Stereo-Based Head Pose Tracking Using Iterative Closest Point and Normal Flow Constraint

Louis-Philippe Morency

AI Technical Report 2003-006 May 2003

Stereo-based Head Pose Tracking using Iterative Closest Point and Normal Flow Constraint

by

Louis-Philippe Morency

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Certified by: Trevor J. Darrell
Assistant Professor
Thesis Supervisor

Accepted by: Arthur C. Smith
Chairman, Department Committee on Graduate Students

Stereo-based Head Pose Tracking using Iterative Closest Point and Normal Flow Constraint

by

Louis-Philippe Morency

Submitted to the Department of Electrical Engineering and Computer Science on May 24, 2002, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

Abstract

In this text, we present two stereo-based head tracking techniques along with a fast 3D model acquisition system. The first tracking technique is a robust implementation of stereo-based head tracking designed for interactive environments with uncontrolled lighting. We integrate fast face detection and drift reduction algorithms with a gradient-based stereo rigid motion tracking technique. Our system can automatically segment and track a user's head under large rotation and illumination variations. Precision and usability of this approach are compared with previous tracking methods for cursor control and target selection in both desktop and interactive room environments.

The second tracking technique is designed to improve the robustness of head pose tracking for fast movements. Our iterative hybrid tracker combines constraints from the ICP (Iterative Closest Point) algorithm and normal flow constraint. This new technique is more precise for small movements and noisy depth than ICP alone, and more robust for large movements than the normal flow constraint alone. We present experiments which test the accuracy of our approach on sequences of real and synthetic stereo images.

The 3D model acquisition system we present quickly aligns intensity and depth images, and reconstructs a textured 3D mesh. 3D views are registered with shape alignment based on our iterative hybrid tracker. We reconstruct the 3D model using a new Cubic Ray Projection merging algorithm which takes advantage of a novel data structure: the linked voxel space. We present experiments to test the accuracy of our approach on 3D face modelling using real-time stereo images.

Thesis Supervisor: Trevor J. Darrell
Title: Assistant Professor

Acknowledgments

First and foremost, I would like to thank my research advisor, Trevor Darrell, who with his infinite wisdom has always been there to guide me through my research. You instilled in me the motivation to accomplish great research and I thank you for that.

I would like to thank my treasured ”officemate” and friend Neal Checka. You always been there for me and I appreciate it greatly. I'm really glad to be your friend.

I would like to thank David Demirdjian for your permanent *bonne humeur*. It was really great work with you and I hope you will stay longer with us.

I would like to thank my accomplice Ali Rahimi. I will remember the many nights we spent together writing papers and listening to Joe Dassin and Goldorak.

I would like to thank Marc Parizeau, my undergraduate teacher from Laval University, Quebec city, who first introduced me to research and shared with me his passion for computer vision.

Thanks to Jean-Francois Roberge who first introduced me 15 years ago to object oriented programming and encouraged me since then to continue my studying in computer sciences.

Merci chere maman pour tout ce que tu as fait pour moi. Durant plus de 20 annnees (et encore aujourd'hui), tu m'as donne ton amour et ta sagesse. Tu m'as appris a partager et a respecter les autres. Je ne pourrai te le dire assez souvent: Merci maman.

Merci cher papa de m'avoir encourage a toujours aller plus loin. C'est grace a toi si j'ai fait ma demande a MIT. Durant ma derniere annee a Quebec, j'ai decouvert ce que la complicite pere-fils signifiait. Merci pour tout ce que tu as fait pour moi.

And finally, I would like to thank the one I love, Tracy Anne Hammond Zeigler, ma chere Titi. Even when I was grumpy or tired, you always took the time to listen me. I shared with you the most beautiful moment of my life and I'm expecting a lot more to come. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Head pose or gaze is a potentially powerful and intuitive pointing cue if it can be obtained accurately and non-invasively. In interactive environments, like 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. In addition, this technology can be important as a hands-free mouse substitute for users with disabilities or for control of gaming environments.

When interacting directly with users, robustness and efficiency are key requirements for a successful system. Interactive environments often include dynamic video projection across multiple large screens, and thus have illumination levels which can change spontaneously. A head tracking system for such environments must be able to handle variations of illumination and large head rotations. In addition, the system should be fast enough to maintain transparent interaction with the user.

Head pose tracking can also be used for building textured 3D models by stitching together synchronized range and intensity frames from stereo cameras. The head pose tracking performs the first step of 3D model acquisition by registering frames to recover their relative positions in the real world. A registration step is necessary because the shape of most objects cannot be observed from only one view: we must scan the object from several directions and bring these scans into registration.

Because frames can rarely be brought into exact registration, a second step, the merging phase, is required to resolve these conflicts by forcing points to lie on a 2D manifold. Range from real-time stereo provides the basis for a modelling tool that is small and hand-held, requires one-time only calibration, is completely passive, produces almost instant 3D models, and provides real-time feedback as the model is being acquired.

In this text, we present two different techniques for head pose tracking using stereo cameras. The first technique is based on the rigid stereo motion tracking technique proposed by Harville *et al.* [17] called ZBCCE which combines the Normal Flow Constraint (NFC) (also called Brightness Change Constraint Equation (BCCE)) with a Depth Constant Constraint Equation (DCCE). This intensity- and depth-based technique is relatively insensitive to illumination variation. Since it is based on real-time 3D observations, it can be more accurate than previous approaches that presumed approximate models. The complete tracking system relies on an online drift reduction algorithm based on Rahimi *et al.*[33] and an automatic initialization technique using fast face detection [43].

The performance of the ZBCCE tracking system was evaluated on a shape tracing task and a selection task. We compared this tracker performance with published reports and side-by-side implementations of two other systems. We evaluated tracing accuracy with small and large head rotations and with different levels of lighting variation. We also compared the performance of the ZBCCE tracker with that of a head-mounted inertial sensor. Results from this user study showed that the ZBCCE tracking system is accurate and fast but does not handle fast movement.

The second tracking technique is an iterative hybrid tracker designed to improve the robustness of our head pose tracker for fast movements. Our new tracking approach jointly aligns images using a Normal Flow gradient Constraint (NFC) and an Iterative Closest Point (ICP) algorithm. This new framework has the precision of the former with the robustness of the latter. Our implementation of ICP finds correspondences between two 3D point clouds using a 4-dimensional search space (3D euclidian space + 1D color/brightness) and minimizes the distance between each 3D point and the tangential plane of its corresponding point. This framework does not include DCCE since the point-to-plane distance used in ICP is a function of the normal vector which is computed using the depth gradient (same as DCCE).

To date, most ICP algorithms have been tested on very precise 3D data sets from a laser scanners [32] or other range scanning methods. We are interested in tracking data from relatively noisy optical stereo range data captured at modest frame rates. To evaluate this new hybrid tracking technique, we performed head pose tracking experiments with real image sequences. We compared our iterative hybrid tracker with each individual tracking techniques: ICP and the normal flow constraint.

Finally, we present an efficient solution for 3D model acquisition using the new iterative head pose tracker to register 3D views. The outcome of the registration phase is a 3D mesh transformed to a canonical pose where each vertex corresponds to a valid image pixel. Due to noise in the imager and imperfect registration, the vertices will not lie on a 2D manifold, but will instead form a fuzzy cloud around the desired surface. To solve this problem, we introduce a new merging algorithm, called Cubic Ray Projection, which non-rigidly deforms each mesh so that vertices are forced toward a 2D manifold. To facilitate the creation of connected meshes from unstructured range data, we use a linked voxel space during the merging process. The linked voxel space is easily turned into a connected mesh for rendering. The system presented is extremely fast and when used with a real-time stereo camera, it is possible to capture 3D models interactively and unobtrusively. Many 3D views are merged together, reducing noise in the final model.

1.1 Related Work

Several authors have recently proposed face tracking for pointer or scrolling control and have reported successful user studies [41, 26]. In contrast to eye gaze [46], users seem to be able to maintain fine motor control of head gaze at or below the level needed to make fine pointing gestures[1]. However, performance of the systems reported to date has been relatively coarse and many systems required users to manually initialize or reset tracking. They are generally unable to accurately track large rotations under rapid illumination variation (but see [27]), which are common in interactive environments (and airplane/automotive cockpits).

Many techniques have been proposed for tracking a user's head based on passive visual observation. To be useful for interactive environments, tracking performance must be accurate enough to localize a desired region, robust enough to ignore illumination and scene variation, and fast enough to serve as an interactive controller. Examples of 2-D approaches to face tracking include color-based [45], template-based [26] and eigenface-based [16] techniques. Techniques using 3-D models have greater potential for accurate tracking but require knowl-

[1]Involuntary microsaccades are known to limit the accuracy of eye-gaze based tracking[25].

edge of the shape of the face. Early work presumed simple shape models (e.g., planar[4], ellipsoidal[2], or cylindrical[27]). Tracking can also be performed with a 3-D face texture mesh [36] or 3-D face feature mesh [44].

Very accurate shape models are possible using the active appearance model methodology [7], such as was applied to 3-D head data in [5]. However, tracking 3-D active appearance models with monocular intensity images is currently a time-consuming process, and requires that the trained model be general enough to include the class of tracked users.

The problem of estimating 3D rigid body motion has been studied extensively in the computer vision and graphics fields. The well-known Iterative Closest Point (ICP) algorithm, introduced by Chen and Medioni [6] and Besl and McKay [3], has been used extensively in the graphics literature to merge 3D laser range scans. In the vision literature much progress has been made on gradient-based parametric motion estimation techniques which aggregate pointwise normal flow constraints [4, 20, 24].

ICP finds corresponding points between two 3D point clouds and tries to minimize the error (usually the euclidian distance) between the matched points. Chen and Medioni minimize this error based on a point-to-plane distance, while Besl and McKay minimize the direct euclidian distance between the matched points (point-to-point). Rusinkiewicz and Levoy [34] present a extensive survey of many variants of ICP. Godin *et al.*[13] first used color to filter matched points during ICP. While other methods [11, 37] have incorporated color information in the distance function of the matching process, no solution has been suggested that uses color/brightness during the error minimization process.

The normal flow is 3D vector field which can be defined as the component of the 2D optical flow that is in the direction of the image gradient[42]. When 3D observations are directly available, such as from optical stereo or laser range finders, a normal flow constraint can be expressed directly to estimate rigid body motion [39]. Harville *et al.*[17] combined normal flow constraint with a depth gradient constraints to track rigid motion. Gradient-based approaches use color/brightness information during the minimization process and have proved to be accurate for sub-pixel movements[1].

Many algorithms have been proposed for registering range data. These differ notably in the energy function minimized during registration, and whether the registration procedure ensures global consistency.

The method of Stoddart and Hilton [40] minimizes a function corresponding to the energy stored in a spring system connecting corresponding points across frames. This algorithm provides global consistency, but requires correspondences to be known.

The registration algorithm of [15] brings each point of a scan as close as possible to its closest point on the model acquired so far, thus avoiding the need for correspondences. However, this method does not produce a globally consistent model. Accumulated registration errors against the model eventually cause the model to become inconsistent (see [35] for a discussion).

The Iterative Closest Point (ICP) framework proposed by Besl and McKay [3] iteratively assigns correspondences and then minimizes the resulting distance metric by rigidly transforming the scans [32, 6]. Chen and Medioni [6] employ this technique to minimize the distance between each point of a scan and the closest tangent plane in the corresponding scan. They perform this minimization jointly over the pose of all scans. Because each iteration must involve all pairs of corresponding points, the optimization is expensive.

To reduce the complexity of this minimization, Pulli [32] first aligns scans pairwise, obtaining relative pose estimates between many redundant pairs of scans. Global consistency is obtained by assigning each frame a pose such that the pairwise relative alignments are minimally perturbed. This optimization is fast as it does not require correspondences to be recomputed at each iteration of the optimization and only matches up frame poses poses instead of individual points.

In contrast to these head tracking systems, our system is robust to strong illumination changes, automatically initializes without user intervention, and can re-initialize automatically if tracking is lost (which is rare). In addition, it can track head pose under large rotations and does not suffer from drift.

Our approach uses the combined depth and intensity constraint of [17] to obtain relative pose changes between each frame and several other base frames. The pose changes describe the rigid transformation required for bringing each frame into registration with its base frames. The global registration method we present is based on [33] and is similar in structure to [32] in that, during global registration, poses are relaxed to find a registration which is consistent with the measured pairwise pose changes.

1.2 Organization of Thesis

The following chapter describes the main components of our ZBCCE head pose tracking system. We review the pose change estimation algorithm of [17] and the global pose consistency algorithm of [33].

In chapter 3, we present our experimental paradigm and interaction task. We evaluate the spatial accuracy and temporal resolution of the ZBCCE head pose tracker, compare it to previously reported systems, and conclude with a discussion of these results.

Chapter 4 describes the iterative framework used for 3D view registration. Section 4.3 presents the closest point matching process and point-to-plane error function, two important components of ICP. Section 4.4 reviews the normal flow constraint and shows how inverse calibration parameters can be used to do find correspondence. Then, section 4.5 describes the hybrid error functions.

