STEREO BASED 3D HEAD POSE TRACKING USING THE SCALE INVARIANT FEATURE TRANSFORM

by

Batu Akan

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

Sabancı University
January 2008

STEREO BASED 3D HEAD POSE TRACKING USING
THE SCALE INVARIANT FEATURE TRANSFORM

APPROVED BY:

Prof. Dr. Aytül Erçil _____
(Thesis Supervisor)

Assistant Prof. Dr. Müjdat Çetin _____
(Thesis Co-Advisor)

Assistant Prof. Dr. Selim Balcısoy _____

Associate Prof. Dr. Mustafa Ünel _____

Assistant Prof. Dr. Hakan Erdoğan _____

DATE OF APPROVAL: _____

ACKNOWLEDGEMENTS

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

I am also grateful to Assist. Prof. Hakan Erdoğan and Assist. Prof. Selim Balcısoy for their participation in my thesis committee.

I would like to thank my friends Serhan Coşar, Ali Özgür Argunşah, Ali Baran Çürüklü and other members of the VPALAB for helping me to tackle the problems I had during this thesis.

Special thanks to Tuğba Karagöz for all the encouragement and support she has provided throughout the thesis.

January 2008 Batu Akan

ABSTRACT

STEREO BASED 3D HEAD POSE TRACKING USING
THE SCALE INVARIANT FEATURE TRANSFORM
Batu Akan

In this thesis a new stereo-based 3D head tracking technique, based on scale-invariant feature transform (SIFT) features, that is robust to illumination changes is proposed. Also two major tracking techniques, one based on normal flow constraints (NFC) and a 3D registration-based method, based on iterative closest point (ICP) algorithm, are reviewed and compared against the proposed technique. A 3D head tracker is very important for many vision applications. The resulting tracker output parameters can be used to generate a stabilized view of the face that can be used as input to many existing 2D techniques such as facial expression analysis, lip reading, eye tracking, and face recognition.

Our system can automatically initialize using a simple 2D face detector. We extract salient points from the intensity images using SIFT features and match them between frames. Together with the depth image and the matched features we obtain 3D correspondences. Using the unit quaternion method, we recover the 3D motion parameters. Our proposed method outperforms both NFC and ICP on translations; and performs as good as NFC on rotations. Experimentally, the proposed system is less likely to drift than NFC and ICP over long sequences and is robust to illumination changes. We present experiments to test the accuracy of our SIFT-based 3D tracker on sequences of synthetic and real stereo images.

ÖZET

ÖLÇEKTEN BAĞIMSIZ ÖZNİTELİK DÖNÜŞÜMÜ KULLANARAK
STEREO KAMERA İLE ÜÇ BOYUTLU KAFA TAKİBİ
Batu Akan

Bu tezde üç boyutlu (3B) kafa takibi için ölçekten bağımsız öznitelik dönüşümüne (SIFT) dayalı bir yöntem önerilmektedir. Önerilen yöntemin, düzlem dışı öteleme ve dönmelere karşı gürbüz olduğu belirlenmiş aynı zamanda görüntüdeki ani değişen aydınlanma farklarından da etkilenmediği gözlenmiştir. Bunun yanı sıra optik akış yöntemine dayalı, Normal Flow Constraint ve 3B çakıştırma yöntemi olan tekrarlı en yakın nokta algoritmasını (ICP) değerlendirdik ve önerdiğimiz yöntem ile karşılaştırmasını yaptık. Kafa takibi, bir çok bilgisayarla görme uygulaması için önemli bir süreçtir. Eğer kafanın üç boyutlu uzaydaki yeri ve duruşu bilinirse, yüz tanıması, ifade analizi, dudak okuması gibi problemleri, 3B kafa izleyicisi tarafından oluşturulan dengelenmiş imgeleri kullanarak çözmek daha muhtemeldir.

Önerdiğimiz sistem 2B bir yüz sezicisi kullanarak özişler bir biçimde başlamaktadır. Birbirini takip eden video imgelerinde SIFT öznitelik noktaları bulunur ve birbirlri ile eşleştirilir. Eşleştirilen noktalar, derinlik bilgisi de kullanılarak 3B ilişki kümesi oluşturulur. Birim kuaterniyon yöntemi ile 3B katı devinim hesaplanır. Önerdiğimiz SIFT yöntemi ötelemelerde NFC ve ICP yöntemlerin daha iyi sonuç verdi ve dönmelerde ise NFC benzer bir başarım gösterdi. Aynı zamanda önerilen yöntem uzun videolarda sürüklenmeden daha az etkilenmekte ve zamana bağlı aydınlanma değişikliklerine göre gürbüzdür. Önerilen SIFT tabanlı yöntemin başarısı sentetik ve stereo kamera ile çekilmiş gerçek görüntüler üzerinde denenip var olan diğer yöntemlerle karşılaştıması yapıldı.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

This thesis presents a novel method for 3D head tracking with 6 degrees of freedom. The proposed method is robust to in and out of plane rotations and translations, and illumination changes.

1.1 Motivations

Head Tracking is a very important task for several applications of computer vision. If head location and 3D pose are known, tasks such as face recognition, facial expression analysis, lip reading, etc., are more likely to be solved using a stabilized image generated through the 3D head tracker. One of the applications of 3D head tracking is the development of intelligent vehicles (Figure 1.1), in which one goal is the design of a smart system to actively control the driver before he/she becomes too drowsy, tired or distracted. The pose of the head can reveal numerous clues about alertness, drowsiness or whether the driver is comfortable or not. Also head pose is a powerful pointing cue; determining the head pose is fundamental in vision driven user interfaces. In public kiosks or airplane cockpits, head pose estimation can be used for direct pointing when hands and/or feet are otherwise engaged or as complementary information when the desired action has many input parameters [2]. Hands-free cursor control can be an important control for users with disabilities and is very promising for gaming environments. Furthermore head tracking can be used

for development of very low bit-rate model-based video coding for video-telephone applications.

A successful head tracking application should be robust to significant head motion, change in orientation and scale. The tracker must also be able to handle variations in illumination. Furthermore, the system should work at near video frame rate and be fast enough to maintain interaction with the user. Such requirements make the problem of 3D head tracking even more challenging.

Figure 1.1: Application of 3D head tracking in a vehicle environment.

1.2 Related Work

Several techniques have been proposed for free head motion and head tracking. The proposed techniques can be grouped into 2D and 3D approaches. 2D approaches to face tracking include color based techniques [3] where the background and the objects to be tracked are grouped into several color based groups. Template-based [4] and eigenface-based [5] techniques have also been proposed. In template based techniques the face is tracked by matching the face template within a small search region in each frame with the assumption that small 3D head rotation appears as 2D translation of the facial image [6]. Eigenface-based methods use a statistically-based 3D head model to further constrain the estimated 3D structure. Also tracking

of salient points, features or 2D image patches for recovering 3D head parameters have been proposed. The outputs of the 2D trackers can be processed and joined together to recover 3D location, orientation and facial pose [7]. However using 3D model-based techniques offer better accuracy over 2D methods, but they require knowledge of the shape of the face. The early methods used simple shape models such as planar [8], ellipsoidal [9] or cylindrical [7] surfaces to represent the head. Also a 3D texture mesh [10] or a 3D feature mesh [11, 12] can be used to fill in the missing 3D data.

Methods like active appearance models [13, 14] or active shape models provide very accurate shape models for 3D head tracking. Heavy computational requirements of 3D active appearance models with monocular images makes these methods less favorable for real time applications. Also these methods often require a separate model for every individual that uses the system or the trained model to be general enough for every user, therefore an unfavorable training session is involved in order to generate the appearance models.

Many of the proposed 2D tracking methods perform tracking on the image plane with very little to none out of plane translations and rotations. Those methods which can perform tracking with 6 degrees of freedom often require lots of computation time, due to missing depth information; therefore these methods are not suitable for real time usage. Also intensity based 2D tracking methods suffer severely from time varying illumination changes. Until recently there has not been much research on head tracking using 3D data. With the development of laser scanners and stereo cameras, real time depth information became available. Depth information, surface parameters or point clouds can be extracted easily and surface or shape models can be generated in real time. Therefore the tracking system can easily adapt to different users without prior training. Also the depth image acquired through stereo cameras or laser scanners are not affected from illumination changes thus enabling the algorithms that use depth image to be more robust to illumination changes. The proposed methods of estimating 3D rigid body motion using disparity information can be grouped into two families: optical flow based, and registration based method. In the thesis we will review normal flow constraint (NFC) and iterative closest point (ICP) methods and propose a scale invariant feature transform (SIFT) based

tracking method for 3D head tracking.

Normal flow constraint (NFC) [15] is an extension of the original optical flow algorithm [16]. Optical flow is the two-dimensional vector field which is the projection of the three-dimensional motion onto an image plane [17]. It is often required to use complex 3D models or non-linear estimation techniques to recover the 3D motion when depth information is not available. However when such observations are available from devices such as laser range finders or stereo cameras, 3D rigid body motion can be estimated using linear estimation techniques. Furthermore, combining brightness and depth constraints tend to provide more accuracy for subpixel movements and provides robustness against illumination changes[15, 2].

The iterative closest point (ICP) algorithm introduced by Chen and Medioni [18] and Besl and McKay [19] has been used to merge and stitch laser range scans. The ICP algorithm tries to find corresponding point sets between two given surfaces or point clouds and estimates a best transformation that minimizes the distance between the matched points using Horn's unit quaternion method [20]. Chen and Medioni try to minimize a point-to-plane distance, whereas Besl and McKay try to minimize the point-to-point Euclidean distance. Over the years many derivations of the original ICP algorithm have been proposed, affecting all phases of the algorithm from the selection and matching of points to the minimization strategy. Rusinkiewicz and Levoy [21] present an extensive survey on many derivations of the ICP algorithm. Often the most tweaked part of the algorithm is the description of the correspondence function. Features such as color/intensity [22], normal vector directions are often added to make a better correspondence assignment. ICP has been used for the purpose of head tracking by Simon [23] and Morency [2]. Morency reported that ICP performed well on coarse movements especially in translations but was not as successful as optical flow based methods on fine movements and rotations.

Scale invariant feature transform (SIFT) introduced by Lowe [24] is one of several computer vision algorithms for extracting distinctive features from images. SIFT features are often used in object recognition, because of the descriptors invariance to scale and orientation. Such abilities of SIFT can be easily adapted for object tracking in the image plane as well. At run-time, SIFT features are extracted from

the current frame, matched against the SIFT descriptors of the previous frame, resulting in a set of correspondences. The rigid body motion then can be recovered using Random Sample Consensus (RANSAC) algorithm. [25, 26, 27, 6].

1.3 Contributions

A new head tracking algorithm based on SIFT features used together with a stereo camera input has been proposed, developed, and tested. The algorithm provides one to one point correspondences, which were missing in the original ICP algorithm, and brings the current frame into registration with the previous frame in order to obtain the relative pose change between the frames. The algorithm yields better motion estimates both in translation and rotation then ICP.

1.4 Organization of the thesis

In chapter 2 a review of the Normal Flow Constraint (NFC) algorithm is presented. Chapter 3 presents a review of the iterative closest point algorithm (ICP). Chapter 4 provides a brief review of the scale invariant feature transform algorithm and shows how it can be used for 6DOF rigid body tracking, with the help of optical stereo data. Chapter 5 presents experimental results of the algorithm over both synthetic and real video sequences.

CHAPTER 2

Normal flow constraint algorithm (NFC)

In this chapter, the structure of the gradient-based differential tracking algorithm called NFC [15] is reviewed. NFC is a 3D extension of the original optical flow algorithm proposed by Horn [16]. Optical flow is the apparent motion of the image projection of a physical point in an image sequence, where the velocity at each pixel location is computed under the assumption that projection's intensity remains constant [6]. Similar constraints can be derived when the disparity image is also available. Combining the outputs of brightness constancy constraint equation (BCCE) with depth change constraint equation (DCCE), the NFC algorithm tries to recover the pose change between two consecutive frames.

2.1 Brightness Constancy Constraint Equation

A 3D point in space is represented by its coordinate vector $\vec{X} = [X\ Y\ Z]^T$ and the 3D velocity of this point is represented as $\vec{V} = [V_x\ V_y\ V_z]^T$. When this point is projected onto the camera image plane using some projection model, the point will be mapped to 2D image coordinates $\vec{x} = [x\ y]^T$ and the motion of the 3D point in space will induce a corresponding 2D velocity vector onto the camera image plane $\vec{v} = [v_x\ v_y]^T$.

Assume that $I(x, y, t)$ represents the brightness of a point (x, y) in the image plane

at time t. It can be assumed that the brightness of a particular point in the pattern remains constant even though the point has moved; therefore an equation can be derived that relates the change in image brightness at a point to the motion of the brightness pattern. This assumption is only partially true in practice. In situations such as occlusions, disocclusion, changes in intensity due to changes in lighting, the displacement of pixel patches does not represent physical movement of points in space. For example a rotating uniform sphere with uniform color would seem fixed to an observer. The assumption may be expressed for frames at t and $t+1$ as follows:

$$I(x, y, t) = I(x + v_x(x, y, t), y + v_y(x, y, t), t + 1) \tag{2.1}$$

where $I(x, y, t)$ represents the image intensity, and $v_x(x, y, t)$ and $v_x(x, y, t)$ are the x and y components of the 2D velocity vector at time t. Taylor expansion of the righthand side of Equation (2.1) is

$$\begin{aligned} I(x, y, t) = I(x, y, t) + I_x(x, y, t)v_x(x, y, t) \\ +I_y(x, y, t)v_y(x, y, t) + I_t(x, y, t) \end{aligned} \tag{2.2}$$

where $I_x(x, y, t)$,$I_y(x, y, t)$ and $I_t(x, y, t)$ are the image gradients with respect to x, y and t as a function of space and time. Canceling out the $I(x, y, t)$ terms and rearranging Equation (2.2) into a matrix form yields to the commonly used optical flow equation:

$$-I_t = [I_x \; I_y] \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{2.3}$$

The above equation constraints the velocities in the 2D image plane. However we are interested in the 3D-world velocities. Therefore, for a perspective projection camera with focal length f, the relationship between the real world velocities and the image plane velocities can be derived from the perspective camera equation: $x = \frac{fX}{Z}$ and $y = \frac{fY}{Z}$. Taking derivatives with respect to time yields

$$\begin{aligned} v_x = \frac{dx}{dt} = \frac{f}{Z}V_x - \frac{x}{Z}V_z \\ v_y = \frac{dy}{dt} = \frac{f}{Z}V_y - \frac{y}{Z}V_z \end{aligned} \tag{2.4}$$

when written in matrix form

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{2.5}$$

By substituting the lefthand side of equation (2.5) for $\vec{v}$ into equation (2.3), we obtain the brightness constraint equation for 3D object velocities:

$$\begin{aligned} -I_t &= \frac{1}{Z}\left[I_x \; I_y\right]\begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix}\vec{V} \\ &= \frac{1}{Z}\left[fI_x \; fI_y \; -(xI_x + yI_y)\right]\vec{V} \end{aligned} \tag{2.6}$$

Any rigid body motion can be expressed as an object undergoing instantaneous translation $\vec{T} = [t_x \; t_y \; t_z]^T$ and instantaneous rotation $\Omega = [w_x \; w_y \; w_z]^T$ where Ω represents the orientation of the axis of rotation and $|\Omega|$ is the magnitude of rotation per unit time. For small rotations $\vec{V}$ can be approximated as:

$$\vec{V} \approx \vec{T} + \Omega \times \vec{X} = \vec{T} - \vec{X} \times \Omega \tag{2.7}$$

The cross product of two vectors can be written as the product of a skew-symmetric matrix and a vector, therefore $\vec{X} \times \Omega$ can be rearranged as

$$\vec{X} \times \Omega = \hat{X}\Omega, \text{ where } \hat{X} = \begin{bmatrix} 0 & -Z & Y \\ Z & 0 & -X \\ -Y & X & 0 \end{bmatrix}$$

Equation (2.7) can be rearranged into a matrix form

$$\vec{V} = \mathbf{Q}\vec{\phi} \tag{2.8}$$

where $\vec{\phi} = [\vec{T}^T \; \vec{\Omega}^T]^T$ is the instantaneous motion vector and matrix the $\mathbf{Q}$ is

$$\mathbf{Q} = [I_{3\times 3} \vdots \; -\hat{X}] = \begin{bmatrix} 1 & 0 & 0 & 0 & -Z & Y \\ 0 & 1 & 0 & Z & 0 & -X \\ 0 & 0 & 1 & -Y & X & 0 \end{bmatrix}$$

When the right hand side of equation (2.8) is substituted into equation (2.6), a linear equation which relates pixel intensity values to rigid body motion parameters is obtained for a single pixel.

$$-I_t = \frac{1}{Z}\left[fI_x \; fI_y \; -(xI_x + yI_y)\right]\mathbf{Q}\vec{\phi} \tag{2.9}$$

This is the generic brightness constraint used in many of the previous approaches [15] regarding 3D motion and pose tracking. When 3D world coordinates are not

known, one needs non-linear estimation techniques to solve for the motion or the estimation can be simplified to a linear system using 3D models when a shape prior of the object being tracked is known. Simple models such as planar[8], ellipsoidal[9], cylindrical[7] or a 3D feature mesh[11, 12] can be used to fill in the missing 3D data. If 3D-world coordinates are available, it is relatively easy to solve this equation system and non-linearities can be avoided. Not using 3D shape models reduces any errors introduced in the latter class of approaches.

