TWO CASE STUDIES ON VISION-BASED MOVING OBJECTS MEASUREMENT

A Thesis

by

JI ZHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2011

Major Subject: Computer Engineering

TWO CASE STUDIES ON VISION-BASED MOVING OBJECTS MEASUREMENT

A Thesis

by

JI ZHANG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2011

Major Subject: Computer Engineering

ABSTRACT

Two Case Studies on Vision-based Moving Objects Measurement. (August 2011)

Ji Zhang, B.E., Tsinghua University

Co–Chairs of Advisory Committee: Dr. Dezhen Song
Dr. Wei Yan

In this thesis, we presented two case studies on vision-based moving objects measurement.

In the first case, we used a monocular camera to perform ego-motion estimation for a robot in an urban area. We developed the algorithm based on vertical line features such as vertical edges of buildings and poles in an urban area, because vertical lines are easy to be extracted, insensitive to lighting conditions/shadows, and sensitive to camera/robot movements on the ground plane. We derived an incremental estimation algorithm based on the vertical line pairs. We analyzed how errors are introduced and propagated in the continuous estimation process by deriving the closed form representation of covariance matrix. Then, we formulated the minimum variance ego-motion estimation problem into a convex optimization problem, and solved the problem with the interior-point method. The algorithm was extensively tested in physical experiments and compared with two popular methods. Our estimation results consistently outperformed the two counterparts in robustness, speed, and accuracy.

In the second case, we used a camera-mirror system to measure the swimming motion of a live fish and the extracted motion data was used to drive animation of fish behavior. The camera-mirror system captured three orthogonal views of the fish. We also built a virtual fish model to assist the measurement of the real fish. The fish model has a four-link spinal cord and meshes attached to the spinal cord. We projected the fish model into three orthogonal views and matched the projected views with the real views captured by the camera. Then, we maximized the overlapping area of the fish in the projected views and the real views. The maximization result gave us the position, orientation, and body bending angle for the fish model that was used for the fish movement measurement. Part of this algorithm is still under construction and will be updated in the future.

To my parents

ACKNOWLEDGMENTS

I would like to thank my committee co-chairs, Dr. Song and Dr. Yan, my committee member, Dr. Gutierrez-Osuna, my project collaborators, Dr. Rosenthal and Dr. Johnson, for their valuable guidance and support through my master's program.

Also, thanks to my friends and colleagues in the NetBot lab, the Viz Lab, the Rosenthal Lab at Texas A&M University, and the Jerry Johnson Lab at Brigham Young University for their insightful suggestions and inputs to my research projects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Research on vision-based measurements has gained extensive attention over the recent decade, partly because of the fast developing technology on digital video cameras and the increasing processing power. In this thesis, we present two case studies on vision-based measurement of moving objects. In the first case, we use a monocular camera to measure the ego-motion of a robot moving in urban area. We employ vertical line features and build an algorithm to estimate the robot 2D ego-motion in an incremental format. In the second case, we use a camera-mirror system to measure the swimming motion of a live fish in a tank. The camera-mirror system creates three orthogonal views of the fish. We then build a virtual fish model and reconstruct its swimming motion from the camera video. The rest of this thesis is organized as follows. We present the two case studies in Chapter II and Chapter III, respectively. The conclusion and future work are in Chapter IV.

A. Vision-based Measurement of Robot Ego-motion

In Chapter II, we perform vision-based measurement on the ego-motion of a moving robot. We work on the particular scenario that a small robot with limited onboard computation capability moves in an urban area. Due to the fact that tall buildings in urban areas along the road often form a deep valley and block GPS signals, and wheel encoders and low cost inertial measurement units (IMU) usually cannot provide enough accuracy for the ego-motion estimation, we propose a vision-based method that is suitable for working in urban area.

This thesis follows the style of *IEEE Transactions on Robotics*.

We realize that urban area has an abundant amount of vertical lines. Vertical lines are very easy to extract from camera images because they are all parallel to each other and perpendicular to the ground plane. Manipulating vertical lines in camera images only needs very low computational cost which allows the algorithm runs on low cost onboard computers with limited computation capability. Also, vertical lines are sensitive to the camera horizontal movement because they are perpendicular to the ground. Based on the above consideration, we propose a 2D ego-motion estimation method that estimates the robot horizontal movement using vertical line features.

The method works in an incremental estimation format and minimizes the estimation error variance. We first derive an incremental estimation algorithm based on vertical line pairs. We analyze how errors are introduced and propagated in the continuous estimation process by deriving the closed form representation of covariance matrix. Then, we formulate the minimum variance ego-motion estimation problem into a convex optimization problem and solve the problem with the interior-point method. The algorithm is extensively tested in physical experiments and compared with a point feature based method and a line feature based method. Our estimation results consistently outperform the two counterparts in robustness, speed, and accuracy.

B. Vision-based Measurement of Fish Swimming Motion

In Chapter III, we perform vision-based measurement on the swimming motion of a live fish. The motivation of this work rises from the fact that video playback has become a useful tool for biological study on animal visual communication and its related behaviors. Currently, video playbacks are generated in two ways: (1) hand connecting live video sequences of the animal and (2) using 3D animation tools to create artificial movement of a virtual animal. Both of them are labor-intensive and also often result in confounding artifacts on the animal motion, shape, and texture.

The work in this thesis presents a way of automatically tracking and reconstructing the swimming motion of a live fish. The implementation of this work can reduce labor time for manual video recording, manipulating, and video playback creating. Also since the fish swimming motion is reconstructed from a real fish, it can generate more natural and fish-like swimming motion than existing methods.

To reconstruct the swimming motion, the fish is housed in a tank. We build a camera-mirror system to capture three orthogonal views of the fish in the tank. We also build a virtual fish model from measurements of the real fish. The fish model has a four-link spinal cord and meshes attached to the spinal cord. We project the fish model onto three orthogonal views and match the projected views with the real views captured by the camera. Then, we maximize the overlapping area of the fish in the projected views and the real views. That way, we reconstruct the fish swimming motion.

CHAPTER II

VISION-BASED MEASUREMENT OF ROBOT EGO-MOTION

We report our development of a monocular visual odometry system based on vertical lines such as vertical edges of buildings and poles in the urban area. Since vertical lines are easy to be extracted, insensitive to lighting conditions/shadows, and sensitive to robot movements on the ground plane, they are excellent landmarks. We derive an incremental visual odometry method based on the vertical line pairs. We analyze how errors are introduced and propagated in the continuous odometry process by deriving the closed form representation of covariance matrix. We formulate the minimum variance ego-motion estimation problem and present two different algorithms. The two algorithms have been extensively tested in physical experiments. The error aware odometry method has also been compared with two popular methods and consistently outperforms the two counterparts in robustness, speed, and accuracy. The relative errors of the odometry are less than 2% in physical experiments.

A. Introduction

When a small robot travels in urban area, tall buildings along road side form a deep valley and often block GPS signals. Wheel encoders and low cost inertial measurement units (IMU) cannot provide enough accuracy for ego-motion estimation. Visual odometry becomes an important supplemental motion estimation method for the robot. Although capable of providing motion estimation for all six degrees of freedom, existing visual odometry methods require extensive computation and cannot be trivially scaled down to be implemented on a low power computation platform. We are interested in designing a light-weighted planar motion estimation scheme for

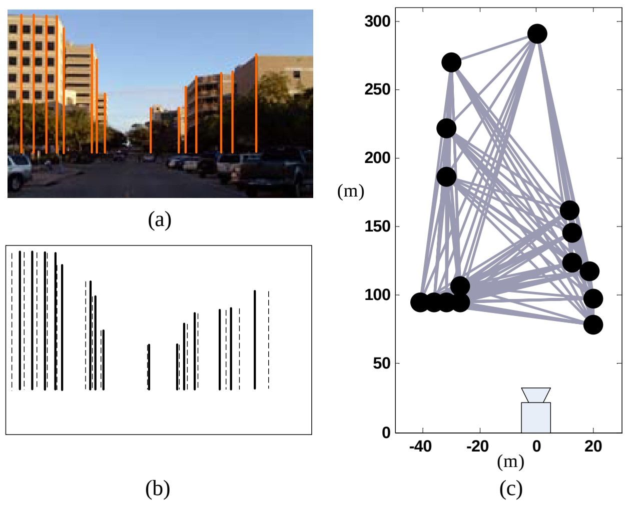


Fig. 1. An illustration of monocular visual odometry using vertical line pairs. (a) An image frame taken by the robot with vertical lines highlighted in orange color. (b) Corresponding vertical lines in two consecutive frames after the robot moving forward along the optical axis direction of the camera by 10 meters. (c) A top view of the vertical edges (black dots in the figure) in (a) and potential choice of pairs (edges between black dots).

those low power platforms.

Urban environments often offer a rich set of structured features. As illustrated in Fig. 1(a), building edges and poles are common features in urban area. Those vertical lines are insensitive to lighting conditions and shadows. They are parallel to each other and to the gravity direction. Extracting parallel lines using the gravity direction as a reference can be done quickly and accurately in a low power computation platform. Moreover, vertical lines are sensitive to robot motion on the ground plane. Hence vertical lines are natural choices for landmarks.

Here we present a visual odometry method based on paired vertical lines for a robot equipped with a single camera. We first show a single pair of vertical lines can provide a minimal solution for estimating the robot ego-motion up to similarity. Since there often exist multiple vertical edges in urban scenes (Fig. 1(b)), there are multiple vertical line pairs. Different choices of the vertical line pairs affect the ego-motion estimation accuracy. We analyze how errors are introduced and propagated in the continuous odometry process by deriving the recursive and closed form representation of error covariance matrix. We formulate the minimum variance ego-motion estimation problem and present an algorithm that outputs weights for different vertical line pairs. The resulting visual odometry method is tested in physical experiments and compared with two existing methods that are based on point features and line features, respectively. Our result outperforms the two counterparts in robustness, accuracy, and speed. The relative errors of our method are less than 2% in experiments.

This chapter combines our two previous conference papers [1, 2] with a comprehensive and complete approach and more experimental results. The rest of this chapter is organized as follows. First we review related work in Section B. We formulate the vertical line pair-based ego-motion estimation problem in Section C. Modeling and analysis of ego-motion estimation for a single vertical line pair are p-resented in Section D. Building on the result, we present the variance minimization method to aggregate motion estimation results from multiple vertical line pairs in Section E. We summarize the two resulting algorithms in Section F. The algorithms are extensively tested in physical experiments in Section G before we conclude the chapter in Section H.

B. Related Work

Visual odometry [3] utilizes images taken from on-board camera(s) to estimate robot motion. It can be viewed as a supplementary method when GPS signals are challenged. Visual odometry has many successful applications including aerial vehicles [4], underwater vehicles [5], legged robots [6], and ground mobile robots [7, 8]. Visual odometry is closely related to simultaneous localization and mapping (SLAM) [9] and can be viewed as a building block for Visual SLAM [10–18]. A better visual odometry method can certainly increase SLAM performance.

Although our work is based on a regular pinhole camera system that follows a minimalism design to save power usage, visual odometry and visual SLAM can be performed with different sensor combinations such as omnidirectional cameras, stereo vision systems, and laser range finders.

An omnidirectional camera has been a popular choice for odometry applications [19]. Scaramuzza and Siegwart propose a real-time visual odometry algorithm for estimating the vehicle ego-motion using the omnidirectional camera [3, 20]. Also using an omnidirectional camera, Wongphati et al. propose a fast indoor SLAM method [21]. New vertical line detection and matching methods have been developed for omnidirectional cameras [22, 23]. In our system, we use a regular camera because the regular camera distributes pixels into space more evenly than that of an omnidirectional camera and hence can achieve better accuracy.

A very popular sensor in visual odometry is the stereo vision system [24, 25]. Nister et al. develop a visual odometry system to estimate the motion of a stereo head or a single camera on a ground vehicle [26, 27]. The stereo vision-based visual odometry used on Mars rovers is a well-known example [28–30]. Laser range finders [31,32] and sonars [33] that provide proximity data can also be used in assisting visual odometry or SLAM. Inspired by the fact that a human can perform odometry with a single eye, we focus on monocular vision-based approaches.

A different way of classifying visual odometry and visual SLAM is based on what kind of features or landmarks have been used. Point features, such as Harris Corners, scale-invariant feature transformation (SIFT) points [34], speed up robust feature (SURF) points [35], or the recently proposed CENter SURround Extremas (CenSurE) feature points [36], are the most popular ones in visual odometry [26] since they are readily available and well developed in computer vision literature. However, point features usually contain a large number of noisy data and must be combined with filtering methods such as RANdom SAmple Consensus (RANSAC) [37, 38] to allow correct correspondence across frames to be found. Such combination usually results in very high computation cost.

Moreover, a point feature mathematically is a singularity in the feature space and sometimes might not have an actual geometric meaning, which could lead to problems if serving as a landmark because we are unsure how robust such singularity would be under different lighting/shadow conditions. Humans do not view a scene as isolated points and are still capable of performing odometry tasks. Lines are often used by humans in estimating distance. Easy to be extracted [39], lines are inherently robust and insensitive to lighting conditions or shadows. Due to this characteristic, line features see many successful applications in visual SLAM [11, 40–44], structure from motion [45, 46], indoor localization [47], scene analysis [48], and camera orientation estimation [49, 50].