Chapter 5 present results that show how our iterative hybrid tracker can reliably track sequences from optical stereo data that neither technique alone could track.

Chapter 6 describes our novel 3D model reconstruction algorithm called Cubic Ray Projection, which is applied after frames have been globally registered using our iterative hybrid tracker. We then show how our system can be used to build 3D models of human heads.

Chapter 2

Stereo-based Head Pose Tracker using ZBCCE

Our ZBCCE head pose tracking system has three main components. Its core is an algorithm for instantaneous depth and brightness gradient tracking called ZBCCE [17], combined with two other modules for initialization, and stabilization/error-correction. For initialization we use a fast face detection scheme to detect when a user is in a frontal pose, using the system reported in [43]. To minimize the accumulation of error when tracking in a closed environment, we rely on a scheme which can perform tracking relative to multiple base frames [33].

The following subsections describe the initialization and basic differential tracking algorithm which recovers the rotation and translation of an object between two time steps t and r, given images sets $\{I_t, Z_t\}$ and $\{I_r, Z_r\}$. The last subsection explains how to use multiple base frames to reduce drift.

2.1 Initialization with Face Detection

When it first comes online, the tracker scans the image for regions which it identifies as a face using the face detector of [43]. As soon a face has been consistently located near the same area for several frames, the tracker switches to tracking mode. The face detector is sensitive only to completely frontal heads, making it possible for the tracker to assume that the initial rotation of the head is aligned with the camera coordinate system. The face detector provides the tracker an initial region of interest, which is updated by the tracker as the subject moves around. Since depth information is readily available from the stereo camera, the initial pose parameters of the head can be fully determined by 2D region of the face with the depth from stereo processing.

When we observe erratic translations or rotations from the tracker, the tracker automatically reinitializes by reverting to face detection mode until a new target is found. This occurs when there is occlusion or rapid appearance changes.

2.2 Pose Parameters

Our tracker process two image sets as input: the new image set $\{I_t, Z_t\}$ grabbed at time t and the reference image set $\{I_r, Z_r\}$. The reference image set can be either the image set grabbed at time t-1, the first image set, or any relevant image set between time 0 and time t-1 [33] (see section 2.4).

The goal of the tracker is to find the rigid pose change $\{\mathbf{R}, \vec{\mathbf{t}}\}$ between the two image sets, where $\mathbf{R}$ is a 3x3 rotation matrix and $\vec{t}$ is a 3D translation vector . A transformation $\vec{\delta}$ represented by 6 parameters vector $[\ \vec{\omega}\quad \vec{t}\ ]^t$ is computed. In this vector, $\vec{\omega}$ is the instantaneous rotation (3 parameters) and $\vec{t}$ is the translation (3 parameters). The current pose estimation is updated as follow:

$$\mathbf{R^{new}} = \mathbf{R^{old}}\mathbf{R}^{(\delta)} \tag{2.1}$$

$$\vec{t}^{new} = \vec{t}^{old} + \vec{t}^{\,(\delta)} \tag{2.2}$$

where $\mathbf{R}^{(\delta)}$ is the 3x3 matrix representing the rotation $\omega^{(\delta)}$. Initially, $\mathbf{R^0}$ is set to the identity matrix and $\vec{t}^{\,0}$ is set to 0.

2.3 ZBCCE Differential Tracking

To recover the motion between two frames, we apply the traditional Normal Flow Constraint (NFC) [19] (also called Brightness Change Constraint Equation (BCCE)) jointly with the Depth Change Constraint Equation (DCCE) of [17] on range and intensity imagery of stereo camera. As shown in [42], the NFC can be expressed as:

$$-\frac{\partial I_{ri}}{\partial t} = \nabla I_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \vec{V} \tag{2.3}$$

where $\vec{V} = \left[\ \frac{\partial x_{ri}}{\partial t}\quad \frac{\partial y_{ri}}{\partial t}\quad \frac{\partial z_{ri}}{\partial t}\ \right]$ is the velocity of the object, $\nabla I_{ri} = \left[\ \frac{\partial I_{ri}}{\partial u_{ri}}\quad \frac{\partial I_{ri}}{\partial v_{ri}}\ \right]$ is the image gradient, and $\frac{\partial I_{ri}}{\partial t}$ is the time gradient.

$\frac{\partial I_{ri}}{\partial u_{ri}}$ and $\frac{\partial I_{ri}}{\partial v_{ri}}$ are computed directly from the referential image I_r. The time gradient is approximated by:

$$\frac{\partial I_{ri}}{\partial t} = I_{ti} - I_{ri} \tag{2.4}$$

For a perspective projection where $u_{ri} = f\frac{x_{ri}}{z_{ri}}$ and $v_{ri} = f\frac{y_{ri}}{z_{ri}}$, we can find the Jacobian matrix:

$$\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}} = \begin{bmatrix} \frac{f}{z_{ri}} & 0 & -f\frac{x_{ri}}{z_{ri}^2} \\ 0 & \frac{f}{z_{ri}} & -f\frac{y_{ri}}{z_{ri}^2} \end{bmatrix} \tag{2.5}$$

Since the object is rigid, the velocity V can be expressed as:

$$\vec{V} = \begin{bmatrix} \mathbf{I} & -\hat{q}_{ri} \end{bmatrix} \vec{\delta} \tag{2.6}$$

where $\mathbf{I}$ is a 3x3 identity matrix and $\hat{q}_{ri}$ is the skew matrix of the vector $\vec{q}_{ri}$. By rearranging the equation, we get a linear system:

$$\varepsilon_{NFC} = \|\mathbf{A_{NFC}}\vec{\delta} - \vec{b}_{NFC}\|^2 \tag{2.7}$$

where each line is defined as follow

$$\vec{A}_i = \nabla I_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \begin{bmatrix} \mathbf{I} & -\hat{q}_{ri} \end{bmatrix} \tag{2.8}$$

$$b_i = -\frac{\partial I_{ri}}{\partial t} \tag{2.9}$$

The DCCE of [17] uses the same functional form as equation (2.3) to constrain changes in depth. But since depth is not preserved under rotation, the DCCE includes an adjustment term:

$$-\frac{\partial Z_{ri}}{\partial t} = \nabla Z_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \vec{V} - V_z \tag{2.10}$$

where $\nabla Z_{ri} = \begin{bmatrix} \frac{\partial Z_{ri}}{\partial u_{ri}} & \frac{\partial Z_{ri}}{\partial v_{ri}} \end{bmatrix}$ is the depth gradient and $V_z = \frac{\partial z_{ri}}{\partial t}$ is the flow towards the Z direction induced by δ. By rearranging the equation, we get a linear system similar to NFC:

$$\varepsilon_{DCCE} = \|\mathbf{A_{DCCE}}\vec{\delta} - \vec{b}_{DCCE}\|^2 \tag{2.11}$$

where each line is defined as follow

$$\vec{A}_i = \nabla Z_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \begin{bmatrix} \mathbf{I} & -\hat{q}_{ri} \end{bmatrix} \tag{2.12}$$

$$b_i = -\frac{\partial Z_{ri}}{\partial t} + V_z \tag{2.13}$$

Note that the DCCE is robust to lighting changes since lighting does not affect the depth map. We combine the NFC and DCCE into one function optimization function with a weighted sum:

$$\delta^* = \arg\min_{\delta} \epsilon_{NFC}(\delta) + \lambda\epsilon_{DCCE}(\delta)$$

We can rewrite this equation as one linear system:

$$\arg\min_{\vec{\delta}} \left\| \begin{bmatrix} \mathbf{A_{NFC}} \\ \lambda\mathbf{A_{DCCE}} \end{bmatrix} \vec{\delta} - \begin{bmatrix} \vec{b}_{NFC} \\ \lambda\vec{b}_{DCCE} \end{bmatrix} \right\|^2$$

The only unknown variables are the pose parameters, since Z is available from the depth maps. This linear system can be solved using a least-squares method or any robust estimator. For an approximate way to optimize this function, see [17], where one iteration of Newton-Raphson is shown to be adequate for tracking. To reduce the influence of outliers, we use a M-estimator to minimize the system [21].

2.4 Drift Reduction

Given a method for computing the pose difference δ_s^t between frames I_r and I_t, one approach for estimating the pose ξ_t of frame I_t relative to the first frame I_0 is to accumulate the pose difference between adjacent frames I_r and I_{r+1}, for $r = 0..t-1$. But since each pose change measurement is noisy, the accumulation of these measurements becomes noisier with time, resulting in unbounded drift.

To curb this drift, we compute the pose change between I_t and several base frames. When the trajectory of the target crosses itself, its pose change is computed against recently acquired scans as well as past scans near the current pose. These pose differences are combined to not only obtain a more robust and drift-free pose estimate of the current scan, but also to adjust the pose of past frames by incorporating knowledge about the closed trajectory.

Several authors have proposed an optimization framework to implement this technique [33, 32, 29]. Poses are assigned to each scan so that the predicted pose changes between pairs of scans are as similar as possible to the observed pose changes. Assuming a function $d(\xi_r, \xi_t)$ which returns the pose change between two poses, we wish to minimize for all poses ξ_i:

$$\sum_{(r,t)\in P} \|\delta_r^t - d(\xi_r, \xi_t))\|^2_{\Lambda_r t}$$

where P is the set of frame indices between which pose changes have been computed, and $\|.\|^{\Lambda}$ is the Mahalanobis distance. Poses are parameterized using local rotations so that $d(\xi_r, \xi_t) = \xi_r - \xi_t$. Optimizing (2.4) involves solving a sparse linear system, which can be performed efficiently using conjugate gradient descent, for example. For more details, see [33].

Chapter 3

Cursor Control User Study using ZBCCE

To evaluate the ZBCCE stereo-based head pose tracker we performed two series of experiments, the first in a desktop screen environment and the second in an interactive room with large projection screens. In the following subsection, we describe the tracking systems used in this user study. We then present the experimental setups and results for both experiments.

3.1 Tracking Techniques

We compared side-by-side the stereo motion tracker of section 2 with a 2D tracker based on normalized cross-correlation, and a head-mounted inertial rotation sensor. The following sections describe each tracker in more detail.

3.1.1 Stereo-Motion Tracker

The ZBCCE stereo-motion head tracker is a standalone system which takes video-rate intensity and range imagery from a stereo camera such as the SRI Small Vision System [10] camera and locates and tracks heads in real-time. The SRI camera software produces 320x240 pixel resolution intensity and range images at 15 fps. The tracker runs on a 1.5 Ghz Pentium 4 running a Windows operating system, and takes advantage of Intel's SIMD architecture through the Intel Performance Library. This tracker uses the rigid motion stereo algorithm described

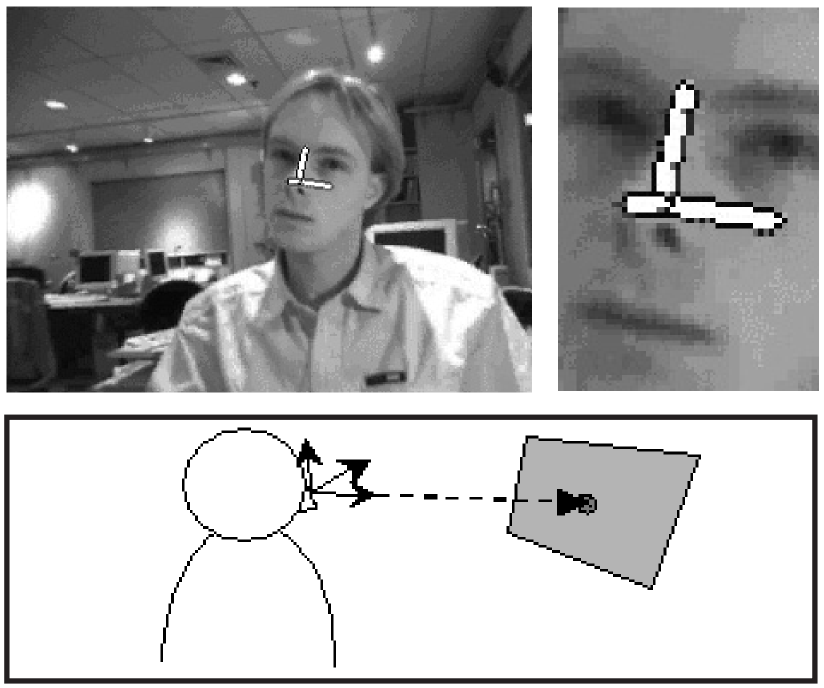


Figure 3.1: Example of pointer control using a head tracking system. In the top-left image, a set of axes are overlaid on the user's face to show the estimated pose. A close-up of the face is shown in the top-right. The intersection of the frontal face ray with a display screen is used to control a pointer, as shown in the lower image.

above, together with face detection and drift reduction (with 2 past frames).

As in [41], we use the tracked head position to infer a point of intersection of a "face ray" with the control or display surface, and use this to set a pointer target (see figure 3.1).