2.2 Depth Change Constraint Equation

Assuming that video rate depth information is available for every pixel in the intensity image, similar expressions can be derived using the disparity image. Therefore any changes in the depth image over time can be related to rigid body motion. A point on the rigid body surface, located at (x, y) at time t will be at location $(x + v_x, y + v_y)$ at time $t + 1$. The depth values of any particular point in the image space and time should remain the same unless the particular point goes under a depth translation between frames t and $t + 1$. In mathematical terms, this can be expressed in a way similar to equation (2.1)

$$Z(x, y, t) + V_z(x, y, t) = Z(x + v_x(x, y, t), y + v_y(x, y, t), t + 1) \tag{2.10}$$

where $Z(x, y, t)$ represents the disparity image and $v_x(x, y, t)$ and $v_x(x, y, t)$ are the x and y components of the 2D velocity vector at time t. Following the same steps that are used to derive the brightness constancy constraint equation, an analogous depth constancy constraint equation can be derived. Rewriting the first order Taylor series expansion of the righthand side of Equation (2.10) in matrix form we obtain

$$-Z_t = [Z_x \; Z_y] \begin{bmatrix} v_x \\ v_y \end{bmatrix} - V_z \tag{2.11}$$

By substituting the 3D world velocities into Equation (2.11) using the perspective projection model yields:

$$-Z_t = [Z_x \; Z_y] \frac{1}{Z} \begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix} \vec{V} - V_z \tag{2.12}$$

Since any rigid body motion can be expressed as an object undergoing instantaneous translation $\vec{T}$ and an instantaneous rotation Ω, substituting the 3D velocity vector $\vec{V}$ with $Q\vec{\phi}$ as shown previously, produces

$$-Z_t = \frac{1}{Z}[fZ_x \; fZ_y \; -(Z + xZ_x + yZ_y)]\mathbf{Q}\vec{\phi} \tag{2.13}$$

The derived formulation above is the analogous form of the equation (2.9) that relates the change in image brightness at a point to the rigid body motion. Brightness constancy constraint equation depends on the assumption that the brightness of a particular point in the pattern remains constant. Since this assumption is only true under some conditions, the outcome is an approximation at best, whereas equation (2.13) makes use of change in the disparity image which reflects the true dynamics of the motion and the disparity image is not affected by changes in illumination.

2.3 Orthographic Projection

In most cases of interest, camera projection can be modeled as orthographic projection without introducing much error into the system. Such an approach would simplify the constraint equations, reducing the computational load.

Deriving the analogous versions of Equations (2.9) and (2.13) are straightforward. All occurrences of image plane x and y are replaced with their real world counterparts X and Y. Therefore any real world velocities will be equivalent to their image plane counterparts; $v_x = V_x$ and $v_y = V_y$.

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} \tag{2.14}$$

Inserting the simplified orthographic projection matrix into Equations (2.6) and (2.12) yields to the orthogonal projection analogs of the Equations (2.9) and (2.13):

$$-I_t = [I_x \; I_y \; 0]\mathbf{Q}\vec{\phi} \tag{2.15}$$

$$-Z_t = [Z_x \; Z_y \; -1]\mathbf{Q}\vec{\phi} \tag{2.16}$$

2.4 Shifting the World Coordinate System

Since Euler rotations are defined around the origin, translating 3D coordinates to the centroid $\vec{X_o} = [X_o\ Y_o\ Z_o]$ would increase the numerical stability of the solution. Such shift in the coordinate system would only affect the matrix $\mathbf{Q}$ and the motion parameter vector $\vec{\phi}$ will compensate the shift. In this case we can rewrite Equation (2.9) as

$$-I_t = \frac{1}{Z}\left[fI_x\ fI_y\ -(xI_x + yI_y)\right]\mathbf{Q}'\vec{\phi}' \tag{2.17}$$

where

$$\mathbf{Q\prime} = \begin{bmatrix} 1 & 0 & 0 & 0 & (Z-Z_o) & -(Y-Y_o) \\ 0 & 1 & 0 & -(Z-Z_o) & 0 & (X-X_o) \\ 0 & 0 & 1 & (Y-Y_o) & -(X-X_o) & 0 \end{bmatrix}$$

and $\vec{\phi} = [\vec{T\prime}^T\ \vec{\Omega\prime}^T]^T$

2.5 Least Squares Solution

In the previous sections brightness and depth constancy constraint formulations were derived. These formulations try to approximate a single pixel's velocity as it undergoes instantaneous translation and rotation. Since these constraint equations are linear they can be stacked up in a matrix formulation $b_I = \mathbf{H_I}\vec{\phi}$ across N pixels which belong to the object that is being tracked, where $\vec{b_I} \in \Re^{N\times 1}$ is the temporal intensity derivative and $\vec{H_I} \in \Re^{N\times 6}$ is the constraint matrix for brightness values. $\vec{\phi}$ is the motion vector that is to be solved for. A similar formulation can be obtained for depth constraints $b_D = \mathbf{H_D}\vec{\phi}$. Given that $N > 6$, the least squares method can be used to solve for the motion parameter vector $\vec{\phi}$ independently for both systems. Combining the two equations into a single equation,

$$\vec{b} = \mathbf{H}\vec{\phi},\ \text{where}\ \mathbf{H} = \begin{bmatrix} \mathbf{H_I} \\ \lambda\mathbf{H_D} \end{bmatrix}, \vec{b} = \begin{bmatrix} \vec{b_I} \\ \lambda\vec{b_D} \end{bmatrix} \tag{2.18}$$

in order to solve for a combined vector $\vec{\phi}$, where λ is the scaling factor for depth constraints. In situations where the disparity image is more reliable than the intensity image such as fast changing illumination conditions, values higher than 1 should

be chosen for λ. In other situations where it is known that intensity image is more reliable than the disparity image, values smaller than 1 should be chosen for λ. The least-squares solution for the equation above is:

$$\vec{\phi} = \mathbf{H}^{\dagger}\vec{b} = (\mathbf{H}^{T}\mathbf{H})^{-1}\mathbf{H}^{\mathbf{T}}\vec{b} \tag{2.19}$$

where $\mathbf{H}^{\dagger}$ is the pseudo inverse. The least squares solution gives out the motion vector for a set of pixels that belong to the object of interest. These pixels are selected from the images where both intensity and depth images are well defined.

CHAPTER 3

Iterative Closest Point Algorithm (ICP)

3.1 Motivation and Problem Definition

Rigid body or head tracking problem can be thought of as a registration problem [28]. If a transformation can be found that brings the current facial surface into alignment with the previous surface, then the pose of the head can be updated based on the previous pose and calculated transformation.

In order to represent the facial surfaces, low level primitives such as point clouds or triangle meshes are well suited. For any intensity and disparity image set $\{I_t, Z_t\}$ acquired at time t, a point cloud, based on point coordinates $\{x_t, y_t, z_t, I_t\}$ can be built and can be brought into registration with a previously acquired cloud. Extraction of this point cloud and the necessary data are explained in Section 5.1.

The algorithm takes two sets of 3D point clouds namely P and X and tries to find a rigid transformation defined as rotation $\mathbf{R}$ and a translation $\mathbf{t}$ that minimizes the least square distances between the corresponding points of P and X. The total mean-squares distance can be expressed as

$$e(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_i \|(\mathbf{R}p_i + \mathbf{t}) - c(p_i)\|^2, p_i \in P \tag{3.1}$$

where N is the total number of pairs and the c is the correspondence that associates

every point p_i to a corresponding point in set X such that,

$$c : P \rightarrow X | \forall p_i \in P, y_i = c(p_i) \in X \tag{3.2}$$

The correspondence function c is not known because the corresponding points of a surface captured at two different times cannot be obtained from a stereo camera. If the correspondence function is known then the transformation $(\mathbf{R}, \mathbf{t})$ that minimizes Equation (3.1) can be calculated in closed form. Since c is unknown, solving for $\mathbf{R}$ and $\mathbf{t}$ is a complex minimization process. Although the correspondence function c is unknown, it can be approximated by an estimated function $\hat{c}$ and by decomposing Equation (3.1) into several steps that reduce the matching error, it is possible to solve for $\mathbf{R}$ and $\mathbf{t}$ iteratively.

The main idea behind an iterative approach is to use the estimated function $\hat{c}$ to calculate point correspondences to calculate $\mathbf{R}_k$ and $\mathbf{t}_k$ for the current iteration based on $\mathbf{R}_{k-1}$ and $\mathbf{t}_{k-1}$ from the previous iteration. The total transformation ($\mathbf{R}$, $\mathbf{t}$) is the accumulation of $\mathbf{R}_k$ and $\mathbf{t}_k$ from all iterations and can be expressed as follows: $\mathbf{R}_{k+1} = \Delta\mathbf{R}_k\mathbf{R}_k$ and $\mathbf{t}_{k+1} = \Delta\mathbf{t}_k + \mathbf{t}_k$, where $\mathbf{R}_k$ and $\mathbf{t}_k$ are initially set to the 3×3 identity matrix and $[0\ 0\ 0]^T$ respectively. Using an iterative approach and an estimated function $\hat{c}$, the complex geometric point matching function (3.1) can be replaced with a simpler one,

$$e(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k) = \frac{1}{N}\sum_{i}^{N} \| \left[\Delta\mathbf{R}_k(\mathbf{R}_k\mathbf{p}_i + \mathbf{t}_k) + \Delta\mathbf{t}_k\right] - \hat{c}(\mathbf{p}_i)\|^2 \tag{3.3}$$

The estimated function $\hat{c}$ has to be chosen in such a way that at every iteration complex geometric point matching converges to a minimum.

3.2 Closest Point Matching

The original ICP algorithm assumes that the correspondence function $\hat{c}$ which associates every point of the set P with a point with the smallest Euclidean distance in the set X

$$\hat{c}(\mathbf{p}) = \underset{x \in X}{\operatorname{argmin}}\, d(\mathbf{p}, \mathbf{x}) = \underset{x \in X}{\operatorname{argmin}} \|\mathbf{p} - \mathbf{x}\|^2 \tag{3.4}$$

is a good approximation of the unknown function c. If the calculated closest points reflect the real correspondences, then the algorithm will converge faster, however it is not always sufficient to establish good correspondences using the Euclidean distances.

Often better correspondences can be found if more discriminative features are used, such as color or intensity information which are usually available with 3D data. Also secondary or calculated features such as surface normals or curvature can be extracted from range images or triangle meshes.

3.2.1 Color

Using color information together with surface geometry improves the performance of the ICP algorithm, since it helps to avoid ambiguous cases where the surface geometry is not always sufficient for a successful correspondence. Several authors have reported performance improvements when ICP is used with color features [28, 22, 29, 30]. Since intensity information is already available, it can be easily integrated into the distance function,

$$d(\mathbf{p}, \mathbf{x}) = \|\mathbf{p} - \mathbf{x}\|^2 + \alpha(I_p - I_x)^2 \tag{3.5}$$

where α is a normalizing constant for the intensity value. Figure 3.1 shows how color helps to find better correspondences.

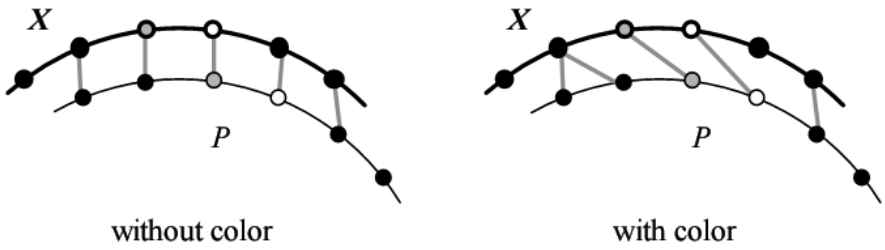

Figure 3.1: Influence of color or intensity on matching (taken from [1])

3.2.2 Surface orientation

In addition to color or intensity information, other geometric features such as surface normals can also be used in order to improve the coupling and more generally the registration. The normal vector for an arbitrary point can be computed from the depth image gradients:

$$\vec{n_{ti}} = \begin{bmatrix} \frac{\partial Z_t}{\partial u_{ti}} & \frac{\partial Z_t}{\partial v_{ti}} & 1 \end{bmatrix} \tag{3.6}$$

In order to add the normals to the distance computation, a similar approach can be followed as in color driven matching. For a normal vector $\tilde{\mathbf{n}} = (u, v, w)$ the distance can be stated as:

$$d(\mathbf{p}, \mathbf{x}) = \left[\|\mathbf{p} - \mathbf{x}\|^2 + \alpha(u_p - u_x)^2 + \beta(v_p - v_x)^2 + \gamma(w_p - w_x)^2 \right] \tag{3.7}$$

Figure 3.2 shows how using surface normals help to find better correspondences.

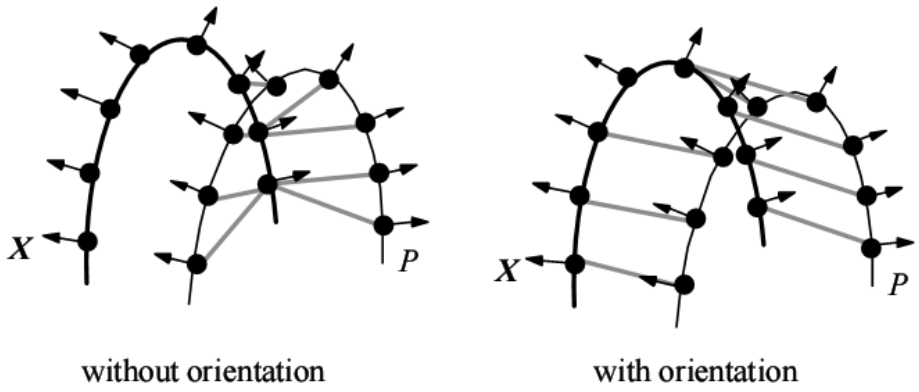

Figure 3.2: Influence of surface normals on matching (taken from [1])

3.2.3 Performance optimization using kd-trees

Searching for closest points is the most exhaustive part of the algorithm and thus it can be optimized for speed when special search structures such as trees, projections

or clusters are used. In this work, usage of kD-trees is selected for their ability to integrate several features in the closest point search. The main advantage of the kD-tree is that it can store multi-dimensional data and can search the 'real' closest point whereas cluster and projection based methods give only close approximations to the closest point. A review of the other methods can be found in [31].

A kD-tree is a space-partitioning data structure for organizing points in a k-dimensional space. kD-trees constitute a useful data structure for several applications, such as searches involving a multidimensional search key, such as range searches and nearest neighbor searches. kD-trees use only splitting planes, that are perpendicular to one of the coordinate system axes, in order to divide the entire space according to one of the feature space dimensions, which are usually used in a cyclic order. Every splitting plane is represented as a node in the kD-tree and goes through one of the feature points.

3.2.3.1 Constructing a kd-tree

The canonical method for constructing a kd-tree has the following two constraints:

- As one moves down the tree, one cycles through the axes used to select the splitting planes. For example, the root would have an x-aligned plane, the root's children would both have y-aligned planes, the root's grandchildren would all have z-aligned planes, and so on.
- At each step, the point selected to create the splitting plane is the median of the points being put into the kd-tree, with respect to their coordinates in the axis being used.

The constraints above lead to a balanced kd-tree. The better the kd-tree is balanced, the faster the closest point search will be. An example for constructing a kd-tree is presented in Figure 3.3. Among the points, point A is selected as the root of the kd-tree because it is the median of the points along the x-axis. Now all points

on the left-hand side of A possess smaller x coordinates and all the points on the right-hand side of A have greater x coordinates. The median point E according to the next key, in this case y, is added to the tree as a left child of A. The construction of the tree continues iteratively until all point are assigned to a node in the tree.

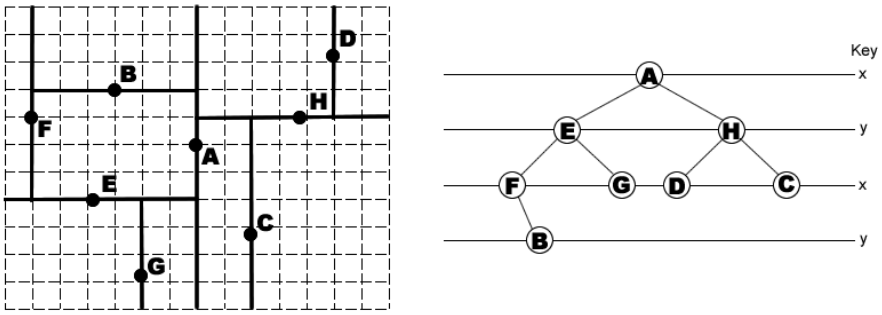

Figure 3.3: Example of a 2D kd-tree construction

The generalization to k-dimensions is straightforward. The splitting lines are replaced by hyperplanes in k-dimensions. The complexity for building a kd-tree from N points is $O(NlogN)$ and uses a storage $\theta(N)$.

3.2.3.2 Closest point search in a kd-tree

The nearest neighbor (NN) algorithm, to find the nearest neighbor to a given target point not in the tree, relies on the ability to discard large portions of the tree by performing a simple test. To perform the NN calculations, the tree is searched in a depth-first fashion, refining the nearest distance. First the root node is examined with an initial assumption that the smallest distance to the next point is infinite. The subdomain (right or left), which is a hyperrectangle containing the target point, is searched. This is done recursively until a final minimum region containing the node is found. The algorithm then (through recursion) examines each parent node, seeing if it is possible for the other domain to contain a point that is closer. This is performed by testing for the possibility of intersection between the hyperrectangle

and the hypersphere (formed by the target node and the current minimum radius). If the rectangle that has not been recursively examined yet, does not intersect this sphere, then there is no way that the rectangle can contain a point that is a better nearest neighbor. This is repeated until all domains are either searched or discarded, thus leaving the nearest neighbor as the final result. Searching for the nearest point is an $O(NlogN)$ operation.

