 2010*, pages 568–581, 2010. → pages 3, 4, 7, 13, 14, 16, 17, 19, 20

[71] L. Valgaerts, C. Wu, A. Bruhn, H.-P. Seidel, and C. Theobalt. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph.*, 31(6):187, 2012. → pages 8

[72] A. Varol, M. Salzmann, E. Tola, and P. Fua. Template-free monocular reconstruction of deformable surfaces. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1811–1818. IEEE, 2009. → pages 6

[73] S. Vedula, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27 (3):475–480, 2005. → pages 4, 7

[74] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001. → pages 1

[75] J. Wang, S. Zhu, and Y. Gong. Resolution enhancement based on learning the sparse association of image patches. *Pattern Recognition Letters*, 31(1): 1–10, 2010. → pages 10

[76] Y. Wang and O. Lee. Active mesh: A feature seeking and tracking image sequence representation scheme. *Image Processing, IEEE Transactions on*, 3(5):610–624, 1994. → pages 5, 6

[77] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. *Efficient dense scene flow from sparse or dense stereo data*. Springer, 2008. → pages 4, 7, 20

[78] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime Performance-Based Facial Animation. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2011)*, 30(4), July 2011. → pages 22, 23, 39, 59

[79] H. Wendland. *Scattered data approximation*, volume 17. Cambridge University Press, 2005. → pages 12, 62, 63

[80] J. Xiao, S. Baker, I. Matthews, and T. Anade. Real-time combined 2D + 3D active appearance models. In *Computer Vision and Pattern Recognition, 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–535. IEEE. → pages 6

[81] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, 19(11): 2861–2873, 2010. → pages 10

[82] A. L. Yarbus. Eye movements during the examination of complicated objects. *Biofizika*, 6:52–56, 1960. → pages 1

[83] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000. → pages 23

[84] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H.-P. Seidel. Complementary optic flow. *Energy minimization methods in computer vision and pattern recognition*, pages 207–220, 2009. → pages 5, 6

[85] H. Zimmer, A. Bruhn, and J. Weickert. Optic flow in harmony. *International Journal of Computer Vision*, 93(3):368–388, 2011. → pages 5, 6