3.1.2 Inertial Rotation Sensor

We used evaluated tracking in comparison to an inertial rotation sensor, using InterSense's Intertrax^2 [22]. The manufacturer reports that it is able to measure changes in rotations in three axes with 0.02 degrees of precision. Our software samples the tracker at about 100 samples per second, though the tracker is reported to have a 256 Hz internal sampling rate; it was attached to the test PC via USB. The documentation for the tracker reports zero jitter, which after some experimentation, we concluded was the result of a hysteretic filter. Based on a patent filed by the manufacturer, the inertial sensor may combine various sources of inertial measurement such as the earth's magnetic and gravitational fields[12].

In contrast to the vision-based tracker of section 2 which automatically tracks after detecting a face, the Intertrax tracker must be manually initialized to provide it with a reference frame. The head-mounted tracker is equipped with a reset button which must be pushed before the user begins each experiment in order to define the initial coordinate system and to reset the accumulated drift.

3.1.3 Normalized Cross-Correlation Tracker

We evaluated 2D tracking techniques to explore the importance of stereo observations for robust real-time tracking. We used a side-by-side implementation of 2D normalized correlation tracking similar to that proposed in [26]. (We also compared published reports of other 2D trackers, as reported below.) The normalized cross-correlation tracker works in two phases, similar to the stereo tracker described above: first, a face detector [43] locates a face and reports a region of interest which represents the bounding box of a face. Second, the correlation tracker takes a snapshot of the resulting region, scales its magnitude to 1, and uses it as the template in its tracking phase[14].

Once a template is acquired, for each new image, the correlation tracker scans a 70 by 30 pixel region around the location where the face was originally found. For each candidate location (u, v), it computes the similarity measure:

$$\epsilon(u, v) = \sum_x \sum_y \|\tilde{I}_t(x + u, y + v) - \tilde{T}(x, y)\|^2, \tag{3.1}$$

where $\tilde{T}$ is the magnitude-normalized face template acquired during detection and $\tilde{I}_t$ is the magnitude-normalized current image.

The correlation tracker reports the value of (u, v) which minimizes (3.1). Typically, this displacement would be scaled by constants in u and v and used as the location of the pointer on the screen. However, because the domain of ϵ is integers and the resolution of the camera is low, the approach is insensitive to small motion. As such, the pointer's precision suffers.

Instead, we resolve the motion (u, v) to sub-pixel resolutions, by approximating the D by D pixel neighborhood around the minimum

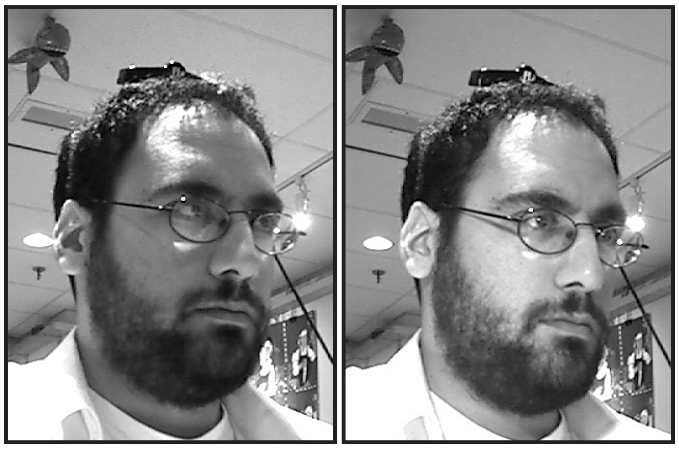


Figure 3.2: Brightness change during the lighting variation experiment. Left: lamp on. Right: lamp off.

of ϵ by a second order polynomial $\hat{\epsilon}$. Then instead of reporting the minimum of $\epsilon(u, v)$, the correlation tracker reports the minimum of $\hat{\epsilon}$.

3.2 Desktop Experiment

The desktop experiment involved 8 experiments per subject. Each subject tested the three tracking techniques described in section 3.1. Each of the trackers was tested in small-screen and wide-screen mode. The former allows the user to trace the rectangle using small head motions. The latter simulates a larger screen which requires larger head rotations to navigate. In addition, the correlation tracker and the stereo motion tracker were tested in the small-screen mode under abruptly varying lighting conditions (see figure 3.2).

As shown in figure 3.3, users sat about 50 cm away from a typical 17" screen, subtended a horizontal angle of about 30 degrees and a vertical angle of about 20 degrees. The screen displayed a black background and a white rectangular path drawn in the middle. The task was to use head pose to move a 2D pointer around the screen to trace the rectangular path as accurately as possible. Users were allowed to take as much time as they liked, as long as they were able to complete the path eventually. Thus, we suggest that the dominant feature under observation is the tracker's accuracy in mapping the user's head to a

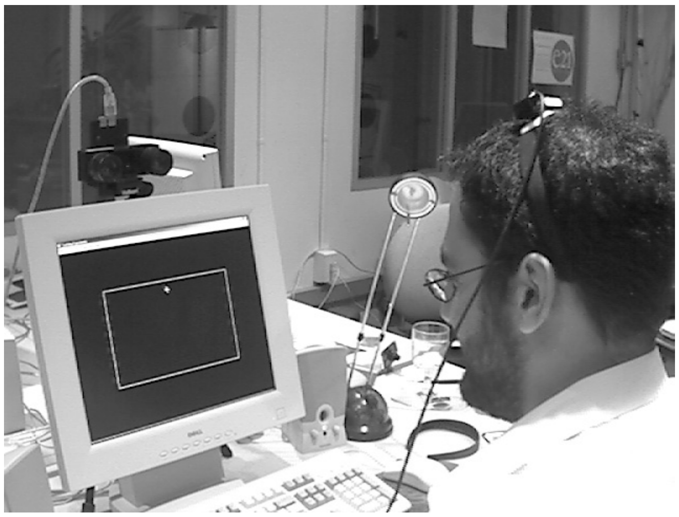


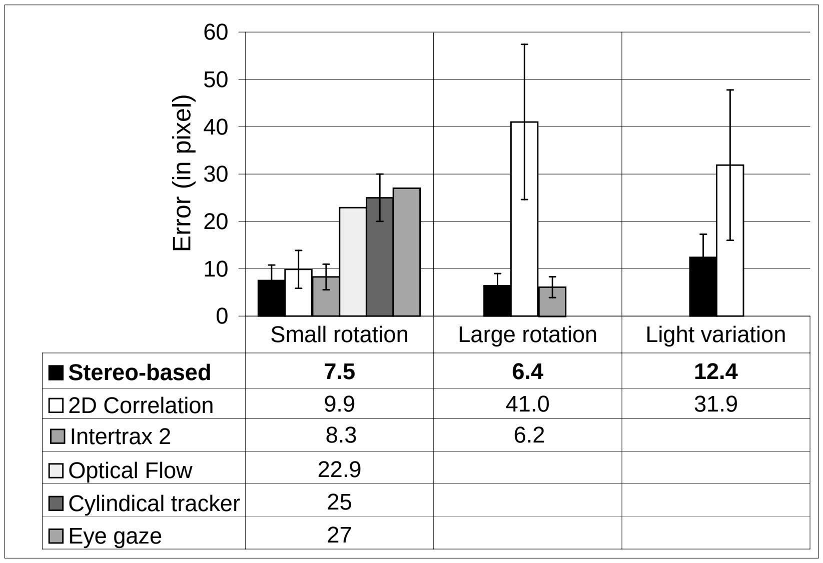
Figure 3.3: A user during the desktop experiment. The SRI stereo camera is placed just over the screen and the user is wearing the Intertrax^2 device on his head.

2D location.

3.2.1 Results

The first three rows of figure 3.4 compares the accuracy of the ZBCCE stereo motion tracker with the 2D normalized cross-correlation tracker and the Intertrax^2 tracker. The histogram shows the average error and standard deviation of 4 subjects. The average error is computed as the average distance in pixels between every point on the cursor trajectory and the closest point on the given rectangular path. The three last rows of the same figure compares our results with some published system: an optical flow tracker[23], cylindrical tracker[27], and an eye gaze tracker[46]. We can observe from figure 3.4 that the stereo-based tracker perform better for large rotation and light variation then the 2D correlation tracker. The stereo-based tracker gives similar accuracy results then the inertial rotation sensor for small and large rotations.

Figure 3.5 shows typical pointer trajectories for each scenario. It took an average of 50 seconds to trace each rectangle. We observe from the trajectories that the stereo-based tracker can be used accurately point on a screen.



	Small rotation	Large rotation	Light variation
■ **Stereo-based**	**7.5**	**6.4**	**12.4**
□ 2D Correlation	9.9	41.0	31.9
▨ Intertrax 2	8.3	6.2	
□ Optical Flow	22.9		
▩ Cylindical tracker	25		
▨ Eye gaze	27		

Figure 3.4: Comparison of average error on tracing task of the desktop experiment. The error bars in the histogram represent the standard deviation between user results.

In a desktop environment, small rotations are sufficient to drive a cursor, since the angle subtended by the screen tends to be small. This situation serves as a baseline where all three trackers can be compared under moderate conditions. Under the small rotation scenario, all trackers showed similar deviation from the given trajectory, with an average deviation of 7.5 pixels for the stereo motion tracker, 9.8 pixels for the normalized cross-correlation tracker, and 8.3 pixels for the inertial tracker. Note that the drift of the inertial sensor becomes significant during the last quarter of its trajectory (figure 3.5), forcing subjects to compensate for its error with exaggerated movements.

Navigating a pointer on a wide screen (multiple monitors, projection screens, cockpits) requires larger head rotations. As expected, the correlation tracker loses track of the subject during rotations beyond 20 degrees, because the tracker is initialized on the appearance of the frontal face only. It incurred an average error of 41.0 pixels. The stereo motion tracker, however, successfully tracks the head as it undergoes large rotations, with an average error of 6.4 pixels. The Intertrax^2 tracker shows an average error of 6.2 pixels. Note that due to

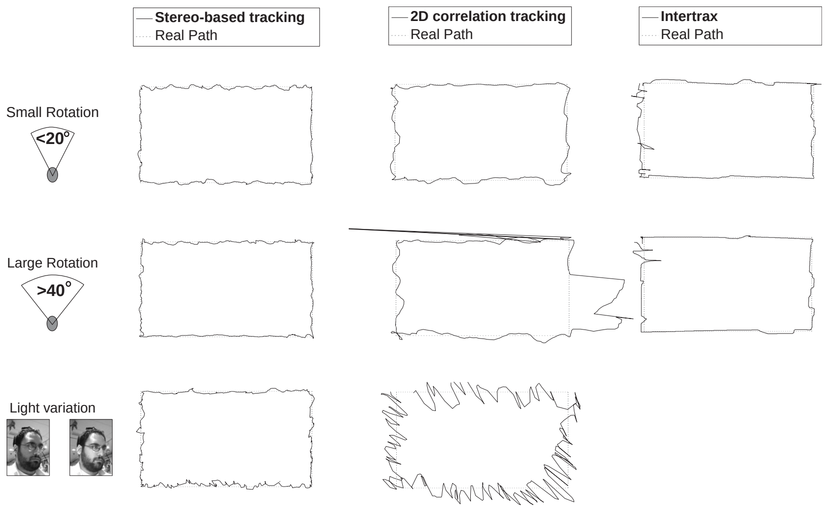


Figure 3.5: Typical trajectories for all three trackers when users perform small rotations (first row), large rotations (second row) and under light variation (last column). The trajectory starts from the upper left corner of the rectangle and ends in the same location.

the accumulated drift of the inertial sensor, typical users had difficulty controlling the cursor in the last portion of the trajectory.

Under varying lighting conditions (the light was modulated at about 1/2 Hz), the normalized cross-correlation tracker lost track of the target regardless of the degree of rotation, yielding an average error of 31.9 pixels as opposed to its 9.9 pixels under unvarying lighting. The stereo motion tracker did suffer slightly, averaging an error rate of 12.4 pixels as opposed to its initial error of 7.5 pixels under normal lighting conditions. This is only a factor 1.6 increase in average error, compared to the correlation tracker's factor of 3.2 loss of performance.

3.2.2 Discussion

The inertial rotation sensor Intertrax^2 is accurate for a short period of time, but it accumulates noticeable drift. Approximately after 1 minute of use of the tracker, subjects were often forced to contort their bodies significantly in order to compensate for the drift. The normalized cross-correlation tracker appears to be suitable for situations involving small head rotations and minimal illumination changes.

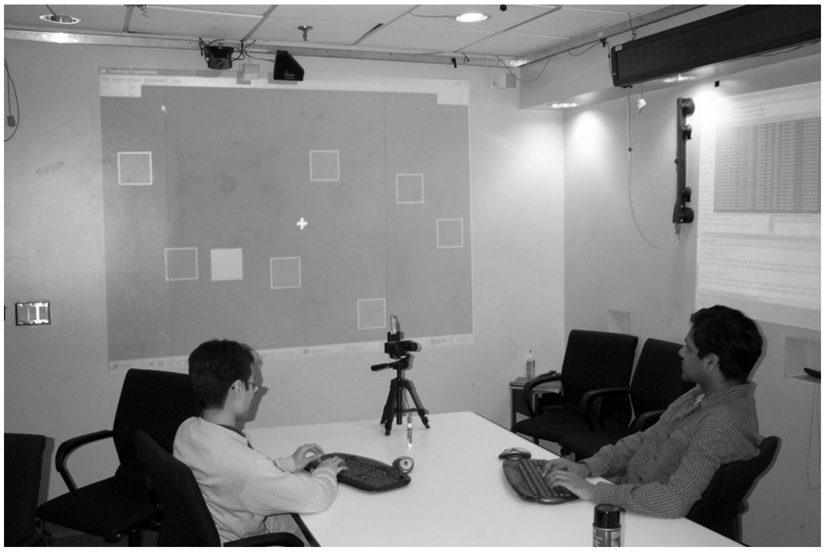


Figure 3.6: Setup for the room experiment. The SRI stereo camera is placed on the table.