Vertical lines are a special class of lines and widely exist in urban environments. Earth gravity forces us to construct buildings with vertical edges. They are inherently parallel to each other and dramatically reduce the feature extraction difficulties [51]. Moreover, they are very sensitive to robot motion on the ground. All of those properties make vertical lines perfect for visual odometry applications. Building on the existing work, we aim to develop a new systematic method to utilize those advantages, which can dramatically reduce the computation cost of odometry without sacrificing accuracy.

C. Problem Definition

We want to estimate the robot motion on the horizontal plane. The robot periodically takes frames to estimate its ego-motion in each step. To setup this ego-motion estimation problem, we begin with assumptions.

1. Assumptions

1. We assume that the initial step of the robot motion is known as a reference. This is the requirement for the monocular vision system. Otherwise the ego-motion estimation is only up to similarity.

2. We assume that the vertical lines, such as poles and building vertical edges, are stationary.

3. We assume that the camera lens distortion is removed by calibration and the camera follows the pinhole camera model with square pixels and a zero skew factor. If not, we can use intrinsic parameters from pre-calibration to correct the discrepancy.

4. For simplicity, we assume the camera image planes are perpendicular to the horizontal plane, and parallel to each other. If not, we can use homography matrices [38] constructed from vanishing points [49] to rotate the image planes to satisfy the condition.

2. Notations and Coordinate Systems

In this chapter, all coordinate systems are right hand systems (RHS). For camera coordinate systems (CCS), we define z-axis as the optical axis of the camera, and let y-axis point upward toward the sky. The optical axis is always parallel to the

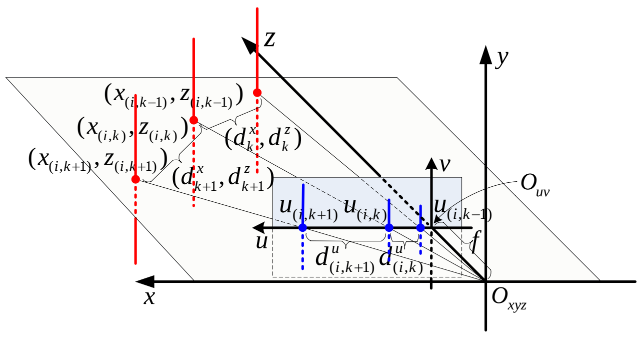


Fig. 2. Superimposed CCSs $x-y-z$ and ICSs $u-v$ for the vertical line i over frames $k-1$, k and $k+1$.

$x-z$ plane which is horizontal. The corresponding image coordinate system (ICS) is defined on the image plane parallel to the $x-y$ plane of CCS with its u-axis and v-axis parallel to x-axis and y-axis, respectively. The optical axis intersects ICS at its origin on the image plane. To maintain RHS, the x-axis of CCS and its corresponding u-axis in ICS must point left (see Fig. 2).

Since image planes are perpendicular to the horizontal plane, and parallel to each other, the corresponding CCSs are iso-oriented during computation. Therefore, the robot ego-motion on the horizontal plane in different CCSs is equivalent to the displacement of vertical lines in a fixed CCS in the opposite direction. The $x-y-z$ coordinate in Fig. 2 illustrates the superimposed CCSs for three consecutive frames $k-1$, k and $k+1$, respectively. At time k, $k \in \mathbb{N}^+$, let $(x_{(i,k-1)}, z_{(i,k-1)})$, $(x_{(i,k)}, z_{(i,k)})$, and $(x_{(i,k+1)}, z_{(i,k+1)})$ be the (x, z) coordinate of the intersection between the corresponding vertical line i and the $x-z$ plane for frames $k-1$, k, and $k+1$, respectively. Let (d^x_k, d^z_k) be the vertical line i's displacement from frame $k-1$ to k, we have $d^x_k = x_{(i,k)} - x_{(i,k-1)}$, $d^z_k = z_{(i,k)} - z_{(i,k-1)}$.

The $u-v$ coordinate in Fig. 2 shows the corresponding superimposed ICSs for frames $k-1$, k and $k+1$. Let $u_{(i,k-1)}$, $u_{(i,k)}$, and $u_{(i,k+1)}$ be the u-coordinate of the intersections between vertical line i and u-axis in frames $k-1$, k, and $k+1$, respectively. Let $d^u_{(i,k)}$ be vertical line i's displacement in ICS from frame $k-1$ to k, we have $d^u_{(i,k)} = u_{(i,k)} - u_{(i,k-1)}$. With the above notations and coordinate systems defined, we will describe our task.

3. Problem Description

Define n as the number of corresponding vertical lines in three consecutive frames $k-1$, k, and $k+1$. Define $I = \{1, 2, ..., n\}$ as the index set of the lines. Let $\mathbf{u}_i = [u_{(i,k-1)}, u_{(i,k)}, u_{(i,k+1)}]^T$ be vertical line i's u-coordinate in frames $k-1$, k, and $k+1$. Given the robot displacement in pervious step, $\mathbf{d}_k = [d^x_k, d^z_k]^T$, we can calculate the displacement of the current step $\mathbf{d}_{k+1} = [d^x_{k+1}, d^z_{k+1}]^T$ using the corresponding vertical line positions in the three images, $\mathbf{u}_{1:n} = \{\mathbf{u}_i, i \in I\}$, as follows,

$$\mathbf{d}_{k+1} = \boldsymbol{F}(\mathbf{d}_k, \mathbf{u}_{1:n}), \tag{2.1}$$

where function $\boldsymbol{F}(\cdot)$ will be determined later in the chapter.

Eq. (2.1) provides a recursive format for us to estimate the robot ego-motion that is represented by $\mathbf{d}_{k+1}$. However, in each step of calculation, errors are brought into the system. We do not know the actual values of $\mathbf{d}_k$ and $\mathbf{u}_i$, which are defined as $\mathbf{d}^*_k$ and $\mathbf{u}^*_i$, respectively. $\mathbf{d}_k$ and $\mathbf{u}_i$ are measurements of $\mathbf{d}^*_k$ and $\mathbf{u}^*_i$, respectively. As a convention in this chapter, we use starred notation a^* to indicate the true value of variable a and define error value e^a of a as $e^a = a^* - a$. Hence we have $\mathbf{e}^{\mathbf{d}}_k = \mathbf{d}^*_k - \mathbf{d}_k$, $\mathbf{e}^{\mathbf{d}}_{k+1} = \mathbf{d}^*_{k+1} - \mathbf{d}_{k+1}$, and $\mathbf{e}^{\mathbf{u}}_i = \mathbf{u}^*_i - \mathbf{u}_i$, where $\mathbf{e}^{\mathbf{u}}_i = [e^u_{(i,k-1)}, e^u_{(i,k)}, e^u_{(i,k+1)}]^T$ describes the measurement error from line segment extraction for line i in frames $k-1$, k, and $k+1$.

Define $\Sigma^{\mathbf{d}}_k$ and $\Sigma^{\mathbf{d}}_{k+1}$ as the covariance matrices for $\mathbf{e}^{\mathbf{d}}_k$ and $\mathbf{e}^{\mathbf{d}}_{k+1}$, respectively. At time k, $\Sigma^{\mathbf{d}}_k$ is known from the previous step. $\Sigma^{\mathbf{d}}_{k+1}$ is influenced by the errors from the previous step, namely, $\Sigma_k^{\mathbf{d}}$, and the new measurement errors $\mathbf{e}_i^{\mathbf{u}}$. For measurement error $\mathbf{e}_i^{\mathbf{u}}$, we assume that each vertical line follows independent and identical Gaussian distribution with zero mean and a variance of σ_u^2. The covariance matrix Σ^u of $\mathbf{e}_i^{\mathbf{u}}$ is a diagonal matrix, $\Sigma^u = \text{diag}(\sigma_u^2, \sigma_u^2, \sigma_u^2)$.

To measure how $\Sigma_{k+1}^{\mathbf{d}}$ changes, we use its trace $\sigma_{k+1}^2 = \text{Tr}(\Sigma_{k+1}^{\mathbf{d}})$ as a metric. Hence our incremental error aware motion estimation problem becomes,

Definition 1 *Given $\mathbf{d}_k$ with $\Sigma_k^{\mathbf{d}}$ and new measurements ($\mathbf{u}_i$, $i \in I$) with Σ^u, derive $\mathbf{F}(\cdot)$ and $\Sigma_{k+1}^{\mathbf{d}}$ while minimizing σ_{k+1}^2 with respect to design options.*

There are two design options: a minimal solution using a single vertical line pair and a multiple vertical line pair-based solution. We begin with the minimal solution.

D. Deriving a Minimum Solution with a Single Vertical Line Pair

With two equations for two unknowns, a pair of vertical lines can offer us a minimum solution for the ego-motion estimation. The minimum solution is a foundation for multiple vertical line-based solutions. The minimum solution can also help us understand how factors, such as locations of vertical lines and relative positions of the lines, affect the solution quality.

Let $\mathbf{u}_i$ and $\mathbf{u}_j$ be the input pair of vertical lines where $i \in I$, $j \in I$, and $i \neq j$. Then (2.1) can be rewritten as,

$$\mathbf{d}_{k+1} = \mathbf{F}_{\mathbf{s}}(\mathbf{d}_k, \mathbf{u}_i, \mathbf{u}_j), \tag{2.2}$$

where $\mathbf{F}_{\mathbf{s}}(\cdot)$ is the motion estimation function for the minimum solution.

1. Deriving $\mathbf{F_s}(\cdot)$

Define f as the camera focal length in units of camera pixel width. Since the camera has square pixels and a zero skew factor, we can reduce the camera to the simple pinhole camera model to obtain the following relationship between $(x_{(l,k)},\ z_{(l,k)})$ and $u_{(l,k)}$, $l = i, j$,

$$u_{(l,k)} = \frac{f x_{(l,k)}}{z_{(l,k)}}, \quad l = i, j. \tag{2.3}$$

Combining (2.3) with $x_{(i,k-1)} = x_{(i,k)} - d^x_k$, $x_{(i,k+1)} = x_{(i,k)} + d^x_{k+1}$, $z_{(i,k-1)} = z_{(i,k)} - d^z_k$, and $z_{(i,k+1)} = z_{(i,k)} + d^z_{k+1}$, we have

$$u_{(i,k-1)} = \frac{f x_{(i,k-1)}}{z_{(i,k-1)}} = \frac{f(x_{(i,k)} - d^x_k)}{z_{(i,k)} - d^z_k}, \tag{2.4}$$

$$u_{(i,k)} = \frac{f x_{(i,k)}}{z_{(i,k)}}, \tag{2.5}$$

$$u_{(i,k+1)} = \frac{f x_{(i,k+1)}}{z_{(i,k+1)}} = \frac{f(x_{(i,k)} + d^x_{k+1})}{z_{(i,k)} + d^x_{k+1}}. \tag{2.6}$$

Combining (2.4-2.6) to eliminate $x_{(i,k)}$ and $z_{(i,k)}$, we have

$$d^x_{k+1} + a_i d^z_{k+1} = b_i, \tag{2.7}$$

where $a_i = -\frac{u_{(i,k+1)}}{f}$, $b_i = \frac{u_{(i,k+1)} - u_{(i,k)}}{u_{(i,k)} - u_{(i,k-1)}}(d^x_k - \frac{u_{(i,k-1)}}{f} d^z_k)$.

Similarly, we have the following for vertical line j,

$$d^x_{k+1} + a_j d^z_{k+1} = b_j, \tag{2.8}$$

where $a_j = -\frac{u_{(j,k+1)}}{f}$, $b_j = \frac{u_{(j,k+1)} - u_{(j,k)}}{u_{(j,k)} - u_{(j,k-1)}}(d^x_k - \frac{u_{(j,k-1)}}{f} d^z_k)$.

Combine (2.7) and (2.8), we have the $\mathbf{F_s}(\cdot)$ function

$$\mathbf{d}_{k+1} = \mathbf{F_s}(\mathbf{d}_k, \mathbf{u}_i, \mathbf{u}_j) = \mathbf{M}^{-1}_{k+1}\mathbf{M}_k \mathbf{d}_k, \tag{2.9}$$

where

$$\mathbf{M}_k = \begin{bmatrix} f(u_{(i,k+1)} - u_{(i,k)}) & -u_{(i,k-1)}(u_{(i,k+1)} - u_{(i,k)}) \\ f(u_{(j,k+1)} - u_{(j,k)}) & -u_{(j,k-1)}(u_{(j,k+1)} - u_{(j,k)}) \end{bmatrix},$$

$$\mathbf{M}_{k+1} = \begin{bmatrix} f(u_{(i,k)} - u_{(i,k-1)}) & -u_{(i,k+1)}(u_{(i,k)} - u_{(i,k-1)}) \\ f(u_{(j,k)} - u_{(j,k-1)}) & -u_{(j,k+1)}(u_{(j,k)} - u_{(j,k-1)}) \end{bmatrix}.$$

2. Computing Jacobian Matrices

When errors are brought into the system, (2.2) becomes

$$\mathbf{d}_{k+1} + \mathbf{e}^{\mathbf{d}}_{k+1} = \mathbf{F}_{\mathbf{s}}(\mathbf{d}_k + \mathbf{e}^{\mathbf{d}}_k, \mathbf{u}_i + \mathbf{e}^{\mathbf{u}}_i, \mathbf{u}_j + \mathbf{e}^{\mathbf{u}}_j). \tag{2.10}$$

Since we are interested in how errors propagate, we want to derive the following relationship from (2.10),

$$\mathbf{e}^{\mathbf{d}}_{k+1} = \boldsymbol{G}(\mathbf{e}^{\mathbf{d}}_k, \mathbf{e}^{\mathbf{u}}_i, \mathbf{e}^{\mathbf{u}}_j). \tag{2.11}$$

When errors are small, function $\boldsymbol{G}$ can be approximated by a linear expression

$$\mathbf{e}^{\mathbf{d}}_{k+1} = \boldsymbol{P}_{(i,j)}\mathbf{e}^{\mathbf{d}}_k + \boldsymbol{Q}_{(i,j)}\mathbf{e}^{\mathbf{u}}_i + \boldsymbol{Q}_{(j,i)}\mathbf{e}^{\mathbf{u}}_j, \tag{2.12}$$

where $\boldsymbol{P}_{(i,j)} = \partial\mathbf{F}_{\mathbf{s}}/\partial\mathbf{e}^{\mathbf{d}}_k$, $\boldsymbol{Q}_{(i,j)} = \partial\mathbf{F}_{\mathbf{s}}/\partial\mathbf{e}^{\mathbf{u}}_i$ and $\boldsymbol{Q}_{(j,i)} = \partial\mathbf{F}_{\mathbf{s}}/\partial\mathbf{e}^{\mathbf{u}}_j$ are Jacobian matrices. Note that $\boldsymbol{Q}_{(i,j)}$ and $\boldsymbol{Q}_{(j,i)}$ are for vertical line i and j, respectively.

