2010

Markerless Tracking Using Polar Correlation Of Camera Optical Flow

Prince Gupta
University of Central Florida

MARKERLESS TRACKING USING POLAR CORRELATION OF CAMERA OPTICAL FLOW

by

PRINCE GUPTA

B.Tech. Uttar Pradesh Technical University, 2008

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
in the School of Electrical Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2010

Major Professor: Niels da Vitoria Lobo

ABSTRACT

We present a novel, real-time, markerless vision-based tracking system, employing a rigid orthogonal configuration of two pairs of opposing cameras. Our system uses optical flow over sparse features to overcome the limitation of vision-based systems that require markers or a pre-loaded model of the physical environment. We show how opposing cameras enable cancellation of common components of optical flow leading to an efficient tracking algorithm that captures five degrees of freedom including direction of translation and angular velocity. Experiments comparing our device with an electromagnetic tracker show that its average tracking accuracy is 80% over 185 frames, and it is able to track large range motions even in outdoor settings. We also present how opposing cameras in vision-based inside-looking-out systems can be used for gesture recognition. To demonstrate our approach, we discuss three different algorithms for recovering motion parameters at different levels of complete recovery. We show how optical flow in opposing cameras can be used to recover motion parameters of the multi-camera rig. Experimental results show gesture recognition accuracy of 88.0%, 90.7% and 86.7% for our three techniques, respectively, across a set of 15 gestures.

ACKNOWLEDGMENTS

I would like to thank my father Rajeev Gupta, my mother Priti Gupta, my sister Saloni Jain and my uncle and aunt, Sanjeev Gupta and Manisha Gupta for helping me throughout my numerous years of college. Furthermore, I would like to thank my faculty advisor, Dr. Nies da Vitoria Lobo, for his guidance and support. I would like to thank the members of my committee, Dr. Joseph J. LaViola Jr. and Dr. Mubarak Shah for their direction and guidance. I would also like to thank Daniel Gabriel, Phillip Napieralski and Jon Harter for their help at various levels in my work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Motion tracking is a critical aspect of many virtual and augmented reality applications and there is a wide variety of different tracking technologies and approaches. Current tracking technologies include accelerometers, gyroscopes, electromagnetic trackers, infra-red trackers, GPS and sonic trackers. Each of these approaches have limitations: IMU and GPS systems are generally expensive, electromagnetic trackers have a low range and accelerometers and gyroscopes give only three dimensional data. As such, hybrid approaches use multiple sensor modalities to limit the individual sensor weaknesses [46].

One approach that has gained in popularity in recent years is vision-based tracking. Cameras are inexpensive, have large range, and provide raw images which are rich in information. Additionally, there are other auxiliary benefits of using cameras, such as face recognition that could be used to identify a user and personalize a system using their profile. Vision-based tracking systems can be classified into two genres: Inside-looking-out, in which the optical sensor is placed on the moving

object and the scene is stationary, example [45], and Outside-looking-in, in which the optical sensor is stationary and observes the moving object, example [11].

Traditionally, vision-based tracking requires markers as reference points or a pre-loaded model of the physical environment. Unfortunately, markers can clutter the physical environment and preloaded models can be time-consuming to create. We present a real-time, markerless, vision-based tracking system employing a rigid orthogonal configuration of two pairs of opposing cameras. We show how opposing cameras enable cancellation of common components of optical flow, which leads to an efficient tracking algorithm. Our prototype is low cost, requires no setup, obtains high accuracy and has large range span.

Our prototype tracker is an example of inside-looking-out optical tracking system. Other related approaches [27, 28] use omnidirectional cameras that give a 360° view of the environment. Omnidirectional cameras are generally expensive, whereas our device prototype is built using inexpensive off-the-shelf webcams. We can extract tracking information by exploiting cameras placed on two ends of a diameter and facing away from each other. Our approach is based on polar correlation of optical flow to calculate egomotion, in this case, the rotation and translation of a camera between every two frames. Our tracking system can be used in various applications like 3D spatial interaction, body tracking, and in robotic applications for large terrain tracking.

Handheld devices used in gesture recognition applications are often built using accelerometers (for example, the Nintendo Wii Remote), gyroscopes or electromagnetic trackers. A less common, yet promising approach are vision-based inside-looking-out systems. Markerless inside-looking-out camera systems have to recover motion parameters. Recovering motion parameters of a moving camera is similar to the autonomous robotic problem, in the sense that both have to solve for the egomotion of the camera. It is advantageous to use a multi-camera based approach to solve this problem (i.e., recovering motion parameters) because it increases the field of view and also introduces additional constraints that could be used to recover the motion parameters. In [7, 20, 21, 22, 26], SFM and egomotion algorithms are developed for multi camera navigation tasks. Other related approaches [27, 28] use omnidirectional cameras that give a 360° view of the environment. Omnidirectional cameras are generally expensive, whereas our device prototype is built using inexpensive off-the-shelf webcams. For the limited task of gesture recognition, where it is not necessary to obtain perfect egomotion, a multi-camera vision-based device has significant potential.

We present three different techniques for obtaining motion parameters at different levels of completed recovery using our device. Fully complete recovery is to obtain rotation (with magnitude) and direction of translation. The first technique, Pattern Matching, gives an estimate of direction of translation and Euler angles of rotation by limiting the possible values to a small quantized set of values. The second

technique, Antipodal Regions, gives an estimate of the direction of translation and the axis of rotation of the rig (without magnitude of rotation). In the third technique, Polar Correlation, we are able to make a fully complete recovery, giving us the direction of translation and angular velocity of the rig.

SECTION 1.1 Motivation

Importance of a good tracking system in virtual reality and augmented reality applications can never be overstated. Characteristics of a good tracking system include:

1. Should give absolute position and orientation.
2. Should run in real-time, with low latency.
3. Should attain good levels of accuracy. Not necessarily the levels of accuracy of a mechanical tracker, but atleast good enough in the perceptual domain.
4. Should have large range. Though the range requirements really depends on the application. An applications may need tracking inside a lab setting or in a building or across a whole city.
5. Should be low cost. As this really limits the affordability of the technology.
6. Should be able to work in both indoor and outdoor environments.

7. Should be markerless, as markers can clutter the space.

8. Should not require external installations and setup. Portability and ease of setup are critical if it is desired for a tracking technology to be able to run in somebody's living room someday.

9. It would be good to be small and wireless.

The work presented in this thesis is our first step in realization of a good tracking system that covers the above characteristics.

SECTION 1.2 Thesis Overview

In CHAPTER 2, we present work related to single camera and multi-camera vision-based systems. CHAPTER 3 describes the details behind our tracking algorithm. In CHAPTER 4, we present the three different algorithms for recovering motion parameters. CHAPTER 5 provides some implementation details and CHAPTER 6 presents a set of experiments done to evaluate our tracker's performance and the experiments done to evaluate the performance of the techniques on gesture recognition. Finally, CHAPTER 7 concludes the thesis.

CHAPTER 2: RELATED WORK

There are two main categories of work related to ours: vision-based outside-looking-in systems and vision-based inside-looking-out systems.

Outside-looking-in systems

There are many vision-based outside-looking-in user interface systems. In [24], presents a single camera and a two camera approach for on-screen item selection by finger pointing, without any 3D inference, rather only by image morphing and and line intersection. In [37], a hand gesture interface is presented using two cameras. In [47], a system for hand pointing and gesture recognition is presented. It uses optical flow techniques to model the motion of the hand. In [48], a system for tracking the 3D position of a finger using single camera is presented, to be used as a visualization tool or a user input interface.

For gaming and other applications: In [9], a perceptual user interface for a responsive dialog-box agent is presented, by recognizing user acknowledgement from

head gestures. Sony's EyeToy, [11], and Microsoft's NATAL Project are examples of commercial vision-based outside-looking-in systems in gaming applications. In [13], several vision algorithms are described for vision controller graphics applications like vision-based computer games, hand signal recognition system and a television set controlled by hand gestures. In [12], specialized algorithms tailored for particular hardware is presented for game interactions. In [15, 17], a computer vision and hearing based user interface is presented, specifically for children's games. In [40], a markerless multi-camera motion caputer system is presented. In [44], theoretical work on perceptual user interface is presented along with algorithms for head tracking, hand tracking and full body tracking.

As examples of other approaches, [32] uses multiple stationary cameras to observe the user's interaction, [36] utilizes a glove with colored markers that assist in tracking, [4] tracks faces for enhanced interaction, and [8, 23] are real-time markerless object tracking systems for augmented reality applications.

The difference in these approaches and our approach is that they employ stationary cameras and we use a rigid set of moving cameras as a controller. From a Computer Vision point of view these two approaches are similar because the fundamental algorithms needed to process the motion in the camera images remain the same. For example, the same optical flow algorithm is applied to image frames whether only a scene object is moving or the camera is moving.

Inside-looking-out systems

There are fewer inside-looking-out systems for user interface applications, even though there are many algorithms for structure from motion (SFM) and simultaneous localization and mapping (SLAM), both used in robotics or general purpose vision applications.

Single camera SLAM methods include single camera [10] and multiple camera [19] approaches. In [16], the eight-point algorithm is presented for structure from motion. In [33], an algorithmic solution to classical five point pose problem is presented. The algorithm is suited for numerical implementation that also corresponds to the inherent complexity of the problem. Though due to the heuristic and ad hoc nature of the procedure it applies, to implement it is not so easy for non-expert users.In [25], a much easier algorithm based on hidden variable resultant technique is presented. In [39], a system for interactively browsing and exploring large unstructured collections of photographs is presented. The underlying solution for structure from motion is formulated as a minimization problem, as a non-linear least squares problem and solved with algorithms such as Levenberg-Marquardt, [34].