3.3 Best Transformation

Once the point correspondence set $c(\mathbf{p}_{i,k}, \mathbf{y}_{i,k})$ has been created between surfaces P and X, the best transformation is computed by minimizing the mean squared error of the couplings $\{\mathbf{p}_{i,k}, \mathbf{y}_{i,k}\}$ as a function of $\Delta\mathbf{R}_k$ and $\Delta\mathbf{t}_k$.

$$e(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|(\Delta\mathbf{R}_k\mathbf{p}_{i,k} + \Delta\mathbf{t}_k) - \mathbf{y_{i,k}}\|^2 \tag{3.8}$$

In order to make the equations more readable, let us drop out the iteration index k and in order to be able to regroup the terms in error function, let us minimize the sum of squared errors instead of their mean value, which produces the same result. With this simplification, the error function under consideration becomes

$$e(\Delta\mathbf{R}, \Delta\mathbf{t}) = \sum_{i=1}^{N} \|(\Delta\mathbf{R}\mathbf{p}_i + \Delta\mathbf{t}) - \mathbf{y}_i\|^2 \tag{3.9}$$

3.3.1 Matching Centroids

The error function (3.9) can be factored out into the following sums

$$e(\Delta\mathbf{R}, \Delta\mathbf{t}) = \sum_{i=1}^{N} \|\Delta\mathbf{R}\mathbf{p}_i - \mathbf{y}_i\|^2 + 2\Delta\mathbf{t} \cdot \sum_{i=1}^{N} (\Delta\mathbf{R}\mathbf{p}_i - \mathbf{y}_i) + \sum_{i=1}^{N} \|\Delta\mathbf{t}\|^2 \tag{3.10}$$

The first and the last term only depends on $\Delta\mathbf{R}$ and $\Delta\mathbf{t}$, but the second term is mixed. If it can be set to zero, then the error function e can be optimized for $\Delta\mathbf{R}$

and $\Delta\mathbf{t}$ separately. In order to set the mixed term to zero, the data sets $\mathbf{P}_k$ and $\mathbf{Y}_k$ are translated to their centroids.

$$\begin{aligned} \mu_p = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i, \mu_y = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i \\ \mathbf{p}'_i = \mathbf{p}_i - \mu_p, \mathbf{y}'_i = \mathbf{y}_i - \mu_y, \end{aligned} \tag{3.11}$$

Now the error function (3.9) can be written as

$$e(\Delta\mathbf{R}, \Delta\mathbf{t}) = \sum_{i=1}^{N} \|(\Delta\mathbf{R}\mathbf{p}'_i + \Delta\mathbf{t}) - \mathbf{y}'_i\|^2, \text{with } \mathbf{t}' = \Delta\mathbf{R}\mu_p - \mu_y + \Delta\mathbf{t} \tag{3.12}$$

and factored out into the following sums

$$e(\Delta\mathbf{R}, \Delta\mathbf{t}) = \sum_{i=1}^{N} \|\Delta\mathbf{R}\mathbf{p}'_i - \mathbf{y}'_i\|^2 + 2\Delta\mathbf{t} \cdot \sum_{i=1}^{N}(\Delta\mathbf{R}\mathbf{p}'_i - \mathbf{y}'_i) + \sum_{i=1}^{N} \|\Delta\mathbf{t}'\|^2 \tag{3.13}$$

Because the measurements are now translated to their centroids, the second term sums up to zero and thus can be dropped. The first and the third term cannot be set to zero therefore they must be minimized separately. For the total error to be minimized, the third term must be minimal or zero so the best translation vector can be defined as

$$\Delta\mathbf{t} = \mu_y - \Delta\mathbf{R}\mu_p \tag{3.14}$$

Finally, in order to find the best rotation matrix $\Delta\mathbf{R}$, the following equation needs to be minimized.

$$e(\Delta\mathbf{R}) = \sum_{i=1}^{N} \|\Delta\mathbf{R}\mathbf{p}'_i - \mathbf{y}'_i\|^2 \tag{3.15}$$

Once the best rotation matrix $\mathbf{R}$ is found that minimizes the Equation (3.15), the best translation vector can be calculated using Equation (3.14). Several methods have been proposed to solve for the values of $\mathbf{R}$ and $\mathbf{t}$ that minimize e, such as steepest decent, and also closed form solutions like singular value decomposition (SVD), dual number quaternions and unit quaternions [1] [32]. Extensive evaluations and comparisons of these methods can be found in [32].

In this work, the quaternion method proposed by Faugeras [33] and Horn [20] has been chosen and used. This method is reviewed in the following sections.

3.3.2 Quaternions

This section provides a basic definition of quaternions and basic properties that are necessary to derive the formulation for computing the best transform.

In mathematics, quaternions are a non-commutative extension of complex numbers. A quaternion can be thought of as a complex number with four components, with one real component and three imaginary components (i, j, k). Quaternions can be represented as

$$\begin{aligned} \dot{\mathbf{q}} =& q_0 + q_x i + q_y j + q_z k \text{ with} \\ i^2 =& j^2 = k^2 = ijk = -1 \end{aligned} \tag{3.16}$$

It is also possible to think of quaternions as a vector with four components. The first component being the scalar and the remaining three being the imaginary part:

$$\dot{\mathbf{q}} = (q_0, q_x, q_y, q_z) = (q_0, \mathbf{q}) \tag{3.17}$$

The conjugate of a quaternion can be defined as

$$\dot{\mathbf{q}}^* = (q_0, -\mathbf{q}) \tag{3.18}$$

and a point in 3D space is represented as a purely imaginary quaternion:

$$\dot{\mathbf{p}} = (0, p_x, p_y, p_z) = (0, \mathbf{p}) \tag{3.19}$$

The product of two quaternions is defined as

$$\begin{aligned} \dot{\mathbf{q}}\dot{\mathbf{r}} =& (q_0 r_0 - \mathbf{q} \cdot \mathbf{R}, q_0 \mathbf{R} + r_0 \mathbf{q} + \mathbf{q} \times \mathbf{R}) \\ =& (q_0 r_0 - q_x r_x - q_y r_y - q_z r_z, q_0 r_z, q_0 r_x + q_x r_0 + q_y r_z - q_z r_y, \\ & q_0 r_y - q_x r_z + q_y r_0 + q_z r_x, q_0 r_z + q_x r_y - q_y r_x + q_z r_0) \end{aligned} \tag{3.20}$$

and one can note that quaternion multiplication is not commutative. It is possible to associate an orthogonal matrix $\mathbf{Q}$ to a quaternion.

$$\mathbf{Q} = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{bmatrix} \text{ and } \bar{\mathbf{Q}} = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix} \tag{3.21}$$

Such an association makes it possible to express quaternion product as multiplication of a matrix $\mathbf{Q}$ with the quaternion.

$$\dot{\mathbf{q}}\dot{\mathbf{r}} = \mathbf{Q}\dot{\mathbf{r}} \neq \dot{\mathbf{r}}\dot{\mathbf{q}} = \bar{\mathbf{Q}}\dot{\mathbf{r}} \text{ and } \dot{\mathbf{q}}\dot{\mathbf{r}}^* = \bar{\mathbf{Q}}^T\dot{\mathbf{r}} \tag{3.22}$$

The magnitude $|\dot{\mathbf{q}}|^2$ of a quaternion can be defined as:

$$\begin{aligned} |\dot{\mathbf{q}}|^2 &= \dot{\mathbf{q}}\dot{\mathbf{q}}^* = \dot{\mathbf{q}}^*\dot{\mathbf{q}} \\ &= \left(q_0^2 + \|\mathbf{q}\|^2, 0\right) = \left(\|\dot{\mathbf{q}}\|^2, 0\right) = (\dot{\mathbf{q}} \cdot \dot{\mathbf{q}}, 0) \end{aligned} \tag{3.23}$$

A quaternion can be referred as a unit quaternion if its magnitude is 1.

Rotations and Quaternions

A rotation on a data point can be expressed by unit quaternions if there exists a way to transform purely imaginary quaternions to purely imaginary quaternions. Such a transformation must preserve the length of the data vector; furthermore the dot and cross product must also be preserved [20]. It can be shown that the composite product

$$\dot{\mathbf{p}}' = \dot{\mathbf{q}}\dot{\mathbf{p}}\dot{\mathbf{q}}^* \tag{3.24}$$

is purely imaginary given that $\dot{\mathbf{p}} = (0, p_x, p_y, p_z)$ is the data point to be rotated and $\dot{\mathbf{q}}$ is a unit quaternion. This can be proved by extracting the rotation matrix defined by the quaternion $\dot{\mathbf{q}}$ using the properties presented in (3.22):

$$\dot{\mathbf{p}}' = \dot{\mathbf{q}}\dot{\mathbf{p}}\dot{\mathbf{q}}^* = (\mathbf{Q}\dot{\mathbf{p}})\dot{\mathbf{q}}^* = \bar{\mathbf{Q}}^T(\mathbf{Q}\dot{\mathbf{p}}) = (\bar{\mathbf{Q}}^T\mathbf{Q})\dot{\mathbf{p}} \tag{3.25}$$

The matrix $\bar{\mathbf{Q}}^T\mathbf{Q}$ is in the form

$$\bar{\mathbf{Q}}^T\mathbf{Q} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \tag{3.26}$$

where $\mathbf{R}$ is always a 3×3 rotation matrix which is defined by the unit quaternion $\dot{\mathbf{q}}$. $\mathbf{R}$ has the following values

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3.27}$$

The upper-right elements of $\bar{\mathbf{Q}}^T\mathbf{Q}$ is zero therefore $\dot{\mathbf{p}}'$ is purely imaginary.

3.3.3 Best Rotation

The previous section summarizes some basic properties of quaternions and shows that a rotation can be represented using quaternions. The minimization function in 3.15 can be rewritten using quaternions for any rotation $\Delta\mathbf{R}$.

$$e(\Delta\mathbf{R}) = \sum_{i=1}^{N} \|\Delta\mathbf{R}\mathbf{p}'_i - \mathbf{y}'_i\|^2 = \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i\dot{\mathbf{q}}^* - \dot{\mathbf{y}}'_i\|^2 = e(\dot{\mathbf{q}}) \tag{3.28}$$

In order to allow the grouping of different terms into one multiplication the error function is multiplied with the square magnitude of the unit quaternion $\dot{\mathbf{q}}$ which does not change its value.

$$e(\dot{\mathbf{q}}) = \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i\dot{\mathbf{q}}^* - \dot{\mathbf{y}}'_i\|^2\|\dot{\mathbf{q}}\|^2 \tag{3.29}$$

which can be factorized and simplified into

$$\begin{aligned} e(\dot{\mathbf{q}}) &= \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i\dot{\mathbf{q}}^*\dot{\mathbf{q}} - \dot{\mathbf{y}}'_i\dot{\mathbf{q}}\| \text{ with } \|\dot{\mathbf{q}}\dot{\mathbf{n}}\|^2 = \|\dot{\mathbf{q}}\|^2\|\dot{\mathbf{n}}\|^2 \\ &= \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i - \dot{\mathbf{y}}'_i\dot{\mathbf{q}}\|^2 \text{ with } \dot{\mathbf{q}}^*\dot{\mathbf{q}} = \|\dot{\mathbf{q}}\|^2 = (1, \mathbf{0}) \end{aligned} \tag{3.30}$$

Since the imaginary part of the error quaternion $\dot{\mathbf{e}}$ is a zero vector only the real part of $\dot{\mathbf{e}}$ is observed and can be regrouped using properties described in (3.23)

$$\begin{aligned} e &= \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i - \dot{\mathbf{y}}'_i\dot{\mathbf{q}}\|^2 = \sum_{i=1}^{N} (\dot{\mathbf{q}}\dot{\mathbf{p}}'_i - \dot{\mathbf{y}}'_i\dot{\mathbf{q}}).(\dot{\mathbf{q}}\dot{\mathbf{p}}'_i - \dot{\mathbf{y}}'_i\dot{\mathbf{q}}) \\ &= \sum_{i=1}^{N} (\dot{\mathbf{q}}\dot{\mathbf{p}}'_i)(\dot{\mathbf{q}}\dot{\mathbf{p}}'_i) - 2(\dot{\mathbf{q}}\dot{\mathbf{p}}'_i).(\dot{\mathbf{y}}'_i\dot{\mathbf{q}}) + (\dot{\mathbf{y}}'_i\dot{\mathbf{q}}).(\dot{\mathbf{y}}'_i\dot{\mathbf{q}}) \end{aligned} \tag{3.31}$$

and when rewritten in matrix representation as described in (3.21),

$$\begin{aligned} e(\dot{\mathbf{q}}) &= \sum_{i=1}^{N} (\bar{\mathbf{P}}'_i\dot{\mathbf{q}}).(\bar{\mathbf{P}}'_i\dot{\mathbf{q}}) - 2(\bar{\mathbf{P}}'_i\dot{\mathbf{q}}).(\mathbf{Y}'_i\dot{\mathbf{q}}) + (\mathbf{Y}'_i\dot{\mathbf{q}}).(\bar{\mathbf{P}}'_i\dot{\mathbf{q}}) \\ &= \sum_{i=1}^{N} (\bar{\mathbf{P}}'_i\dot{\mathbf{q}})^T(\bar{\mathbf{P}}'_i\dot{\mathbf{q}}) - 2(\bar{\mathbf{P}}'_i\dot{\mathbf{q}})^T(\mathbf{Y}'_i\dot{\mathbf{q}}) + (\mathbf{Y}'_i\dot{\mathbf{q}})^T(\bar{\mathbf{P}}'_i\dot{\mathbf{q}}) \\ &= \sum_{i=1}^{N} \dot{\mathbf{q}}^T\bar{\mathbf{P}}_i'^T\bar{\mathbf{P}}'_i\dot{\mathbf{q}} - 2\dot{\mathbf{q}}^T\bar{\mathbf{P}}_i'^T\mathbf{Y}'_i\dot{\mathbf{q}} + \dot{\mathbf{q}}^T\mathbf{Y}_i'^T\bar{\mathbf{P}}'_i\dot{\mathbf{q}} \\ &= \sum_{i=1}^{N} \dot{\mathbf{q}}^T(\bar{\mathbf{P}}_i'^T\bar{\mathbf{P}}'_i - 2\bar{\mathbf{P}}_i'^T\mathbf{Y}'_i + \mathbf{Y}_i'^T\mathbf{Y}'_i)\dot{\mathbf{q}} \end{aligned} \tag{3.32}$$

Finally the minimization problem can now be stated as

$$\begin{aligned} e(\dot{\mathbf{q}}) &= \sum_{i=1}^{N} \dot{\mathbf{q}}'^T\mathbf{A}_i\dot{\mathbf{q}} = \dot{\mathbf{q}}'^T\mathbf{A}\dot{\mathbf{q}} \\ \text{where } \mathbf{A}_i &= \bar{\mathbf{P}}_i'^T\bar{\mathbf{P}}'_{\mathbf{i}} - 2\bar{\mathbf{P}}_i'^T\mathbf{Y}'_i + \mathbf{Y}_i'^T\mathbf{Y}'_i \text{ and } \mathbf{A} = \sum_{i=1}^{N}\mathbf{A_i} \\ \mathbf{A} &= \sum_{i=1}^{N} \|\mathbf{p}_i\|^2\mathbf{I} - 2\mathbf{B} + \|\mathbf{y}_i\|^2\mathbf{I} \end{aligned} \tag{3.33}$$

where

$$\mathbf{B} = \begin{bmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}+S_{zy} & S_{zx}+S_{zx} & S_{xy}+S_{yx} \\ S_{yz}+S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yz} & S_{zx}+S_{xz} \\ S_{zx}+S_{xz} & S_{xy}+S_{yz} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}+S_{yx} & S_{zx}-S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix}$$

At this point it is convenient to introduce a 3×3 cross-correlation matrix:

$$\zeta_{\mathbf{py}} = \sum_{i=1}^{N} \mathbf{p}'_i \cdot \mathbf{y}'_i \tag{3.34}$$

The individual elements of matrix $\zeta_{\mathbf{py}}$ can be identified as

$$\zeta_{\mathbf{py}} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{3.35}$$

where

$$S_{mn} = \sum_{i=1}^{N} p'_{i,m} \cdot y'_{i,n} \text{with } m, n \in (x, y, z) \tag{3.36}$$

The matrix $\zeta_{\mathbf{py}}$ contains all the information required to obtain the matrix $\mathbf{A}$ in the minimization function. Once the elements of $\zeta_{\mathbf{py}}$ are known, the matrix $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \begin{bmatrix} trace(\zeta_{py}) & \Delta^T \\ \Delta & \zeta_{py} + \zeta_{py}^T - trace(\zeta_{py})\mathbf{I_3} \end{bmatrix} \tag{3.37}$$

where $\mathbf{I_3}$ is a 3×3 identity matrix and Δ is a column vector formed by the cyclic elements of the anti symmetric matrix $\mathbf{C_{ij}} = \zeta_{py} - \zeta_{py}^T$ and the elements of Δ are defined as $\Delta = [\mathbf{C_{23}}\ \mathbf{C_{31}}\ \mathbf{C_{12}}]^T$.

Using Lagrange multipliers technique, Equation (3.33) is equivalent to

$$\begin{aligned} \text{find } min_{\dot{\mathbf{q}}} L = \left[\dot{\mathbf{q}}^T \mathbf{A} \dot{\mathbf{q}} + \lambda(1 - \|\dot{\mathbf{q}}\|^2)\right] \\ \text{with constraint } \|\dot{\mathbf{q}}\|^2 = 1 \end{aligned} \tag{3.38}$$

where the above equation can be solved by setting the partial derivatives of L to zero

$$\frac{\partial L}{\partial \dot{\mathbf{q}}} = \mathbf{A}\dot{\mathbf{q}} + (\dot{\mathbf{q}}^T \mathbf{A})^T - 2\lambda\dot{\mathbf{q}} = 0 \tag{3.39}$$

Since $\mathbf{A}$ is a symmetric matrix, Equation (3.39) becomes

$$\begin{aligned} 2\mathbf{A}\dot{\mathbf{q}} - 2\lambda\dot{\mathbf{q}} = 0 \\ \mathbf{A}\dot{\mathbf{q}} = \lambda\dot{\mathbf{q}} \end{aligned} \tag{3.40}$$

which states that $\dot{\mathbf{q}}$ is an eigenvector of the matrix $\mathbf{A}$. Combining Equations (3.33) and (3.40) we can come up with the following relation

$$e_{min}(\dot{\mathbf{q}}) = \dot{\mathbf{q}}^T \mathbf{A} \dot{\mathbf{q}} = \dot{\mathbf{q}}^T \lambda \dot{\mathbf{q}} = \lambda \dot{\mathbf{q}}^T \dot{\mathbf{q}} = \lambda \|\dot{\mathbf{q}}\|^2 = \lambda \tag{3.41}$$

This equation lads us to conclude that the unit quaternion that minimizes the error function e is the eigenvector corresponding to the smallest eigenvalue of matrix $\mathbf{A}$.

3.3.4 Summary of quaternion method

The unit quaternion method to calculate the optimal rigid transformation which minimizes the coupling error can be summarized as follows:

(1) Translate point clouds to the origin.

(2) Build matrix $\mathbf{A}$

(3) Calculate the smallest eigenvalue of $\mathbf{A}$ and its corresponding eigenvector and use it as quaternion $\dot{\mathbf{q}}$

(4) Calculate the best rotation matrix $\Delta\mathbf{R}$ using $\dot{\mathbf{q}}$

(5) Calculate the best translation vector $\Delta\mathbf{t}$ using Equation (3.14)

3.4 Iteration Termination

ICP is a computationally intensive operation; therefore the number of iterations should be kept as low as possible. Various approaches have been proposed in literature that automatically stop the iterations of the algorithm. These methods are:

- Absolute error criterion:
  Iterations stop when the mean square error, described in Equation (3.1), falls below a threshold, $e < \tau$.
- Error change criterion:
  Iterations stop when the error change falls below a threshold $e_{k-1} - e_k < \tau$.
- Pose change criterion: The iteration stops when the calculated rigid transformation $(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k)$ at iteration k is relatively small and ineffective [32].