Obtaining Jacobian matrices is necessary for studying how errors propagate. It is possible to take an algebraic approach. However, it is more intuitive to use a geometric approach, which helps understand the error propagation process.

Fig. 3 illustrates the geometric approach. Let l_i be the line described by (2.7), which intersects with d^x-axis at b_i with angle α_i. Recalling that a_i is defined in (2.7), we have $\tan\alpha_i = -1/a_i = f/u_{(i,k+1)}$. Similarly, let l_j be the line described by (2.8), which intersects with d^x-axis at b_j with angle α_j. Also, we have $\tan\alpha_j = 1/a_j =$

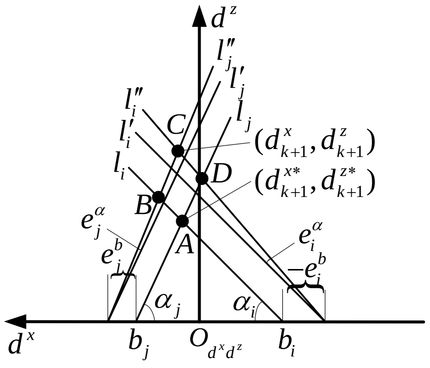


Fig. 3. Computing Jacobian matrices using a geometric approach. Point A at $(d^{x*}_{k+1}, d^{z*}_{k+1})$ is the unknown true location of the displacement.

$-f/u_{(j,k+1)}$. l_i and l_j intersect at point A, which is the robot displacement $\mathbf{d}_{k+1}$.

Let e^α_i, e^b_i be the parameter errors of α_i, b_i, respectively. Due to the existence of e^b_i, l_i shifts to l'_i, where l'_i is a line parallel to l_i. Due to the existence of e^α_i, l'_i shifts to l''_i, where l''_i is a line intersects with l'_i on d^x-axis. Let e^α_j and e^b_j be the parameter errors of α_j and b_j, respectively. Similarly, we have lines l'_j and l''_j. Accordingly, the intersection between l_i, l_j becomes that of l''_i, l''_j, locating at point C, which is the estimated displacement $\mathbf{d}_{k+1}$. The difference between C and A is the robot ego-motion estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$.

Let B be the intersection between l_i and l''_j and D be the intersection between l''_i and l_j. Since e^α_i and e^α_j are both very small, we can approximate $ABCD$ as a parallelogram. Thus, we have

$$e^x_{k+1} = |\overline{AB}|\cos\alpha_i - |\overline{AD}|\cos\alpha_j, \tag{2.13}$$

$$e^z_{k+1} = |\overline{AB}|\sin\alpha_i + |\overline{AD}|\sin\alpha_j. \tag{2.14}$$

From the geometry relationship, we have

$$|\overline{AD}| = -\frac{e^b_i \sin(\alpha_i)}{\sin(\alpha_i+\alpha_j)} + \frac{e^\alpha_i (b_j - b_i)\sin(\alpha_j)}{\sin^2(\alpha_i+\alpha_j)}. \tag{2.15}$$

Let e^a_i be the parameter error of a_i in (2.7), Since $a_i = -1/\tan\alpha_i$, we have $e^a_i = e^\alpha_i/\sin^2\alpha_i$. At the same time, applying $\tan\alpha_i = f/u_{(i,k+1)}$ and $\tan\alpha_j = -f/u_{(j,k+1)}$, (2.15) becomes

$$|\overline{AD}| = \eta/\sin\alpha_j, \tag{2.16}$$

where $\eta = \frac{-fe^b_i}{u_{(i,k+1)}-u_{(j,k+1)}} + \frac{f^2e^a_i(b_j-b_i)}{(u_{(i,k+1)}-u_{(j,k+1)})^2}$. Similarly, we have

$$|\overline{AB}| = \mu/\sin\alpha_i, \tag{2.17}$$

where $\mu = \frac{fe^b_j}{u_{(i,k+1)}-u_{(j,k+1)}} + \frac{f^2e^a_j(b_j-b_i)}{(u_{(i,k+1)}-u_{(j,k+1)})^2}$. Substituting (2.16) and (2.17) into (2.13) and (2.14), and using $\tan\alpha_i = f/u_{(i,k+1)}$ and $\tan\alpha_j = -f/u_{(j,k+1)}$, we have

$$e^x_{k+1} = \mu u_{(i,k+1)}/f + \eta u_{(j,k+1)}/f, \tag{2.18}$$

$$e^z_{k+1} = \mu + \eta. \tag{2.19}$$

Recalling a_i in (2.7), we have

$$e^a_i = -e^u_{(i,k+1)}/f. \tag{2.20}$$

From (2.4) and (2.5), we have $u_{(i,k)} - u_{(i,k-1)} = (fd^x_k - u_{(i,k-1)}d^z_k)/z_{(i,k)}$. Recall that $d^u_{(i,k)}$ is the u-displacement of vertical line i in the superimposed ICS from frame $k-1$ to k, we have $d^u_{(i,k)} = u_{(i,k)} - u_{(i,k-1)}$. After deriving b_i in (2.7) and substituting the above equations, we have

$$\begin{aligned} e^b_i =& e^u_{(i,k+1)}\frac{z_{(i,k)}}{f} - e^u_{(i,k)}\frac{z_{(i,k)}}{f}(1+\frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}) \\ &+ e^u_{(i,k-1)}\frac{z_{(i,k-1)}}{f}\frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} + (e^x_k - \frac{u_{(i,k-1)}}{f}e^z_k)\frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}. \end{aligned} \tag{2.21}$$

Substituting (2.20), (2.21), b_i, and b_j into the expression of η in (2.16), and applying the same substitution that we used in (2.21), we have the expression of η, and similarly, the expression of μ in (2.17). Then, we substitute η and μ into (2.18) and (2.19). Finally, we can obtain the Jacobian matrices shown in (2.22) and (2.23) in the following,

$$\boldsymbol{P}_{(i,j)} = \frac{\begin{bmatrix} \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(j,k+1)} - \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} u_{(i,k+1)} & -\frac{d^u_{(i,k+1)}}{d^u_{(i,k)} f} u_{(j,k+1)} u_{(i,k-1)} + \frac{d^u_{(j,k+1)}}{d^u_{(j,k)} f} u_{(i,k+1)} u_{(j,k-1)} \\ \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} f - \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} f & -\frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(i,k-1)} + \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} u_{(j,k-1)} \end{bmatrix}}{u_{(j,k+1)} - u_{(i,k+1)}}, \tag{2.22}$$

$$\boldsymbol{Q}_{(i,j)} = \frac{\begin{bmatrix} \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(j,k+1)} z_{(i,k-1)} & -(1 + \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}) u_{(j,k+1)} z_{(i,k)} & u_{(j,k+1)} z_{(i,k+1)} \\ \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} f z_{(i,k-1)} & -(1 + \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}) f z_{(i,k)} & f z_{(i,k+1)} \end{bmatrix}}{u_{(j,k+1)} - u_{(i,k+1)}}. \tag{2.23}$$

3. Sensitivity Analysis

With Jacobian matrices ready, we can analyze how errors are introduced and propagated over the computation. The first analysis we conduct is to study which dimension of the ego-motion estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$ is more suspectable to the error introduced by line detection. In this case, matrix $\boldsymbol{Q}_{(i,j)}$ is scrutinized. We have the following result.

Theorem 1 *Let $Q^{gh}_{(i,j)}$ be the (g,h)-th entry of $\mathbf{Q}_{(i,j)}$. If the camera horizontal field of view (HFOV) $\leqslant 50^\circ$, then $|Q^{1h}_{(i,j)}/Q^{2h}_{(i,j)}| \leqslant 0.46$, $h = 1, 2, 3$.*

Proof From (2.23), we have

$$Q^{1h}_{(i,j)}/Q^{2h}_{(i,j)} = u_{(j,k+1)}, \quad h = 1, 2, 3. \tag{2.24}$$

Since the HFOV $\leqslant 50^\circ$, we have

$$-\tan 25^\circ \leqslant \frac{x_{(j,k+1)}}{z_{(j,k+1)}} \leqslant \tan 25^\circ. \tag{2.25}$$

Combining (2.25) with (2.3), we have

$$-0.46 \leqslant u_{(j,k+1)} \leqslant 0.46. \tag{2.26}$$

Thus

$$|Q^{1h}_{(i,j)}/Q^{2h}_{(i,j)}| \leqslant 0.46, \quad j = 1, 2, 3. \tag{2.27}$$

■

This theorem indicates that the introduced error in x-direction is smaller than that in z-direction. The result could also be explained by Fig. 3, where point C only moves inside $\angle BAD$. Since the HFOV $\leqslant 50^\circ$, angles α_i and α_j are bounded inside set $[65^\circ, 90^\circ]$. Hence the quadrilateral $ABCD$ is long in d^z-direction and narrow in d^x-direction. Since a regular camera has HFOV less than 50°, the conclusion is that the depth error is at least twice more than the lateral error.

Another interesting question is how the ego-motion estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$ relates to the position of the vertical line pair. In other words, if there are many vertical line pairs available in the scene, how to find the pair that provides the most accurate ego-motion estimation. Define $\delta^u_{k+1} = u_{(i,k+1)} - u_{(j,k+1)}$ as the distance between the two vertical lines in ICS. Recall that $z_{(i,k+1)}$ is the depth of vertical line i. We have,

Theorem 2 *$\partial|Q^{2h}_{(i,j)}|/\partial|\delta^u_{k+1}| \leqslant 0$, $\partial|Q^{gh}_{(i,j)}|/\partial z_{(i,k+1)} \geqslant 0$, $g = 1, 2$, $h = 1, 2, 3$.*

Proof The first step is to prove $\partial|Q^{2h}_{(i,j)}|/\partial|\delta^u_{k+1}| \leqslant 0$, $h = 1, 2, 3$. We prove the inequality for the case of $h = 1$ because other cases can be proved similarly. From (2.23), we have

$$Q^{21}_{(i,j)} = -\frac{d^u_{(i,k+1)} z_{(i,k-1)}}{d^u_{(i,k)}(u_{(i,k+1)} - u_{(j,k+1)})}. \tag{2.28}$$

Since $\delta^u_{k+1} = u_{(i,k+1)} - u_{(j,k+1)}$, we have

$$\frac{\partial Q^{21}_{(i,j)}}{\partial \delta^u_{k+1}} = \frac{d^u_{(i,k+1)} z_{(i,k-1)}}{d^u_{(i,k)} {\delta^u_{k+1}}^2} = -\frac{Q^{21}_{(i,j)}}{\delta^u_{k+1}}. \tag{2.29}$$

For the case when $Q^{21}_{(i,j)} \geqslant 0$ and $\delta^u_{k+1} > 0$, or the case when $Q^{21}_{(i,j)} < 0$ and $\delta^u_{k+1} < 0$, we have

$$\partial \left|Q^{21}_{(i,j)}\right| / \partial \left|\delta^u_{k+1}\right| = -Q^{21}_{(i,j)} / \delta^u_{k+1} \leqslant 0. \tag{2.30}$$

For the case when $Q^{21}_{(i,j)} \geqslant 0$ and $\delta^u_{k+1} < 0$, or the case when $Q^{21}_{(i,j)} < 0$ and $\delta^u_{k+1} > 0$, we have

$$\partial \left|Q^{21}_{(i,j)}\right| / \partial \left|\delta^u_{k+1}\right| = Q^{21}_{(i,j)} / \delta^u_{k+1} \leqslant 0. \tag{2.31}$$

Thus,

$$\partial \left|Q^{21}_{(i,j)}\right| / \partial \left|\delta^u_{k+1}\right| \leqslant 0. \tag{2.32}$$

The second step is to prove $\partial |Q^{gh}_{(i,j)}| / \partial z_{(i,k+1)} \geqslant 0$, $g = 1, 2$, $h = 1, 2, 3$. Here we only prove the inequality for $g = 1$ and $h = 1$. All other cases with different g and h values can be proved similarly. From (2.23), we have

$$Q^{11}_{(i,j)} = -\frac{d^u_{(i,k+1)} u_{(j,k+1)} z_{(i,k-1)}}{d^u_{(i,k)} (u_{(i,k+1)} - u_{(j,k+1)})}. \tag{2.33}$$

Substituting $z_{(i,k-1)} = z_{(i,k+1)} - d^z_k - d^z_{k+1}$ into (2.33), we have

$$\frac{\partial Q^{11}_{(i,j)}}{\partial z_{(i,k+1)}} = \frac{-d^u_{(i,k+1)} u_{(j,k+1)}}{d^u_{(i,k)} (u_{(i,k+1)} - u_{(j,k+1)})} = \frac{Q^{11}_{(i,j)}}{z_{(i,k-1)}}. \tag{2.34}$$

Since $z_{(i,k-1)} > 0$, when $Q^{11}_{(i,j)} \geqslant 0$, we have

$$\partial \left|Q^{11}_{(i,j)}\right| / \partial z_{(i,k+1)} = Q^{11}_{(i,j)} / z_{(i,k-1)} \geqslant 0. \tag{2.35}$$

When $Q^{11}_{(i,j)} < 0$ we have

$$\partial \left|Q^{11}_{(i,j)}\right| / \partial z_{(i,k+1)} = -Q^{11}_{(i,j)} / z_{(i,k-1)} > 0. \tag{2.36}$$

Thus,

$$\partial \left|Q^{11}_{(i,j)}\right| / \partial z_{(i,k+1)} \geqslant 0. \tag{2.37}$$

■

This theorem indicates that the ego-motion estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$ grows as the depth of the vertical line $z_{(i,k+1)}$ increases. Also the depth error, which is the d^z-direction of $\mathbf{d}_{k+1}$, decreases as $|\delta^u_{k+1}|$ increases. From Theorem 1, we know that the depth error dominates the lateral error. Therefore, choosing the vertical line pair with short depth and a large distance between the two lines can improve the accuracy of the ego-motion estimation.