The stereo motion tracker is robust to lighting variations because it largely relies on depth information, which is unaffected by the illumination changes. In addition, it can track arbitrarily large transformations without suffering from drift due to the drift reduction algorithm described in section 2.4.

3.3 Interactive Room Experiment

As shown in figure 3.6, the second experiment was run in an interactive room with large projection screens. Users were sitting about 1.8 meters away from a 2.1m x 1.5m projection screen, subtended a horizontal angle of about 100 degrees and a vertical angle of about 80 degrees. Subject were asked to perform two tasks: the tracing task described in section 3.2 and a selection task where the user must reach different colored squares without touching the red squares. A short interview was performed following the experiment to obtain feedback from the subject about the usability of these head trackers.

	Average error (in pixel)	Standard deviation (in pixel)
Small rotation	6.3	0.4
Large rotation	6.1	0.6
Light variation	11.5	3.1

Table 3.1: Experimental results of the stereo-based tracker inside the interactive room.

3.3.1 Results and Discussion

With more then 90 degrees of rotation to reach both sides of the screens, the limitations of the normalized cross-correlation tracker appeared clearly. Subjects could not use the tracker without unnaturally translating their heads over long distances to move the cursor correctly.

The stereo-based tracker was successful on both the tracing task and the selection task. Table 3.1 presents the average errors and standard deviation for the tracing task of 3 subjects.

The interviews after the second experiment showed that users don't like a linear mapping between the head pose and the cursor position. For slow movement of the head, the ratio cursor distance by head movement should be smaller to give more precision on small selections. For fast movement of the head, the ratio should be larger to give more speed on large displacement. These observations corroborate Kjeldson results[26].

Chapter 4

Iterative Hybrid tracker

To improve the robustness of our head pose tracker for fast movement, we designed a second tracking technique: the iterative hybrid tracker. In this new tracking framework, we integrate an ICP 3D euclidian error function with the normal flow constraint, creating a hybrid registration error metric yielding a tracker which is both robust and precise. The ICP approach matches points in 4 dimensions (3D + brightness) and minimizes the euclidian distance between corresponding points. Empirically, we have found that ICP robustly handles coarse motion. The NFC (Normal Flow Constraint) approach matches points based on the inverse calibration parameters and find the transformation between corresponding points based on their appearance and their 3D position. As shown in Section 4.5, this method is more precise for small movement since it searches the pose parameter space using a gradient method which can give sub-pixel accuracy.

4.1 Preprocessing

The new image set $\{I_t, Z_t\}$ is preprocessed in concert with known camera calibration information to obtain the 3D vertex set Ψ_t of $i := 1..m$ vertices $\vec{v}_{ti} = \{\vec{p}_{ti}, \vec{n}_{ti}, I_{ti}\}$ where $\vec{p}_{ti}$ is the 3D point coordinates in the camera reference, $\vec{n}_{ti}$ is the normal vector of the surface projected by Z_t at point $\vec{p}_{ti}$ and I_{ti} is the brightness value of the point $\vec{p}_{ti}$ as specified by the intensity image I_t. The normal vector $\vec{n}_{ti}$ is computed from the depth image gradients:

$$\vec{n}_{ti} = \left[\begin{array}{ccc} \frac{\partial Z_t}{\partial u_{ti}} & \frac{\partial Z_t}{\partial v_{ti}} & 1 \end{array} \right] \tag{4.1}$$

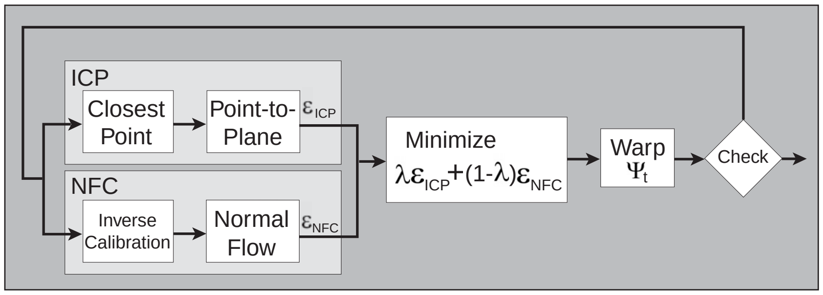


Figure 4.1: Hybrid tracker structure.

where u_{ri} and v_{ri} are the 2D image coordinates of Z_t.

4.2 Iterative Hybrid Tracker Structure

As shown in figure 4.1, our hybrid tracker iterates a joint error minimization process until convergence. At each iteration two error function are minimized in the same linear system. The iteration process can be divided into 5 distinct steps: Match, Error Function, Minimization, Warping and Convergence check.

- The Match stage finds corresponding points between the 3D image sets. In the hybrid tracker we use two matching techniques: closest point and inverse calibration. These techniques are described in more details in sections 4.3.1 and 4.4.1.

- Given the two sets of correspondences, we compute two error functions: point-to-plane and normal flow constraint. These two error functions relate the corresponding point sets to the pose parameters. As shown in Section 4.3.2 and 4.4.2, each error function can be locally approximated as linear problems in terms of the motion parameters:

$$\varepsilon_{ICP} = \|\mathbf{A_{ICP}}\vec{\delta} - \vec{b}_{ICP}\|^2 \tag{4.2}$$
$$\varepsilon_{NFC} = \|\mathbf{A_{NFC}}\vec{\delta} - \vec{b}_{NFC}\|^2 \tag{4.3}$$

- The Minimization stage estimates the optimal transformation $\vec{\delta}^*$ between the matched points using the combined error function:

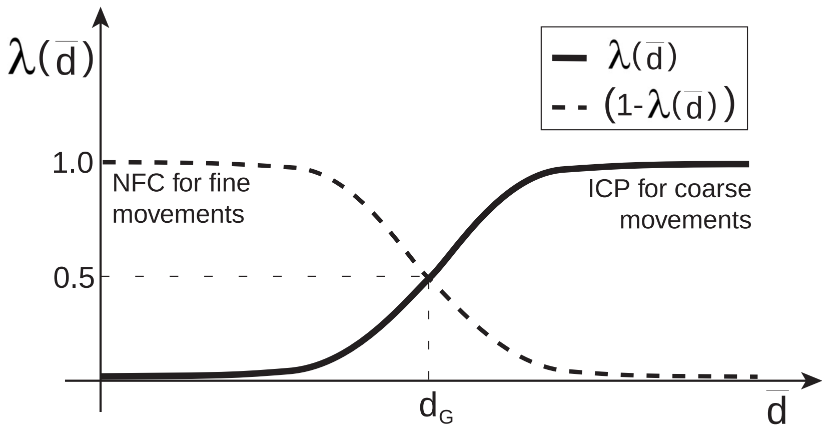


Figure 4.2: Plot of the sigmoidal function $\lambda(\overline{d})$ used in equation 4.4. Notice that as the average distance between matched points $\overline{d}$ decrease, NFC error function has more weight, and vice-versa.

$$\vec{\delta}^{*} = \arg\min_{\vec{\delta}} [\lambda(\overline{d})\varepsilon_{ICP} + (1 - \lambda(\overline{d}))\varepsilon_{NFC}] \tag{4.4}$$

where $\overline{d}$ is the average distance between matched points and $\lambda(\overline{d})$ is a sigmoid function which arbitrates the importance of the ICP error function over the normal flow error function as alignment improves (see figure 4.2). Section 4.5 discusses in more details how the sigmoid function $\lambda(\overline{d})$ is computed.

- The Warping stage warps the 3D vertex set Ψ_t according to the new estimated transformation $\vec{\delta}$. The warping is done by updating the $\vec{p}_{ti}$ and $\vec{n}_{ti}$ of each vertex as follows:

$$\vec{n}_{ti}{}' = \mathbf{R}^{(\delta)}\vec{n}_{ti} \quad \vec{p}_{ti}{}' = \mathbf{R}^{(\delta)}\vec{p}_{ti} + \vec{t}^{\,(\delta)} \tag{4.5}$$

- The Convergence Check stage computes the convergence factor ϵ by averaging the distance D between warped 3D points $\vec{p}_{ti}{}'$ and referential 3D points $\vec{q}_{ri}$:

$$\epsilon = \frac{1}{n}\left(\sum_{i=1}^{n} D(\vec{p}_{ti}{}', \vec{q}_{ri})\right) \tag{4.6}$$

If the difference between the convergence factor ϵ of two consecutive iterations is smaller then a threshold value τ, then convergence is reached. The 3D view registration is completed when convergence is reached or, in the case of non-convergence, when a maximum number N_I of iterations is performed.

4.3 ICP Error Function

To compute the ICP error function, the matching stage searches for closest points in a 4-dimensional space composed of the 3D euclidian space and 1D for brightness. An exhaustive search for matching closest points makes large displacements easier to track. A k-d tree is used to accelerate the matching process [38]. As suggested by Rusinkiewicz and Levoy [34], we use a point-to-plane error function to align the matched points.

4.3.1 Closest Point with k-d Tree

Among the earliest ICP distance functions proposed was the 3D euclidian distance [3]. This function doesn't take into account color or intensity information which may be available. As Godin *et al.*[13], we take advantage of intensity information and use a 4D space (X,Y,Z,E) where E is the brightness value from a intensity image I_r. When I_r is a color image, Godin *et al.* [13] suggests using the hue channel as the brightness measure.

To accelerate the matching process we use a k-d tree and an Approximate Nearest Neighbor algorithm [30]. The k-d tree is created with the values $\{\vec{x}_r, \vec{y}_r, \vec{z}_r, \vec{I}_r\}$ of the referential image set. The same k-d tree is used throughout all the iterations. The matching process finds, for each vertices $\vec{v}_{ti}$ of the current 3D vertex set Ψ_t, the closest node of the k-d tree $\{x_{ri}, y_{ri}, z_{ri}, I_{ri}\}$ that minimizes the 4D distance function:

$$\|\vec{q}_{ri} - \vec{p}_{ti}\| + k\|I_{ri} - I_{ti}\| \tag{4.7}$$

where k is a constant to normalize the brightness value.

4.3.2 Point-to-Plane

The point-to-plane method [6] minimizes the distance between a point $\vec{q}_{ri}$ and the tangential plane of the corresponding point $\vec{p}_{ti}$:

$$D_{Plane}(\vec{q}_{ri}, \vec{p}_{ti}) = \vec{n}_{ti}(\vec{q}_{ri} - (\mathbf{R}\vec{p}_{ti} - \vec{t})) \tag{4.8}$$

By approximating the rotation $\mathbf{R}$ with an instantaneous rotation ω and rearranging the equation 4.8 adequately, we obtain the following linear system:

$$\varepsilon_{ICP} = \|\mathbf{A_{ICP}}\vec{\delta} - \vec{b}_{ICP}\|^2 \tag{4.9}$$

where each line is defined as follow

$$\vec{A}_i = \begin{pmatrix} \vec{n}_{ti} \times \vec{q}_{ri} \\ -\vec{n}_{ti} \end{pmatrix} \tag{4.10}$$

$$b_i = \vec{n}_{ti} \cdot (\vec{p}_{ti} - \vec{q}_{ri}) \tag{4.11}$$

Compared with the point-to-point method [3], the point-to-plane converges faster but requires extra preprocessing to compute the normals (see [34] for more details).

4.4 NFC Error Function

The normal flow constraint is a gradient-based approach which can estimate sub-pixel movements accurately. During the matching stage, we use an inverse calibration method to find corresponding points which belong on the same projective ray. This provides the correspondence needed to compute the temporal gradient term of the normal flow constraint.

4.4.1 Inverse Calibration

The inverse calibration approach [31] searches for corresponding points of $\vec{p}_{ti}$ by projecting the 3D point from the 3D coordinate system of Υ_t to the referential depth image Z_r coordinate system:

$$\begin{bmatrix} \vec{u}_{ri} \\ 1 \end{bmatrix} = \mathbf{C} \begin{bmatrix} \vec{p}_{ti} \\ 1 \end{bmatrix} \tag{4.12}$$

where $\mathbf{C}$ is a 3x4 projection matrix that relate 3D coordinate system of $\vec{p}_{ti}$ to the 2D image coordinate $\vec{u}_{ri} = [\ u_{ri} \quad v_{ri}\ ]$. This matrix is based on the stereo camera or laser scanner parameters.