3.5 ICP Algorithm Overview

The ICP algorithm to register two point clouds is formulated in a procedural description as follows:

- input: Two sets of point sets, $P = \{p_i\}$ with N_p points and $X = \{x_i\}$ with N_x points.
- output: A transformation $(\mathbf{R}, \mathbf{t})$ which registers P to X
- initialization: $k = 0$, $P_0 = P$, $R_0 = I$ and $t_0 = [0\ 0\ 0]^T$
- iteration k:

(1) Compute closest points:
Use the squared Euclidean distance

$$d(p, x) = \|p - x\|^2 \tag{3.42}$$

to compute a correspondence set $C^{(k)}(\tilde{p_i}, y_i)$ of closest point with size N being the number of points in set P. The closest point y_i is defined as follows

$$y_i = \hat{c}(\tilde{p_i}) = \min_{x \in X} d(\hat{p_i}, x) \text{ with } \hat{p_i} = \mathbf{R}p_i + \mathbf{t} \tag{3.43}$$

(2) Compute registration $(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k)$:
Define a mean error function of couplings in set $c^{(k)}(\tilde{p_i}, y_{i,k})$ as a function of $(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k)$.

$$e(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|(\Delta\mathbf{R}_k\mathbf{p}_{i,0} + \Delta\mathbf{t}_k) - \mathbf{y_{i,k}}\|^2 \tag{3.44}$$

and calculate the rigid transformation $(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k)$ such that

$$e_k = \underset{\Delta\mathbf{R}_k, \Delta\mathbf{t}_k}{\operatorname{argmin}} e(\Delta\mathbf{R}_k, \Delta\mathbf{t}_k) \tag{3.45}$$

(3) Apply the registration:
Apply the transformation obtained to get the next point set $P_{k+1} = \{\mathbf{p}_{i,k+1}\}$ defined as

$$\mathbf{p}_{i,k+1} = \mathbf{R}_k\mathbf{p}_{p,0} + \mathbf{t}_k \tag{3.46}$$

and update $\mathbf{R}_{k+1} = \Delta\mathbf{R}_k\mathbf{R}_k$ and $\mathbf{t}_{k+1} = \Delta\mathbf{t}_k + \mathbf{t}_k$

(4) Stop iterating if the matching has converged to a desired error rate or if the maximum number of iterations is reached. Set $\mathbf{R} = \mathbf{R}_k$ and $\mathbf{t} = \mathbf{t}_k$

The block diagram of the algorithm is presented in Figure 3.4

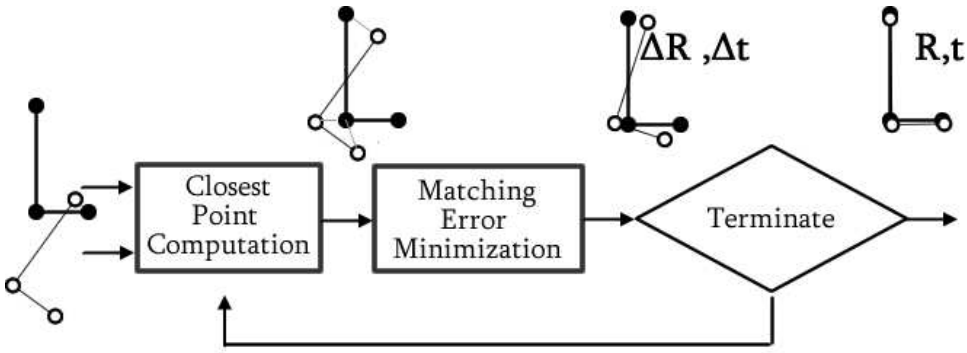

Figure 3.4: Block Diagram of ICP Algorithm

CHAPTER 4

Stereo-based Head Pose Tracker using the Scale Invariant Feature Transform

In the previous chapter, the ICP algorithm is explained. The major drawback of the ICP algorithm is its iterative nature. Since the corresponding points between the current and previous frames are not known, the correspondence function is estimated and the registration problem is solved iteratively. If the correspondence function is known, then the registration problem can have a closed form solution. To solve this problem we have extracted and matched the Scale Invariant Feature Transform (SIFT) features between consecutive frames providing us with the missing correspondence function. If good correspondences are computed using the SIFT algorithm, then the registration problem can be solved in a single step avoiding any iterations.

In the next section, a brief description of the SIFT algorithm is given and the steps of 3D rigid body motion tracking using SIFT and the unit quaternion method is explained.

4.1 Scale Invariant Feature Transform

Scale-invariant feature transform (SIFT) is one of several computer vision algorithms for extracting distinctive features from images [24]. SIFT interest points are based

on a Difference of Gaussian detector. Its high-dimensional descriptor vector relies on gradient histograms. In addition to key point location, Lowe's method [24] also provides a way to extract features which are invariant to scale, rotation, and translation. The name Scale-invariant Feature Transform was chosen, as the algorithm transforms image data into scale-invariant coordinates. Following are the major stages of computation used to generate the set of image features:

- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- Keypoint descriptor

4.1.1 Detection of scale-space extrema

The first stage of keypoint detection is to determine location and scales that can be repeatably assigned under differing views of the same object. Detection of these locations can be done by searching for stable features across all possible scales, using a continuous function scale known as scale space [34]. Using the Gaussian function as the scale-space kernel, the scale space of an image can be defined as a function $L(x, y, \sigma)$, which is produced from the convolution of the image, $I(x, y)$ with a variable-scale Gaussian, $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4.1}$$

where $*$ is the convolution operator in x and y and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{4.2}$$

In order to detect stable keypoint locations in the scale-space, difference-of-Gaussian (DoG) function $D(x, y, \sigma)$ is used, which is computed from the difference of two nearby scales that are separated by a constant multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{4.3}$$

Figure 4.1 shows an efficient way to construct $D(x, y, \sigma)$. In the left column, the initial image is incrementally convolved by Gaussians of different scales, which are separated by a constant factor k in the scale space to build an image pyramid. Each octave of scale space is divided into an integer number s, of intervals, so $k = 2^{1/s}$. Adjacent image scales are subtracted to produce the DoG images as shown on the right. Once one octave is complete, the Gaussian image in the current octave that has a variance of 2σ is sub-sampled. It is often problematic to downsize an image by an arbitrary scale; therefore scaling down can be mimicked by convolving the image with a Gaussian and sub-sampling by taking every second pixel in each row and column.

Once the DoG images are computed, the local maxima and minima are to be detected. In the scale-space, each pixel has 26 neighbors, eight in the current scale, nine in scale below and nine in the scale above (See Fig.4.2). Each pixel is compared with its 26 neighbors and is selected as a local extrema if its larger than all of its neighbors or smaller then all of them. Since most pixels are eliminated after a few comparisons, the cost of detecting is much lower than building of the pyramid.

4.1.2 Keypoint localization

Once a set of potential keypoints have been found in the image by comparing a pixel to its neighbors in the scale space, for each keypoint, its sub-pixel and sub-space location (x, y, σ) are determined. This information allows points with low contrast to be rejected. Furthermore, keypoints that lie on edges are needed to be removed as well, because edges are poor keypoints since their location cannot be determined well along the edge. Poorly defined peaks in the DoG function will have a large principal curvature across the edge but a small one in the perpendicular direction. The 2×2 Hessian matrix $\mathbf{H}$ can be computed at the location and the scale of the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{4.4}$$

The derivatives are estimated by taking differences of neighboring sample pixels. The eigenvalues of $\mathbf{H}$ are proportional to the principal curvature of D. Based on the

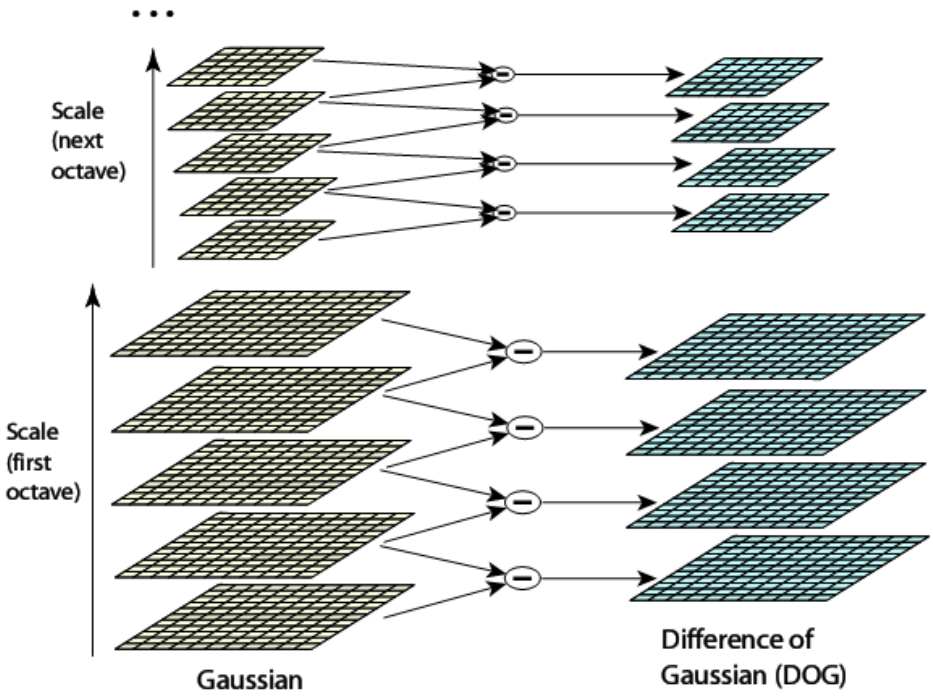

Figure 4.1: Difference of Gaussians are computed from a pyramid of Gaussians. Adjacent Gaussian images are subtracted to produce a difference of Gaussian(DoG) images.

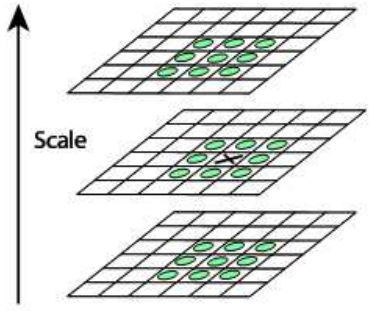

Figure 4.2: Maxima and minima of the DoG images are detected by comparing the pixel of interest by its 26 neighbors of the current and adjacent scales.

proportion of the eigenvalues, edge like features can be detected and eliminated.

4.1.3 Orientation Assignment

Finally by calculating the dominant orientation of each keypoint, the keypoint descriptor represented with this orientation becomes invariant to image rotation. First the Gaussian image in the octave with the closest scale is selected, so that all computations are carried out in a scale-invariant manner. For each keypoint the gradient magnitude, $m(x, y)$ and orientation, $\theta(x, y)$ are computed using pixel differences for the current scale L(x,y).

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \theta(x,y) &= tan^{-1}((L(x,y+1) - L(x,y-1))/L(x+1,y) - L(x-1,y))) \end{aligned} \tag{4.5}$$

An orientation histogram is build using the gradient orientations within a region around the keypoint. The histogram contains 36 bins, separating the 360 degree orientations into regions of 10 degrees. Each sample is weighted by its gradient magnitude before being added to the histogram. Significant peaks in the histogram correspond to dominant directions in the gradient. The highest peak in the histogram is selected to be the orientation of the keypoint, however if other local peaks within 80% of the highest peak are detected, a new keypoint with that orientation is created. For any keypoints with multiple peaks, multiple keypoints will be created. This improves the orientation invariance of the SIFT algorithm.

4.1.4 Local Image Descriptor

The previous operations extract image location, scale and orientation for keypoints from a given image. The next step is to describe a local image region in a manner which is invariant to scale and orientation as well as to changes in illumination and to 3D viewpoint.

Figure 4.3 shows how keypoint descriptors are computed. In a region around the

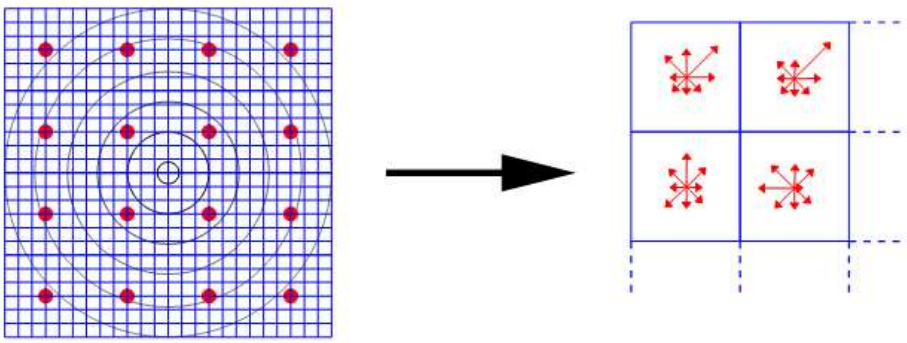

Figure 4.3: SIFT Descriptor. For each pixel around the keypoint gradient magnitudes and orientations are computed. These samples are weighted by a Gaussian and accumulated into 16 orientation histograms for the 16 subregions.

keypoint, image gradient magnitudes and orientations are computed. The gradient magnitudes are weighted by a Gaussian centered on the keypoint location. In a typical application the window is divided into $4 \times 4 = 16$ subregions and for each subregion, an orientation histogram is build and the histograms are placed at the center of the subregion. Boundary effects, in which the descriptor abruptly changes as a subregion shifts smoothly from one histogram to another or from one orientation to another, are avoided by interpolation where each gradient votes for an orientation in its neighboring histograms. The vote is weighted by $1 - d$ for each dimension, where d is the distance to the histogram. The coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation in order to maintain orientation invariance. The 4×4 descriptors computed from a 16×16 sample array provide a 128-dimensional descriptor array.

Finally, in order to reduce the effects of illumination changes, the feature vector is normalized to unit length. Since a change in image contrast where each pixel value is multiplied by a constant will also reflect the gradients by the same constant, normalizing the feature vector will remove the effect of contrast change. A brightness change where a constant is added to every image pixel will not change the gradient magnitudes since they are computed from pixel differences. Therefore, affine changes in illumination do not affect the normalized keypoint descriptor. In order to reduce the affects of non-linear illumination changes, which can occur due to camera change or due to a change in illumination orientation or amount that affect 3D surfaces, the influence of large gradient magnitudes in the unit feature vector are thresholded to be

no longer than 0.2 and then the feature vector is renormalized to unit length. This is because nonlinear illumination changes are more likely to effect gradient magnitudes rather than gradient orientations, therefore by thresholding the magnitudes, more emphasis is put onto orientations. The value 0.2 is determined experimentally by Lowe [24].

4.2 Matching

The SIFT features for the current and the previous frame are extracted, creating two sets of features F_1 and F_2. Feature matching is then performed between the features in each of the two sets. For each feature in F_1, the nearest neighbor is found in F_2, where the nearest neighbor is defined using some distance measure based on feature descriptors or other properties such as scale and orientation. In order to speed up the matching, nearest-neighbor algorithm on a 128 dimensional kd-tree is applied.

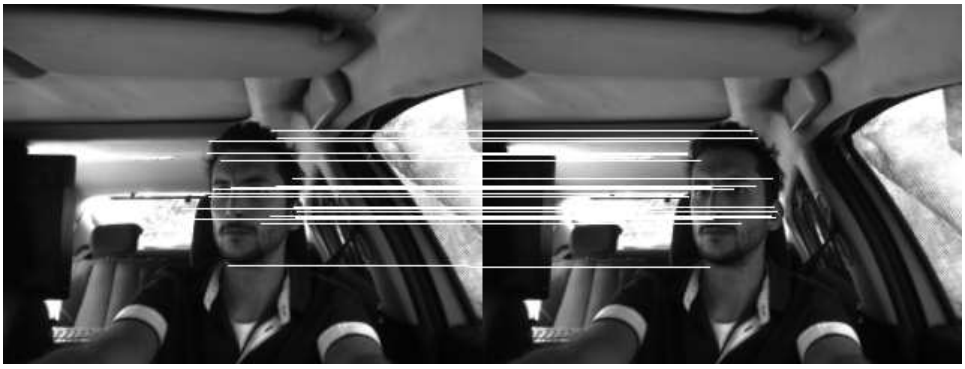

Figure 4.4: Matching of keypoints

4.3 Estimating Motion Parameters

Once the SIFT descriptors are matched, 3D point correspondence set $C(\mathbf{p}_i, \mathbf{y}_i)$ is generated by back projecting the descriptor pixels using the disparity image to get

the real world coordinates for the pixel. The best transformation is then computed by minimizing the mean squared error of the couplings $\{\mathbf{p}_i, \mathbf{y}_i\}$ as a function of $\mathbf{R}$ and $\mathbf{t}$. The error function can be minimized using the unit quaternion method explained in Section 3.3

$$e(\mathbf{R}, \mathbf{t}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|(\mathbf{R}\mathbf{p}_i + \mathbf{t}) - \mathbf{y_i}\|^2 \tag{4.6}$$

The point sets $\mathbf{p}_k$ and $\mathbf{y}_k$ are translated to their centroids

$$\begin{aligned} \mu_p = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i, \mu_y = \frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i \\ \mathbf{p}'_i = \mathbf{p}_i - \mu_p, \mathbf{y}'_i = \mathbf{y}_i - \mu_y, \end{aligned} \tag{4.7}$$

Translating the point sets to the origin drops out the $\mathbf{t}$ in the minimization function and by representing the rotation $\mathbf{R}$ using quaternions the following error function is obtained

$$e(\mathbf{R}) = \sum_{i=1}^{N} \|\mathbf{R}\mathbf{p}'_i - \mathbf{y}'_i\| = \sum_{i=1}^{N} \|\dot{\mathbf{q}}\dot{\mathbf{p}}'_i\dot{\mathbf{q}}^* - \dot{\mathbf{y}}'_i\|^2 = e(\dot{\mathbf{q}}) \tag{4.8}$$

From the new point sets defined around the origin, a 3×3 cross-correlation matrix is build.

$$\zeta_{\mathbf{py}} = \sum_{i=1}^{N} \mathbf{p}'_i \cdot \mathbf{y}'_i \tag{4.9}$$

The individual elements of the matrix $\zeta_{\mathbf{py}}$ can be identified as

$$\zeta_{\mathbf{py}} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{4.10}$$

The matrix $\mathbf{Q}$ is formed using the elements of $\zeta_{\mathbf{py}}$ and the unit quaternion that minimizes the error function e is the eigenvector corresponding to the smallest eigenvalue of matrix $\mathbf{Q}$.