Multi camera In [7], a multi-camera 6 DOF motion estimation system is presented. The approach first calculates 5 DOF using five point algorithm of Nister [33] and

then calculates the scale of translation using a multi-camera scale constraint.

In [20], a multi-camera motion estimation algorithm using global optimization technique is presented. It computes an optimal estimate of the essential matrix by searching the rotation space and an optimal solution for translation using linear programming and Branch and Bound algorithm. For multi camera systems [35] presents the Generalized Camera Model (GCM). In [26], an algorithm is presented which uses the GCM to formulate the Generalized Epipolar Constraint (GEC) and then solves it to get rotation and translation. In [21], two algorithms in the context of GCM are presented and compared, the first with a linear solution and the second a geometric algorithm. In [22], a multi-camera rig is developed and a motion estimation algorithm is presented assuming a spherical imaging system for the multi-camera rig (i.e., the cameras share a common center of projection). An approach for computing egomotion using optical flow is described in [43], but the experimental results show that the technique works for only very small motions, which is not practical for user interface application. A multi-camera 6 DOF pose tracking algorithm is presented in [41], but tested only on synthetic data.

In [2], a real-time tracking system using a cluster of outward-looking custom integrated circuits as smart optical sensors is presented. In [45], LED panels in a room ceiling are used to provide markers for tracking; this cumbersome setup limits its applicability as a convenient tracking system.

In [18], a subspace method for recovering the observer's motion and the depth structure of the scene is presented. The algorithm involves splitting the motion equations into separate equations for translational direction, rotation velocity and relative depth. It was shown that the resultant equations can be solved successively, starting with the equations for translational direction. In [42], Simoncelli presents a linear structure from motion algorithm using spherical cameras. Spherical projection has the advantage of treating all viewing directions homogeneously, and it allows writing the optical flow equation in a simple form. In [14], we present an algorithm to recover motion parameters using the concept of Polar Correlation of optical flow. The approach presented is similar to [18] and [42]. The primary difference is that we show that we can cancel rotation terms in opposite cameras in a multi-camera rig arrangement. In [5], the instantaneous model of optical flow is presented. And in [30], it was shown that translational optical flow lies in the direction of the line connecting the point and the focus of expansion. These work form the theoretical foundation of our work.

In [28], a technique for computing direction of translation and axis of rotation using the antipodal points, for omnidirectional cameras, is presented. We generalize the technique for opposing cameras in a multi camera rig, using the instantaneous model for cameras shifted from the rig center. Our technique is a generalization of [28] because it is not restricted to using omnidirectional cameras, does not require the center of projection of each camera to be at the center of the rig and it can also

be applied to a pair of opposing cameras which are displaced from the rig center by different amounts.

For a vision based controller to be adopted in user interface applications it must function in real-time, have long range, be accurate, be convenient to use and be low cost. Our approach works in real time and does not require markers, making it a practical tracking approach for virtual and augmented reality applications.

CHAPTER 3: TRACKING ALGORITHM

The schematic design of our device is shown in Figure 3.1 (a). Our device is designed as a multi camera rig with four cameras C_k (for $k = 1$ to 4), placed as a rigid orthogonal configuration of two pairs of opposing cameras. Figure 3.1 (b) shows the position s_k and orientation m_k of each camera with respect the rig coordinate system. Figure 3.1 (c) shows a prototype of the device that we built using off-the-shelf webcams, for testing purposes. This section presents our algorithm for conducting five degrees of freedom tracking, by computing the direction of translation and angular velocity. This information is integrated over time to get position information of the device at each time instant.

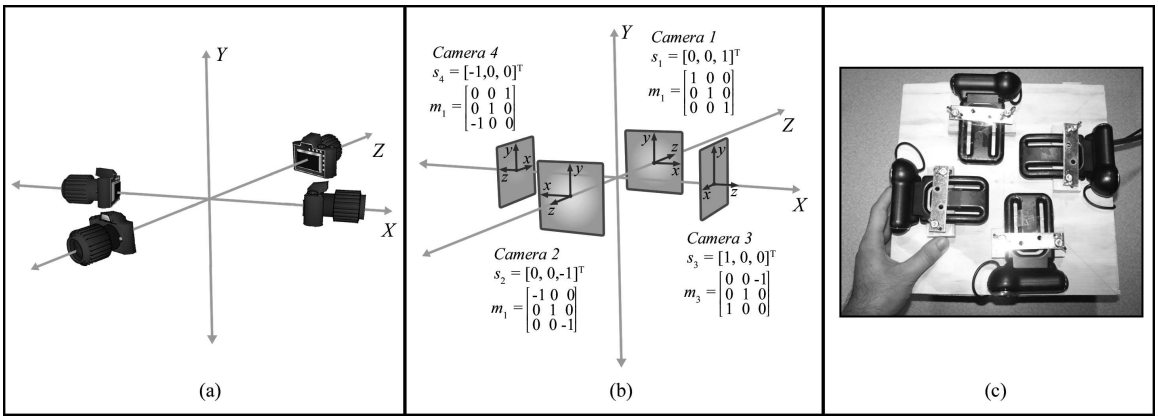

Figure 3.1: (a) Schematic diagram of the rig, (b) Position and orientation of each camera in the rig, (c) Prototype of the device.

SECTION 3.1 Direction of Translation

SECTION 3.1.1 Instantaneous Model of Optical Flow

Given two successive images of a scene, the motion of each pixel in the first image to the second image is defined as a vector $[u, v]^T$, called Optical Flow, where u and v are velocity components in x and y direction respectively. Using the instantaneous model of optical flow [5], for a camera C_k the optical flow vector $[u^k, v^k]^T$ at point $P(x, y)$ can be written as:

$$u^k = u_t^k + u_r^k, \qquad v^k = v_t^k + v_r^k, \tag{3.1}$$

$$u_t^k = \frac{-t_x^k + xt_z^k}{Z}, \quad v_t^k = \frac{-t_y^k + yt_z^k}{Z}, \tag{3.2}$$

$$u_r^k = \omega_x^k xy - \omega_y^k(x^2+1) + \omega_z^k y, \quad v_r^k = \omega_x^k(y^2+1) - \omega_y^k xy - \omega_z^k x, \tag{3.3}$$

where $[u_t^k, v_t^k]^T$ are the translational components and $[u_r^k, v_r^k]^T$ are the rotational components of optical flow, $t^k = [t_x^k, t_y^k, t_z^k]^T$ is the translation and $\omega^k = [\omega_x^k, \omega_y^k, \omega_z^k]^T$ is the angular velocity of camera C_k and Z is the z component (depth) of the 3D point corresponding to the image point $P(x, y)$.

SECTION 3.1.2 Instantaneous Model of Optical Flow for Shifted Cameras

Figure 3.1 (b) shows the position and orientation of each camera in the rig. Following [43], for a camera shifted from the origin:

$$t^k = m_k[(\omega \times s_k) + T], \tag{3.4}$$

$$\omega^k = m_k \omega, \tag{3.5}$$

where t^k is the translation and ω^k is the angular velocity of camera C_k, placed at position s_k with orientation m_k, and $T = [T_x, T_y, T_z]^T$ is the translation and $\omega = [\omega_x, \omega_y, \omega_z]^T$ is the angular velocity of the rig. As shown in Figure 3.1 (b), for camera 1:

$$s^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad m^1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

Substituting equation (3.6) in equations (3.4) and (3.5), we get:

$$t^1 = \begin{bmatrix} \omega_y + T_x \\ -\omega_x + T_y \\ T_z \end{bmatrix}, \quad \omega^1 = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{3.7}$$

Using equation (3.1), and substituting equation (3.7) in equations (3.2) and (3.3), we get:

$$u^1 = \frac{-\omega_y - T_x + xT_z}{Z} + \omega_x xy - \omega_y(x^2 + 1) + \omega_z y, \tag{3.8}$$

$$v^1 = \frac{\omega_x - T_y + yT_z}{Z} + \omega_x(y^2 + 1) - \omega_y xy - \omega_z x. \tag{3.9}$$

Equations (3.8) and (3.9) represent the optical flow in camera 1 in terms of the rig motion parameters T and ω. Similarly equations for camera 2 can also be derived, and they are:

$$u^2 = \frac{-\omega_y + T_x - xT_z}{Z} - \omega_x xy - \omega_y(x^2 + 1) - \omega_z y, \tag{3.10}$$

$$v^2 = \frac{-\omega_x - T_y - yT_z}{Z} - \omega_x(y^2 + 1) - \omega_y xy + \omega_z x. \tag{3.11}$$