E. Error Aware Ego-motion Estimation Using Multiple Vertical Line Pairs

There are often multiple vertical lines in the scene. For n vertical lines, there are $n(n-1)/2$ pairs. Each pair is capable of providing a minimum solution. The intuition is that we should be able to combine those solutions to yield a motion estimation with minimal error variance. To achieve this, we first define the final motion estimation result as the weighted sum of the minimum solutions from all possible pairs. Plugging (2.2) in, the new recursive ego-motion estimation function is

$$\mathbf{d}_{k+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} \mathbf{F}_{\mathbf{s}}(\mathbf{d}_k, \mathbf{u}_i, \mathbf{u}_j), \tag{2.38}$$

where $w_{(i,j)}$ is the weight of vertical line pair (i, j). $w_{(i,j)}$'s are standardized,

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} = 1, \tag{2.39}$$

$$w_{(i,j)} = w_{(j,i)} \geqslant 0,\ i \in I,\ j \in I, \text{and } i \neq j. \tag{2.40}$$

We want to compute a set of $w_{(i,j)}$ to minimize σ^2_{k+1},

$$\{w_{(i,j)}, \forall i, j \in I, i \neq j\} = \arg \min_{\{w_{(i,j)}\}} \sigma^2_{k+1}. \tag{2.41}$$

To solve this optimization problem, we need to derive the closed form of σ^2_{k+1}. Let us begin with deriving the expression of the estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$ and its covariance matrix $\Sigma^{\mathbf{d}}_{k+1}$. We know that $\mathbf{e}^{\mathbf{d}}_{k+1}$ has two parts,

$$\mathbf{e}^{\mathbf{d}}_{k+1} = \mathbf{e}^{p}_{k+1} + \mathbf{e}^{m}_{k+1}, \tag{2.42}$$

where $\mathbf{e}^{p}_{k+1}$ is the estimation error propagated from the previous step $\mathbf{e}^{\mathbf{d}}_{k}$ and $\mathbf{e}^{m}_{k+1}$ is introduced from the measurement errors of the current step $\mathbf{e}^{\mathbf{u}}_{i}$, $i \in I$. From (2.12) and (2.38), we have the expressions of $\mathbf{e}^{p}_{k+1}$ and $\mathbf{e}^{m}_{k+1}$ as,

$$\mathbf{e}^{p}_{k+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} \boldsymbol{P}_{(i,j)} \mathbf{e}^{\mathbf{d}}_{k} = \boldsymbol{T}\mathbf{e}^{\mathbf{d}}_{k}, \tag{2.43}$$

where $\boldsymbol{T} = \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} w_{(i,j)} \boldsymbol{P}_{(i,j)}$, and

$$\mathbf{e}^{m}_{k+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} (\boldsymbol{Q}_{(i,j)} \mathbf{e}^{\mathbf{u}}_{i} + \boldsymbol{Q}_{(j,i)} \mathbf{e}^{\mathbf{u}}_{j}) = \sum_{i=1}^{n} (\sum_{j=1, j\neq i}^{n} w_{(i,j)} \boldsymbol{Q}_{(i,j)}) \mathbf{e}^{\mathbf{u}}_{i} = \sum_{i=1}^{n} \boldsymbol{S}_{i} \mathbf{e}^{\mathbf{u}}_{i}, \tag{2.44}$$

where $\boldsymbol{S}_i = \sum\limits_{j=1, j\neq i}^{n} w_{(i,j)} \boldsymbol{Q}_{(i,j)}$. In the above equations, $\boldsymbol{T}$ and $\boldsymbol{S}_i$ are just the Jacobian matrices corresponding to $\mathbf{e}^{\mathbf{d}}_{k}$ and $\mathbf{e}^{\mathbf{u}}_{i}$, respectively. With the error relationship, we can derive the covariance matrices.

Similar to (2.42), the covariance matrix $\Sigma^{\mathbf{d}}_{k+1}$ of the estimation error $\mathbf{e}^{\mathbf{d}}_{k+1}$ also has two parts because errors propagated from the previous step are independent of the measurement errors in the current step. Hence,

$$\Sigma^{\mathbf{d}}_{k+1} = \Sigma^{p}_{k+1} + \Sigma^{m}_{k+1}, \tag{2.45}$$

where Σ_{k+1}^{p} and Σ_{k+1}^{m} are corresponding to $\mathbf{e}_{k+1}^{p}$ and $\mathbf{e}_{k+1}^{m}$, respectively.

Recall that the covariance matrix $\Sigma_k^{\mathbf{d}}$ of $\mathbf{e}_k^{\mathbf{d}}$ in (2.43) is known from the previous step. Recall that the covariance matrix Σ^u of $\mathbf{e}_i^{\mathbf{u}}$ in (2.44) is a diagonal matrix, $\Sigma^u = \mathrm{diag}(\sigma_u^2, \sigma_u^2, \sigma_u^2)$. Using the covariance matrices with (2.43) and (2.44), we have

$$\Sigma_{k+1}^{p} = \boldsymbol{T}\Sigma_k^{\mathbf{d}}\boldsymbol{T}^T, \tag{2.46}$$

$$\Sigma_{k+1}^{m} = \sum_{i=1}^{n} \boldsymbol{S}_i \Sigma^u \boldsymbol{S}_i^T = \sigma_u^2 \sum_{i=1}^{n} \boldsymbol{S}_i \boldsymbol{S}_i^T. \tag{2.47}$$

Therefore, Σ_{k+1}^{d} and its trace can be obtained,

$$\Sigma_{k+1}^{\mathbf{d}} = \Sigma_{k+1}^{p} + \Sigma_{k+1}^{m} = \boldsymbol{T}\Sigma_k^{\mathbf{d}}\boldsymbol{T}^T + \sigma_u^2 \sum_{i=1}^{n} \boldsymbol{S}_i \boldsymbol{S}_i^T, \tag{2.48}$$

$$\sigma_{k+1}^2 = \mathrm{Tr}(\Sigma_{k+1}^{\mathbf{d}}) = \mathrm{Tr}(\boldsymbol{T}\Sigma_k^{\mathbf{d}}\boldsymbol{T}^T) + \sigma_u^2 \sum_{i=1}^{n} \mathrm{Tr}(\boldsymbol{S}_i \mathbf{S}_i^T). \tag{2.49}$$

With the closed form of σ_{k+1}^2 derived, we can solve the problem defined in (2.41). Let us define vector $\boldsymbol{w} = [w_1, ..., w_{n(n-1)/2}]^T$ with its g-th entry obtained as follows,

$$w^g = w_{(i,j)}, \text{ where} \begin{cases} i = 1, ..., n-1, \ j = i+1, ..., n, \\ g = (i-1)(n-i/2) + j - i. \end{cases} \tag{2.50}$$

Vector $\boldsymbol{w}$ is our decision vector for the optimization problem in (2.41), which can be rewritten as,

$$\min_{\boldsymbol{w}} \sigma_{k+1}^2 = \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w}, \text{subject to:} -\boldsymbol{w} \leqslant \mathbf{0}, \text{and } \boldsymbol{c}^T \boldsymbol{w} = 1, \tag{2.51}$$

where $\boldsymbol{c} = \mathbf{1}_{n(n-1)/2 \times 1}$ is a vector with all elements being 1 and $\boldsymbol{A}$ is an $n(n-1)/2 \times n(n-1)/2$ matrix obtained from (2.49).

Let us detail how to obtain each entry for $\boldsymbol{A}$, which actually represents the correlations between the vertical line pairs. $\boldsymbol{A}$ also consists of two parts $\boldsymbol{A} = \boldsymbol{A}_p + \boldsymbol{A}_m$, where $\boldsymbol{A}_p$ is the error propagation from the previous step and $\boldsymbol{A}_m$ is newly introduced in the current step. Define A_p^{gh} as the (g,h)-th entry of $\boldsymbol{A}_p$. Similarly, A_m^{gh} is the (g,h)-th entry of $\boldsymbol{A}_m$. Then $\boldsymbol{A}_p$ and $\boldsymbol{A}_m$ are obtained from (2.49) as follows,

$$\begin{cases} A_p^{gg} = \mathrm{Tr}(\boldsymbol{P}_{(i,j)}\Sigma_k^{\mathbf{d}}\boldsymbol{P}_{(i,j)}^T), & i=r,\ j=m, \\ A_p^{gh} = A_p^{hg} = \mathrm{Tr}(\boldsymbol{P}_{(i,j)}\Sigma_k^{\mathbf{d}}\boldsymbol{P}_{(r,m)}^T), & \text{otherwise}, \end{cases} \tag{2.52}$$

$$\begin{cases} A_m^{gg} = \sigma_u^2(\mathrm{Tr}(\boldsymbol{Q}_{(i,j)}\boldsymbol{Q}_{(i,j)}^T) + \mathrm{Tr}(\boldsymbol{Q}_{(j,i)}\boldsymbol{Q}_{(j,i)}^T), & \\ & i=r,\ j=m, \\ A_m^{gh} = A_m^{hg} = \sigma_u^2\mathrm{Tr}(\boldsymbol{Q}_{(i,j)}\boldsymbol{Q}_{(r,m)}^T), & i=r,\ j\neq m, \\ A_m^{gh} = A_m^{hg} = \sigma_u^2\mathrm{Tr}(\boldsymbol{Q}_{(j,i)}\boldsymbol{Q}_{(m,r)}^T), & i\neq r,\ j=m, \\ A_m^{gh} = A_m^{hg} = \sigma_u^2\mathrm{Tr}(\boldsymbol{Q}_{(j,i)}\boldsymbol{Q}_{(r,m)}^T), & j=r,\ j\neq m, \\ A_m^{gh} = A_m^{hg} = 0, & \text{otherwise}, \end{cases} \tag{2.53}$$

where

$$\begin{cases} i=1,...,n-1,\ j=i+1,...,n, \\ r=i,...,n-1,\ m=j,...,n, \\ g=(i-1)(n-i/2)+j-i, \\ h=(r-1)(n-r/2)+m-r. \end{cases}$$

Each diagonal entry of $\boldsymbol{A}$ is exactly the estimation error variance of each single vertical line pair. Defining it as $\sigma_{(i,j)}^2$, we have

$$\sigma_{(i,j)}^2 = A^{gg} = \mathrm{Tr}(\boldsymbol{P}_{(i,j)}\Sigma_k^{\mathbf{d}}\boldsymbol{P}_{(i,j)}^T) + \sigma_u^2(\mathrm{Tr}(\boldsymbol{Q}_{(i,j)}\boldsymbol{Q}_{(i,j)}^T) + \mathrm{Tr}(\boldsymbol{Q}_{(j,i)}\boldsymbol{Q}_{(j,i)}^T), \tag{2.54}$$

for vertical line pair (i,j). This can be simply proved by degenerating (2.49) into the case of containing only two vertical lines. Since $\boldsymbol{A}$ is positive definite, the feasible set in the optimization problem in (2.51) is convex. Hence, this problem is a quadratic convex optimization problem. For such a problem, it is well studied and has various solving methods [52]. In this chapter, we use the well-known interior-point method [53] to solve it.

F. Algorithms

The above analysis implies two different algorithms: best single pair (BSP) and minimum variance ego-motion estimation (MVEE) with multiple pairs. The BSP selects the best pair from multiple vertical line pairs using the results in sensitivity analysis in Section D. The MVEE uses the weights computed from solving the optimization problem (2.41).