CHAPTER 5

Experimental Results

5.1 Tracker Structure

Figure 5.1 presents the block diagram of the whole tracker system. Prior to running the tracking algorithms, a fast face detector system initializes the tracker and a face model is build around the 3D real world points found by the face detector. One of the NFC, ICP or SIFT algorithms are executed together with the pose reseting algorithm to estimate the pose change between the previous and the current frame.

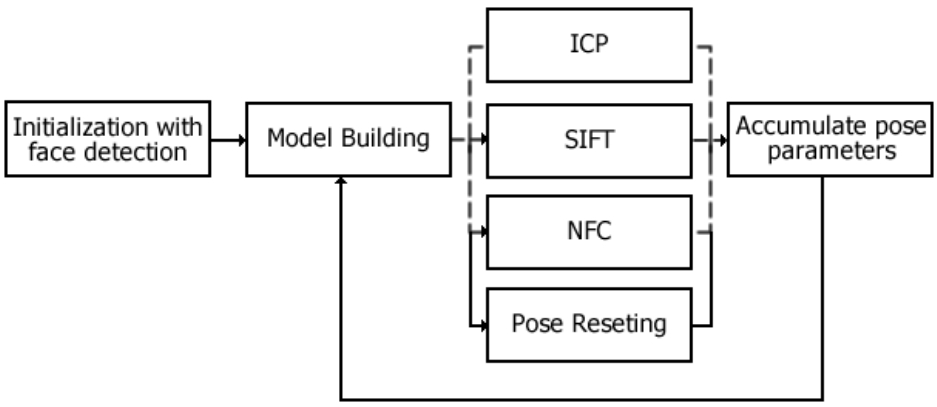

Figure 5.1: Block diagram of the tracker system

5.1.1 Face Detection

At the beginning of the motion estimation algorithm, a fast face detector [35] scans the intensity image for face regions. The face detector is trained to detect only frontal faces; therefore it can be assumed that the initial rotation of the head is aligned with the camera. The initial region for the head is detected by the face detector at the first frame and then in each step the region is updated based on the tracker output. For this work we assumed that the face of the user is towards the camera and all the experimentation data has been collected with this assumption.

5.1.2 Model Building

Once the face detector gives a rough estimate of where the head is located in the intensity image, 3D world coordinates of the center of the head is extracted using the disparity image and a point cloud is formed by selecting the pixels that are within a selected range to the center of the head. Also a face mask, cropped intensity and disparity images are stored to be used by the tracking algorithms. Figure 5.2 shows the output of the face detection algorithm and the head model extracted using disparity and intensity images. One advantage of building such a model based on depth thresholding is that it provides robustness to partial occlusions. The occluded points appear as missing data in the face model.

5.1.3 Pose Parameters

The goal of the tracker is to find rigid body pose change parameters $\vec{\phi} = [\vec{t}, \vec{\Omega}]^T$ between two models, where $\vec{\Omega}$ is the instantaneous rotation vector represented with three Euler angles and $\vec{t}$ is the translation vector in x, y and z directions. Since the pose changes are additive, the current pose estimation is updated as follows:

$$\begin{aligned} \phi_{\Omega}^{\text{new}} &= \phi_{\Omega}^{\text{old}} + \phi_{\Omega} \\ \phi_{t}^{\text{new}} &= \phi_{t}^{\text{old}} + \phi_{t} \end{aligned} \tag{5.1}$$

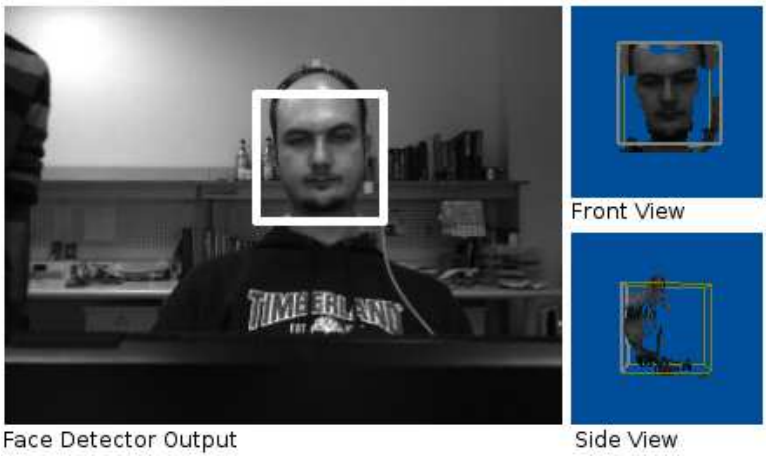

Figure 5.2: Result of face detection and model extraction.

where δ_t is initially set to the head's position vector acquired through face detection and δ_w is initially set to zero.

5.1.4 Pose Reseting

All the tracking algorithms, either reviewed or proposed in this thesis are differential methods that can only estimate the pose change between consecutive frames. In order to obtain the pose of the head at a given time t relative to the first frame, pose differences between adjacent frames need to be accumulated. However since each pose change measurement is noisy, the accumulation of these measurements becomes noisier with time, potentially resulting in an unbounded drift. [2]

In order to overcome this problem, we use a simple and effective method for resetting the pose parameters when the head pose of the user is close to its original pose. We compute the change in appearance between the first frame and the current frame. If the change in their appearance is smaller than a threshold, we reset the pose back to its original state at time t_0. We compute the L_2 distance between the current and the first frame after aligning the two intensity images and selecting the pixels within the face mask.

5.2 System Evaluation

We have tested the proposed SIFT based stereo tracker on both synthetic and real video sequences captured using a Bumblebee [36] stereo camera, which can deliver images of size 320 by 240 pixels at 15 frames per second, from Point Grey Research. We compared the results with those of the NFC, and the ICP algorithms. For NFC and ICP algorithms all the points/pixels that belong to the head are used for registration. Also for the ICP algorithm, the correspondence function uses both geometry and intensity information as given in Equation 3.5. Since The motivating application of this thesis is on driver fatigue detection in a vehicle environment, the collected data are, with a user sitting on a chair, doing free head movement. In order to get a decent resolution of the user's face, the users were asked to sit about 80cm away from the camera which is also a reasonable distance for a vehicle environment.

Without special optimizations, the SIFT tracker can update the pose changes at around 2 frames per second on a Celeron 1500Mhz Laptop with 512MB of RAM. Table 5.1 presents the time complexity comparison of the SIFT algorithm against NFC and ICP algorithms.

Table 5.1: Computational complexity of the 3 tracking algorithms.

Algorithm	**Frames per second**
NFC	30 fps
ICP (at most 10 iterations)	0.3 fps
SIFT	2 fps

5.2.1 Test results over synthetic video

In order to perform a quantitative evaluation of the optical flow and 3D pose estimation algorithms in a noise free environment with precise ground truth data available, several synthetic image sequences were generated. Using a 3D texture mapped head model and animating according to predefined motion parameters, we

were able to generate video sequences that were realistic enough to do performance tests on the tracker. Disparity images are rendered using Bumblebee's back projection API, therefore the resulting disparity image is compatible with images captured from the Bumblebee. Figure 5.3 shows sample frames from the sequences. Six sequences were recorded; 3 for translations in x,y,z axis and 3 for rotations around x,y,z axis. Translations were performed as 10cm in every direction, twice in a row. Rotations were performed as 25, 60, 35 degrees around x,y,z axis respectively. Figures 5.5 present the estimated translation and rotation parameters during tracking compared to ground truth. Table 5.2 shows the mean error and variance of each tracker over synthetic data. As shown in the Table, the proposed SIFT tracker performs better than NFC in all translation cases. Although ICP performs better than SIFT in X-axis translation, SIFT is more reliable considering other cases. In rotation examples, NFC outperforms both ICP and SIFT, but SIFT tracker's results are close to NFC's results. Also as seen in Figure 5.5 NFC method can easily drift because it uses image gradients to estimate the motion rather than well defined correspondences like SIFT does.

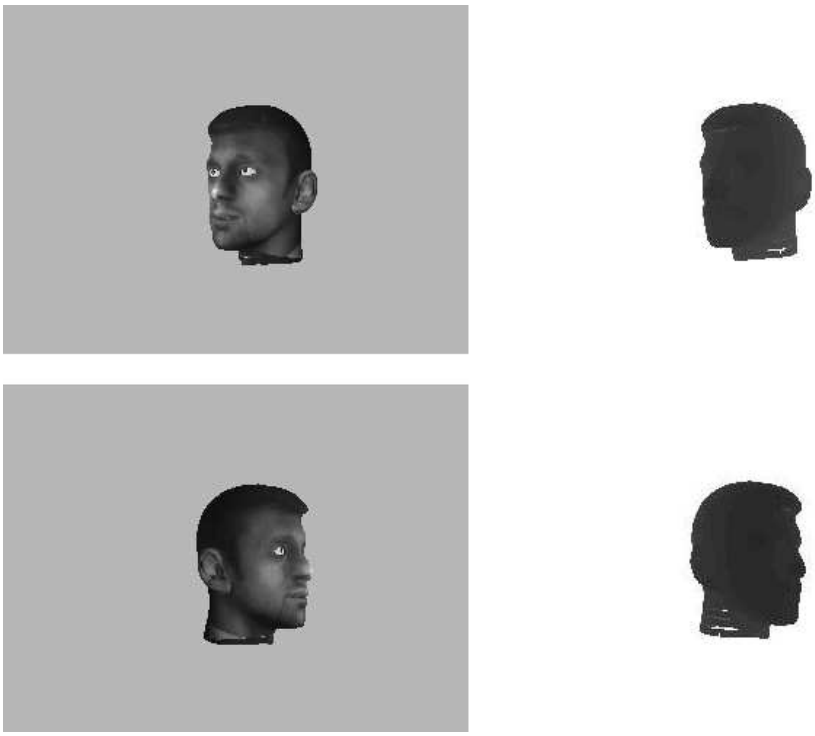

Figure 5.3: Sample video frames from computer generated sequences. The left column shows intensity images and the right column shows corresponding disparity image.

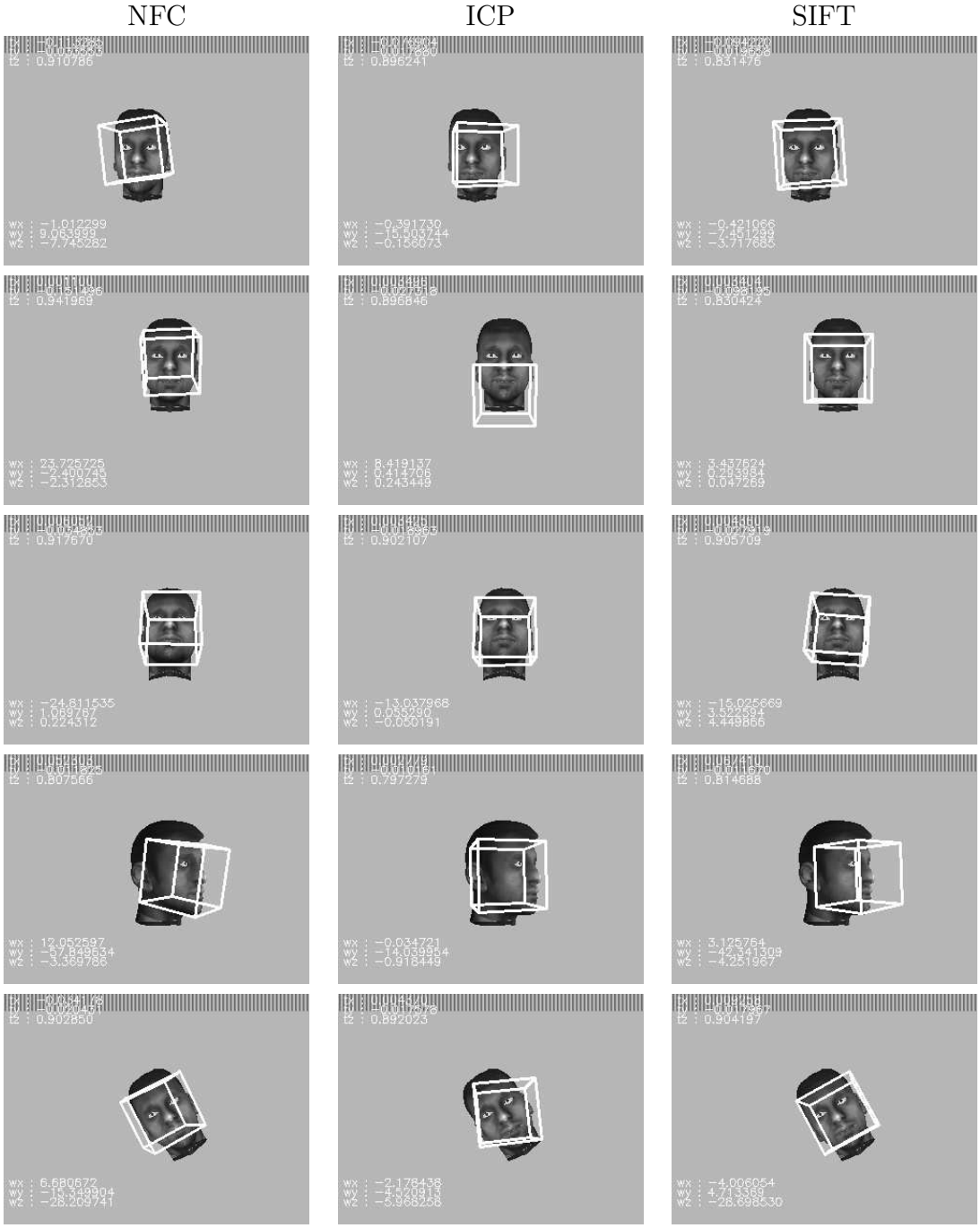

Figure 5.4: Head tracking results for synthetic images sequences. Each row represents tracking results at different frames

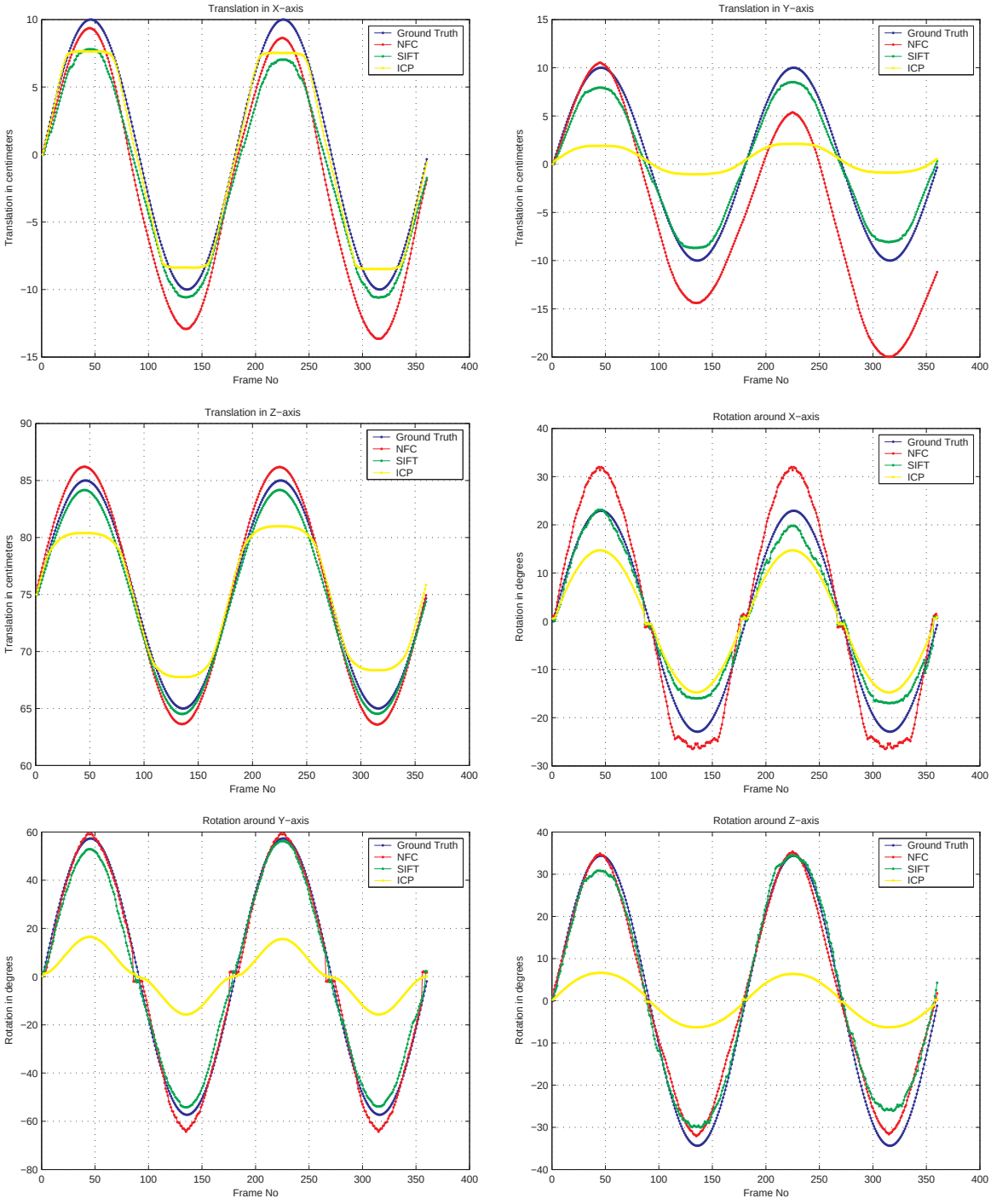

Figure 5.5: Head pose estimation plots for synthetic image sequences.