SECTION 3.1.3 Polar Correlation

Consider four symmetric points of the form $Q^0(x, y)$, $Q^1(-x, y)$, $Q^2(-x, -y)$ and $Q^3(x, -y)$. Let the flow vector at these symmetric points for camera C_k be $[u^k_{Q^i}, v^k_{Q^i}]^T$ (for $i = 0$ to 3). The equations for flow vectors at these symmetric points in camera 1 and camera 2 can be obtained by substituting the coordinates of these points in terms of x and y in equations (3.8) and (3.9) for camera 1 and equations (3.10) and (3.11) for camera 2. The equations for optical flow at these symmetric points in camera 1 are:

$$u^1_{Q^0} = \frac{-\omega_y - T_x + xT_z}{Z} + \omega_x xy - \omega_y(x^2 + 1) + \omega_z y, \tag{3.12}$$

$$v^1_{Q^0} = \frac{\omega_x - T_y + yT_z}{Z} + \omega_x(y^2+1) - \omega_y xy - \omega_z x, \tag{3.13}$$

$$u^1_{Q^1} = \frac{-\omega_y - T_x - xT_z}{Z} - \omega_x xy - \omega_y(x^2+1) + \omega_z y, \tag{3.14}$$

$$v^1_{Q^1} = \frac{\omega_x - T_y + yT_z}{Z} + \omega_x(y^2+1) + \omega_y xy + \omega_z x, \tag{3.15}$$

$$u^1_{Q^2} = \frac{-\omega_y - T_x - xT_z}{Z} + \omega_x xy - \omega_y(x^2+1) - \omega_z y, \tag{3.16}$$

$$v^1_{Q^2} = \frac{\omega_x - T_y - yT_z}{Z} + \omega_x(y^2+1) - \omega_y xy + \omega_z x, \tag{3.17}$$

$$u^1_{Q^3} = \frac{-\omega_y - T_x + xT_z}{Z} - \omega_x xy - \omega_y(x^2+1) - \omega_z y, \tag{3.18}$$

$$v^1_{Q^3} = \frac{\omega_x - T_y - yT_z}{Z} + \omega_x(y^2+1) + \omega_y xy - \omega_z x. \tag{3.19}$$

We compute a quantity $[L^k_x, L^k_y]$ for camera C_k as:

$$L_x^k = \frac{\sum_{i=0}^{3} u_{Q^i}^k}{4}, \qquad L_y^k = \frac{\sum_{i=0}^{3} v_{Q^i}^k}{4}. \tag{3.20}$$

$$L_x^1 = \frac{-\omega_y - T_x}{Z} - \omega_y(x^2+1), \qquad L_y^1 = \frac{\omega_x - T_y}{Z} + \omega_x(y^2+1). \tag{3.21}$$

Similarly we can derive the equations for $[L_x^k, L_y^k]$ for cameras 2, 3 and 4. For camera 2:

$$L_x^2 = \frac{-\omega_y + T_x}{Z} - \omega_y(x^2+1), \qquad L_y^2 = \frac{-\omega_x - T_y}{Z} - \omega_x(y^2+1). \tag{3.22}$$

Next we compute a quantity $[G_x, G_y, G_z]$ as:

$$G_x = [-L_x^1 + L_x^2]/2, \tag{3.23}$$

$$G_y = [-L_y^1 - L_y^2 - L_y^3 - L_y^4]/4, \tag{3.24}$$

$$G_z = [L_x^3 - L_x^4]/2. \tag{3.25}$$

By substituting equations (3.21) and (3.22) in equation (3.23) we get:

$$G_x = T_x/Z. \tag{3.26}$$

Note that G_x has reduced to a scaled version of T_x. By substituting $[L_x^k, L_y^k]$ for all the four cameras in equations (3.24) and (3.25), we get:

$$G_y = T_y/Z. \tag{3.27}$$

$$G_z = T_z/Z. \tag{3.28}$$

$[G_x, G_y, G_z]$ is the scaled version of translation $T = [T_x, T_y, T_z]^T$ of the rig. The computation of $[G_x, G_y, G_z]$ cancels all the rotation terms and we are left with only translation terms. This is the concept of Polar Correlation, which says that opposing cameras have common component of optical flow, which we show can be canceled out to get the direction of translation of the rig. It should be noted that two pairs of opposing cameras are needed because using camera 1 and 2 only G_x and G_y can be computed, to compute G_z we need another pair of opposing cameras.

SECTION 3.2 Angular Velocity

After computing the scaled version of translation of the rig, we can obtain the translation t_k of each camera C^k using

$$t_k = m_k T, \tag{3.29}$$

where m_k is the orientation of camera C^k and T is the translation of the rig. Using the translation of each camera, we can generate a synthetic translational field $[u_t^k,\ v_t^k]$ for each camera as

$$u_t^k = \frac{-t_x^k + x t_z^k}{Z}, \quad v_t^k = \frac{-t_y^k + y t_z^k}{Z}. \tag{3.30}$$

We generate a synthetic translational flow vector for all the points where we have computed the optical flow. In [30], it was shown that the translational component of optical flow at point P always lie on the line joining the focus of expansion (FOE) and the point P. Therefore, by using the synthetic translational flow at each point we can get the direction of the line joining the point P and the FOE. When a camera C_k is shifted from the rig center, an optical flow vector in that camera has three components, $\vec{o} = \vec{o}_t + \vec{o}_{tr} + \vec{o}_r$, where $\vec{o}_t$ is a translational component due to

translation of the rig, $\vec{o}_{tr}$ is a translational component due to rotation of the rig, [$(\omega \times s_k)$ term in equation (3.4)], and $\vec{o}_r$ is a rotational component due to rotation of the rig. The component of optical flow perpendicular to the line connecting the FOE and the point P has projection of the translational component $\vec{o}_{tr}$ due to the rotation of the rig and rotational component $\vec{o}_r$ due to rotation of rig. This concept is pictorially represented in Figure 3.2. Let $\vec{F}$ be the vector going from the FOE (x_0, y_0) to a point $P(x, y)$ on the image plane. Then,

$$\vec{F} = (x - x_0)\hat{i} + (y - y_0)\hat{j}. \tag{3.31}$$

Let $\vec{F}_p$ be a vector perpendicular to $\vec{F}$, obtained by 90° rotation. Then,

$$\vec{F}_p = (y_0 - y)\hat{i} + (x - x_0)\hat{j}. \tag{3.32}$$

Now we normalize the vector $\vec{F}_p$ to obtain

$$\vec{F}_{pn} = \frac{(y_0 - y)}{\sqrt{(y_0 - y)^2 + (x - x_0)^2}}\hat{i} + \frac{(x - x_0)}{\sqrt{(y_0 - y)^2 + (x - x_0)^2}}\hat{j}. \tag{3.33}$$

The translational component of the optical flow always lies on the line connecting the image point and the FOE. Therefore, $\vec{o}_t$ will always lie on the line joining point

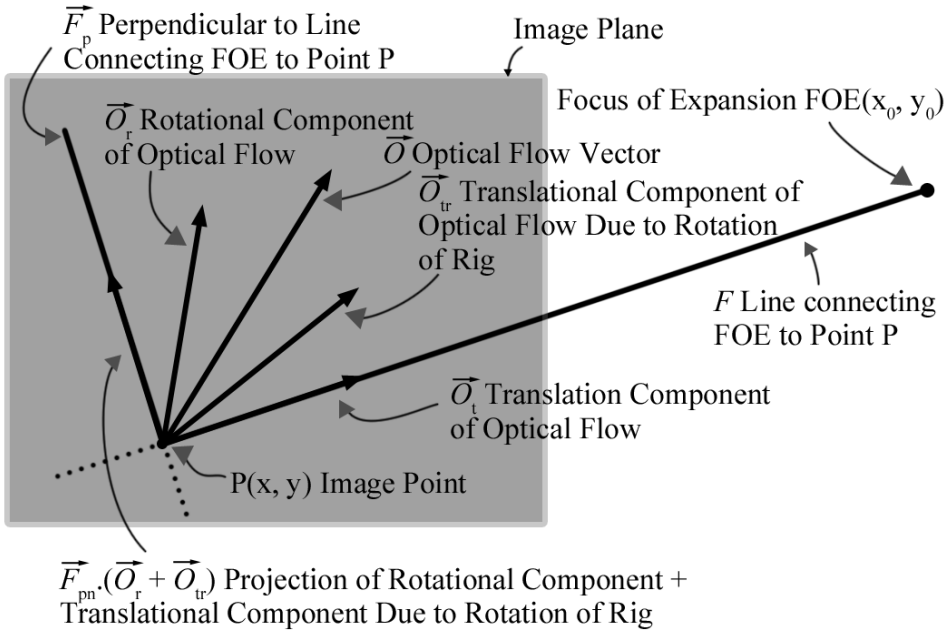

Figure 3.2: Components of optical flow for a camera shifted from the rig center.

P and the FOE. Now we take the component of $(\vec{o}_r + \vec{o}_{tr})$ perpendicular to $\vec{F}$, by taking a dot product between $(\vec{o}_r + \vec{o}_{tr})$ and $\vec{F}_{pn}$. The motive behind normalizing $\vec{F}_p$ is that the component of $(\vec{o}_r + \vec{o}_{tr})$ perpendicular to $\vec{F}$ can be found directly by computing a dot product between $(\vec{o}_r + \vec{o}_{tr})$ and $\vec{F}_{pn}$. Using the computed optical flow over the image frames from the four cameras, on selected interest points, we can obtain the numerical value

$$\eta = (\vec{o}_r + \vec{o}_{tr}).\vec{F}_{pn}, \tag{3.34}$$

by taking the projection of the flow vector perpendicular to the direction of the line connecting the point to the FOE. By extracting the rotational component and

translational component due to the rotation of the rig from equations (3.8) and (3.9) for Camera 1, we get

$$\vec{o_r} = [\omega_x xy - \omega_y(x^2+1) + \omega_z y]\hat{i} + [\omega_x(y^2+1) - \omega_y xy - \omega_z x]\hat{j}, \tag{3.35}$$

$$\vec{o_{tr}} = \frac{-\omega_y}{Z}\hat{i} + \frac{\omega_x}{Z}\hat{j}. \tag{3.36}$$

Substituting equations (3.33), (3.35) and (3.36) in equation (3.34) we get an equation of the form:

$$\omega_x c_1 + \omega_y c_2 + \omega_z c_3 = \eta, \tag{3.37}$$

where c_1, c_2 and c_3 are in terms of x_0, y_0, x, y and Z. Similarly we can also get equations for cameras 2, 3 and 4. Thus each point in a camera gives us one equation in 3 unknowns ω_x, ω_y and ω_z. We choose a uniform constant value for depth Z. Using all the points in all the four cameras we obtain a family of equations of the form:

$$PX = Q, \tag{3.38}$$

where $X = [\omega_x, \omega_y, \omega_z]^T$, P is a $N \times 3$ matrix and Q is a $N \times 1$ vector of known quantities, for a total of N interest points. Pre-multiplying both sides of equation (3.38) by P^T gives us

$$P^T P X = P^T Q. \tag{3.39}$$

Since $P^T P$ is a 3×3 matrix and $P^T Q$ is a 3×1 vector, we get three linear equations in three unknowns, which can be easily solved to get ω_x, ω_y and ω_z, which is the angular velocity of the rig.