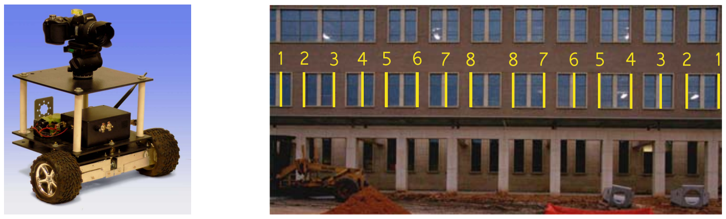
The two algorithms share a common structure as indicated in Algorithm 1. Note that the function $T(n)$ in Algorithm 1 is the complexity of either BSP subroutine in Algorithm 3 or MVEE subroutine in Algorithm 4. For simplicity, we adopt the approximation that $z_{(i,k-1)} \approx z_{(i,k)}$ and $z_{(i,k+1)} \approx z_{(i,k)}$ in the algorithm (see line 4-9). Since $d^z_{(i,k)} \ll z_{(i,k)}$ and $d^z_{(i,k+1)} \ll z_{(i,k)}$ for consecutive image frames, this approximation is reasonable. At line 29, we call either BSP subroutine in Algorithm 3 or MVEE subroutine in Algorithm 4 to obtain ego-motion estimation results. It is not difficult to find that,

Theorem 3 *The computation times for the BSP algorithm and the MVEE algorithm are $O(n^4)$ and $O(n^6)$, respectively.*

At first glance, the computation complexity seems to be high. However, there are $O(n^2)$ pairs to start with. The dominating computation in MVEE is from the use of the interior-point method (IPM) [53] to get $\boldsymbol{w}$, which takes $O(n^6)$ time for the worst case in our IPM implementation, which apparently can be improved. However, the speed is not a concern since n is the number of vertical lines and usually no more than 20. The problem size is still small and our testing results have also confirmed that.

G. Experiments

1. Experiment Setup



(a) (b)

Fig. 4. (a) The camera and the robot used in the experiment. (b) Experiment site from the robot view for the minimum solution. We use the vertical lines on the frontal plane of a building as highlighted in yellow color. The vertical lines are numbered in pairs.

The algorithms are implemented on a Compaq V3000 laptop PC with an Intel 1.6GHz dual core CPU and 1.0G RAM and programmed in MatLab environment. We use a Sony DSC-F828 Camera mounted on a robot in the experiment as shown in Fig. 4(a). The camera HFOV is set to $50°$ with a resolution of 640×480 pixels. The robot is custom made in our lab. The robot measures $50 \times 47 \times 50$ cm^3 in size. The robot has two front drive wheels and one rear cast wheel and uses a typical differential driving structure. The robot can travel at a maximum speed of 50 cm/second.

We define a relative error metric ε for the comparison purpose. Let d_k^{x*} and d_k^{z*} be the true displacements (i.e. *ground truth*) of the robot in x- and z-directions at step k, respectively, which are obtained using a tape measurer in our experiments. Recall that the corresponding outputs of visual odometry are d_k^x and d_k^z. ε is defined as

$$\varepsilon = \sqrt{\varepsilon_x^2 + \varepsilon_z^2}. \tag{2.55}$$

where ε_x and ε_x are relative errors in x- and z- directions, respectively,

$$\varepsilon_x = \frac{|\sum_k d_k^x - \sum_k d_k^{x*}|}{\sum_k \sqrt{(d_k^{x*})^2 + (d_k^{z*})^2}}, \quad \varepsilon_z = \frac{|\sum_k d_k^z - \sum_k d_k^{z*}|}{\sum_k \sqrt{(d_k^{x*})^2 + (d_k^{z*})^2}}.$$

This metric describes the ratio of the ego-motion estimation error in comparison to the overall distance traveled.

During the experiments, we employ Gioi et al.'s method to extract the line segments from the images [54]. The vertical lines are found using an inclination angle threshold [51] and vanishing points. Then, we employ the vanishing point method [49] for vertical and horizontal lines to construct homographies that project images into the iso-oriented ICSs with their $u-v$ planes parallel to the vertical lines, which allows us to align the ICSs at frames $k-1$, k, and $k+1$ for step $k+1$. The correspondence between lines in adjacent frames is found by directly matching pixels of vertical stripes at the neighboring region of the vertical lines.

2. Validating the Minimum Solution and the BSP Algorithm

We first validate the minimum solution and its sensitivity analysis results in the physical experiment. The experiment site is in front of a building on Texas A&M University campus (see Fig. 4(b)). We use eight pairs of vertical lines on the frontal plane of the building. In Fig. 4(b), two lines with the same number belong to the same pair. The relative distance between the two lines in each pair is defined as δ. During the experiment, the camera has to face the building frontal plane to obtain edge position readings. Hence, the z-direction is the direction perpendicular to the frontal plane and x-direction is the direction parallel to the frontal plane.

During each trial, the robot moves along a straight line with 11 incremental steps and the step length of 0.5m. The robot takes images at each of the 12 positions introduced by the 11-step movement. Recall that the robot displacement of the first step is given as a reference. For the subsequent 10 steps, we compute the robot ego-motion using (2.9) and compare it with the ground truth. Each trial is the average of the outcome of the 10 steps.

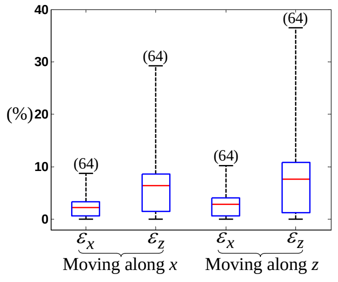
The combination of four different experimental conditions are tested in different trials:

- C1: Two different robot headings including x- and z-directions,
- C2: Eight different relative distance settings δ between the vertical line pair,
- C3: Eight different depth of vertical line pair z. The initial positions of the robot with respect to the building frontal plane are from eight different depth settings ranging from 35m to 70m with 5m intervals, and
- C4: Camera rotation vs. no camera rotation. For cases without camera rotation, we adjust camera pan and tilt in the experiment to force CCSs to be iso-oriented. For cases with camera rotation, we introduce CCSs with $\pm 10°$ orientational difference.

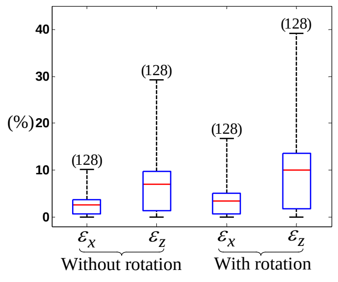
Therefore, we conduct a total of 256 trials in the experiments.

Fig. 5 illustrates experiment results. As shown in Fig. 5(a), regardless of the robot moving directions, the depth direction estimation error ε_z is always over two times as large as the lateral direction error ε_x, which confirms Theorem 1.

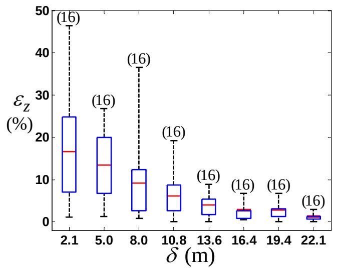
Since the depth error ε_z is the dominating error, we only compare ε_z. Fig. 5(c) illustrates how ε_z changes with respect to different δ settings. It is clear that as δ increases, ε_z decreases. Fig. 5(d) illustrates how ε_z changes as z changes. There is a trend that ε_z decreases as z decreases although the trend is not clear when z is relatively small since factors other than z dominate the error. These results confirm Theorem 2.



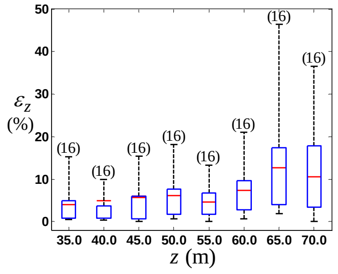
(a)



(b)



(c)



(d)

Fig. 5. Statistical experiment results for the minimum solution. Note that the red line is the mean value, the blue box represents the population ranging from 25 percentile to 75 percentile, and the black dashed intervals indicate the data range. Numbers inside the parentheses are the numbers of trials. (a) ε_z vs. ε_x. (b) Camera rotation vs. no camera rotation. (c) ε_z vs. δ. (d) ε_z vs. z. The number in the parenthesis is the number of trials used to compute the statistics.

Additionally, Fig. 5(b) illustrates how camera rotation impacts ε_x and ε_z. It is clear that there is no significant difference between the two cases for either ε_x or ε_z. The result shows that assuming CCSs are iso-oriented in the analysis is reasonable.

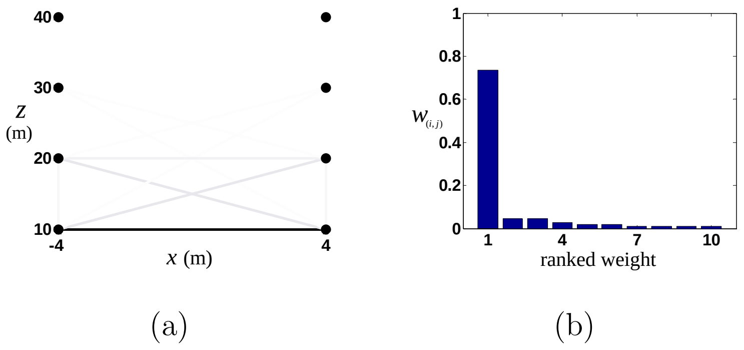


Fig. 6. (a) A top view of vertical lines and corresponding weights for pairs for the sample case. The black dots are vertical lines. The resulting weights for each vertical line pair are presented as edges in grayscale. Darker edge means heavier weight. (b) Weight distribution in decreasing order corresponds to the edges in (a).

3. Validating the MVEE Algorithm

a. A Sample Case in Simulation

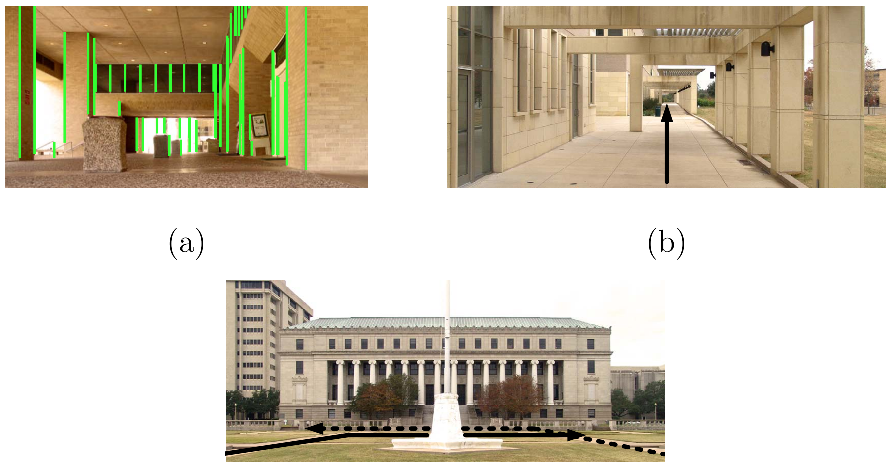
For MVEE validation, we first present a sample case to see how the weights are assigned by the MVEE algorithm for a typical vertical line distribution. We want to know which pairs are more important than others. We setup a scenario that eight vertical lines are symmetrically located along each side of a road as illustrated in Fig. 6(a). This is to simulate the urban case where buildings are evenly distributed on both sides of a road.

Originally, the robot is located at $(x = 0, z = 0)$. Then the robot moves two steps by traveling 1m at a time in z-direction. The first step is given as an initial reference and the MVEE algorithm is executed at the end of the second step.

Fig. 6(a) highlights more heavily weighted pairs by darker edge. The pairs with weights less than 1% of the maximum weight have little contribution to the final ego-motion estimation and are not drawn. It is clear that the vertical line pair that is closest to the camera has the heaviest weight, which is expected according to our results in the minimum solution sensitivity analysis.

Fig. 6(b) illustrates ordered weights by their values. It is clear that only a small part of the vertical line pairs (20%) have their contributions more than 1% of the maximum weighted pair. This result suggests that it is not necessary to track all edges if computation power is limited. In fact, the best pair actually contributes over 70% to the final result, which indicates that BSP is an available choice for cases when computation power is extremely limited.

b. A Comparison of Different Pair Aggregation Methods in Physical Experiments



(a) (b) (c)

Fig. 7. (a) Experiment site 1 from the robot view with vertical edges highlighted in green. (b) and (c) Experiment sites 2 and 3 with robot trajectories highlighted in black.

The MVEE aggregates motion estimation results from multiple vertical line pairs using the variance minimization method. Here we compare MVEE with the results from BSP and a simple equal weighting for all (EWA) pairs. The experiment site is shown in Fig. 7(a). In each trial, the robot moves 31 steps along a zigzagging poly line with a step length of 1 m for odd steps and a step length of 0.5 m for even steps (Fig. 8(a)). We also repeat the experiment with three different camera resolutions: 640×480, 1280×960 and 2560×1920 pixels. With 10 trials for each resolution, we have a total of 30 trials.

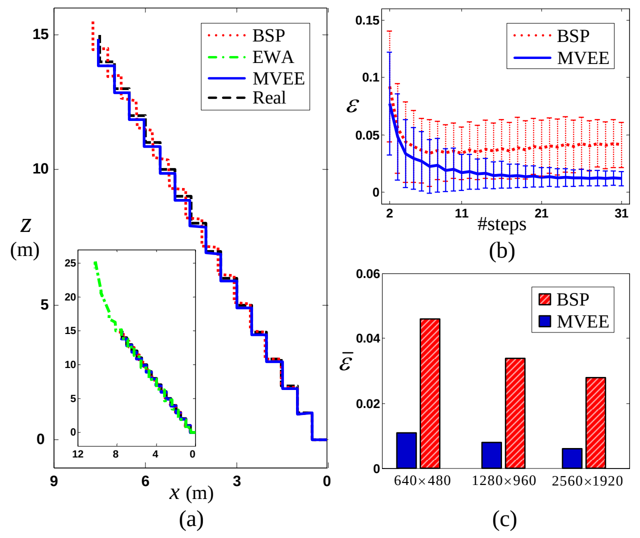


Fig. 8. (a) A comparison of robot trajectories from BSP, MVEE and EWA with ground truth (dashed black poly line). (b) Mean relative error $\bar{\varepsilon}$ and its standard deviation over #steps for both BSP and MVEE. (c) $\bar{\varepsilon}$ vs. camera resolutions.