After projection, two match functions could be used: 1) interpolate the 3D coordinates $\vec{q}_{ri}$ of the corresponding point from the projection value $\vec{u}_{ri}$, or 2) search around the projected point $\vec{u}_{ri}$ in the Z_r image to find the closest point. We used the first method to be compatible with the time gradient term of the normal flow constraint which assumes that the corresponding points are on the same projective ray.

The 3D coordinates $\vec{q}_{ri} = \begin{bmatrix} x_{ri} & y_{ri} & z_{ri} \end{bmatrix}$ are interpolated from the depth image Z_r as follows:

$$z_{ri} = Z_r(\vec{u}_{ri}) \quad , \quad x_{ri} = f\tfrac{u_{ri}}{z_{ri}} \quad , \quad y_{ri} = f\tfrac{v_{ri}}{z_{ri}} \tag{4.13}$$

4.4.2 Normal Flow Constraint

Given 3D input data, the normal flow is the component of the optical flow in the direction of the image gradient. As shown in [42], the normal flow can be expressed as:

$$-\frac{\partial I_{ri}}{\partial t} = \nabla I_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \vec{V} \tag{4.14}$$

where $\vec{V} = \begin{bmatrix} \frac{\partial x_{ri}}{\partial t} & \frac{\partial y_{ri}}{\partial t} & \frac{\partial z_{ri}}{\partial t} \end{bmatrix}$ is the velocity of the object, $\nabla I_{ri} = \begin{bmatrix} \frac{\partial I_{ri}}{\partial u_{ri}} & \frac{\partial I_{ri}}{\partial v_{ri}} \end{bmatrix}$ is the image gradient, and $\frac{\partial I_{ri}}{\partial t}$ is the time gradient. $\frac{\partial I_{ri}}{\partial u_{ri}}$ and $\frac{\partial I_{ri}}{\partial v_{ri}}$ are computed directly from the referential image I_r. The time gradient is approximated by:

$$\frac{\partial I_{ri}}{\partial t} = I_{ti} - I_{ri} \tag{4.15}$$

For a perspective projection where $u_{ri} = f\frac{x_{ri}}{z_{ri}}$ and $v_{ri} = f\frac{y_{ri}}{z_{ri}}$, we can find the Jacobian matrix:

$$\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}} = \begin{bmatrix} \frac{f}{z_{ri}} & 0 & -f\frac{x_{ri}}{z_{ri}^2} \\ 0 & \frac{f}{z_{ri}} & -f\frac{y_{ri}}{z_{ri}^2} \end{bmatrix} \tag{4.16}$$

Since the object is rigid, the velocity V can be expressed as:

$$\vec{V} = \begin{bmatrix} \mathbf{I} & -\hat{q}_{ri} \end{bmatrix} \vec{\delta} \tag{4.17}$$

where $\mathbf{I}$ is a 3x3 identity matrix and $\hat{q}_{ri}$ is the skew matrix of the vector $\vec{q}_{ri}$. By rearranging the equation, we get a linear system similar to the point-to-plane technique (section 4.3.2):

$$\varepsilon_{NFC} = \|\mathbf{A_{NFC}}\vec{\delta} - \vec{b}_{NFC}\|^2 \tag{4.18}$$

where each line is defined as follow

$$\vec{A}_i = \nabla I_{ri} \left[\frac{\partial \vec{u}_{ri}}{\partial \vec{q}_{ri}}\right] \begin{bmatrix} \mathbf{I} & -\hat{q}_{ri} \end{bmatrix} \tag{4.19}$$

$$b_i = -\frac{\partial I_{ri}}{\partial t} \tag{4.20}$$

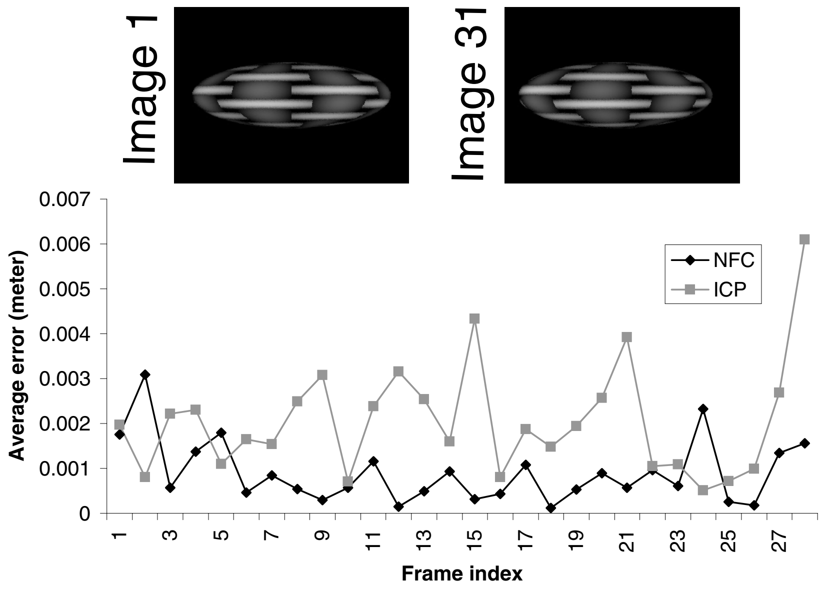


Figure 4.3: Small rotation sequence with synthetic images.

4.4.3 Accuracy comparison for small movements

We compared the performance of NFC and ICP sequential tracking approach on sequences with small movements. The top part of figure 4.3 shows the first and the last frame of a 31 synthetic frame sequence. A rotation of 0.5 degrees occurred between each consecutive frames. Since the sequence is synthetic, we could compare the result of each tracker with the real transformation (0.5 degrees). The average error was computed by warping the referential image by the found transformation and the real transformation and computing the average distance between the two sets of 3D points. The average error for normal flow constraint was 0.898mm, better then the ICP with 2.06mm . The graph in figure 4.3 presents the average error at each frame.

4.5 Hybrid Error Function

At each iteration, the tracking algorithm minimize the hybrid error function to find the optimal pose parameters δ^*. We can rewrite equation 4.4 as one linear system:

$$\arg\min_{\vec{\delta}} \left\| \begin{bmatrix} \lambda(\overline{d})\mathbf{A_{ICP}} \\ (1-\lambda(\overline{d}))\mathbf{A_{NFC}} \end{bmatrix} \vec{\delta} - \begin{bmatrix} \lambda(\overline{d})\vec{b}_{ICP} \\ (1-\lambda(\overline{d}))\vec{b}_{NFC} \end{bmatrix} \right\|^2$$

This linear system can be solved using a least-squares method or any robust estimator. To reduce the influence of outliers, we use a M-estimator to minimize the system [21].

As shown in figure 4.3, the NFC error function is more accurate for the estimation small movement. Since the normal flow constraint approximate the pixel by a plane to compute the intensity gradients $\frac{\partial I_{ri}}{\partial u_{ri}}$ and $\frac{\partial I_{ri}}{\partial v_{ri}}$ of equation 4.14, its accuracy is directly related to the variance of the Gaussian d_G used to compute to compute these gradients. We want a function that increases the importance of NFC when the average distance $\overline{d}$ between matched points decreases, and vice versa. Figure 4.2 shows the sigmoid function that we use:

$$\lambda(\overline{d}) = \frac{1}{1+e^{-c(\overline{d}-d_G)}} \tag{4.21}$$

where c is a constant that determine the slope of the sigmoid function and $\overline{d}$ is the average distance of matched points found during the closest point matching process (see Section 4.3.1). We define the average distance as follow:

$$\overline{d} = \frac{1}{N}\sum_{i=1}^{N} D(\vec{q}_{ri}, \vec{p}_{ri}) \tag{4.22}$$

where N is the number of matched points and D is the euclidian distance between two points.

Chapter 5

Head Pose Experiments

We tested our hybrid tracker with 3 sequences obtained from a stereo camera using the SRI Small Vision System [10]. Tracking was initiated automatically by using a face detector [43]. Without special optimizations, the hybrid sequential tracker can update poses based on observations of 2500 points per frame at 2Hz on a Pentium III 800MHz.

The following sections present tracking results for three sequences recorded in different environments and with different persons. For each sequence, we present intensity and depth image samples from the original sequence. Since the goal is to compare the accuracy of each tracker, no drift reduction algorithm was used during th tracking of all 3 sequences. Then we show the tracking result from 3 different algorithms: ICP alone(see section 4.3, NFC alone (see section 4.4), and the hybrid tracker (see section 4.5). For sequence 2, we compare the convergence of the error factor for each tracking technique.

5.1 Test sequence 1

Figure 5.1 presents some key frames of a sequence where the user move its head in front of the camera. The sequence contains 180 frames which represents approximately 18 seconds since the grabbing rate was about 10 Hz. During the sequence, the user turned his head approximately 40 degrees down, up, left, and right. Then, the user translated his head 30cm, which was equivalent to 25 image pixels.

Figure 5.2 present the tracking results after one iteration. We observe that ICP alone performs well for translation, but has trouble with rotation. We observe the opposite results for NFC alone which handles rotation well, but translation poorly. The hybrid tracker is able to track all the sequence reliably.

5.2 Test sequence 2

Figure 5.3 presents some key frames of a sequence containing 160 frames which represents approximately 16 seconds. During the sequence, the user turned his head left and right (approximately 25 degrees) and then translated his head left and right rapidly, three times.

Figure 5.4 presents the tracking results for sequence 2 after 3 iterations. As observed for sequence 2, the hybrid tracker perform better then ICP or NFC alone. The figure 5.5 shows the average convergence factor of each registration technique. The convergence factor is computed as described in section 4.2. The three techniques converge in less then 3 iterations. The hybrid error function converge to an average distance error 20% smaller then ICP alone and 5% smaller then NFC alone.

5.3 Test sequence 3

Figure 5.6 presents some key frames of a sequence containing 250 frames ($\tilde{2}5$ seconds). During the sequence, the user lawn left, right before to look up and finally lawn left again. Figures 5.7 and 5.8 show tracking results for NFC, ICP and the hybrid tracker. We can observe that the hybrid tracker perform better then ICP and NFC even after 250 frames. We could improve these results by adding some drift reduction algorithm as described in section 2.4. Movies of the above results can be found at http://www.ai.mit.edu/people/lmorency/ .

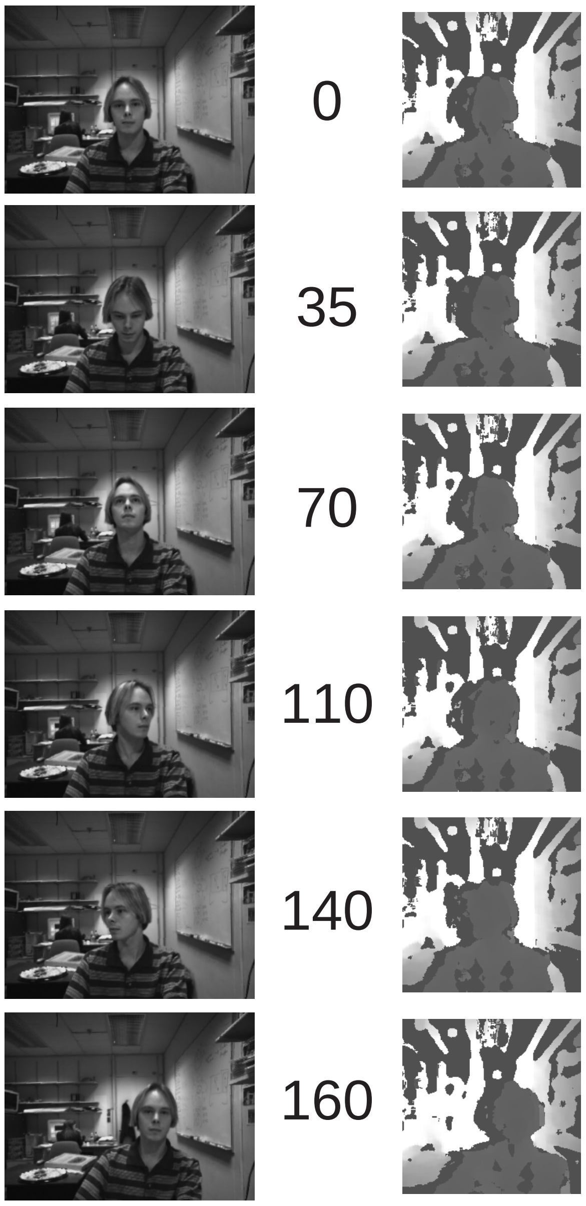


Figure 5.1: Intensity and depth images from sequence 1.