SECTION 3.3 Tracking 3D Position

We assume that initially the multi-camera rig is aligned with and placed at the origin of the world coordinate system. Thus, the multi-camera rig has a starting position $P_0 = (0, 0, 0)$ and orientation $Q_0 = I$ (Identity matrix). We obtain a translation vector T_i from the calculated scaled version of translation and a rotation matrix R_i from the calculated rotation terms for each time frame i. The rotation matrix can be obtained from the calculated rotation terms by treating them as Euler angles $[\Phi_1, \Phi_2, \Phi_3]$ of rotation using the formula:

$$R = \begin{bmatrix} c_3c_2 & c_3s_2s_1 - s_3c_1 & s_3s_1 + c_3s_2c_1 \\ s_3c_2 & c_3c_1 + s_3s_2s_1 & s_3s_2c_1 - c_3s_1 \\ -s_2 & c_2s_1 & c_2c_1 \end{bmatrix}, \tag{3.40}$$

where $c_n = \cos\Phi_n$ and $s_n = \sin\Phi_n$. The position P_i and the orientation Q_i of the multi-camera rig can be calculated using the formulas:

$$P_i = P_{i-1} + Q_{i-1} * T_i, \tag{3.41}$$

$$Q_i = R_i * Q_{i-1}. \tag{3.42}$$

Using equations (3.41) and (3.42) we integrate the calculated relative translation and rotation to obtain the position of the multi-camera rig at each time frame.

CHAPTER 4: RECOVERING MOTION PARAMETERS

In this chapter we present the three different techniques developed for recovering motion parameters of the rig using computed optical flow from the four cameras.

SECTION 4.1 Pattern Matching Technique

The first technique we developed for computing the rig's motion is a qualitative method of pattern matching. This approach matches computed optical flow from the four cameras of the rig with the synthetic optical flow generated using the displacement model for cameras shifted from the rig center. Using this technique, we can recover the rotation angles and the direction of translation, over a pre-defined small quantized set of values.

SECTION 4.1.1 Displacement Model of Optical Flow for Shifted Cameras

In [29], for a camera C_k, image point displacement $[u,\ v]^T$ is formulated as:

$$u^k = \frac{R_1^k \cdot (P - t^k/Z)}{R_3^k \cdot (P - t^k/Z)} - x, \tag{4.1}$$

$$v^k = \frac{R_2^k \cdot (P - t^k/Z)}{R_3^k \cdot (P - t^k/Z)} - y, \tag{4.2}$$

where $t_k = [t_x^k,\ t_y^k,\ t_z^k]^T$ is the translation, R^k is the rotation matrix, the subscript to the rotation matrix denotes each row of the rotation matrix, and Z is the z component (depth) of the 3D point corresponding to the image point P $(x,\ y,\ 1)$. When a camera is shifted from the rig center the motion parameters of the camera are related to the motion parameters of the rig by the following equations:

$$R^k = m_k R {m_k}^T, \tag{4.3}$$

$$t^k = m_k T + m_k(Rs_k - s_k). \tag{4.4}$$

By substituting values of orientation and position for a camera in equations (4.3) and (4.4) and then substituting R^k and t^k in equations (4.1) and (4.2), we can get image point displacement equations for a camera shifted from the rig center in terms of the rig center's motion parameters.

SECTION 4.1.2 Algorithm

Generating Synthetic Flow In generating a synthetic displacement field, the most important parameter that governs the representation is the choice of size of the synthetic image plane. If the size of the image plane is too big, dramatic flow patterns occur, such as drastic curves on the edges of the plane. While these patterns may be realistic for some systems, in practice, optical flow appears as a homogeneous pattern. Such a pattern is achieved by selecting small dimensions for the image plane, as cameras generally have small field of view. Thus, a coordinate range of $[-0.15,\ 0.15]$ is used for the synthetic image plane. A constant value for depth, $Z = 10$, is used for all the points. In practice, it was observed that a small set, $\{-2,\ -1,\ 0,\ 1,\ 2\}$, of translation and rotation values generates an expressive set of optical flow patterns consistent with real optical flow. Using this range, we generated $15,625$ unique optical flow patterns. The generated optical flow is passed through the quadrantization process to get 16 flow vectors corresponding to each quadrant of the four cameras. To optimize the search space we cluster the $15,625$

cases using the k-means algorithm, with $k = \sqrt{15625} = 125$.

INPUT: Flow fields for all the four cameras

OUTPUT: Rotation angles and direction of translation of the rig

- Apply quadrantization to the input flow fields to get 16 flow vectors.
- Find the cluster which has the minimum Euclidean distance between the 16 flow vectors of the input flow field and the cluster center.
- Inside that cluster, find the entry which has the minimum Euclidean distance between it and the 16 flow vectors of the input flow field.
- The motion parameters used to generate that entry is considered to be the motion parameters of the rig that generated the input flow field.

SECTION 4.2 Antipodal Regions Technique

This technique is an extension to [28], by generalizing the approach to multi-camera rigs using the instantaneous model for shifted cameras.

SECTION 4.2.1 Antipodal Theory

For a plane P passing through the center of a sphere S, $P \cap S$ defines a great circle. Consider a pair of antipodal points and a great circle passing through those points.

Following [28], if the projection of optical flow vectors, at the two antipodal points, on the tangent vector to the great circle are in the same direction, then the direction of translation is constrained to be parallel to the great circle, in the direction opposite the direction of the projections of the optical flow. If the projections are in the opposite direction, then the axis of rotation is constrained to lie normal to the plane of the great circle with a direction determined by left hand rule on the projections.

Antipodal Theory for Opposing Cameras Consider two antipodal points, $P\ (x,\ y)$ in Camera 1 and $P'\ (x,\ -y)$ in Camera 2. The equation of optical flow at P' in Camera 2 can be obtained as we obtained equations (3.8) and (3.9) for Camera 1. Optical flow vectors at P and P' with respect to the rig center can be obtained using

$$\vec{o}_{rig} = m_k \vec{o}_k, \tag{4.5}$$

where $\vec{o}_{rig}$ is the optical flow vector with respect to the rig center, m_k is the orientation of camera C^k, and $\vec{o}_k$ is the optical flow vector in camera C^k with respect to the local camera center. Consider a great circle passing through points P and P' such that the tangent vector $\vec{h}$ to it is

$$\vec{h} = 1\hat{i} + 1\hat{j}. \tag{4.6}$$

The projection of optical flow at point P and P', with respect to the rig center, on the tangent to the great circle can be obtained using

$$proj(\vec{o}_{rig}) = \vec{o}_{rig} \cdot \vec{h}. \tag{4.7}$$

Thus, the projection of optical flow vectors at P and P' can be obtained.

$$\begin{aligned} proj(\vec{o}^{p}_{rig}) = [&\frac{-\omega_y - T_x + xT_z}{Z} + \omega_x xy - \omega_y(x^2+1) + \omega_z y] \\ +[&\frac{\omega_x - T_y + yT_z}{Z} + \omega_x(y^2+1) - \omega_y xy - \omega_z x], \end{aligned} \tag{4.8}$$

$$\begin{aligned} proj(\vec{o}^{p'}_{rig}) = [&\frac{\omega_y - T_x + xT_z}{Z} - \omega_x xy + \omega_y(x^2+1) - \omega_z y] \\ +[&\frac{-\omega_x - T_y + yT_z}{Z} - \omega_x(y^2+1) + \omega_y xy + \omega_z x]. \end{aligned} \tag{4.9}$$

Note that in equations (4.8) and (4.9) the translation terms have the same sign and the rotation terms have opposite sign. Following the argument made in [28], if the sign of $proj(\vec{o}^{p}_{rig})$ and $proj(\vec{o}^{p'}_{rig})$ is the same then there must exist a component of translation that is parallel and in opposite direction to $proj(\vec{o}^{p}_{rig})$ and $proj(\vec{o}^{p'}_{rig})$. If the sign is

different, then a component of rotation must exist, with an axis of rotation perpendicular to the great circle.

SECTION 4.2.2 Algorithm

Pre-defined Antipodal Regions and Great Circles Two points on a sphere are antipodal if they are diametrically opposite. That is, if a line can be drawn from a point p to another point p' while passing through the center of the sphere then p and p' are called antipodal points. We define antipodal regions as two regions such that all points contained in one region have a corresponding antipodal point in the opposing region. The center point of an antipodal region is associated with a flow vector, which is the average of optical flow vectors for all the points in that region. Quadrantization is applied to partition the image plane into four regions and to get the average flow vector associated with the center of each region. The antipodal regions are chosen to be the diametrically opposite quadrants after the quadrantization process. For each pair of antipodal regions we pre-defined 6 great circles passing through the center of them, such that they are 30°s apart in sequence.