The experiment results of the three different pair aggregation methods are shown in Fig. 8. With a camera resolution of 640×480 pixels, Fig. 8(a) presents the sample ego-motion estimation results in the form of robot trajectories for one trial. Since EWA is much worse than either BPS or MVEE, it has to be shown in the bigger scale in the small thumbnail at the lower left corner of the figure. The comparison between BPS and MVEE is shown both in the format of the robot trajectories in Fig. 8(a) and in $\bar{\varepsilon}$ over steps in Fig. 8(b). Each $\bar{\varepsilon}$ in Fig. 8(b) is an average of ε over the trials with all camera resolutions at the same step number. Without a surprise, MVEE consistently outperforms BPS at all resolution settings (Fig. 8(c)).

4. Comparison of MVEE with Existing Point and Line-based Odometry Methods

We compare MVEE with two popular ego-motion estimation methods in physical experiments:

- Nister [26]: This method is selected because it is a representative point feature-based method. The method employs Harris corner points as landmarks. This method supports both monocular and stereo configurations. We use its monocular configuration in the experiments.

- L&L [11]: This method is selected because it is a representative line feature-based method. The method is a monocular vision based SLAM method using general line segments as landmarks. We turn off the loop closing for visual odometry comparison purpose.

Both methods estimate 3D robot movements. Since our method is 2D, we only compare the odometry results on the $x - z$ ground plane.

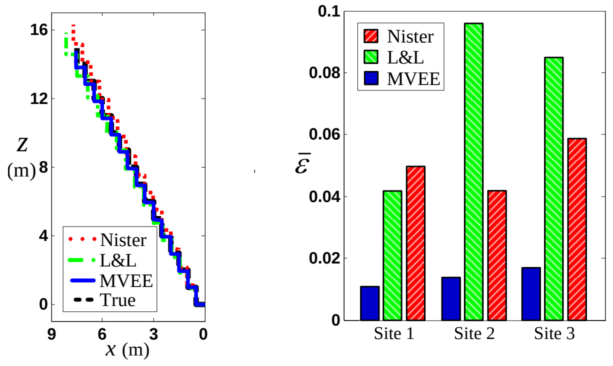
We run tests at three experiment sites (Fig. 7) for all three methods. At each site, the robot moves along a planned trajectory for a certain number of steps. The details about each site are described below:

- Site 1: The same 31-step performed in Fig. 8(a) for the site in Fig. 7(a).

- Site 2: The robot moves toward the depth direction for 51 steps with a step length of 1 m (Fig. 7(b)).

- Site 3: The robot has two trajectories as indicated by the black solid and dashed lines, respectively. Each trajectory has 31 steps along the depth direction followed by 20 steps along the lateral direction with a step length of 1 m (Fig. 7(c)).

We run the robot for 10 trials at each site (for site 3, each trajectory takes 5 trials) which leads to a total of 30 trails.



(a) (b)

Fig. 9. Physical experiment results. (a) A comparison of robot trajectories from the three methods with the ground truth (dashed black poly line). (b) A comparison of $\bar{\varepsilon}$ values for the three methods at each experiment site.

The experiment results of the three methods are shown in Fig. 9. Fig. 9(a) presents a representative sample trial of estimated trajectory comparison at site 1. Fig. 9(b) compares the mean values of ε for the three methods at each site. It is clear that MVEE outperforms its two counterparts in estimation accuracy.

Table 1 compares feature quality and computation speed for the three methods. Each row in Table 1 is the average of the 30 trials. It is obvious that the two line feature based methods, MVEE and L&L, outperform the point feature based Nister method, which conforms to our expectation. MVEE is slightly faster than L&L due to its smaller input sets since vertical lines are a subset of general lines. Note that all implementations are in MatLab and the speed should be much faster if converted to C++ but the factors should remain the same.

For feature quality, it is clear that Nister method employs much more features than MVEE and L&L, while its inliers/total-features ratio is the lowest. On the contrary, MVEE has the least number of features with the highest inlier ratio. This indicates that MVEE is more robust than the other two methods. Overall, MVEE outperforms the other two methods in robustness, accuracy, and speed.

H. Conclusion and Future Work

We reported our development of an incremental error-aware monocular visual odometry method that utilizes vertical edges of buildings in urban area. We derived how to estimate the robot ego-motion using vertical line pairs. To improve the accuracy, we analyzed how errors are introduced and propagated in the continuous odometry process by deriving the recursive and closed form representation of error covariance matrix. We formulated the minimum variance ego-motion estimation problem and presented two algorithms. The resulting visual odometry methods were extensively tested in physical experiments. The proposed odometry method was compared with two popular existing methods and consistently outperforms the two counterparts in speed, robustness, and accuracy.

In the future, we will extend the approach by exploring different combinations of geometric features such as vertical planes, horizontal lines, and points with geometric meanings (e.g. intersections between lines and planes) in visual odometry. We will also look into methods using texture features in combination with geometric features.

Algorithm 1: BSP and MVEE Algorithms

```
input : $\mathbf{d}_k$, $\mathbf{u}_i$, $i \in I$, $f$, $\sigma_u^2$, $\Sigma_k^{\mathbf{d}}$
output : $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$
begin
    for $i = 1$ to $n$ do                                                  $O(n)$
        $d^u_{(i,k+1)} = u_{(i,k+1)} - u_{(i,k)}$;                          $O(1)$
        $d^u_{(i,k)} = u_{(i,k)} - u_{(i,k-1)}$;                            $O(1)$
        $z_{(i,k)} = \frac{(d_k^x - u_{(i,k-1)} d_k^z)}{(u_{(i,k)} - u_{(i,k-1)})}$;   $O(1)$
        $z_{(i,k-1)}, z_{(i,k+1)} = z_{(i,k)}$;                             $O(1)$
    end
    for $i = 1$ to $n - 1$ do                                              $O(n)$
        for $j = i + 1$ to $n$ do                                          $O(n)$
            Calculate $\boldsymbol{P}_{(i,j)}$, $\boldsymbol{Q}_{(i,j)}$, $\boldsymbol{Q}_{(j,i)}$ based on (2.22) and (2.23);   $O(1)$
        end
    end
    Continue on Algorithm 2.
end
```

Algorithm 2: BSP and MVEE Algorithms (Continue)

```
begin
    for $i = 1$ to $n - 1$ do                                              $O(n)$
        for $j = i + 1$ to $n$ do                                          $O(n)$
            $g = (i-1)(n-i/2)+j-i$; $w^g = w_{(i,j)}$, (2.50);             $O(1)$
            for $r = i$ to $n - 1$ do                                      $O(n)$
                for $m = j$ to $n$ do                                      $O(n)$
                    $h = (r-1)(n-r/2)+m-r$;                                $O(1)$
                    Calculate $A_p^{gh}$ using $\boldsymbol{P}_{(i,j)}$, $\boldsymbol{P}_{(r,m)}$ based on (2.52); Calculate
                    $A_m^{gh}$ using $\boldsymbol{Q}_{(i,j)}$, $\boldsymbol{Q}_{(j,i)}$, $\boldsymbol{Q}_{(r,m)}$, $\boldsymbol{Q}_{(m,r)}$ based on (2.53);  $O(1)$
                    $A^{gh} = A_p^{gh} + A_m^{gh}$;                        $O(1)$
                end
            end
        end
    end
    Call BSP subroutine or MVEE subroutine;                                $T(n)$
    Return $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$;
end
```

Algorithm 3: BSP Subroutine

```
1  input : $\mathbf{d}_k$, $\mathbf{u}_i$, $i \in I$, $f$, $\sigma_u^2$, $\Sigma_k^{\mathbf{d}}$, $\boldsymbol{A}$
2  output : $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$
3  begin
4      for $i = 1$ to $n - 1$ do                                              $O(n)$
5          for $j = i + 1$ to $n$ do                                          $O(n)$
6              $g = (i - 1)(n - i/2) + j - i$;                                $O(1)$
7              Record the maximum $A^{gg}$ and the corresponding $i$, $j$ as $i^*$, $j^*$;   $O(1)$
8          end
9      end
10     Calculate $\boldsymbol{T}$ based on (2.43) for $(i^*, j^*)$;          $O(1)$
11     Calculate $\Sigma_{k+1}^{p}$ based on (2.46);                          $O(1)$
12     Calculate $\boldsymbol{S}_{i^*}$, $\boldsymbol{S}_{j^*}$ based on (2.44) for $(i^*, j^*)$;   $O(1)$
13     Calculate $\Sigma_{k+1}^{m}$ based on (2.47) for $(i^*, j^*)$;         $O(1)$
14     Calculate $\Sigma_{k+1}^{\mathbf{d}}$ based on (2.45);                 $O(1)$
15     Compute $\mathbf{d}_{k+1}$ based on (2.9) for $(i^*, j^*)$;            $O(1)$
16     Return $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$;
17 end
```

Algorithm 4: MVEE Subroutine

```
input : $\mathbf{d}_k$, $\mathbf{u}_i$, $i \in I$, $f$, $\sigma_u^2$, $\Sigma_k^{\mathbf{d}}$, $\boldsymbol{w}$, $\boldsymbol{A}$
output : $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$
begin
    Calculate $\boldsymbol{w}$ in (2.51) using IPM;                              $O(n^6)$
    for $i = 1$ to $n-1$ do                                                      $O(n)$
        for $j = i+1$ to $n$ do                                                  $O(n)$
            $g = (i-1)(n-i/2)+j-i$; $w_{(i,j)} = w^g$, (2.50);                   $O(1)$
        end
    end
    Calculate $\boldsymbol{T}$ based on (2.43);                                  $O(n^2)$
    Calculate $\Sigma_{k+1}^p$ based on (2.46);                                  $O(1)$
    for $i = 1$ to $n$ do                                                        $O(n)$
        Calculate $\boldsymbol{S}_i$ based on (2.44);                            $O(n)$
    end
    Calculate $\Sigma_{k+1}^m$ based on (2.47);                                  $O(n)$
    Calculate $\Sigma_{k+1}^{\mathbf{d}}$ based on (2.45);                       $O(1)$
    for $i = 1$ to $n-1$ do                                                      $O(n)$
        for $j = i+1$ to $n$ do                                                  $O(n)$
            Calculate $\boldsymbol{F}(\mathbf{d}_k, \mathbf{u}_i, \mathbf{u}_j)$ based on (2.9);   $O(1)$
        end
    end
    Calculate $\mathbf{d}_{k+1}$ based on (2.38);                                $O(n^2)$
    Return $\mathbf{d}_{k+1}$, $\Sigma_{k+1}^{\mathbf{d}}$;
end
```

Table 1. Feature quality and computation speed comparison.

Methods	Feature			Speed	
	Total	Inliers	Ratio	Time	Factor
Nister	3425	245	7%	15.2s	6.6x
L&L	122	41	34%	3.4s	1.5x
MVEE	59	25	42%	2.3s	1.0x

CHAPTER III

VISION-BASED MEASUREMENT OF FISH SWIMMING MOTION

We present the system and approach to track and reconstruct the movement of a live fish in a tank as shown in Fig. 10. We use a camera-mirror system to capture three orthogonal views of the fish. We build a virtual fish model from measurements of the real fish. The fish model has a four-link spinal cord and meshes attached to the spinal cord. We project the fish model onto three orthogonal views and match the projected views with the real views captured by the camera. We then maximize the overlapping area of the fish in the projected views and the real views, which results in our fish swimming motion reconstruction. Part of this algorithm is still under construction and will be updated in the future.

A. Introduction

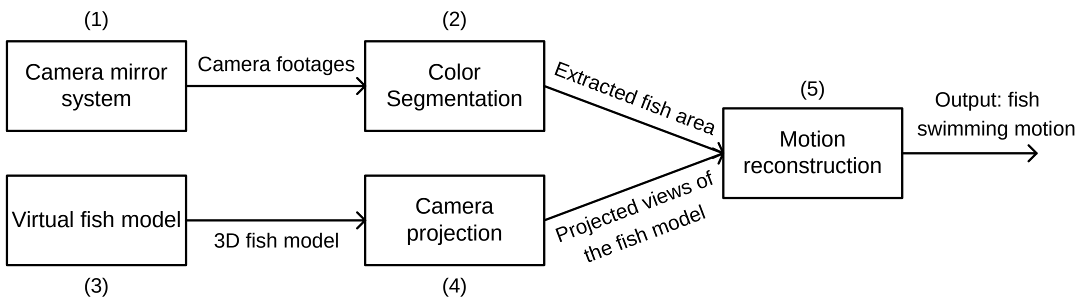


Fig. 10. Illustration of the system. We record camera footages by (1) A camera-mirror system, then we extract the fish area from the video by color segmentation at (2). Meanwhile, a virtual fish model is built at (3) And the fish model is projected to camera views at (4). Finally, we match the projected views and the real views to reconstruct the fish swimming motion at (5).