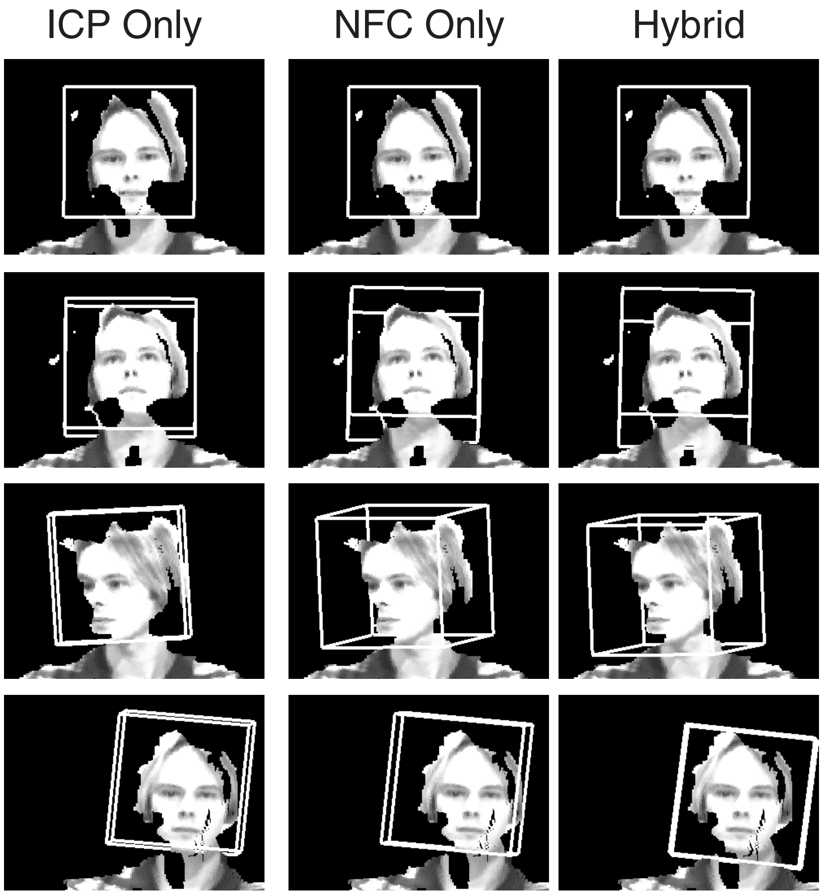


Figure 5.2: Face tracking results of sequence 1. Each row represents tracking results at different frames: 0, 70, 140, and 180.

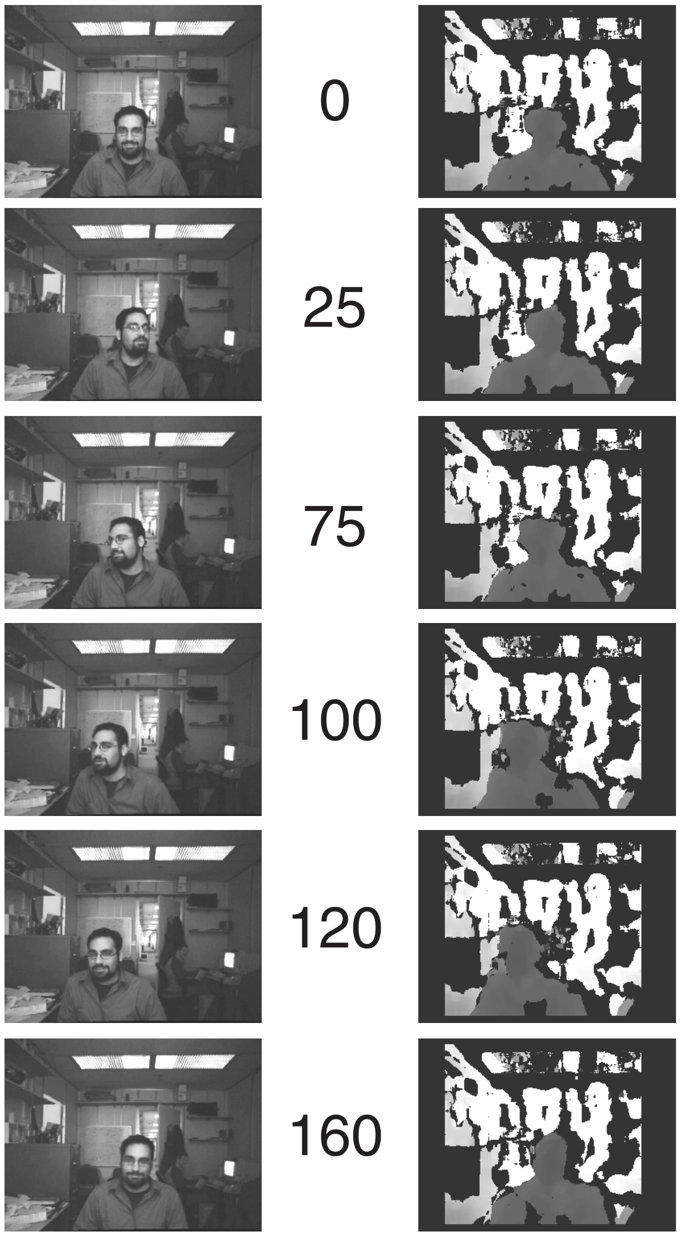


Figure 5.3: Intensity and depth images from sequence 2.

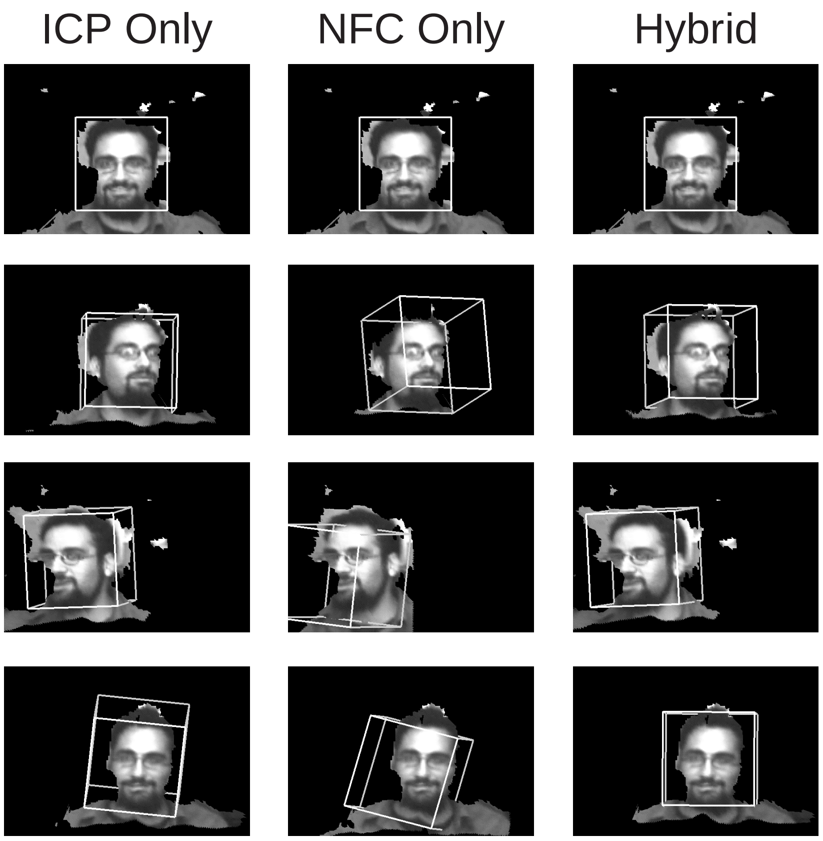


Figure 5.4: Face tracking results of sequence 2. Each row represents tracking results at different frames: 0, 25, 100, and 160.

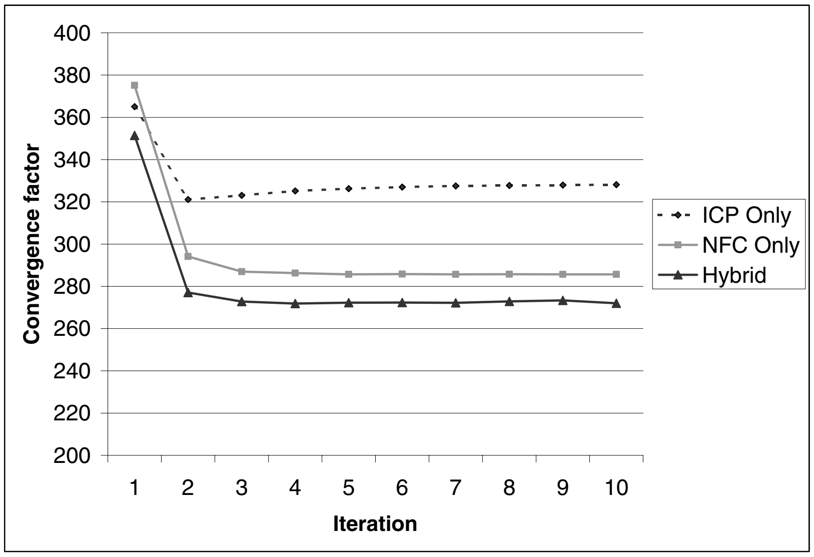


Figure 5.5: Comparison of average convergence factor for 80 frames (sequence 2).

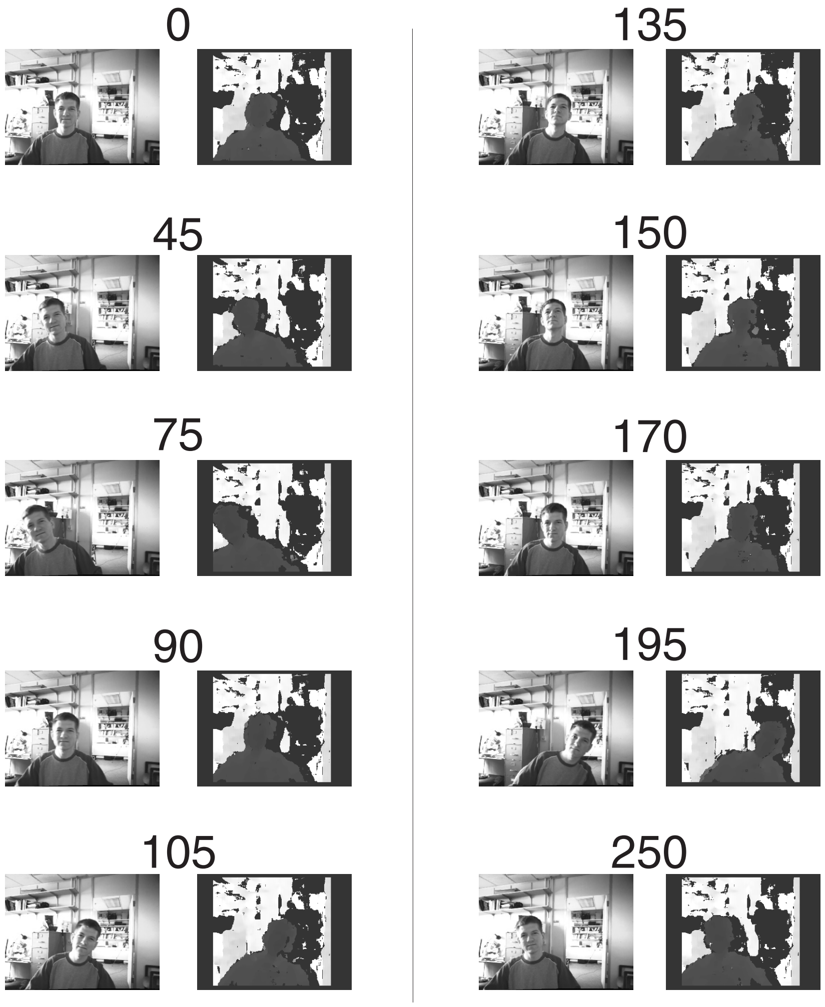


Figure 5.6: Intensity and depth images from sequence 3.

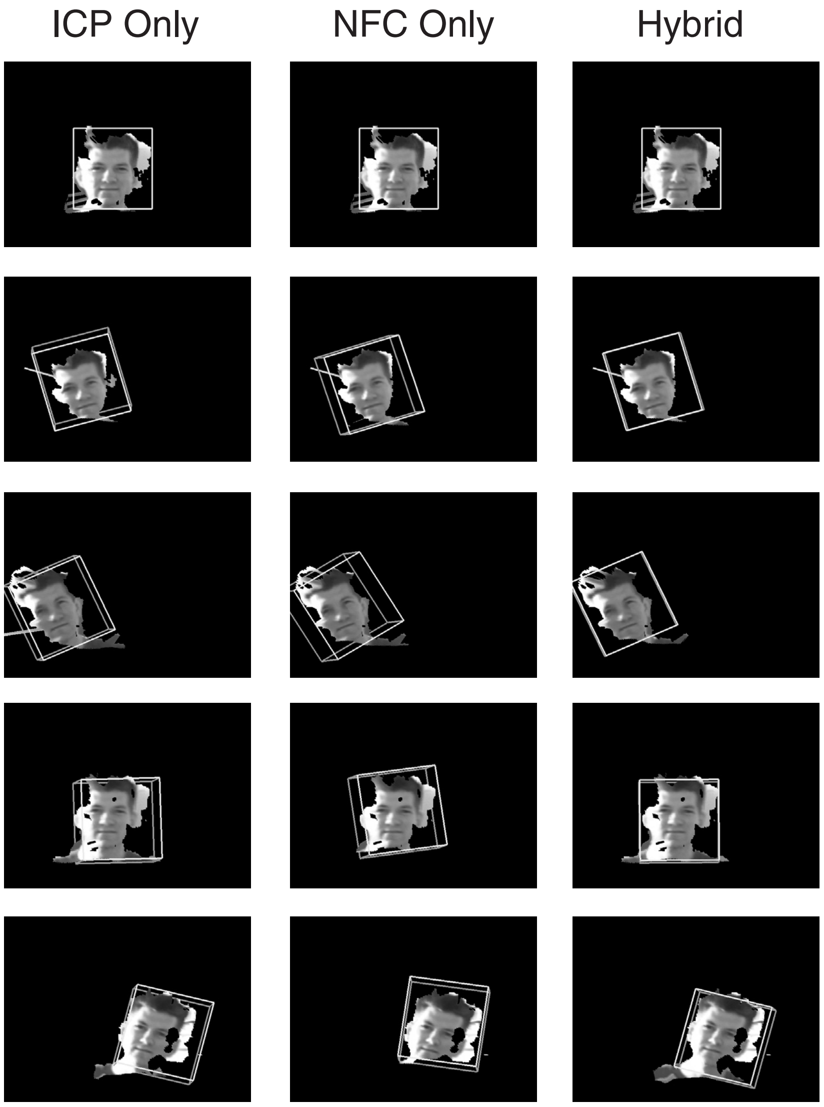


Figure 5.7: Face tracking results for sequence 3. Each row represents tracking results at different frames: 0, 45, 75, 90, and 105.