Table 5.2: Performance results of the trackers over synthetic video sequences

Mean error and variance in cm	NFC		ICP		SIFT	
	Mean	Var	Mean	Var	Mean	Var
Translation in X-Axis	2.21	1.27	**0.84**	**0.43**	1.57	0.59
Translation in Y-Axis	5.20	11.20	5.31	7.09	**1.11**	**0.27**
Translation in Z-Axis	0.85	0.17	1.71	1.86	**0.67**	**0.08**
Mean error and variance in degrees	NFC		ICP		SIFT	
	Mean	Var	Mean	Var	Mean	Var
Rotation around X-Axis	4.42	5.93	5.36	5.36	**2.84**	**3.57**
Rotation around Y-Axis	**2.52**	**4.78**	28.25	151.93	3.78	4.88
Rotation around Z-Axis	**2.56**	**3.20**	17.69	74.10	2.81	4.85

5.2.2 Polhemus Tracker

For testing the performance of our tracker we have used a hardware based tracker, Polhemus Fastrak [37] as the ground truth. The tracker consists of 3 parts: System Electronics Unit (SEU), a transmitter, and a receiver. SEU contains the hardware necessary to generate and sense the magnetic fields, compute position and orientation, and interface with the host computer via a RS-232 or a USB interface. The transmitter contains electromagnetic coils and is responsible for emitting the magnetic fields. The transmitter is the system's reference frame for receiver measurements. The receiver unit contains the magnetic coils that detect the magnetic field emitted by the transmitter. Polhemus is a 3D motion tracker with 6 degrees of freedom. The tracker has an accuracy of 0.076cm RMS with 0.0005 cm resolution for the position and an accuracy 0.15° RMS with an 0.025° resolution for the orientation. In a laboratory environment with computers and metal furniture around, we observed lower accuracy than the specifications. However, the ground truth captured was still visually good enough to evaluate the head tracking algorithms considered in the thesis.

The ground truth obtained through the Polhemus tracker and the outputs of the vision based trackers are expressed in different coordinate systems. In order to compare the outputs of the Polhemus and vision based trackers, the output of the

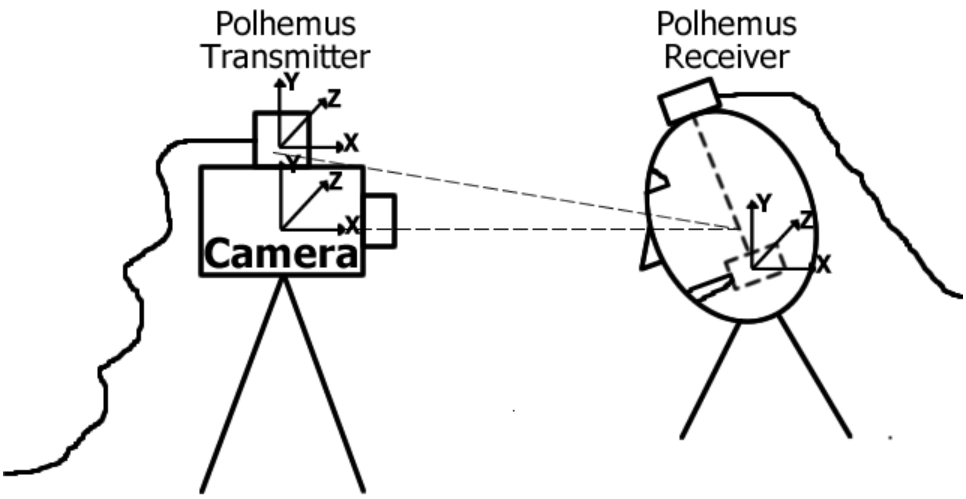

Figure 5.6: Camera and magnetic tracker coordinate systems

Polhemus tracker must be converted to the coordinate system of the vision based trackers. As shown in Figure 5.6, we have positioned the camera and the magnetic transmitter as close as possible to minimize the offset between the two coordinate systems, and aligned them so that the two coordinate systems are parallel to each other. The position vector acquired through the Polhemus tracker is translated along the Polhemus' orientation vector by 10cm so that the ground truth position is roughly aligned with the neck's pivot of rotation. At the initialization and model building step, the center of the head is calculated by both the Polhemus tracker and the vision based trackers, the offset between the two coordinate systems is calculated and the measured offset is added to the magnetic tracker's position vector in order to obtain the Polhemus tracker output in the vision based tracker's coordinate system.

5.2.3 General Performance of the Tracker on Real Data

This experiment was designed to test the performance of the trackers in real video sequences. Six sequences were recorded. In the first three sequences, the user translates his head in x, y and z axis for about 10cm. In the final 3 sequences the user rotates his head around x, y and z axis for more than 20 degrees.

Table 5.3 shows the average error and variance values. Figure 5.7 shows the results of the three trackers against the ground truth data observed using the Polhemus tracker. From the plots it can be observed that SIFT and ICP perform better than NFC for translations; however ICP performs poorly on rotations in line with the observations reported in [28]. NFC performs slightly better than SIFT on rotations. In general we can conclude that SIFT tracking is a solution that performs better than ICP and NFC for translations, and performs as good as NFC in rotations.

Table 5.3: Performance results of the trackers over real video sequences

Mean error and variance in cm	NFC		ICP		SIFT	
	Mean	Var	Mean	Var	Mean	Var
Translation in X-Axis	4.72	15.50	3.12	6.05	**0.66**	**0.54**
Translation in Y-Axis	5.80	21.34	2.63	2.21	**1.11**	**0.69**
Translation in Z-Axis	1.71	1.50	1.52	1.58	**1.21**	**0.67**
Mean error and variance in degrees	NFC		SIFT		ICP	
	Mean	Var	Mean	Var	Mean	Var
Rotation around X-Axis	**2.76**	**4.51**	7.72	26.67	4.67	8.67
Rotation around Y-Axis	**2.72**	**6.44**	12.47	61.28	4.71	12.44
Rotation around Z-Axis	**5.43**	**14.26**	8.66	33.11	4.70	6.85

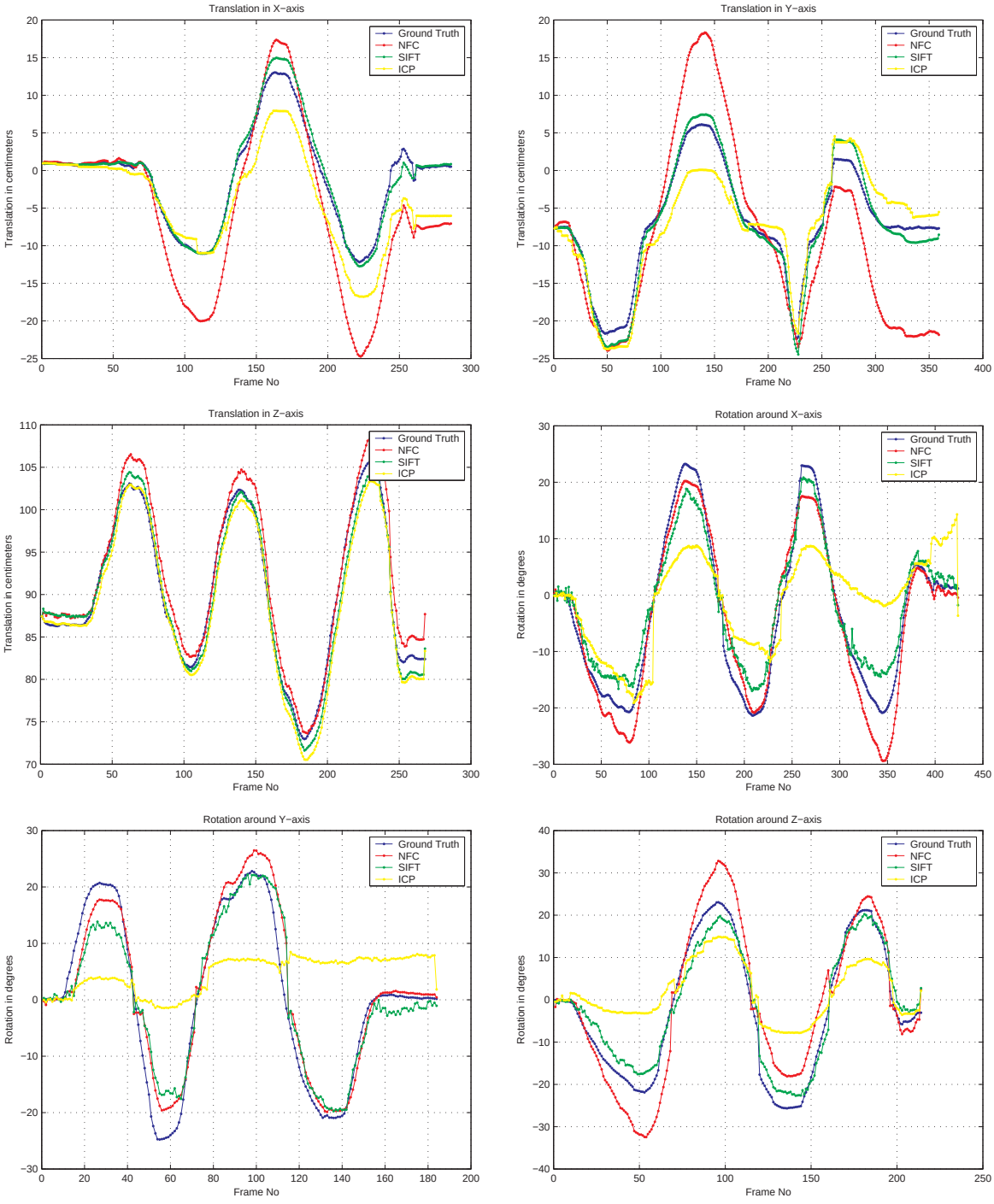

Figure 5.7: Head pose estimation plots for real image sequences.

5.2.4 Performance Under Time Varying Illumination Changes

This set of experiments were conducted to measure the performance of the trackers under time varying illumination conditions. Five sequences were recorded with different subjects and under time varying illumination conditions. During the sequence, the subjects were asked to rotate their head up to 30° along the y axis then the x axis and finally along the z axis with translation up to 10cm in the x axis. Figures 5.8 and 5.11 present some key frames from sequences 1 and 2. Figures 5.9 and 5.12 show some tracker outputs for sequences 1 and 2. Table 5.4 shows the average error and variance values over 5 recorded sequences. As it can be seen in the table both NFC and SIFT are robust to illumination changes. NFC is not affected by illumination changes due to its depth constancy constraint equation and SIFT is not affected because of its illumination invariant features. In this set of experiments, SIFT and NFC perform similar in translations. SIFT perform slightly better than NFC for rotations.

Robustness to illumination changes is very important for applications in real life environments. For example in a vehicle environment, the illumination changes suddenly based on location and the orientation of the vehicle; therefore a head tracking system must be invariant to time varying illumination changes.

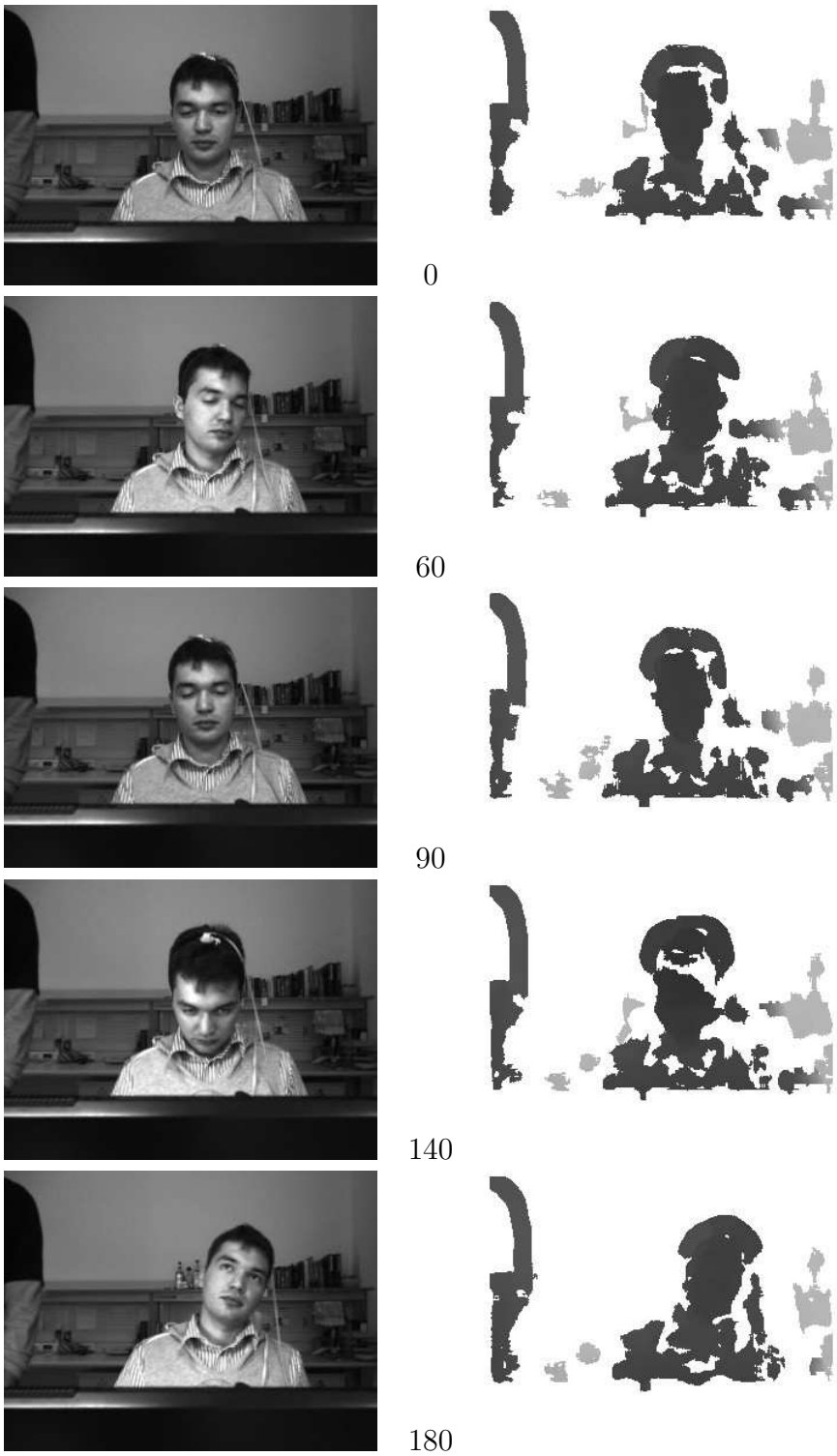

Figure 5.8: Intensity and depth images from time varying illumination change sequence 1.

Figure 5.9: Head tracking results for time varying illumination change sequence 1. Each row represents tracking results at different frames: 0, 60, 90, 140, 180

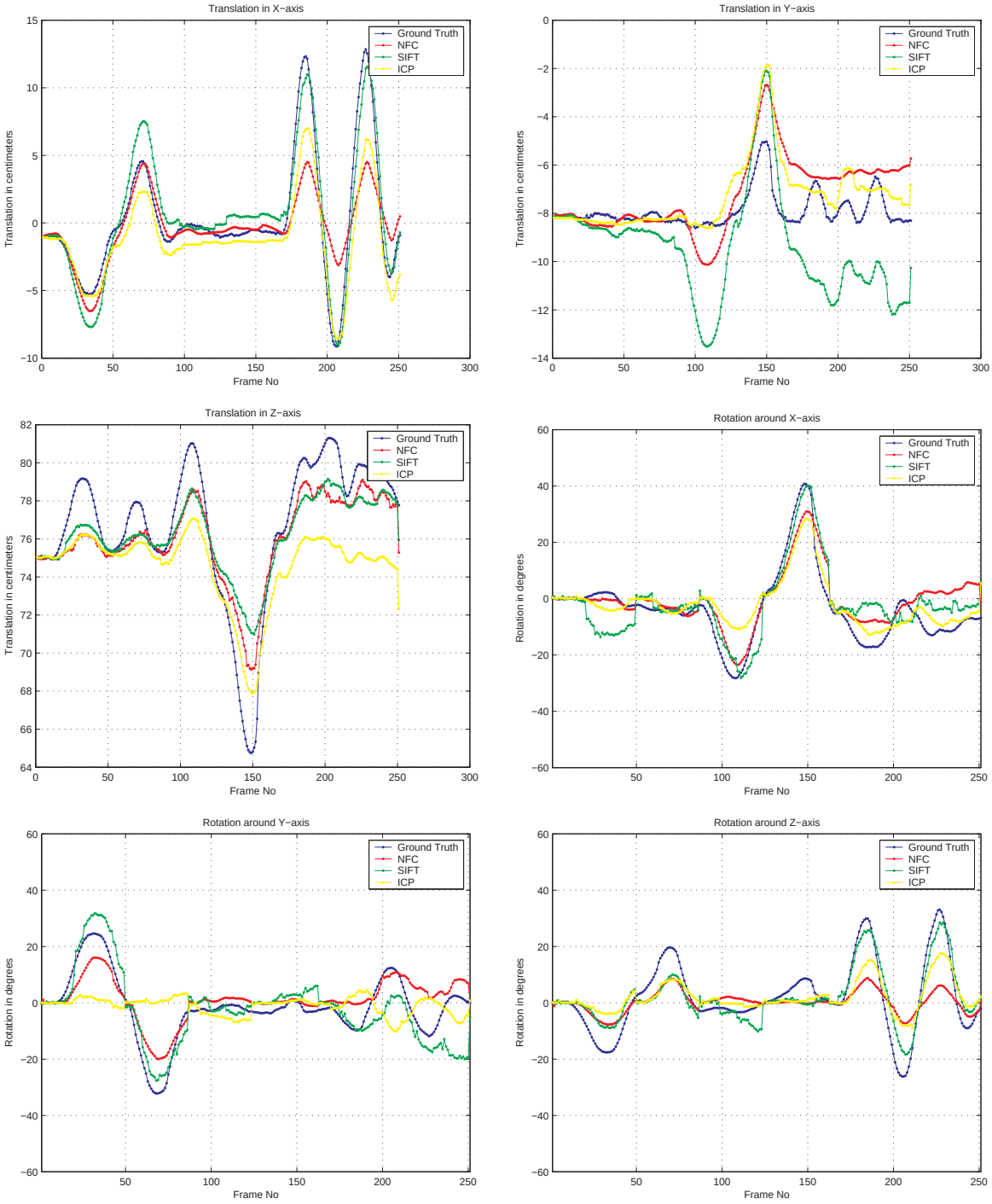

Figure 5.10: Head pose estimation plots for time varying illumination change sequence 1