Computing DOT and AOR Having pre-defined antipodal regions and great circles, and given as input the computed flow fields from all the four cameras we can obtain an estimate of the direction of translation (DOT) and axis of rotation (AOR) using the following: Loop through all antipodal regions (R_i and R_i') with flow vectors $\vec{o}_{R_i}$ and $\vec{o}_{R_i'}$

- Loop through all great circles (C_j)
  - Take the projection of $\vec{o}_{R_i}$ and $\vec{o}_{R'_i}$ onto the tangent vector of C_j
  - If $sign(proj(\vec{o}_{R_i})) = sign(proj(\vec{o}_{R'_i}))$ then the DOT is constrained to be opposite in direction to the projections and parallel to C_j.
  - Else if $sign(proj(\vec{o}_{R_i})) \neq sign(proj(\vec{o}_{R'_i}))$ then the AOR is constrained as the normal vector to C_j in the direction determined by the left hand rule on the projections.
  - If either projection is zero, no constraint is added.

By constraining the DOT and AOR for all the antipodal regions and corresponding great circles, we get an estimate of the DOT and AOR as output.

SECTION 4.3 Polar Correlation Technique

We base this technique on the instantaneous model for cameras shifted from the rig center, and give an estimate of the direction of translation and angular velocity of a moving multi camera rig. The direction of translation and the angular velocity is calculated using the concept of polar correlation of optical flow, SECTION 3.1 and SECTION 3.2.

SECTION 4.3.1 Algorithm

Repeat steps 1 through 9 for each set of four frame from all the four cameras.

- Step 1: Find interest points of the image frames from all the four cameras, [38].
- Step 2: Calculate optical flow on the interest points, [3].
- Step 3: Estimate the flow at the quadrantization points, SECTION 5.3.
- Step 4: Using the estimated flow at the quadrantization points, calculate $[L_x^k, L_y^k]$, equation (3.20).
- Step 5: Calculate $[G_x, G_y, G_z]$, using equation (3.23) through (3.25).
- Step 6: Obtain the direction of translation for each camera, using equation (3.29).
- Step 7: Generate synthetic optical flow at interest point from Step 1 using the direction of translation for each camera from Step 6.
- Step 8: For each interest point, using the direction of line connecting the point and FOE, and the computed flow at that point, calculate the value of η and construct equation (3.34).
- Step 9: Solve the family of equations to get angular velocity, equation (3.39).

CHAPTER 5: IMPLEMENTATION

SECTION 5.1 Device Design

The configuration of the cameras in the device is shown in Figure 3.1 (a). We used off-the-shelf Logitech webcams to build the prototype, as shown in Figure 3.1 (c), making the device low-cost. The cameras are rigidly fixed together and we used a powered usb hub with an extension cable to connect the cameras to the computer. This configuration gave us the ability to move around in a large space with no additional setup time.

SECTION 5.2 Calculating Optical Flow

We use Intel's OpenCV implementation of [3] to compute optical flow, which is a real time, iterative approach using image pyramids. It gives as output a sparse flow field because it calculates optical flow only on selected feature points. The feature points are selected using the approach of [38]. This approach selects points in an image, with specific characteristics. At point (x, y) in an image I, the image intensity gradients are defined as:

$$I_x(x,y) = \frac{I(x+1,y) - I(x-1,y)}{2},\ I_y(x,y) = \frac{I(x,y+1) - I(x,y-1)}{2}. \tag{5.1}$$

Given

$$H = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{5.2}$$

a point is selected if the minimum eigenvalue of matrix H for that point is greater than a threshold. This will give points which are shaped like a corner. These kind of points are helpful to overcome the aperture problem [1] in computing optical flow and let us utilize a markerless implementation. Note that other more robust feature point selection methods, like [31], could also be used, but they are not necessarily real time.

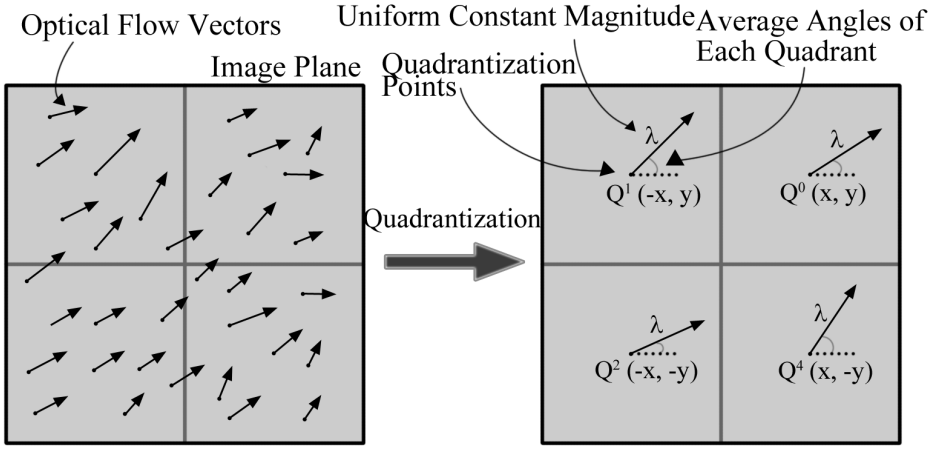

Figure 5.1: Quadrantization Process

SECTION 5.3 Quadrantization

After computing optical flow in each camera, flow vectors from each frame are passed through the Quadrantization step to get an estimate of optical flow at symmetric points $Q^0(x, y)$, $Q^1(-x, y)$, $Q^2(-x, -y)$ and $Q^3(x, -y)$ to use polar correlation. As shown in Figure 5.1 each frame is divided into 4 quadrants. The center points of each quadrant are called Quadrantization Points Q_k^i (for $i = 0$ to 3) for camera C_k. Each quadrantization point is associated with a vector with some uniform constant magnitude λ and angle as the average of all flow vectors' angles in that quadrant.

SECTION 5.4 Procedure

This section enumerates the step by step procedure for implementing the system presented in this work.

1. Build a camera rig with four cameras arranged in a orthogonal configuration of two pairs of opposing cameras. We used Logitech Quickcam Pro webcams. Any other cameras can be used, the camera drivers have to be compatible with OpenCV.

2. We used C Sharp (.NET Framework) for implementation of the algorithm. We also used EmguCV, which is a .NET wrapper for OpenCV. We used XNA Game Studio 3.0 for visualization code, and Bespoke 3DUI Framework for hooking up Nintendo Wii remote (only used for button pressing tasks).

3. The algorithm begins by capturing image from all the four cameras. Then compute optical flow on the images using Pyramidal implementation of Lucas and Kanade algorithm in OpenCV.

4. Apply the Quadrantization process to the computed optical flow to get an estimate of optical flow at the center of the quadrants for all the four cameras.

5. First normalize the estimated optical flow for all the quadrant centers and then give it some constant and uniform magnitude.

6. Average all the normalized estimate of optical flow at the quadrant centers from all the four cameras in all the directions (x, y and z) to get the direction of translation. Multiply this direction of translation with some constant magnitude to get translation of the rig.

7. After obtaining the translation of the rig, pre-multiply it by the orientation of each camera to obtain the translation of each camera. Using the translation of each camera synthetic optical flow can be generated at all the points where we originally calculated optical flow using Bruss and Horn's equations for instantaneous model of optical flow.

8. The generate synthetic optical flow gives the direction of the line connecting the point and the Focus of Expansion (FOE).

9. Take a component of the computed optical flow perpendicular to the direction of the line connection the image point and the FOE. Equate this projected component to the projected equation for rotation.

10. This will give a family of equation in 3 unknowns, solve them to get rotation.

11. We assume that the initial position and orientation of the device coincide with the world coordinate system. After obtaining the translation and rotation of the rig, it can be easily integrated to get the position of the device at each time instant.

CHAPTER 6: EXPERIMENTS AND RESULTS

SECTION 6.1 Motion Reconstruction

To evaluate the accuracy of the device prototype, we compared it to a Polhemus PATRIOT tracker, an electromagnetic (EM) tracker that has position and orientation accuracy of $0.1in$ RMS and 0.75° RMS respectively. The readings from the EM tracker are used as ground truth of the tracked motion. The EM tracker and our device prototype provide measurements in different units. To overcome this issue, the position data from the two devices is normalized using a standard normalization technique [6]. For a given trajectory $S = [s_1, \cdots, s_n]$, Norm(S) is defined as:

$$\left[\left(\frac{s_{1,x}-\mu_x}{\sigma_x}, \frac{s_{1,y}-\mu_y}{\sigma_y}, \frac{s_{1,z}-\mu_z}{\sigma_z}\right), \cdots, \left(\frac{s_{n,x}-\mu_x}{\sigma_x}, \frac{s_{n,y}-\mu_y}{\sigma_y}, \frac{s_{n,z}-\mu_z}{\sigma_z}\right)\right], \tag{6.1}$$

where $s_i = (s_{i,x}, s_{i,y}, s_{i,z})$ is 3D position, μ_x, μ_y and μz are the means and σ_x, σ_y and σ_z are the standard deviations values in x, y and z coordinates respectively. This

normalization makes the distance between two trajectories to be compared invariant to spatial scaling and shifting. For some motion let the trajectory given by the device and the EM tracker be S_n and E_n respectively, for n sample points. Note that the sampling rate of the EM tracker ($\approx 60Hz$) is faster than that of our device ($\approx 16Hz$). However, the trajectory can be recorded with same number of sample points by considering the points with same time stamps. The accuracy of our device compared to the EM tracker is computed using the formulation:

$$A = \left(1 - \frac{\sum_{i=0}^{n} d_i}{n} \right) * 100, \tag{6.2}$$

where d_i is the Euclidean distance between points s_i and e_i for S_n and E_n respectively, obtained after normalization using equation (6.1).