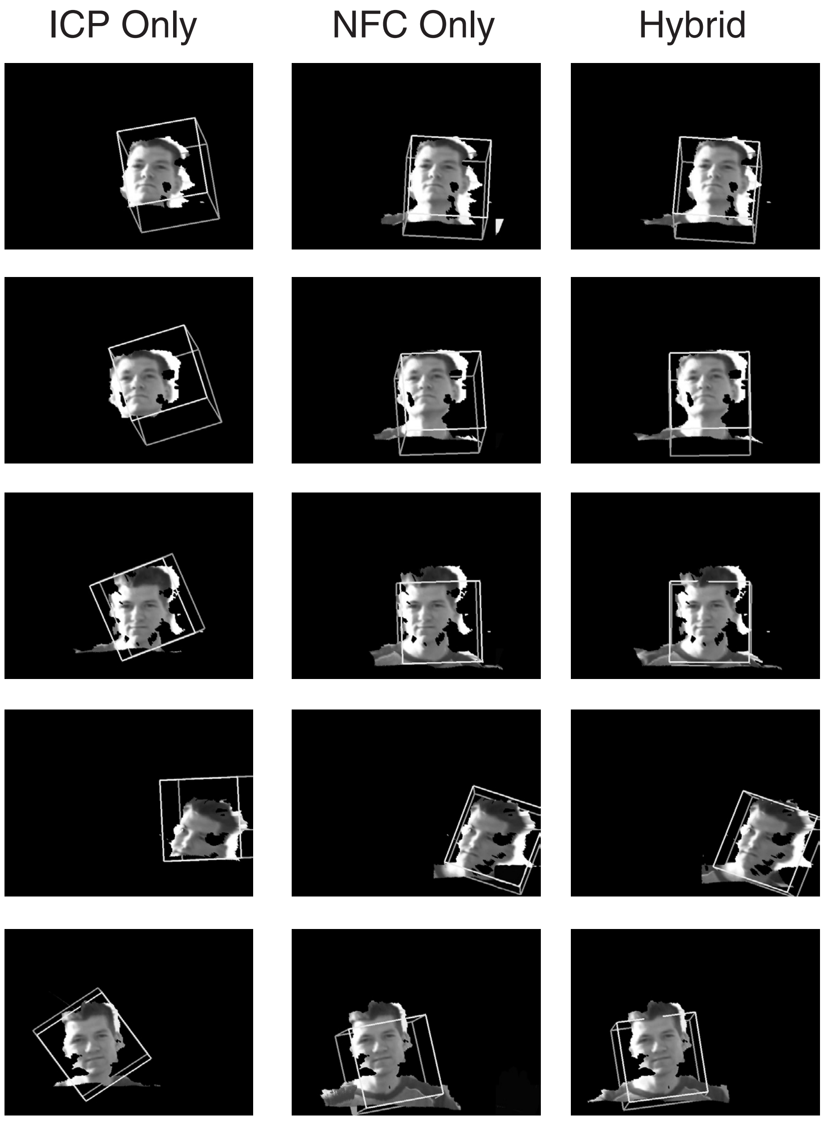


Figure 5.8: Face tracking results for sequence 3. Each row represents tracking results at different frames: 135, 150, 170, 195, and 250.

Chapter 6

3D Model Reconstruction

In this chapter, we present an efficient solution for 3D model acquisition using the iterative hybrid tracker to register 3D views. Once frames have been globally registered, they are non-rigidly deformed during the reconstruction phase to produce a smooth triangular mesh. To construct this mesh, frames are individually converted to meshes by using the pixel adjacency information in the original range scan. Each vertex q on a mesh is assigned the 3D location $\mathbf{X}$, surface normal $\mathbf{n}$ and intensity I of its corresponding point in the registered scan. The uncertainty in these variables is computed by combining the effects of measurement uncertainty and registration error and stored along the other parameters:

$$q = \{[\ \mathbf{X_q} \quad \mathbf{n_q} \quad I_q\ ]; \Lambda_q\}$$

Reconstruction then involves a discretization of these vertices into a linked voxel space (described in the following section), followed by a merging of nearby voxels using the Cubic Ray Projection algorithm of section 6.2. The linked voxel space is finally converted to a triangular mesh and rendered.

6.1 Linked Voxel Space

To maintain an intermediate representation of the final 3D model, we use a voxel space. However, for our purposes, the simple voxel model has two main disadvantages: 1) the connectivity of meshes cannot be represented, and 2) converting this volumetric model to a mesh is difficult[18, 28]. To solve these problems, we use an augmented version of the voxel space called the linked voxel space.

In a linked voxel space, voxels maintain information about their connectivity beyond their immediate neighbors in the space. When converting a mesh to a linked voxel space, edges between vertices of the mesh are converted to links between voxels. In our representation, each voxel v is represented by a vertex q_v centered in the voxel and a list of links L_v, initially empty.

$$v = \{\ q_v \quad L_v\ \}$$

After registration, each frame is converted to a mesh. The mesh is transformed to the pose recovered during the global registration phase, and accumulated into the linked voxel space.

The location of each vertex q in the mesh is quantized and mapped to a voxel v. This voxel v is updated as follows:

- The covariance Λ_v of v is updated with

$$\Lambda_v{}^{new} = \left(\left[\Lambda_v^{old}\right]^{-1} + \Lambda_q^{-1}\right)^{-1}$$

- The mean surface normal $\mathbf{n}_v$ at the voxel is updated with the normal $\mathbf{n}_q$ of q using:

$$\mathbf{n}_v = \Lambda_v{}^{new}\left(\Lambda_q^{-1}\mathbf{n}_q + \left[\Lambda_v{}^{old}\right]^{-1}\mathbf{n}_v\right)$$

  where Λ_q is the uncertainty in the node.

- The intensity value I_v is updated as follow

$$I_v = \Lambda_v{}^{new}\left(\Lambda_q^{-1}I_q + \left[\Lambda_v{}^{old}\right]^{-1}I_v\right)$$

- Each edge i of the vertex q points to a neighboring node q^i which is mapped to a voxel v^i. A link L_{v^i} is added to v if the voxel v^i is not already linked with v.

The mean surface normal of the voxels are used to guide the Cubic Ray Projection merging phase and ultimately become the normals of the final mesh model.

6.2 Cubic Ray Projection

The next stage of reconstruction thins out the voxel space by projecting voxels on one of the six faces of a cube that delimits the voxel space, merging voxels which fall on the same projection ray. As voxels are merged, the link information of the voxels is updated, resulting in a voxel space which can be trivially turned into a mesh.

This process aims to identify voxels which represent the same point on the object being modelled but which have been incorrectly registered by the registration process. We employ the heuristic that if two voxels are nearby, have the same normal, and lie along the same projection ray to the camera, they represent the same point and should be merged. The cube projection algorithm identifies such voxels by quantizing the normal vectors and providing an efficient data structure to aid the search. As a result, merging is an O(n) algorithm, where n is the number of voxels.

Figure 6.1 depicts the merging process. The inverse of the covariance of a voxel is represented by the size of the dot. The arrow shows the direction of the normal. The highlighted line in the figure represent a projection ray to the left face of the cube. Along this ray, only voxels with a normal vector pointing in the direction of left face are selected. Voxels which are nearby and which are mapped to the same location on the surface of the cube are then merged. Merging two voxels involves updating one of them and unlinking the other.

The merging algorithm updates the mean normal, the intensity value and the adjacency information of the voxel with the lowest covariance v_1. The voxel with the highest covariance v_2 is unlinked from the rest of the voxel space. The specifics of the update are similar to the discretization step: 1

- Average normal $\mathbf{n_1}$, intensity I_1 and covariance Λ_1 are updated as described in section 6.1.
- Each links $L^2_{v^i}$ are added to v^1 if the voxel v^i is not already linked with v^1.
- All the links from v^i to v^2 are removed during the unlinking stage. This step is possible since the voxel space is double linked.

Note that the update throws away only a small amount of information: it discards the voxel with the largest covariance, but updates the voxel with the lowest covariance with the former's normal and link information. This voxel merging step is in some ways similar to that of[9] where only one merging plane is used (instead of a cube) and all voxels

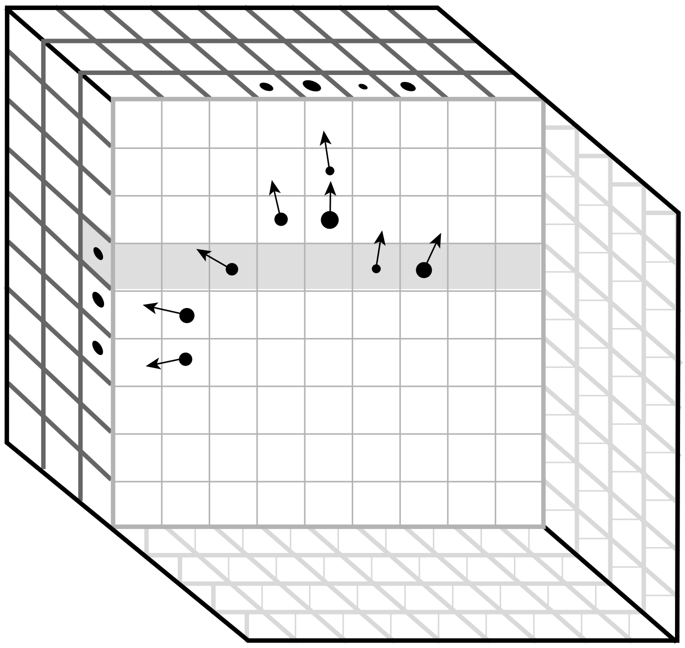


Figure 6.1: Result of voxels merging on one layer of the voxel cube. Occupied voxels are represented by a dot (inverse of the covariance) and an arrow (normal vector). The figure shows the projection of the voxels on two faces of the cube.

along a ray contribute to a weighted average distance to the merging plane (instead of a weighted normal).

Merging all voxels which satisfy the merging criterion results in a representation where no two nearby voxels with similar normals project to the same face of the encompassing cube. This thinned spatial representation has associated adjacency information which make it readily available as a triangular mesh.

6.3 Results

When integrated with a real-time stereo camera, our system makes it possible to capture 3D models interactively and unobtrusively. Using SRI's Small Vision System, we captured about 10 seconds worth of range and intensity frames of a person moving in front of the camera.

Figure 6.2 shows some typical frames from the sequence. The subject rotated his head from left to right, making a 70 degree arc about the vertical axis. Notice that the range information is missing for much of the face and must be recovered by incorporating many views.

The registration step aligns all the 3D views together to create a dense 3D model. Figure 6.3 shows the progression of the 3D model during the registration step. We can observe that the face model is completed as the person turn his head. The registration runs online at the same time as the stereo processing, at about 7 fps on a 1.5Ghz Pentium 4.

The Cubic Ray Projection phase merges all the views into a linked voxel space. This step reduces the number of vertices from 200,000 to 18,000. Figure 6.4 shows the reconstructed 3D voxel space, along with the accompanying texture map. A solid arc of about 180 degrees was recovered from the 70 degrees of rotation. Global registration, 3D reconstruction, and rendering together took less than 1 second.