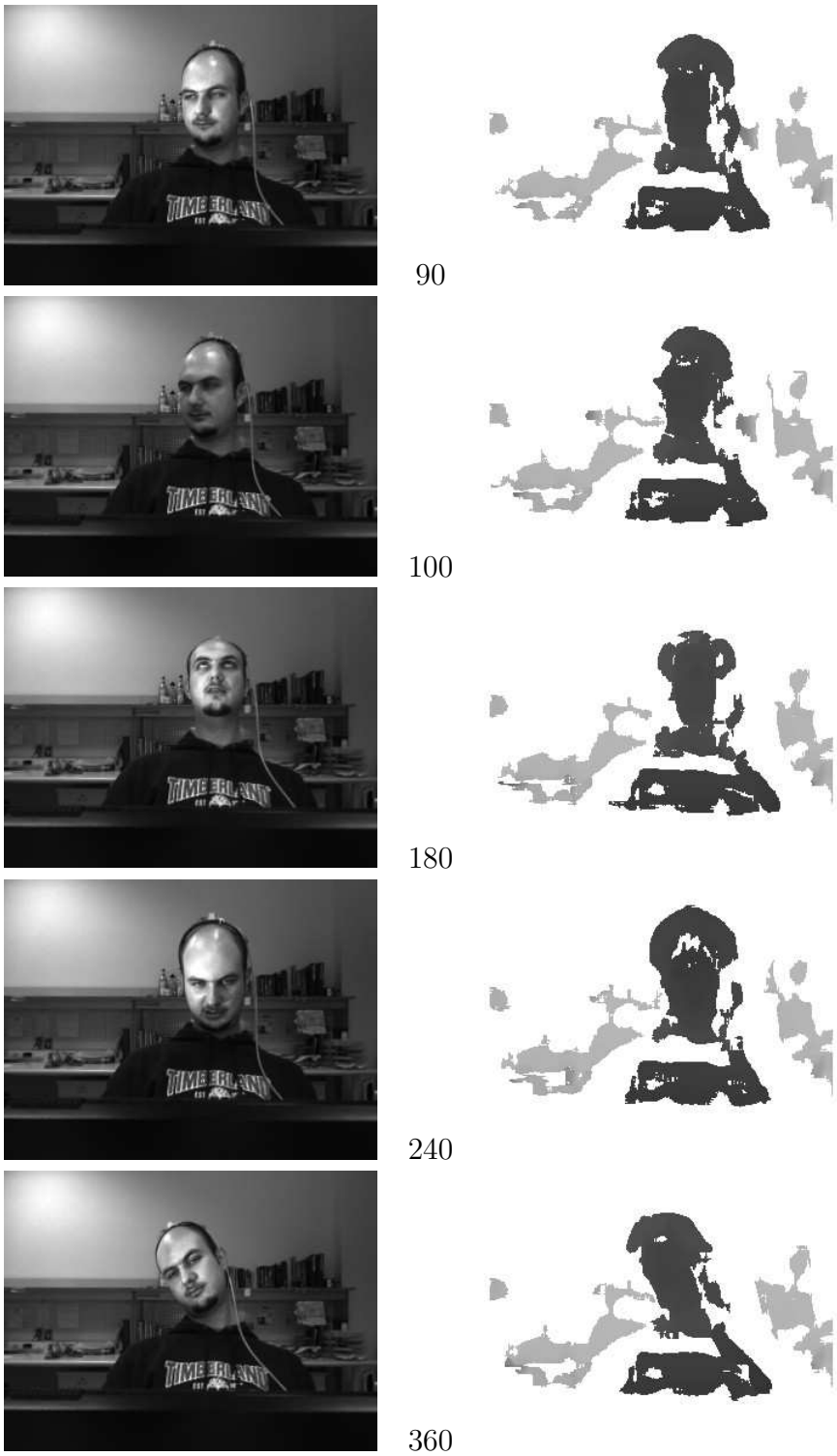

Figure 5.11: Intensity and depth images from time varying illumination change sequence 2.

Figure 5.12: Head tracking results for time varying illumination change sequence 2. Each row represents tracking results at different frames: 90, 100, 180, 240, 360

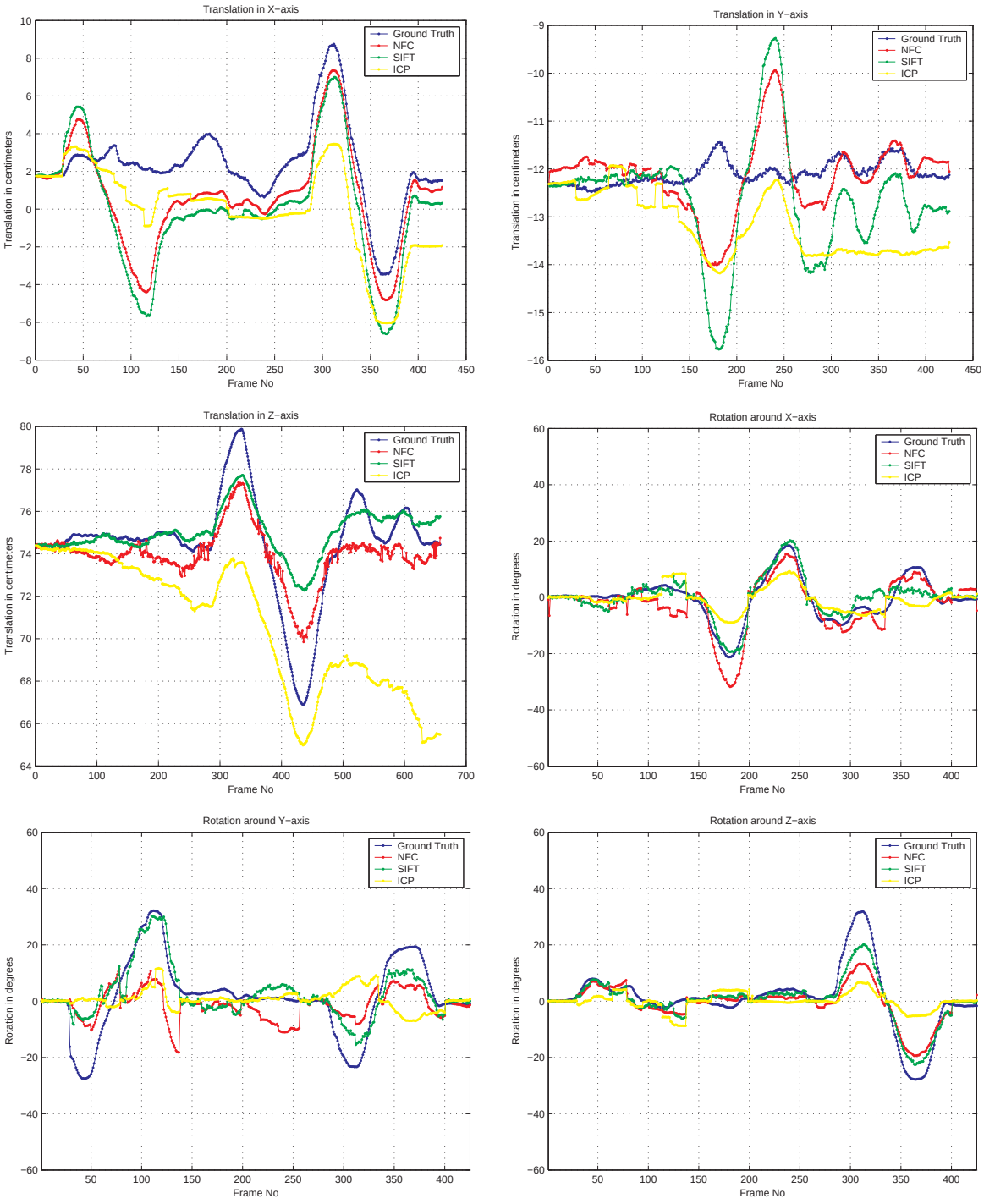

Figure 5.13: Head pose estimation plots for time varying illumination change sequence 2

Table 5.4: Performance results of the trackers under illumination changes.

Mean error and variance in cm	NFC		ICP		SIFT	
	Mean	Var	Mean	Var	Mean	Var
Translation in X-Axis	1.76	2.15	**1.45**	**1.51**	1.65	1.65
Translation in Y-Axis	**0.69**	**0.55**	1.16	1.43	0.98	1.23
Translation in Z-Axis	**1.21**	**1.61**	2.62	4.47	1.16	1.44
Mean error and variance in degrees	NFC		SIFT		ICP	
	Mean	Var	Mean	Var	Mean	Var
Rotation around X-Axis	3.82	13.06	5.39	28.59	**3.14**	**8.15**
Rotation around Y-Axis	6.03	28.84	10.13	107.33	**4.63**	**19.87**
Rotation around Z-Axis	4.23	18.75	4.66	28.98	**3.04**	**6.95**

5.2.5 Performance Under Spatially Varying Illumination

This set of experiments were designed to measure the performance of the trackers under spatially varying illumination conditions. Three sequences were recorded under spatially varying illumination. During the sequence, moving shadows over the user's face were created by inserting objects between the face and the light source, thus creating very large contrast changes. Also since the shadows are moving, it creates an illusion of movement on intensity images even though the head itself is not moving. Figure 5.14 presents some key frames from sequence 1 of this set. As it can be seen from the images, this is a very challenging sequence. Figure 5.15 shows some tracker output frames for sequence 1. Figure 5.16 presents the plots of the 6 axes over time. As it can be seen from the plots, all of the methods did fail tracking the head successfully and after frame 80, the SIFT algorithm totally lost the target. NFC algorithm fails since moving shadows on the face contradicts with the brightness constancy constraint equation and similarly ICP fails because of the correspondence function which incorporates intensity information into the distance metric. While intensities are changing continuously the correspondence function cannot find good correspondences and the algorithm suffers from drift.

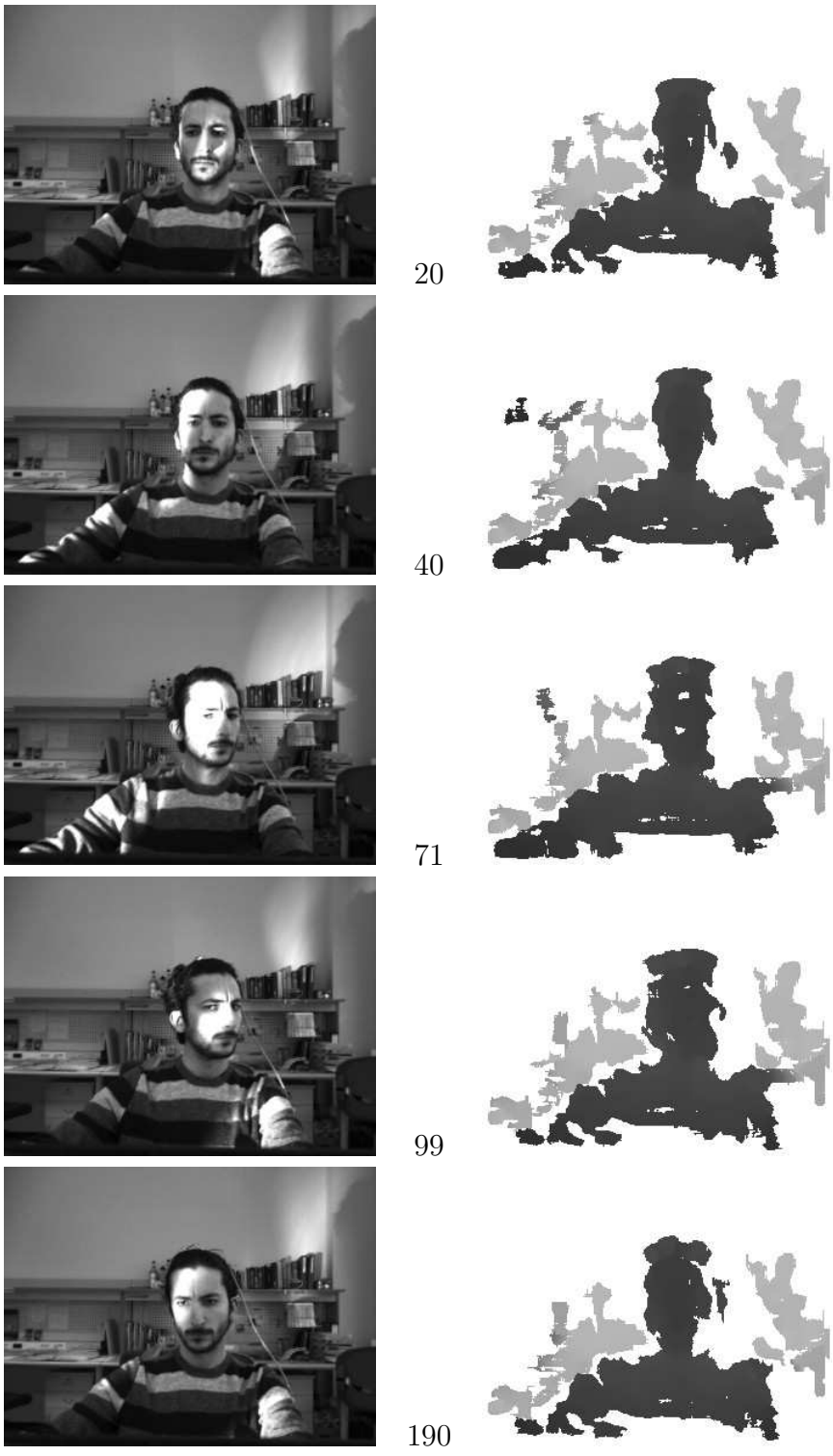

Figure 5.14: Intensity and depth images from spatially varying illumination change sequence 1.

Figure 5.15: Head tracking results for illumination change sequence 1. Each row represents tracking results at different frames: 90, 100, 180, 240, 360

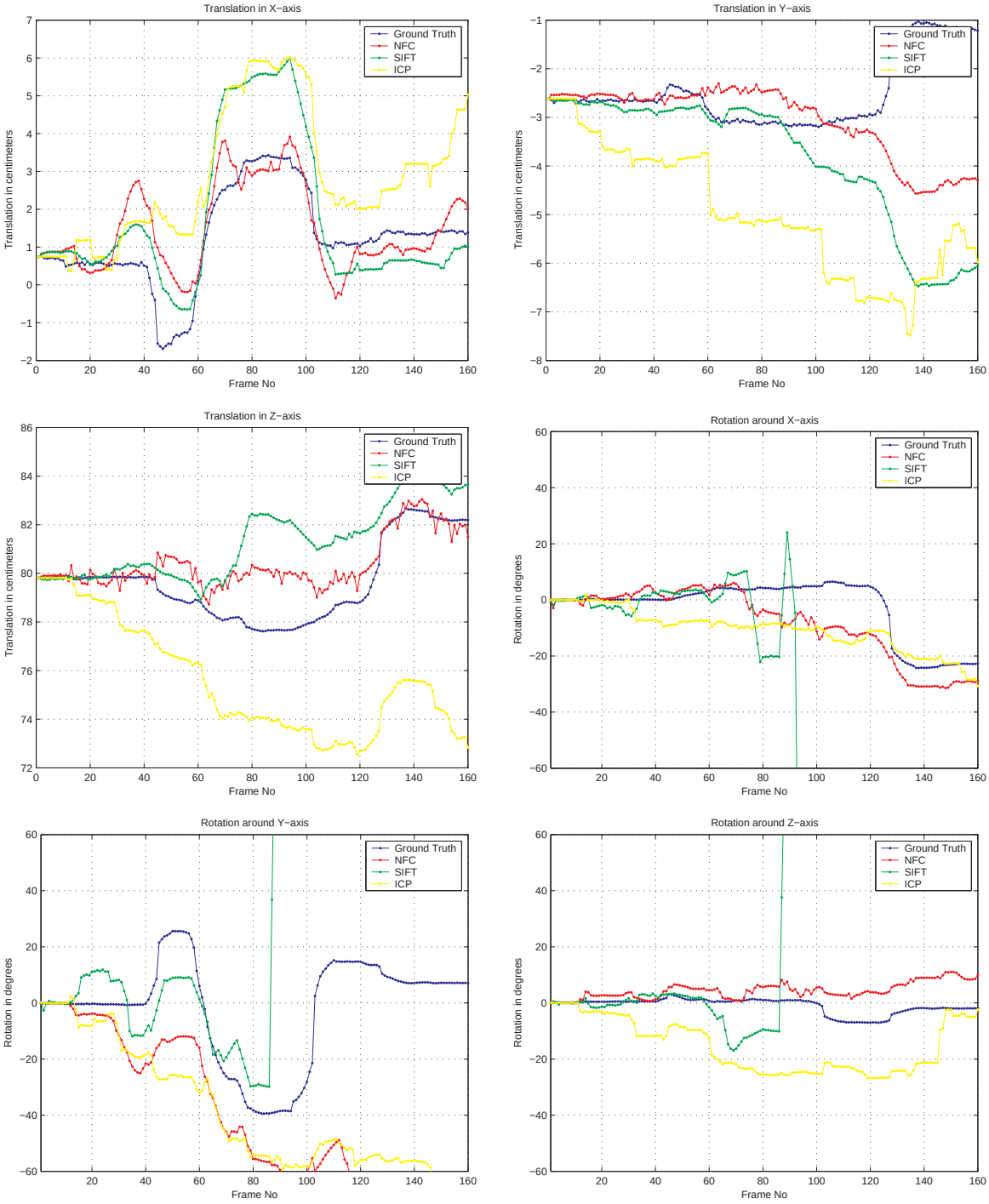

Figure 5.16: Head pose estimation plots for spatial varying illumination change sequence 1

5.2.6 Performance Under Occlusion

This set of experiments were conducted to measure the performance of the trackers under occlusions. Three sequences were recorded with different subjects with partial occlusions involved. During the sequence, the subjects were asked to partially occlude their faces with hands or other objects. Figure 5.17 present some key frames from sequence 1. Figure 5.18 shows some tracker outputs for sequence 1. Figure 5.19, present the estimated translation and rotation parameters during tracking compared to ground truth. Table 5.5 shows the average error and variance over three recorded sequences. As it can be seen in the table, both NFC and SIFT are robust to partial occlusions and both can successfully continue to track. NFC in general performs better than SIFT for occlusion cases.

Table 5.5: Performance results of the trackers over sequences with occlusion.