SECTION 6.1.1 Experiments

All experiments were done on real images. The EM tracker was attached to our device and the trajectories formed by both devices were recorded while making motions. The experiment set consists of small motions (around 2 meters) and large motions (around 20 meters). Note that for large motions, the EM tracker did not have enough range so we show the recorded trajectories of our device.

Experiment Set 1

The first set of experiments consists of random 3D shapes made in a lab setting by moving the device around in the air. The trajectories formed by the EM tracker and our device are compared using the formulation of equation (6.2).

Experiment Set 2

The second set of experiments are done on a larger range than the experiments in set 1. The results are compared with the EM tracker, showing how the EM tracker fails when it goes out of range but our device still tracks accurately.

Experiment Set 3

The third set of experiments were done in a hallway and in an outdoor environment, with large range motions to show how the optical device robustly tracks the motion. These open space settings have extreme lighting conditions and sunlight. The EM Tracker fails in such large range scenarios. The specific motions used in this experiment set were rectangles. We chose rectangles in this case to test right angles and how close is the starting point of the rectangle from the ending point. These measures provide a way to evaluate how the tracker is performing in the absence of direct comparisons with the EM tracker.

SECTION 6.1.2 Discussion

Figure 6.1 shows the trajectories formed by our device in red and the EM Tracker in blue, with total accuracy obtained in each motion instance. Figure 6.2 shows the change in accuracy over time for these instances. It can be seen that the average accuracy of the our device is around 80%, and it is maintained over 185 frames. The frame rate of our device is ≈ 16 Hz, which means that for motions of about 11 seconds the system attains up to 80% accuracy, making the tracker well suited for gesture recognition applications, as typically a gesture lasts less than 5 seconds. Figure 6.3 shows how moving out of the range of the EM tracker cause it to jitter but our device still keeps tracking with good accuracy. Figure 6.4 shows large range motion instances in the hallway and outdoor settings. It can be seen that the our device tracks with reasonable accuracy, though a drift can be seen. The starting and the ending points do not coincide even though the actual motion was made so that the starting and ending points were approximately the same. However, the drift is small as compared to the total range of the motion and the device is able to track the right angles in the rectangles well. Although our experiments show our device tracks fairly well, we recognize that it is still in the prototype stage and more work is needed to improve its accuracy.

We do not explicitly computer the depth of the $3D$ points. Thus, we use a uniform constant value of Z for all the points. We are only inferring the direction of translation, rather than with scale, so the chosen value of Z should not have any effect on the direction of translation. But it might have some effect on the rotation calculation. Thus

an analysis of the algorithm by choosing different Z values would be interesting. Not been able to compute the depth of the 3D points limits the algorithm and results in drift in the tracked motion.

SECTION 6.2 Gesture Recognition

We used a simple and intuitive classifier to test our motion parameter estimation techniques in a gesture recognition application.

SECTION 6.2.1 Feature Extraction

Consider that a gesture motion is made using the device, which is n frames long. Using any one of the three techniques presented, we can get 6 motion parameters for each frame. Let E_G be a $n \times 6$ matrix of all the estimated motion parameters for that given instance of the gesture, using any one of the three techniques. Using E_G we create a $1 \times m$ feature vector. Features are selected based on the criteria that a gesture should be order specific. In order to preserve the chronology of the gesture motions, E_G is divided into i row blocks. From each row block the signed and unsigned average magnitudes for each column are calculated. We do this for $i = 2, 3, and\, 4$. Thus, the total length of the feature vector is $m = 2 \times 6 \times (2 + 3 + 4) = 108$.

SECTION 6.2.2 Gesture Classifier

Given a set of feature vectors corresponding to a set of training gestures, a linear classifier calculates an average representation for each gesture class. Specifically, for each class, the arithmetic mean among all training instances for that class is calculated via element-wise signed averaging. Thus, for 15 gestures, 15 average feature vectors are calculated. Given the feature vector that corresponds to a test gesture, the distance from that vector to each average vector from the training set is calculated. A gesture motion instance is classified as belonging to the class corresponding to the average vector with the minimum distance from its feature vector. The distance between a feature vector and an average feature vector from the training set is calculated using p-norm with $p = 3$.

SECTION 6.2.3 Experiments

We defined a set of 15 gestures to use in our experiments, as shown in Figure 6.5. All experiments were done on real data, in a lab setting and a hallway.

Dataset

The training dataset consists of 600 gesture instances. Two participants performed 40 instances for each of the 15 gesture classes, half of the instances were recorded in a lab setting and the other half in a hallway. The testing dataset consists of 150 gesture

instances. One participant, different from the participants who collected the training dataset, performed 10 instances for each of the 15 gesture classes, half of the instances were recorded in a lab setting and the other half in a hallway.

SECTION 6.2.4 Results and Discussion

Tables 6.2, 6.3 and 6.4 show the confusion tables obtained after training on all gesture instances from the training dataset and testing on all gesture instances of the testing dataset for the Pattern Matching, Antipodal Regions and Polar Correlation techniques, respectively. The Pattern Matching technique achieves an average accuracy of 88.0%, the Antipodal Regions technique achieves an average accuracy of 90.7%, and the Antipodal Regions technique achieves an average accuracy of 86.7%. All three techniques achieve comparable average accuracy and these levels are well suited for gesture recognition applications. Table 6.1 shows the average accuracy achieved with the Pattern Matching technique on different number of cameras and different camera configurations. The results clearly shows that in the Pattern Matching technique case, a multi-camera system is superior to a single camera and the highest average accuracy is achieved with an orthogonal configuration of two pairs of opposing cameras.

Cameras	Accuracy
1	60.00%
1,4	68.33%
1,2	71.33%
1,2,3	74.33%
1,2,3,4	88.00%

Table 6.1: Average accuracy for different camera configurations and number of cameras, using the Pattern Matching Technique.

	B.S.	Ca.	Ci.	F.S.	In.	L.B.	Pi.	Re.	R.B.	S	St.	Tr.	Tw.	V	Z
Back Slash	**50**	50	0	0	0	0	0	0	0	0	0	0	0	0	0
Caret	10	**90**	0	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	**90**	0	0	10	0	0	0	0	0	0	0	0	10
Forward Slash	0	0	0	**100**	0	0	0	0	0	0	0	0	0	0	0
Infinity	0	0	0	0	**90**	0	0	0	10	0	0	0	0	0	0
Left Bracket	0	0	0	0	0	**100**	0	0	0	0	0	0	0	0	0
Pigtail	0	0	0	0	10	0	**70**	0	0	0	0	0	0	0	20
Rectangle	0	0	0	0	0	0	0	**90**	0	0	0	10	0	0	0
Right Bracket	0	0	0	0	0	0	0	0	**100**	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	**100**	0	0	0	0	0
Stab	0	0	0	0	0	0	10	0	0	0	**90**	0	0	0	0
Triangle	0	0	0	0	0	0	10	0	0	20	0	**70**	0	0	0
Twist	0	0	0	0	0	0	0	0	0	0	0	0	**100**	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	**100**	0
Z	10	0	0	0	0	0	0	0	10	0	0	0	0	0	**80**

Table 6.2: Confusion Table for the Pattern Matching Technique; Average Accuracy = 88%.

	B.S.	Ca.	Ci.	F.S.	In.	L.B.	Pi.	Re.	R.B.	S	St.	Tr.	Tw.	V	Z
Back Slash	**50**	50	0	0	0	0	0	0	0	0	0	0	0	0	0
Caret	0	**100**	0	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	**90**	0	0	0	0	0	0	0	0	0	0	0	10
Forward Slash	0	0	0	**100**	0	0	0	0	0	0	0	0	0	0	0
Infinity	0	0	0	0	**90**	0	0	0	0	0	0	0	0	0	10
Left Bracket	0	0	0	0	0	**90**	0	0	0	0	0	0	10	0	0
Pigtail	0	0	0	0	0	0	**80**	0	0	0	0	0	0	0	20
Rectangle	0	0	0	10	0	0	0	**90**	0	0	0	0	0	0	0
Right Bracket	0	0	0	0	0	0	0	0	**100**	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	**100**	0	0	0	0	0
Stab	0	0	0	0	0	0	0	0	0	0	**90**	0	10	0	0
Triangle	0	0	0	0	0	0	0	0	0	0	0	**100**	0	0	0
Twist	0	0	0	10	0	0	0	0	0	0	0	0	**90**	0	0
V	0	0	10	0	0	0	0	0	0	0	0	0	0	**90**	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**100**

Table 6.3: Confusion Table for the Antipodal Regions Technique; Average Accuracy = 90.7%.

	B.S.	Ca.	Ci.	F.S.	In.	L.B.	Pi.	Re.	R.B.	S	St.	Tr.	Tw.	V	Z
Back Slash	**30**	50	0	10	0	0	0	0	0	0	0	0	0	10	0
Caret	0	**100**	0	0	0	0	0	0	0	0	0	0	0	0	0
Circle	0	0	**70**	0	0	0	0	0	0	0	0	0	0	30	0
Forward Slash	0	0	0	**100**	0	0	0	0	0	0	0	0	0	0	0
Infinity	0	0	0	0	**100**	0	0	0	0	0	0	0	0	0	0
Left Bracket	0	0	0	0	0	**100**	0	0	0	0	0	0	0	0	0
Pigtail	0	0	0	0	0	0	**90**	0	0	0	0	0	0	0	10
Rectangle	0	0	0	0	0	0	0	**90**	0	10	0	0	0	0	0
Right Bracket	0	0	0	0	0	0	0	0	**70**	0	0	0	10	20	0
S	0	0	0	0	0	0	0	0	0	**100**	0	0	0	0	0
Stab	0	0	0	0	0	0	0	0	0	0	**100**	0	0	0	0
Triangle	0	0	0	0	0	0	0	0	0	0	0	**100**	0	0	0
Twist	0	10	0	0	0	0	0	0	0	0	0	0	**90**	0	0
V	0	0	10	0	10	0	0	0	10	0	0	0	0	**70**	0
Z	0	0	0	0	10	0	0	0	0	0	0	0	0	0	**90**

Table 6.4: Confusion Table for the Polar Correlation Technique; Average Accuracy = 86.7%.