In biology research, video or animated video playback is a useful tool to study visual communication and its related behaviors of animals. Due to the fast improved technology for generating and manipulating videos, the use of video playback for biological study has significantly increased over the last decade [55, 56]. Currently, video playback is generated in two ways: (1) hand connecting live video sequences of the animal [57] and (2) using 3D animation tools to create artificial movement of a virtual animal [58]. Both of the two ways require close human interaction which often results in confounding artifacts on the animal motion, shape, and texture. Further more, both methods are labor intensive.

In this chapter we present the system setup and approach to automatically track and reconstruct the movement of a live fish in a tank. The work is a combination of the two available video playback generating methods. We use a camera-mirror system to capture three orthogonal views of the fish. We also build a virtual fish model from measurements of the real fish. The fish model has a four-link spinal cord and meshes attached to the spinal cord. We project the fish model onto three orthogonal views and match the projected views with the real views captured by the camera. Then, we maximize the overlapping area of the fish in the projected views and the real views, which results in our fish swimming motion reconstruction. Comparing with the two available methods, our work saves large amount of time on manual video recording, manipulating, and creates video playback based on the motion of a live fish, which can generate more natural and fish-like swimming motion.

Since fish extensively use visual communication and many fish species can be easily housed in a tank for video recording, the technique proposed in this chapter can be greatly helpful for biologists who research on fish visual communication and its related behaviors. Further more, the technique will also favors computer animation specialist working on underwater environment and can be developed into instructional tools for undergraduate and K-12 class students for fish behavior studies.

B. Related Work

This chapter focuses on fish swimming motion tracing and reconstructing, which is related to visual tracking and human/animal motion 3D reconstruction.

Visual tracking is the process of locating a particular object in camera videos, and is widely used in security and surveillance [59,60], traffic control [61,62], medical imaging [63], etc. From the perspective of algorithm output, visual tracking can be classified into blob tracking [64,65], contour tracking [66,67], and feature matching [68, 69]. Blob tracking localizes the interior of an object and computes the center position of the object, contour tracking estimates the boundary of the object, and feature matching identifies particular features (i.e. points, lines) on the object. In our work, we adopt contour tracking because blob tracking which only computes the object center position cannot provide enough information to reconstruct the movement of a live fish, and feature matching is limited to a few fish species because the features on a fish body varies significantly from species to species.

Human 3D reconstruction has been developed for computer animation purpose. The technique usually adopts multiple cameras [70, 71]. Chai, et al, develop a technique of human motion reconstruction with a monocular camera [72]. The technique requires human interference for feature point selection. 3D reconstruction for animals has been implemented on elephant [73], fish [74], extinct animals [75], et al. Our work is an improvement of the pervious works in that we use computer vision technology to reconstruct the fish swimming motion from a live fish.

C. System Configuration

The camera-mirror system is shown in Fig. 11. The system is composed of a rectangular tank, two reflective mirrors, and a video camera. The mirrors and the camera

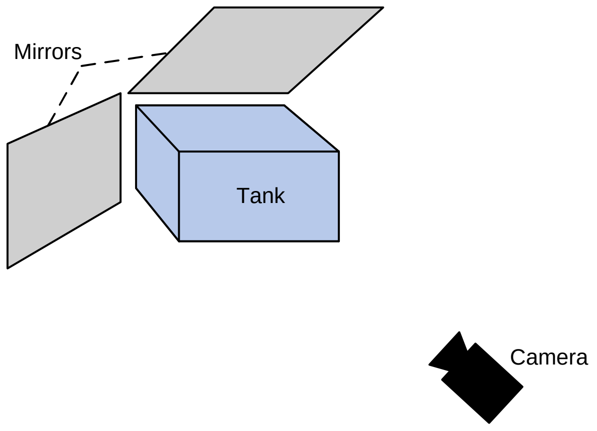


Fig. 11. An illustration of camera-mirror system configuration. The system includes a rectangular tank, two reflective mirrors, and a video camera. The mirrors and the camera can create three orthogonal views of the fish in the tank.

create a live video of the fish in the tank with three orthogonal views. The fish swimming motion is reconstructed from the video.

We use a Panasonic AG-DVX100B video camera for the experiment with the resolution of 720×480 pixels and the frame rate of 30 frames/second. We have two configurations of the tank. The first one is a small sized tank for system testing purpose. The tank is made of plastic board and measures $7 \times 7 \times 5$ cm^3 in size. The second one is a large sized tank for actual experiment purpose. The tank is made of glass board and measures $41 \times 16 \times 16$ cm^3 in size. The two tanks share some common characteristics: (1) Both of them provides a closed space for water only, with no air in the tank. This is because the camera-mirror system takes a top-down view of the tank. Any water surface that is connected with air will create wave on the water surface. The wave will distort the view of the fish in the water. (2) The tanks must have at least one side of the walls openable to allow fish to be put into the tank.

Fig. 12 shows the design of the small tank. The tank has an openable top board and a tube connected with a water reservoir. The top board is not airtight which

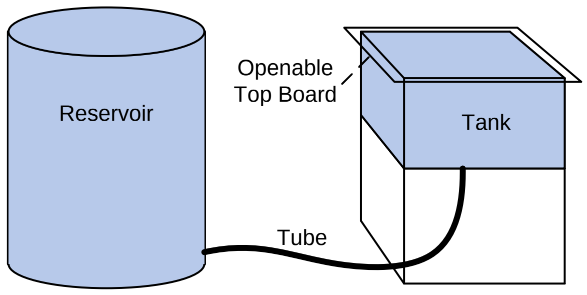


Fig. 12. Design of the small tank. The tank has an openable top board and a tube connected with a water reservoir. The tank, the reservoir, and the tube compose a communicating vessel, which makes the water levels in the tank and the reservoir in the same height.

can allow air goes into/out of the tank. The top of the reservoir is open. The tank, the reservoir, and the tube compose a communicating vessel, which makes the water levels in the tank and the reservoir in the same hight. To start an experiment, we

1. Fill water into the reservoir until the tank is almost full with water;
2. Put a fish into the tank and close the top board;
3. Fill more water into the reservoir to squeeze out any air in the tank.

To finish an experiment, we

1. Slightly lower the position of the reservoir to allow air goes into the tank;
2. Open the top board and take the fish out of the tank;
3. Lower the position of the reservoir more to drill out all the water in the tank.

Fig. 13 shows the design of the large tank. Due to the large size of the tank, it is rather difficult to squeeze out all the air in the tank if we use the same design with the small tank. In our experiment, air bubbles remain in the tank which disturb the

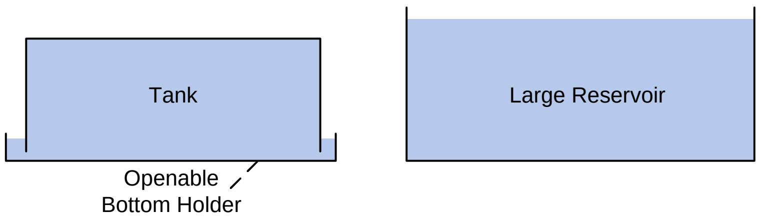


Fig. 13. Design of the large tank. The tank has an openable bottom holder that contains shadow water. The tank is sinked in the shadow water in the bottom holder. Because of atmospheric pressure, the tank fully holds water without any air in the tank. The large reservoir is for the purpose of filling water into the tank.

top-down view. Therefore, we bring in another design for the large tank. As shown in Fig. 13, the tank has an openable bottom holder that contains shadow water. The tank is sinked in the shadow water. Because of atmospheric pressure, the tank can fully hold water without any air in the tank. The design also contains a large reservoir for the purpose of filling water into the tank. To start an experiment, we

1. Put the tank and the bottom holder into the large reservoir and sink them completely into water;

2. Put a fish into the tank and close the bottom holder;

3. Lift the tank with the bottom holder together out of the large reservoir and keep them in level attitude.

To finish an experiment, we

1. keep the tank and the bottom holder in level attitude and put them into the large reservoir;

2. Open the bottom holder and take the fish out of the tank;

3. Lift the tank and the bottom holder out of the large reservoir separately.

For a physical experiment, we first test in the small tank. Then, we use the large tank for final experiment. Both of the tanks are tested in experiment.

D. Camera Calibration

The camera calibration is a pre-processing procedure for each experiment. The procedure corrects the orientation of the image plane and at the same time computes the camera relative position to the tank. We run camera calibration once before each individual experiment. We begin with assumptions.

1. Assumptions

1. We assume the tank is in rectangular shape with the length of its edges known. The interior of the tank is filled with water (refractive index 1.333) and the exterior of the tank is air (refractive index 1.000).

2. We assume that the camera follows the pinhole camera model with square pixels and a zero skew factor. The camera lens distortion is removed. If not, we can use intrinsic parameters from radial distortion calibration to correct the discrepancy.

2. Notations and Coordinate Systems

As shown in Fig. 14, the rectangular illustrates the tank. The camera points at two parallel faces of the tank. We call the face closer to the camera the front face, and the face further away from the camera the back face. Without loss of generality, we define tank coordinate system $x - y - z$ with its origin at the left-bottom vertex of the tank with respect to the camera, on the front face. The x-axis is along with one horizontal edge of the tank pointing rightward, the y-axis is along with one horizontal

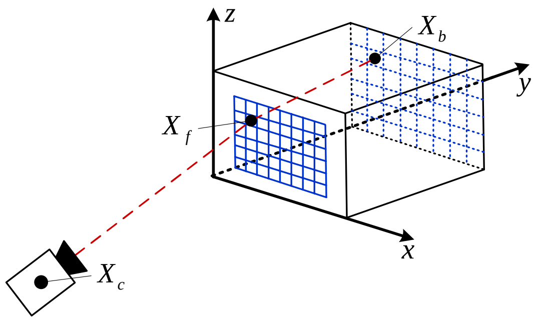


Fig. 14. Definition of tank coordinate system $x-y-z$. The face closer to the camera is the front face, the face further away from the camera is the back face. A pair of points on the front face and the back face that share the same position in the camera image are indicated as X_f and X_b, respectively.

edge of the tank pointing in the inverse direction to the camera position, and the z-axis is along with one vertical edge of the tank pointing upward. With the coordinate system defined, the front face of the tank is on the $x-z$ plane.

Let X_c be the camera optical center, $X_c = (x_c, y_c, z_c)$. Here, since the y-axis is pointing in the inverse direction to the camera, there is always $y_c < 0$. Viewing from the camera optical center, a point on the front face is corresponding to a point on the back face, i.e. the two points overlaps in the camera image. A pair of corresponding points are defined as X_f and X_b, where $X_f = (x_f, 0, z_f)$ is the point on the front face and $X_b = (x_b, y_b, z_b)$ is on the back face. Note that due to the refraction between water, tank wall, and air, X_c, X_f and X_b are not collinear, the three points follow Snell's refraction law [76], which will be discussed later in this chapter.

3. Problem Definition

To calibrate the camera, a grid is attached on the back face of the tank, as illustrated by the blue colored doted grid on the back face (Fig. 14). The grid's measure is known, and we use the horizontal and vertical lines' intersections on the grid for the camera calibration. Let the set of the intersection points be $\mathcal{X}_b$, and let a point in $\mathcal{X}_b$ be X'_b, $X'_b = (x'_b, y'_b, z'_b) \in \mathcal{X}_b$. X'_b in the tank coordinate can be obtained from its relative position on the grid.

Each intersection point on the back face is corresponding to a point on the front face, as illustrated by the blue colored solid grid on the front face (Fig. 14). Let the set of the corresponding points be $\mathcal{X}_f$, and let a point in $\mathcal{X}_f$ be X'_f, $X'_f = (x'_f, 0, z'_f) \in \mathcal{X}_f$. X'_f in the tank coordinate can be obtained from its relative position on the front face in the camera image.

The task of camera calibration is to find the position of the camera optical center X_c using the correspondence between $\mathcal{X}_f$ and $\mathcal{X}_b$, then use X_c to find the point-wise mapping between the front and the back faces. Mathematically, the problem can be defined as

Definition 2 *Given $\mathcal{X}_b$ and a camera image, compute $\mathcal{X}_f$ and estimate X_c, then reversely for any given point X_f on the front face, use X_c to find the corresponding point X_b on the back face.*

4. Locate Intersection Points on Front Face

In this subsection we will compute $\mathcal{X}_f$ in the tank coordinate. For a camera image as Fig. 15(a), we define the image coordinate system $u - v$ with its origin at the left-bottom corner of the image. The u- and v- axis are horizontal and vertical axis pointing rightward and upward, respectively. A point in the $u - v$ coordinate is denoted as $I = (u, v)$.

As shown in Fig. 15(a), since the camera principal axis may not necessarily be perpendicular to the front face, the rectangular shaped front face in the camera image is not necessarily rectangular. We have to correct the image such that the front face

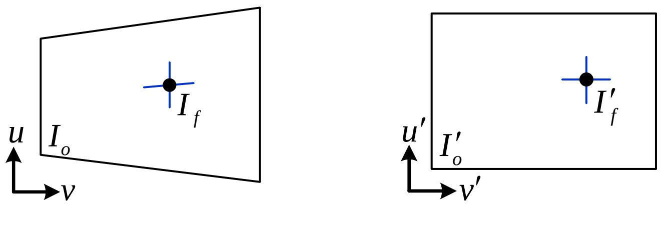


Fig. 15. (a) Before and (b) After image correction. After the correction, the tank front wall appears rectangular in the image.

appears rectangular, as shown in Fig. 15(b). Define the image coordinate system $u' - v'$ for Fig. 15(b) in the same fashion with coordinate $u - v$, a point in the $u' - v'$ coordinate is denoted as $I' = (u', v')$.