Mean error and variance in cm	NFC		ICP		SIFT	
	Mean	Var	Mean	Var	Mean	Var
Translation in X-Axis	1.90	0.91	**0.97**	**0.57**	1.35	0.49
Translation in Y-Axis	0.26	0.06	**0.18**	**0.05**	0.35	0.11
Translation in Z-Axis	0.27	0.07	**0.22**	**0.06**	0.34	0.09
Mean error and variance in degrees	NFC		SIFT		ICP	
	Mean	Var	Mean	Var	Mean	Var
Rotation around X-Axis	2.28	3.44	3.17	8.15	**2.27**	**4.68**
Rotation around Y-Axis	**6.58**	**42.02**	15.13	267.87	7.34	53.60
Rotation around Z-Axis	**1.91**	**2.11**	2.14	4.50	3.28	13.34

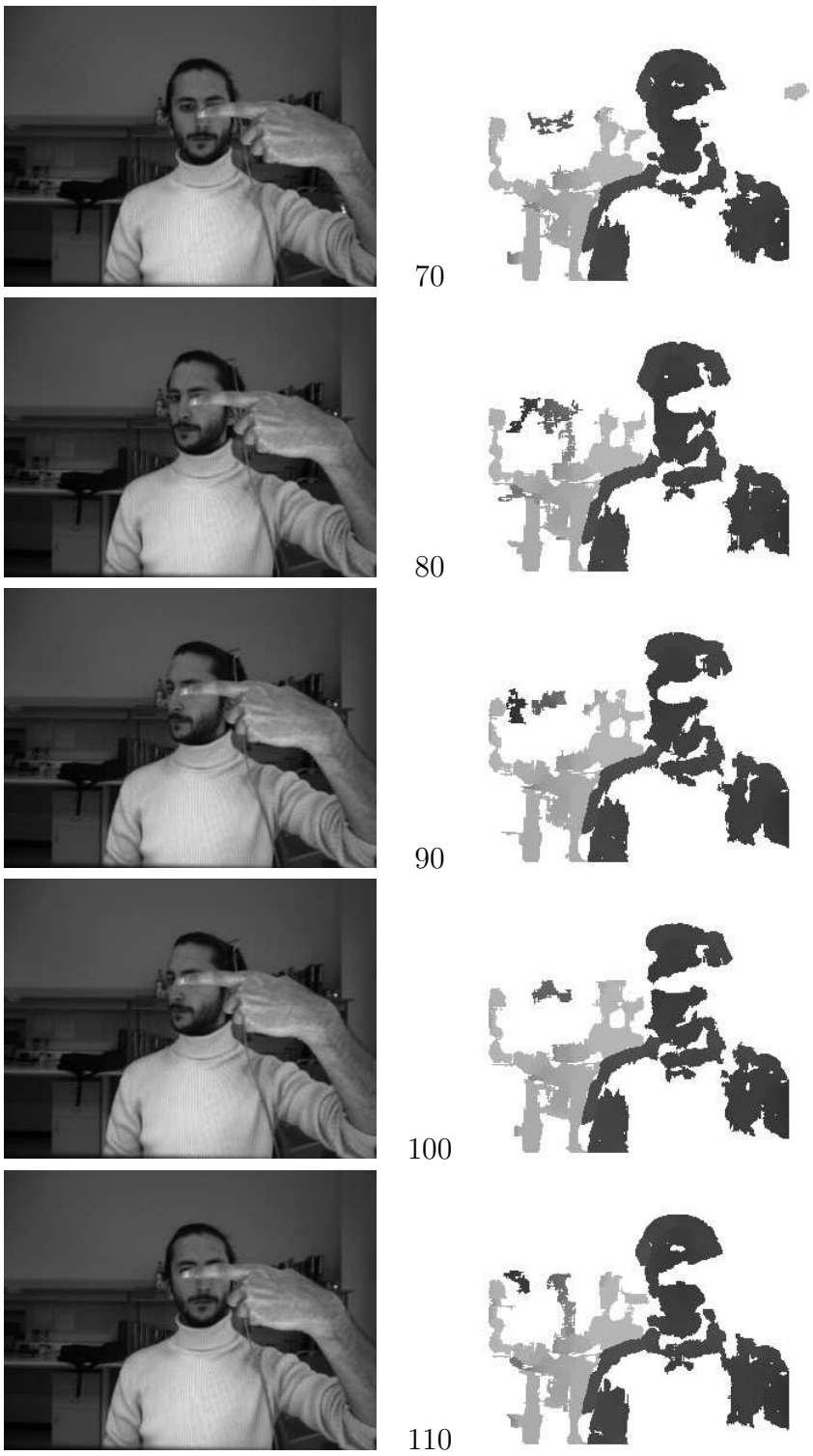

Figure 5.17: Intensity and depth images from sequence 1 including occlusion.

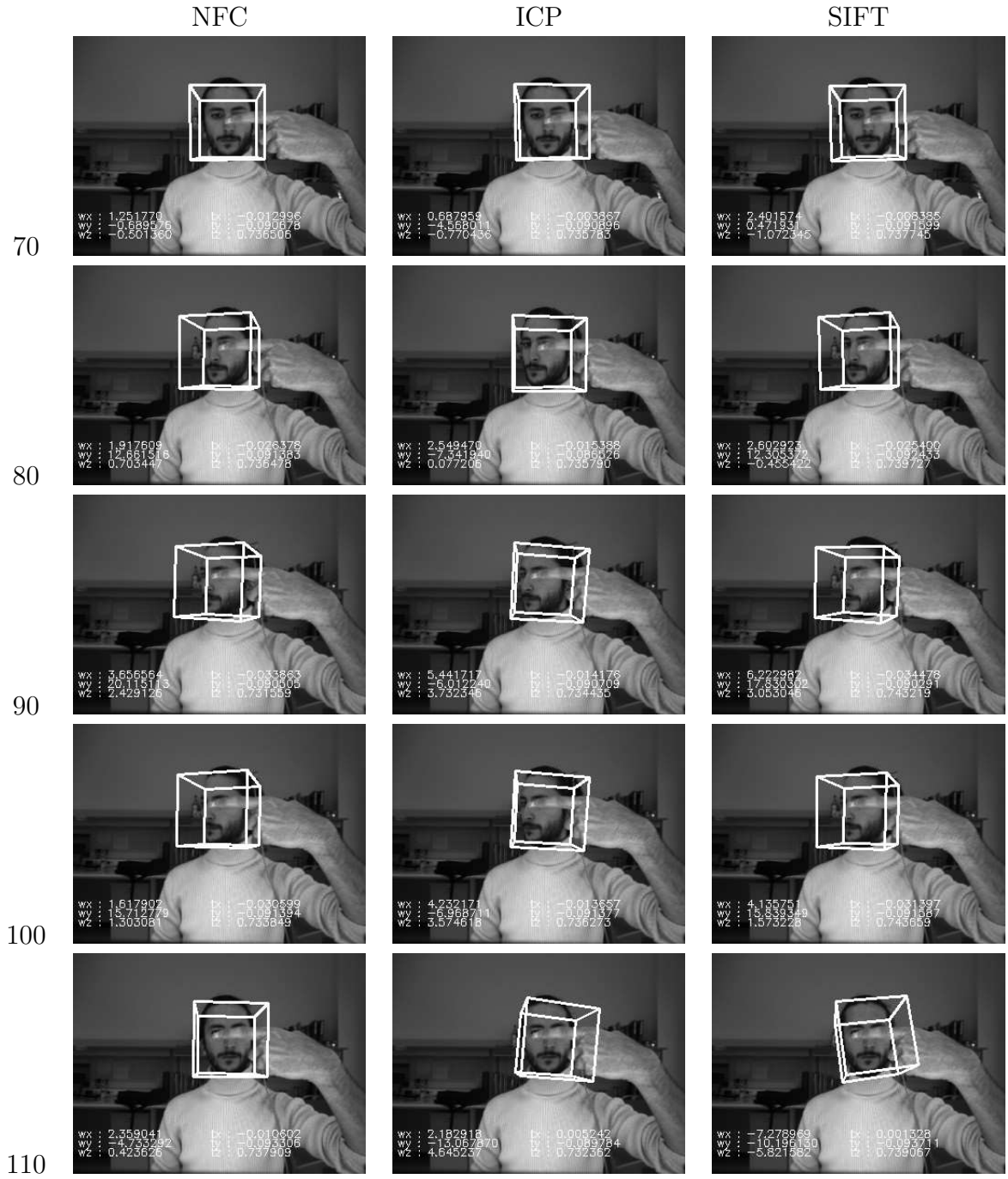

Figure 5.18: Head tracking results for occlusion sequence 1. Each row represents tracking results at different frames: 70, 80, 90, 100, 110

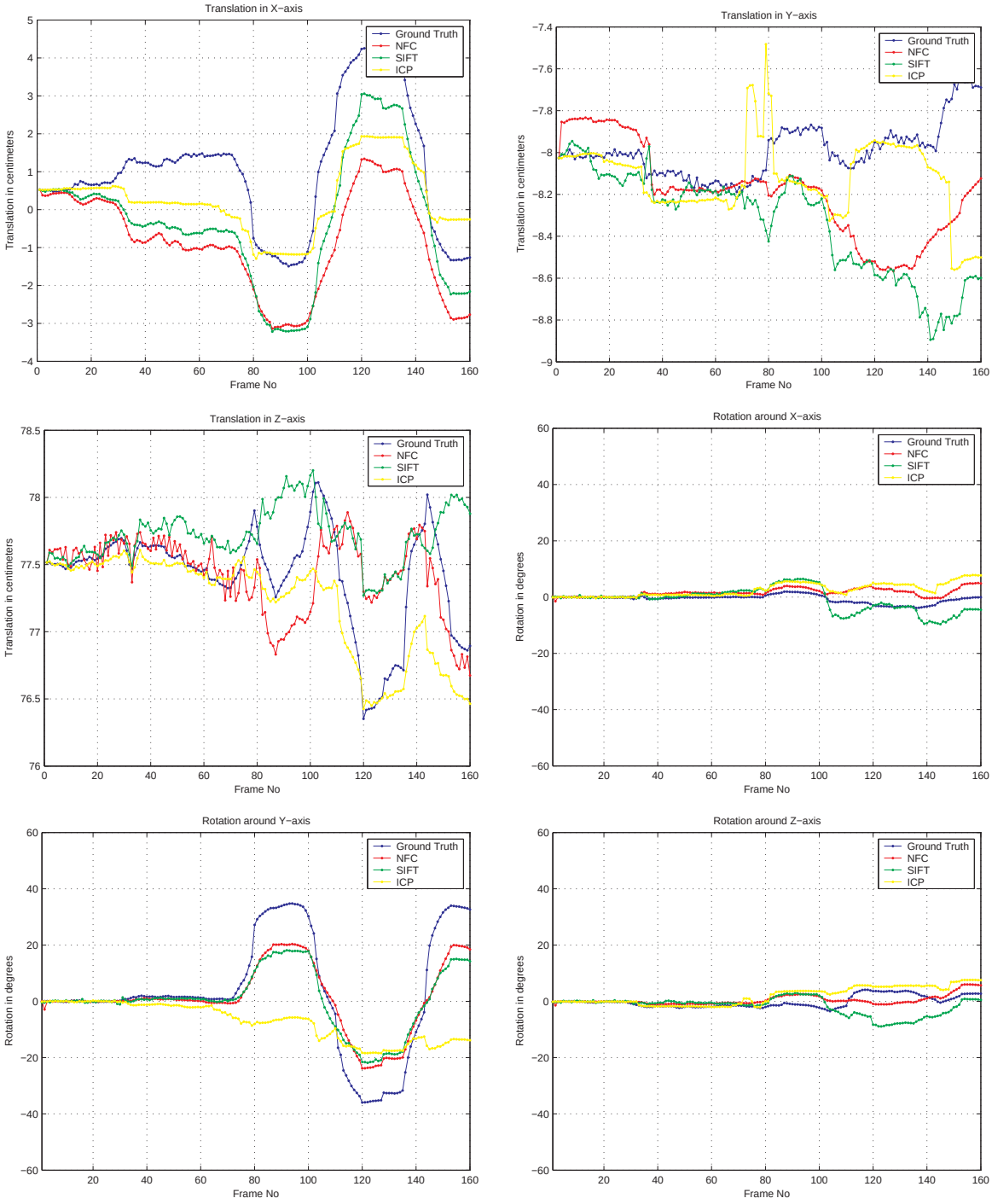

Figure 5.19: Head pose estimation plots for occlusion sequence 1

CHAPTER 6

Summary and Conclusion

In this study we have explored several methods for 3D head tracking using a stereo camera input where disparity information is available together with intensity information. In this thesis we proposed a salient-point based 3D tracking algorithm and reviewed two other methods for rigid body object tracking.

In the first part of the thesis, we explored an optical flow based stereo head tracking system that combines motion information recovered through gradients of both intensity and disparity images. During our tests, we have observed that NFC tracking algorithm is robust to illumination changes and also provides accurate tracking results for rotations, but the tracker can easily lose the target in translations. In the second part, we reviewed a surface registration based algorithm, namely iterative closest point, applied to head tracking on stereo disparity data. Experiments showed that the ICP algorithm was not well suited for tracking rotations, and we also observed that it was not very successful for tracking translations.

In the third part we proposed a 3D tracking algorithm that combines SIFT features with disparity images to estimate the differential head pose. Our proposed algorithm performs as well as NFC algorithm in rotations and outperforms both NFC and ICP alone in translations. Since SIFT features are invariant to linear illumination changes, our algorithm is also not affected by sudden changes in illumination.

6.1 Future Work

Our technique can be improved on several fronts. On initialization part we assumed that the initial pose of the head is aligned with the camera, but this may not always be the case; a precise head pose detection algorithm can be used to initialize the tracker system.

Also since all the trackers we reviewed or proposed are differential trackers, they compute motion only relative to the previous frame. An error made in an arbitrary frame is carried throughout the sequence and these accumulating errors can cause an unbounded drift. A probabilistic framework can be used to estimate the pose changes not only relative to the previous frame, but to several other past frames.

During our experiments we have observed that the NFC algorithm is better suited for sub-pixel movements and rotations and SIFT is robust to coarse translations and rotations. A hybrid approach, combining the outputs of NFC and SIFT based on a confidence function, will result in a more accurate tracking system.

Extracting the SIFT keypoints is a computationally heavy process and results in low rates of frames per second. However exploiting the processing power of the graphics card and the graphic processing unit (GPU), real-time extraction of the SIFT features is possible.

REFERENCES

[1] C. Schütz. Geometric point matching of free-form 3D objects. PhD thesis, DE LUNIVERSITE DE NEUCHATEL POUR LOBTENTION, 1999.

[2] L.P. Morency, A. Rahimi, N. Checka, and T. Darrell. Fast stereo-based head tracking for interactive environments. In Proceedings of Conference on Automatic Face and Gesture Recognition, 2002.

[3] C.R. Wren, A. Azarbayejani, T.J. Darrell, and A.P. Pentland. Real-time tracking of the human body. PAMI, pages 780–785, July 1997.

[4] R. Kjeldsen. Head gestures for computer control. In Second International Workshop on Recognition Analysis and Tracking of Faces and Gestures in Real-Time Systems, pages 62–67, 2001.

[5] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. PAMI, pages 1025–1039, October 1998.

[6] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. Found. Trends. Comput. Graph. Vis., 1(1):1–89, 2005.

[7] M. La Cascia, S. Sclaro, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3d models. PAMI, pages 322–336, April 2000.

[8] M.J. Black and Y. Yacoob. Tracking and recognizing rigidand non-rigidfacial motions using local parametric models of image motion. In International Conference on Computer Vision, pages 374–381, 1995.

[9] S.B., I.E., and A. Pentland. Motion regularization for model-based head tracking. In ICPR96, page C8A.3, August 1996.

[10] A. Schodl, A. Haro, and I. Essa. Head tracking using a textured polygonal model. PUI, 1998.

[11] L. Wiskott, J.M. Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. PAMI, pages 775–779, July 1997.

[12] M. Malciu and F. Preteux. A robust model-based approach for 3d head tracking in video sequences. In Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pages 169–174, 2000.

[13] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In SIGGRAPH99, pages 187–194, 1999.

[14] T.F Cootes, G.J Edwards, and C.J. Taylor. Active appearance model. PAMI, pages 681–684, June 2001.

[15] "M. Harville, A. Rahimi, T. Darrell, G. Gordon, and J. Woodfill". "3d pose tracking with linear depth and brightness constraints". In "International Conference on Computer Vision", pages "206–213", "1999".

[16] B.K.P. Horn and V.G. Schunck. Determining optical flow. 1981.

[17] S. Vedula, S. Baker, R. Collins, T. Kanade, and P. Rander. Three-dimensional scene flow. In International Conference on Computer Vision '99: Proceedings of the International Conference on Computer Vision-Volume 2, page 722, Washington, DC, USA, 1999. IEEE Computer Society.

[18] C. Chen, Y. Hung, and J. Cheng. RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(11):1229–1234, 1999.

[19] P.J. Besl and N.D. McKay. A method for registration of 3d shapes. IEEE Trans. Pattern Anal. Mach. Intell., 14(2):239–256, 1992.

[20] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America, 4:629, April 1987.

[21] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling, pages 145–152, 2001.

[22] G. Godin, M. Rioux, and R. Baribeau. Three-dimensional registration using range and intensity information. SPIE Videometric III, 2350:279–290, 1994.

[23] D.A. Simon. Fast and Accurate Shape-Based Registration. PhD thesis, Carnegie Mellon University, 12 1996.

[24] "D. Lowe". "distinctive image features from scale-invariant keypoints". In "International Journal of Computer Vision", volume "20", pages "91–110", "2003".

[25] M. Fischler and R.Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications ACM, 24(6):381–395, 1981.

[26] R.Haralick, D.Lee, K.Ottenburg, and M.Nolle. Analysis and solutions of the three point perspective pose estimation problem. In Conference on Computer Vision and Pattern Recognition, pages 592–598, June 1991.

[27] L. Quan and Z. Lan. Linear n-point camera pose determination. PAMIe, 21, July 1999.

[28] L.P. Morency. Stereo-based head pose tracking using iterative closest point and normal flow constraint. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, June 2002.

[29] V.Moron, P. Boulanger, T. Masuda, and T. Redarce. Automatic inspection of industrial parts using 3-d optical range sensor. Proc. of SPIE Videometrics, 2598:315–325, 1995.

[30] A. Johnson and S.B. Kang. Registration and integration of textured 3- d data. In International Conference on Recent Advances in 3-D Digital Imaging and Modeling, pages 234–241, Ontario, 1997.

[31] J.H. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Trans. on Mathematical Software, 3(3):209–226, 1977.

[32] Z. Zhang. iterative point matching for registration of free-form curves. Technical Report RR-1658.

[33] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3d objects. Int. J. Rob. Res., 5(3):27–52, 1986.

[34] A.P. Witkin. Scale-space filtering. In International Joint Conference on Artificial Intelligence, pages 1019–1022, 1983.

[35] P. Viola and M. Jones. Robust real-time object detection. International Journal of Computer Vision, 2002.

[36] Point Grey Research. Bumblebee Stereo Camera.

[37] Polhemus Inc. FasTRAK, www.polhemus.com.