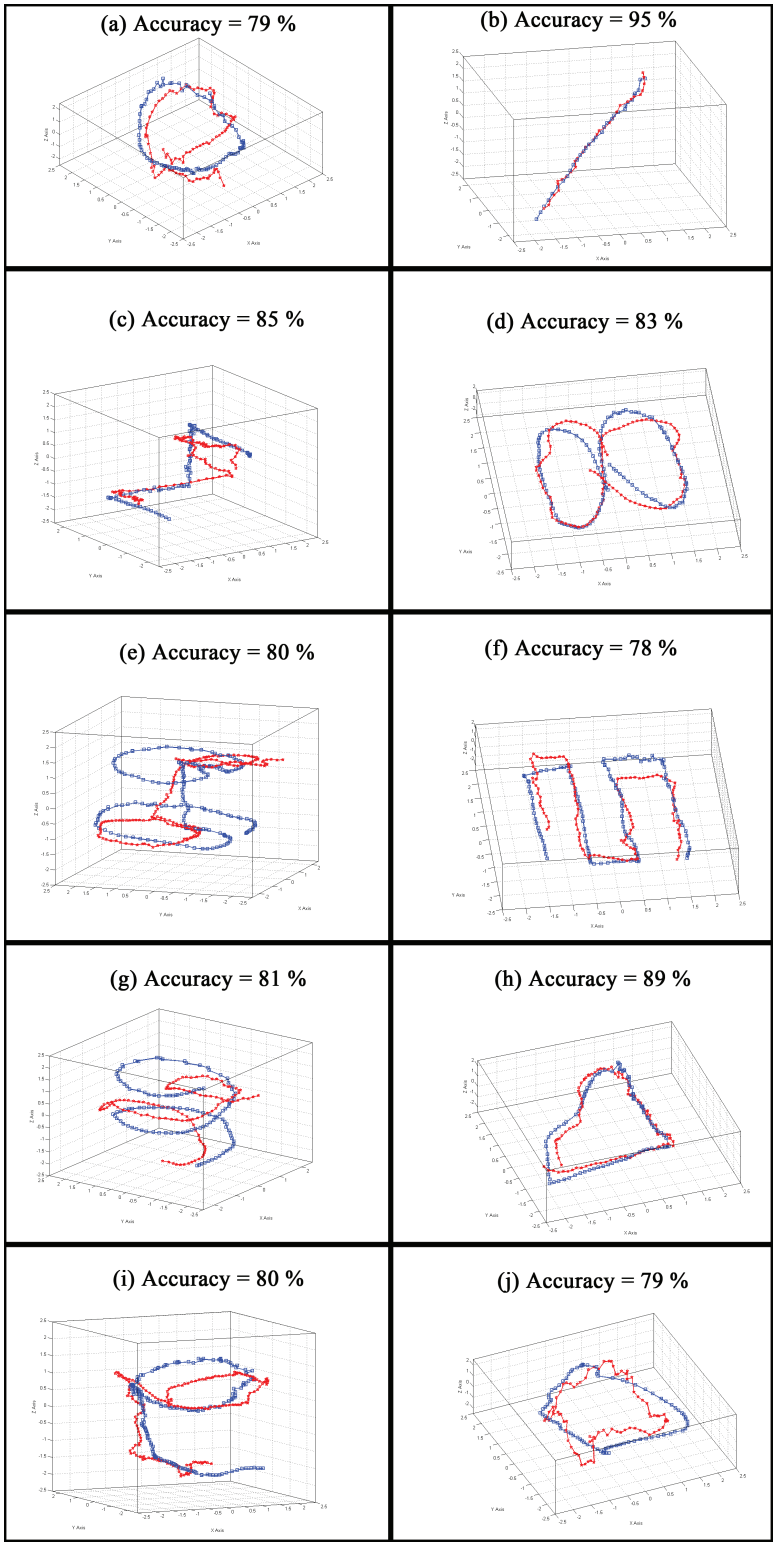

Figure 6.1: Experiments set 1 with accuracy of the optical device as compared to EM tracker, optical device is shown in red and the EM tracker in blue

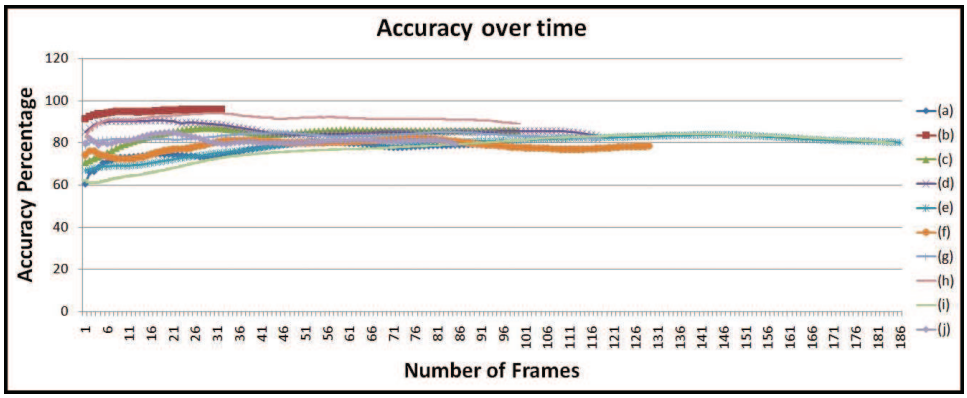

Figure 6.2: Change in accuracy over time of motions instances from experiment set 1

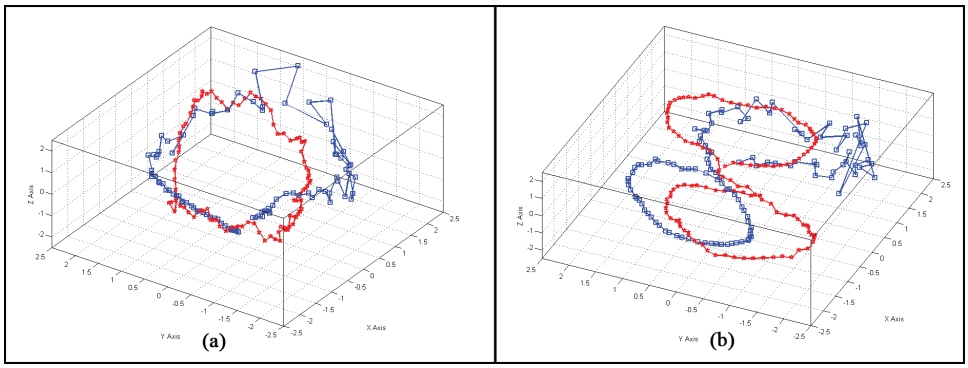

Figure 6.3: Trajectories from experiment set 2 showing how the optical device tracks accurately, when we move out of the range of the EM tracker, and the EM tracker gives jittery data, optical device is shown in red and EM tracker in blue

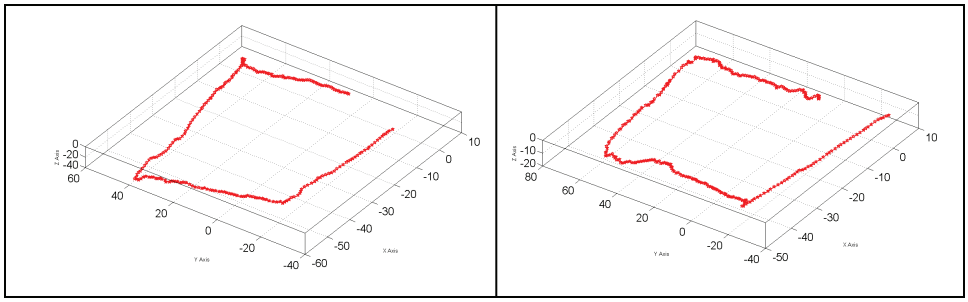

Figure 6.4: Trajectories of large range motion instances from experiment set 3 in outdoor and hallway settings

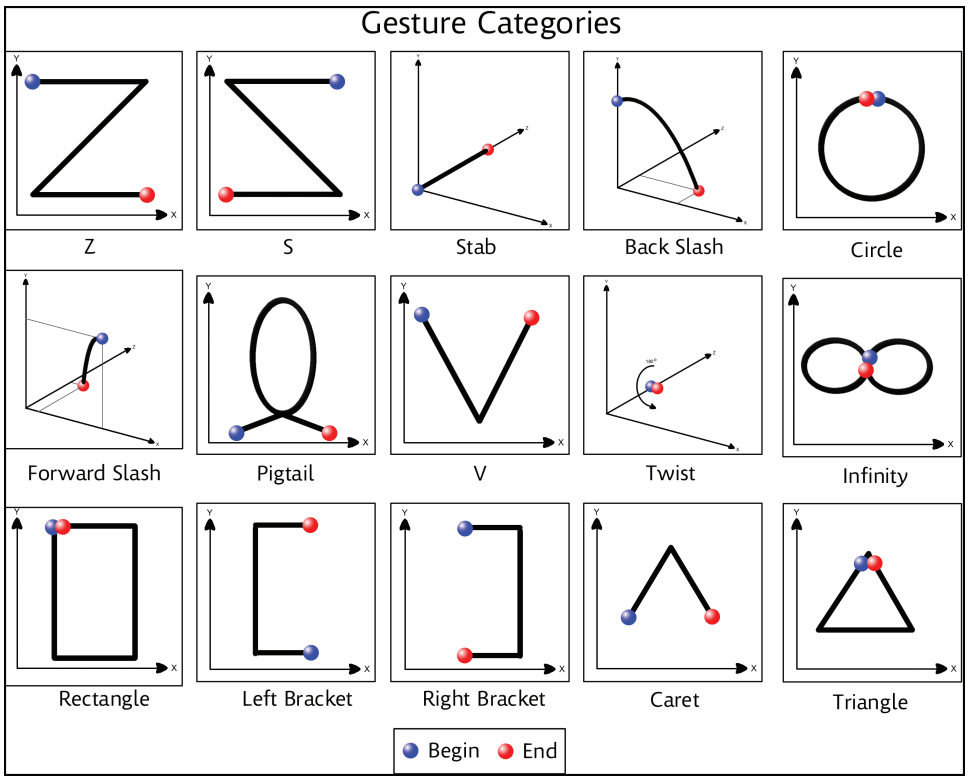

Figure 6.5: Gesture Categories.