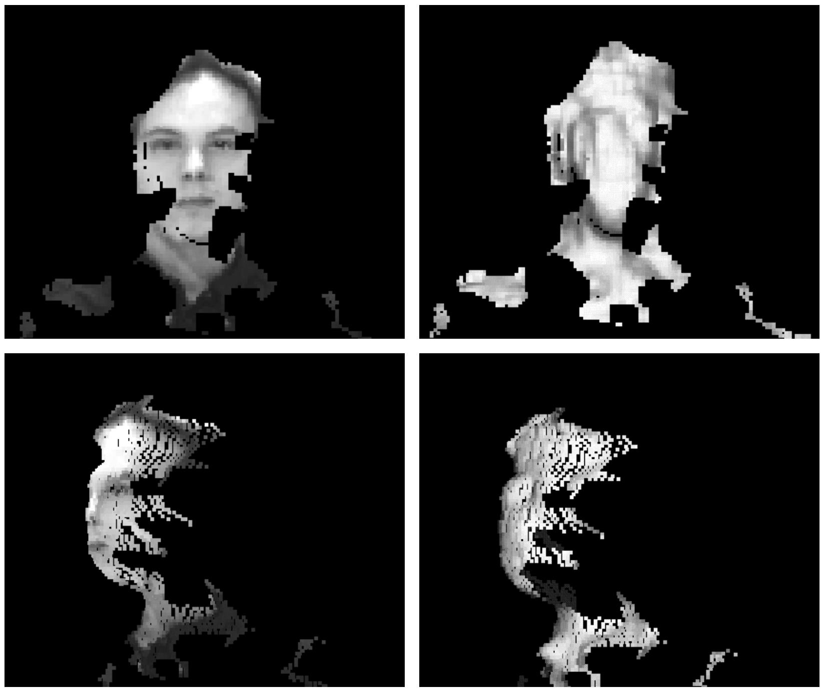


Figure 6.2: Sample images from the sequence. Left: Intensity images. Right: Depth images.

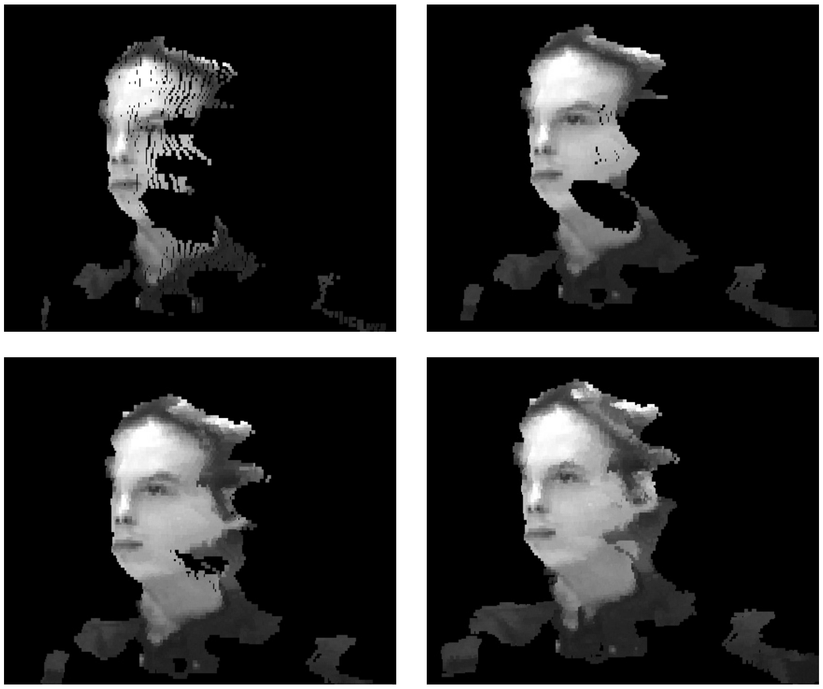


Figure 6.3: Progress of the model acquisition.

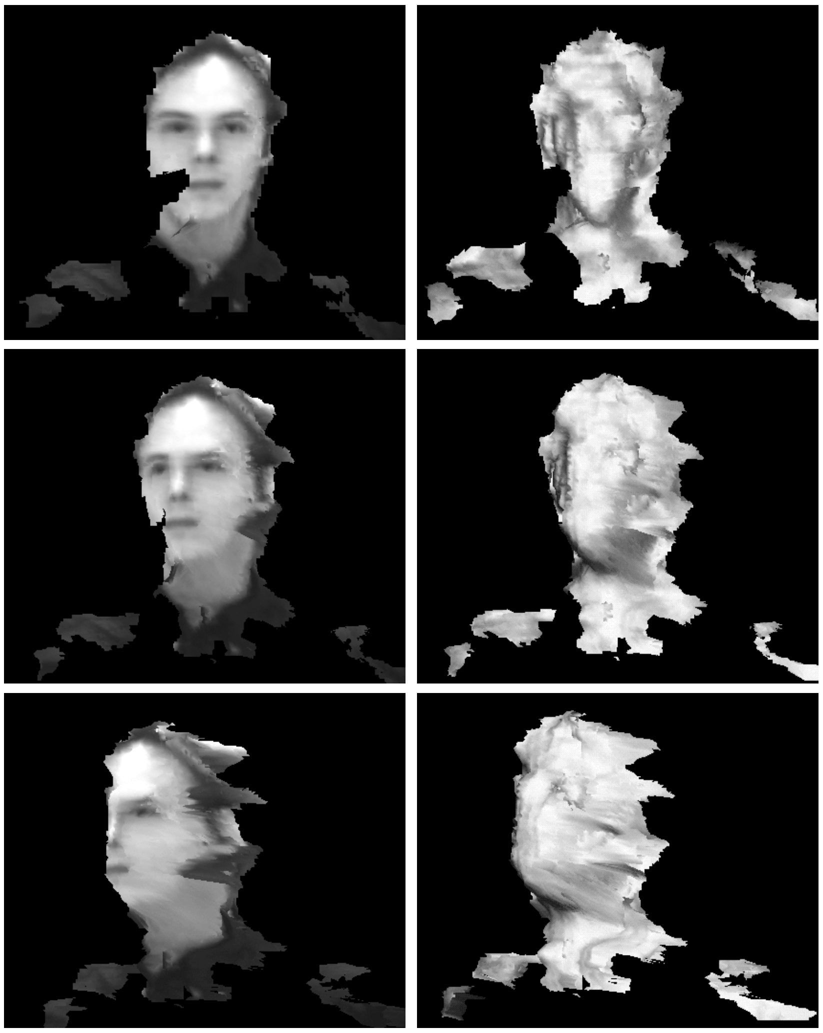


Figure 6.4: Final 3D mesh viewed from different directions.

Chapter 7

Conclusion

In the first part of this text, we presented a stereo head tracking system based on ZBCCE algorithm which requires no manual initialization, does not drift, and has been shown to be accurate at driving cursors and selecting objects. Performance of this tracker was compared against that of a head-mounted inertial sensor and a simple tracker based on normalized cross-correlation. The latter tracker was prone to lighting changes, and the former experienced drift over time. The stereo system was insensitive to these conditions, and was found usable by naive users.

In the second part of this text, we presented a new hybrid 3D view registration framework for tracking 3D pose from noisy 3D stereo images. This second tracking technique is an iterative hybrid tracker designed to improve the robustness of our head pose tracker for fast movement. Our approach integrated the fine tracking ability of a gradient-based normal flow constraint with the robust coarse tracking ability of the ICP algorithm. The stability of our tracker was shown on synthetic sequences with known ground truth and on sequences grabbed from a low-cost stereo camera. Our results indicated that the hybrid approach outperformed either algorithm alone.

Finally, we have demonstrated an efficient technique for producing textured 3D models from range and intensity data using our iterative hybrid tracker. The system uses stereo cameras to obtain synchronized range and intensity frames, and hence does not require subsequent texture alignment. Our algorithm allows the object to be moved around freely to expose different views. The frames first undergo a registration phase (iterative hybrid tracker) which computes relative pose estimates between pairs of frames, and globally solves for the optimal set of poses for all frames. Our registration algorithm uses range as well as intensity data in an image gradient-based approach, compensating for the poor quality of range from correlation-based stereo. The recovered poses are used to warp all frames to a canonical position, and a 3D model reconstruction step merges the registered frames together to build a 3D mesh of the object using a new approach called Cubic Ray Projection and a new data structure called Linked Voxel. We have demonstrated our system by reconstructing a model of a human head as the subject underwent a 70 degree rotation.

We believe that our iterative hybrid tracker when applied to stereo-based head pose tracking can be an important module in designing perceptual interfaces for intelligent environments, cockpit applications, and for disabled users who are not able to use traditional interfaces. Applied to 3D model acquisition, our iterative hybrid tracker could be used for 3D video conference by creating online 3D meshes of human faces.

Bibliography

[1] J.L. Barron, D.J. Fllet, and S.S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

[2] S. Basu, I.A. Essa, and A.P. Pentland. Motion regularization for model-based head tracking. In *ICPR96*, page C8A.3, 1996.

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

[4] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *ICCV*, pages 374–381, 1995.

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

[6] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. In *Proceedings of the IEEE Internation Conference on Robotics and Authomation*, pages 2724–2728, 1991.

[7] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *PAMI*, 23(6):681–684, June 2001.

[8] B. Curless. From range scans to 3d models. *Computer Graphics*, 33(4), november 1999.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30(Annual Conference Series):303–312, 1996.

[10] Videre Design. *MEGA-D Megapixel Digital Stereo Head.* http://www.ai.sri.com/~konolige/svs/, 2000.

[11] J. Feldmar and N. Ayache. affine and locally affine registration of free-form surfaces. *IJCV*, 18(2):99–119, 1996.

[12] Eric M. Foxlin. Inertial orientation tracker apparatus having automatic drift compensation for tracking human head and other similarly sized body. US Patent 5,645,077, US Patent and Trademark Office, Jun 1994.

[13] G. Godin, M. Rioux, and R. Baribeau. Three-dimensional registration using range and intensity information. In *Proceedings of SPIE Videometric III*, volume 2350, pages 279–290, 1994.

[14] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts, 1992.

[15] G.Turk and M. Levoy. Zippered polygon meshes form range images. In *Proceedings of SIGGRAPH vol. 2*, pages 311–318, 1994.

[16] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039, October 1998.

[17] M. Harville, A. Rahimi, T. Darrell, G. Gordon, and J. Woodfill. 3d pose tracking with linear depth and brightness constraints. In *Proceedings of ICCV 99*, pages 206–213, Corfu, Greece, 1999.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Proceedings of SIGGRAPH*, 26(2):71–78, July 1992.

[19] B.K.P. Horn and B.G. Schunck. Determining optical flow. *AI*, 17:185–203, 1981.

[20] B.K.P. Horn and E.J. Weldon, Jr. Direct methods for recovering motion. *IJCV*, 2(1):51–76, June 1988.

[21] P.J. Huber. *Robust statistics*. Addison-Wesley, New York, 1981.

[22] InterSense Inc. *Intertrax 2*. http://www.intersense.com.

[23] Mouse Vision Inc. *Visual Mouse*. http://www.mousevision.com.

[24] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *In ICCV*, September 1999.

[25] R.J.K Jacob. *Eye tracking in advanced interface design*, pages 258–288. Oxford University Press, 1995.

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

[27] M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3d models. *PAMI*, 22(4):322–336, April 2000.

[28] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of SIGGRAPH*, 21(4):163–169, July 1987.

[29] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333–349, 1997.

[30] D. M. Mount and S. Arya. *ANN: Library for Approximate Nearest Neighbor Searching.* http://www.cs.umd.edu, 1998.

[31] P.J. Neugebauer. Geometrical cloning of 3d objects via simultaneous rgistration of multiple range images. In *Proc. Int. Conf. Shape Modeling and Applications*, pages 130–139, 1997.

[32] K. Pulli. Multiview registration for large data sets. In *Proc. 3DIM*, pages 160–168, 1999.

[33] A. Rahimi, L.P. Morency, and T. Darrell. Reducing drift in parametric motion tracking. In *In Proceedings of Internation Conference of Computer Vision*, volume 1, pages 315–322, 2001.

[34] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proc. 3DIM*, pages 145–152, 2001.

[35] Harpreet S. Sawhney, Steve Hsu, and Rakesh Kumar. Robust video mosaicing through topology inference and local to global alignment. In *ECCV*, pages 103–119, 1998.

[36] A. Schodl, A. Haro, and I. Essa. Head tracking using a textured polygonal model. In *PUI98*, 1998.

[37] C. Schtz, T. Jost, and H. Hgli. Multi-featured matching algorithm for free-form 3d surface registration. In *ICPR*, pages 982–984, 1998.

[38] Simon. *Fast and Accurate Shape-Based Registration.* Ph.D. Dissertation, Carnegie Mellon University, 1996.

[39] G. Stein and A. Shashua. Direct estimation of motion and extended scene structure from moving stereo rig. In *Proc. of CVPR*, June 1998.

[40] A. Stoddart and A. Hilton. Registration of multiple point sets. In *IJCV*, pages B40–44, 1996.

[41] K. Toyama. Look,ma - no hands!hands-free cursor control with real-time 3d face tracking. In *PUI98*, 1998.

[42] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV (2)*, pages 722–729, 1999.

[43] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[44] L. Wiskott, J.M. Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *PAMI*, 19(7):775–779, July 1997.

[45] C.R. Wren, A. Azarbayejani, T.J. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, July 1997.

[46] S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. In *CHI99*, pages 246–253, 1999.