CHAPTER 7: FUTURE WORK AND CONCLUSION

Future work includes improving the tracking accuracy by reducing the drift. A limitation of the current system is that we are only able to recover 5 degree of freedom, and thus due to the lack of absolute scale, we can see drift in the tracked motion. This can be overcome if some loop closing mechanism is employed, like by making a database of feature point descriptors of the environment and the system be able to recognize that it has returned to a previously visited location by comparing the input image feature point descriptors to the database.

Also analyzing the performance of the algorithm by dividing the image plane into more than 4 regions. Dividing the image plane into 4 regions was a design decision we made. But it would be interesting to look at the behavior of the algorithm by using more regions. SFM and egomotion algorithms generally works well for certain motion parameters and does not work so well for others. It would be interesting to characterize the error behavior of the algorithm presented over a large range of possible motions.

In conclusion, we presented a markerless, real time, vision-based tracking system that makes use of the novel concept of Polar Correlation of optical flow. Experiments show

that the device has an average accuracy of 80% over 185 frames when compared to an electromagnetic tracker. The prototype of the device is low cost, requires no setup and has a large range span. We have also presented three different techniques for recovering motion parameters at different levels of complete recovery, using optical flow from opposing cameras. These techniques were applied to a gesture recognition application using a simple classifier. The results show the Antipodal Regions technique achieved the highest recognition accuracy level (90.7%) compared with the Pattern Matching and Polar Correlations techniques. We recognize that their is more work to be done to improve the gesture recognition accuracy, by examining more sophisticated gesture classification algorithms and fine tuning our motion parameter recovery techniques. However, we believe that these results provide a great starting point for using opposing cameras in vision-based inside-looking-out systems in gesture recognition applications.

LIST OF REFERENCES

[1] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[2] Gary Bishop and Henry Fuchs. The self-tracker: A smart optical sensor on silicon. *Proceedings, Conference on Advanced Research in VLSI*, 1984.

[3] Jean Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2002.

[4] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

[5] Anna R. Bruss and Berthold K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21(1):3–20, 1983.

[6] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 491–502. ACM, 2005.

[7] Brian Clipp, Jae-Hak Kim, Jan-Michael Frahm, Marc Pollefeys, and Richard Hartley. Robust 6dof motion estimation for non-overlapping, multi-camera systems. *Applications of Computer Vision, IEEE Workshop on*, 0:1–8, 2008.

[8] Andrew I. Comport, Eric Marchand, Muriel Pressigout, and Fran?ois Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, 2006.

[9] James W. Davis and Serge Vaks. A perceptual user interface for recognizing head gesture acknowledgements. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, pages 1–7, New York, NY, USA, 2001. ACM.

[10] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[11] Geanbry Demming. Sony eyetoy ^{TM}: Developing mental models for 3-D interaction in a 2-D gaming environment. In *Computer Human Interaction*, volume 3101, pages 575–582. Springer, 2004.

[12] W. T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma. Computer vision for computer games. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0:100, 1996.

[13] William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michal Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo Kyuma,

Yasunari Miyake, and Ken ichi Tanaka. Computer vision for interactive computer graphics. volume 18, pages 42–53, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[14] Prince Gupta, Niels da Vitoria Lobo, and Joseph J. LaViola Jr. Markerless tracking using polar correlation of camera optical flow. *IEEE Virtual Reality Conference*, 2010.

[15] Perttu Hämäläinen and Johanna Höysniemi. A computer vision and hearing based user interface for a computer game for children. In *Universal Access Theoretical Perspectives, Practice, and Experience*, pages 299–318. Springer Berlin / Heidelberg, 2003.

[16] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2003.

[17] Johanna Höysniemi, Perttu Hämäläinen, Laura Turkki, and Teppo Rouvi. Children's intuitive gestures in vision-based action games. *Communications of the ACM*, 48(1):44–50, 2005.

[18] Allan Jepson and David J. Heeger. Linear subspace methods for recovering translational direction. pages 39–62, 1992.

[19] M. Kaess and F. Dellaert. Visual slam with a multi-camera rig. Technical Report GIT-GVU-06-06, Georgia Institute of Technology, Feb 2006.

[20] Jae-Hak Kim, Hongdong Li, and R. Hartley. Motion estimation for multi-camera systems using global optimization. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[21] Jae-Hak Kim, Hongdong Li, and Richard Hartley. Motion estimation for non-overlapping multi-camera rigs: Linear algebraic and l_∞ geometric solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2009.

[22] Jun-Sik Kim, Myung Hwangbo, and T. Kanade. Motion estimation using multiple non-overlapping cameras for small unmanned aerial vehicles. pages 3076–3081, 2008.

[23] Reinhard Koch, Kevin Koeser, Birger Streckel, and Jan-Friso Evers-Senne. Markerless image-based 3d tracking for real-time augmented reality applications. Montreux, Switzerland, 2005.

[24] M. S. Lee, D. Weinshall, E. Cohen Solal, A. Colmenarez, and D. M. Lyons. A computer vision system for on-screen item selection by finger pointing. In *Computer Vision and Pattern Recognition*, pages I:1026–1033, 2001.

[25] Hongdong Li and Richard Hartley. Five-point motion estimation made easy. *Pattern Recognition, International Conference on*, 1:630–633, 2006.

[26] Hongdong Li, Richard Hartley, and Jae-Hak Kim. A linear approach to motion estimation using generalized camera models. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[27] John Lim and Nick Barnes. Estimation of the epipole using optical flow at antipodal points. *IEEE International Conference on Computer Vision*, 0:1–6, 2007.

[28] John Lim and Nick Barnes. Directions of egomotion from antipodal points. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[29] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[30] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In *Proc. Royal Society London. B208*, pages 385–397, 1980.

[31] David G. Lowe. Object recognition from local scale-invariant features. In *In Proceedings of the International Conference on Computer Vision*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[32] Thomas Moeslund, Moritz Strring, Moritz St Orring, and Erik Granum. Vision-based user interface for interacting with a virtual environment, 2000.

[33] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–777, 2004.

[34] J. Nocedal and S.J. Wright. *Numerical Optimization*. 1999.

[35] R. Pless. Camera cluster in motion: Motion estimation for generalized camera designs. In *In IEEE Robotics and Automation Magazine*, volume 11, pages 39–44, 2004.

[36] A. Rhalibi, M. Merabti, P. Fergus, and Yuanyuan Shen. Perceptual user interface as games controller. In *IEEE Consumer Communications and Networking Conference*, volume 10, pages 1059–1064, 2008.

[37] Jakub Segen and Senthil Kumar. Gesture vr: Vision-based 3d hand interace for spatial interaction. In *In Proceedings of the Sixth ACM International Conference on Multimedia*, pages 455–464, New York, NY, USA, 1998. ACM.

[38] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.

[39] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics (TOG)*, 2006.

[40] Aravind Sundaresan and Rama Chellappa. Markerless motion capture using multiple cameras. *Computer Vision for Interactive and Intelligent Environment*, 0:15–26, 2005.

[41] Sarah Tariq and Frank Dellaert. A multi-camera 6-dof pose tracker. In *In Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 296–297, Washington, DC, USA, 2004. IEEE Computer Society.

[42] I. Thomas and E. P. Simoncelli. Linear structure from motion. Institute for Research in Cognitive Science Technical Report IRCS-94-26, University of Pennsylvania, December 1994.

[43] An-Ting Tsao, Chiou-Shann Fuh, Yi-Ping Hung, and Yong-Sheng Chen. Ego-motion estimation using optical flow fields observed from multiple cameras. In *Computer Vision and Pattern Recognition*, page 457, Washington, DC, USA, 1997. IEEE Computer Society.

[44] Matthew Turk and Matthew Turk. Moving from guis to puis. In *In Proceedings of Fourth Symposium on Intelligent Information*, 1998.

[45] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. High-performance wide-area optical tracking: The hiball tracking system. *Presence: Teleoperators and Virtual Environments*, 10(1):1–21, 2001.

[46] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–38, 2002.

[47] Andrew D. Wilson and Edward Cutrell. Flowmouse: A computer vision-based pointing and gesture input device. In *In Interact*, 2005.

[48] Andrew Wu, Mubarak Shah, and Niels da Vitoria Lobo. A virtual 3d blackboard: 3d finger tracking using a single camera. In *In Fourth IEEE International Conference On Automatic Face And Gesture Recognition*, pages 536–543, 2000.