The image correction is achieved by a homography translation. Since the measure of the front face is known, we can construct a homography matrix $\mathbf{H}$ using the four corner points of the front face [38]. The mapping is up to similarity, where each point I in the $u - v$ coordinate is mapped to a point I' in the $u' - v'$ coordinate by the following relationship,

$$s \begin{bmatrix} u' & v' & 1 \end{bmatrix}^T = \mathbf{H} \begin{bmatrix} u & v & 1 \end{bmatrix}^T \tag{3.1}$$

where s is a scale factor.

Let I_f be the image point of X'_f in the $u - v$ coordinate, and let I_o be the image point of the tank coordinate origin. Using (3.1), the corresponding image points of I_f and I_o in the $u' - v'$ coordinate is obtained, denoted as I'_f and I'_o, $I'_f = (u'_f, v'_f)$ and $I'_o = (u'_o, v'_o)$. X'_f can be computed by its relative position on the front face,

$$\begin{bmatrix} x'_f \\ z'_f \end{bmatrix} = \begin{bmatrix} l_a(u'_f - u'_o)/l_i \\ h_a(v'_f - v'_o)/h_i \end{bmatrix}, \tag{3.2}$$

where l_a and h_a are the width and height of the front face in the tank coordinate, measured before hand, and l_i and h_i are the width and height of the front face in the $u' - v'$ coordinate, measured from the camera image. Computing each intersection on the front face, we obtain $\mathcal{X}_f$.

5. Estimate Camera Optical Center

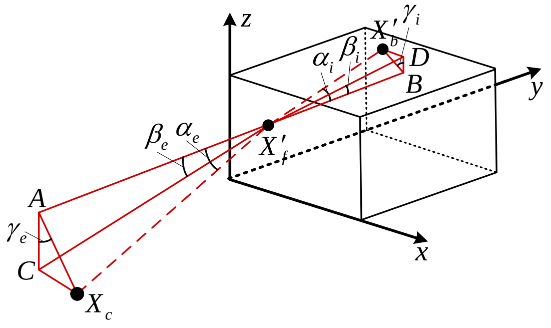


Fig. 16. Camera calibration. A light ray starts from a point on the back face, X_b', through a point on the front face, X_f', into camera optical center, X_c.

In this subsection, we will present a method to recover X_c from $\mathcal{X}_f$ and $\mathcal{X}_b$. As shown in Fig. 16, recall that points X_f' and X_b' are a pair of intersections on the front face and back face, respectively, $X_f' \in \mathcal{X}_f$ and $X_b' \in \mathcal{X}_b$. $\overline{AB}$ is the line segment passing through X_f' and perpendicular to the $x - z$ plane. Points C and D are the projections of X_c and X_b' onto the plane passing through $\overline{AB}$ and parallel to the $y - z$ plane. α_i, β_i, γ_i, α_e, β_e, and γ_e are six angles defined in the figure.

According to Snell's refraction law, α_i and α_e follow the following relationship,

$$n_a \sin \alpha_e = n_w \sin \alpha_i. \tag{3.3}$$

Since $\triangle ACX_c$ and $\triangle BDX_b'$ are a pair of similar triangles, we have

$$\gamma_e = \gamma_i, \tag{3.4}$$

then, from geometry relationship, we have

$$\tan\beta_e = \tan\alpha_e \cos\gamma_e. \tag{3.5}$$

Plugging (3.3) and (3.4) into (3.5), we have

$$\tan\beta_e = \frac{(n_w/n_a)\sin\alpha_i\cos\gamma_i}{\sqrt{1-(n_w/n_a)^2\sin^2\alpha_i}}. \tag{3.6}$$

Replacing $\sin\alpha_i$, $\cos\gamma_i$, and $\tan\beta_e$ in (3.6) by the expressions of X_c, X'_f, and X'_b, where

$$\tan\beta_e = \frac{\overline{AC}}{\overline{AX'_f}} = \frac{z'_f - z_c}{-y_c}, \tag{3.7}$$

$$\sin\alpha_i = \frac{\overline{BX'_b}}{\overline{X_f X'_b}} = \frac{\sqrt{(x'_b - x'_f)^2 + (z'_b - z'_f)^2}}{\sqrt{(x'_b - x'_f)^2 + {y'_b}^2 + (z'_b - z'_f)^2}}, \tag{3.8}$$

$$\cos\gamma_i = \frac{\overline{BD}}{\overline{BX_b}} = \frac{z'_b - z'_f}{\sqrt{(x'_b - x'_f)^2 + (z'_b - z'_f)^2}}, \tag{3.9}$$

we obtain a linear equation of y_c and z_c,

$$a_1 y_c + z_c = b_1, \tag{3.10}$$

where

$$a_1 = -\frac{n_w(z'_b - z'_f)}{\sqrt{(n_a^2 - n_w^2)((x'_b - x'_f)^2 + (z'_b - z'_f)^2) - n_w^2 {y'_b}^2}},$$
$$b_1 = z'_f.$$

Eq. (3.10) is derived by projecting X_c and X'_b to C and D, respectively. Similarly, we project X_c and X'_b onto a plane passing through $\overline{AB}$ and parallel to the $x-y$ plane, and we obtain a linear equation of x_c and y_c as

$$x_c + a_2 y_c = b_2, \tag{3.11}$$

where

$$a_2 = -\frac{n_w(x'_b - x'_f)}{\sqrt{(n_a^2 - n_w^2)((x'_b - x'_f)^2 + (z'_b - z'_f)^2) - n_w^2 {y'_b}^2}},$$

$$b_2 = x'_f.$$

Combine (3.10) and (3.11), we have the following equation,

$$\mathbf{A} \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T = \mathbf{b}, \tag{3.12}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & a_1 & 1 \\ 1 & a_2 & 0 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

This indicates that for a pair of corresponding X'_f and X'_b, we can derive two linear equations as (3.12). Doing this for each pair of points in $\mathcal{X}_f$ and $\mathcal{X}_b$, we have a stack of equations. Recovering X_c from the equations set is a standard procedure using SVD. Therefore, X_c is computed.

E. Extract Fish Area from Video

The fish area in each image frame of the video is extracted using color segmentation. In this task, the image background is a material located on the back of the tank. We choose a unique blue colored material because this color is significantly different from the colors on the fish body. With the material setup, the color of the image background is known. The color segmentation is simply a process of looking for an area in the image frame where the color is the most different from the background.

Before the color segmentation, we first boost the color difference between the fish area and the background using consecutive frame information. As in our observation, it is very rare that the fish body stays exactly statistic for a moment of time. Therefore, the following equation is employed to utilize the fish body movement to boost the color difference,

$$C^b_{(u,v,k)} = 0.5C_{(u,v,k)} + 0.5C_{(u,v,k+1)} + 0.2|C_{(u,v,k)} - C_{(u,v,k+1)}|, \tag{3.13}$$

where $C_{(u,v,k)}$ and $C_{(u,v,k+1)}$ are the pixel color for frame k and $k+1$ respectively, with the image coordinate of (u, v), and $C^b_{u,v,k}$ is the corresponding boosted pixel color.

The color segmentation is then performed using a color threshold, $(C^{blue} - C^{thre}, C^{blue} + C^{thre})$, where C^{blue} is the background color and C^{thre} is the color threshold. Any pixel in the image frame whose color is located in $(C^{blue} - C^{thre}, C^{blue} + C^{thre})$ is regarded as background, and otherwise perspective fish area.

A connectivity check is performed finally. Since we know the size of the fish in the image frame, for any exacted area significantly smaller than the fish size, we treat them as noise. The exacted areas that pass the connectivity check are regarded as fish areas. A sample color segmentation result is shown in Fig. 17.

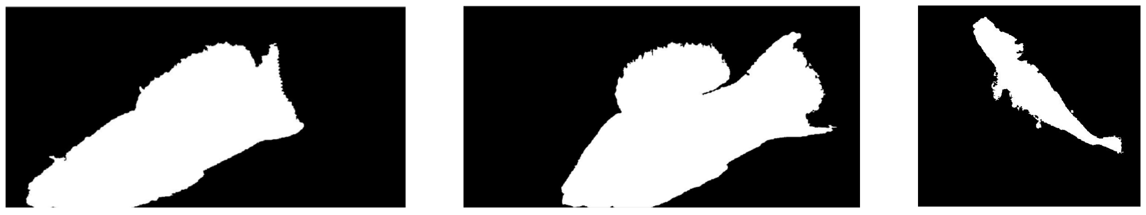


Fig. 17. A sample color segmentation result with the fish area shown in white color and the background in black color. (a) Front view. (b) Left view. (c) Top view.

F. Build Virtual Fish Model

We build a virtual fish model in the same size and shape as the real fish for the swimming motion reconstruction. Fig. 18 shows the virtual fish bone model. The

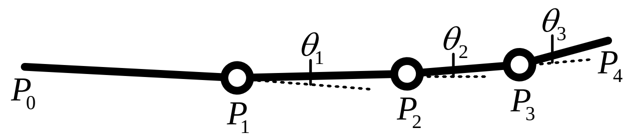


Fig. 18. Top-down view of the virtual fish bone model. The bone model has four links and three pan DOFs for each connection of the links.

bone model has four rigid links denoted as P_0P_1, P_1P_2, P_2P_3, and P_3P_4. The relative length of the four links are determined based on [77]. The bone model has three pan DOFs on each link connection. Define θ_i, $i = 1, 2, 3$ as the intersection angle between link $P_{i-1}P_i$ and P_iP_{i+1} as illustrated in Fig. 18. The counterclockwise direction is the positive direction for θ_i.

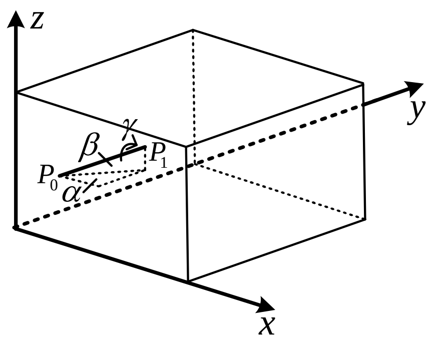


Fig. 19. Position and orientation of the virtual fish model in the tank coordinate. The model has three parameters to determine the position and three parameters to determine the orientation.

To determine the position and orientation of the virtual fish model in the tank coordinate, we need six parameters: three parameters for the position and three parameters for the orientation. Let X_f be the tank coordinate of P_0, $X_f = \{x_f, y_f, z_f\}$. Let α, β, and γ be respectively the pan, tilt, and rotation angles of the first link of the virtual fish bone model, P_0P_1. As illustrated in Fig. 19, α, β, and γ are defined in Euler order. With the position and orientation determined, the virtual fish model has totally nine DOFs indicated as θ_1, θ_2, θ_3, x_f, y_f, z_f, α, β, and γ.

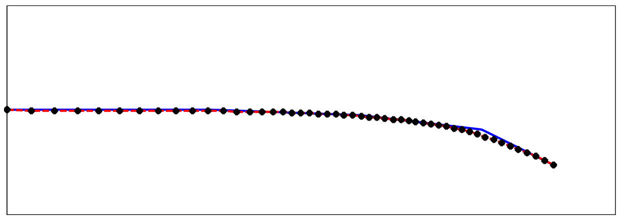


Fig. 20. B-spline generated from the four-link bone model. The blue solid lines represent the bone model, the red dashed curve represents the b-spline curve, and the black crosses are the b-spline curve key points.

With the virtual fish bone model built, we use third-order b-spline to interpolate the five vertices of the four-link bone model, which generates a smooth curve as shown in Fig. 20. The blue solid lines represent the bone model, the red dashed curve represents the b-spline curve, and the black dots are the b-spline curve key points.

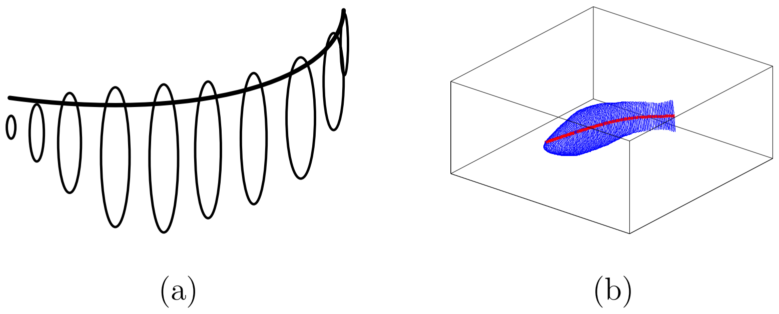


Fig. 21. (a) Ellipses attached to the b-spline curve. The size of the ellipses are measured from the real fish. The ellipses are perpendicular to the b-spline curve and evenly distribute along the curve. (b) Virtual fish model in the tank. The red curve is the b-spline curve and the blue area is the 100 ellipses.

Finally, we attach a number of 100 ellipses on to the b-spline curve. As illustrated in Fig. 21, the size of the ellipses are measured from the real fish. The 100 ellipses are perpendicular to the b-spline curve and evenly distributed along the curve. With the ellipses attached, the virtual fish is model is complete as shown in Fig. 21.

G. Reconstruct Fish Swimming Motion

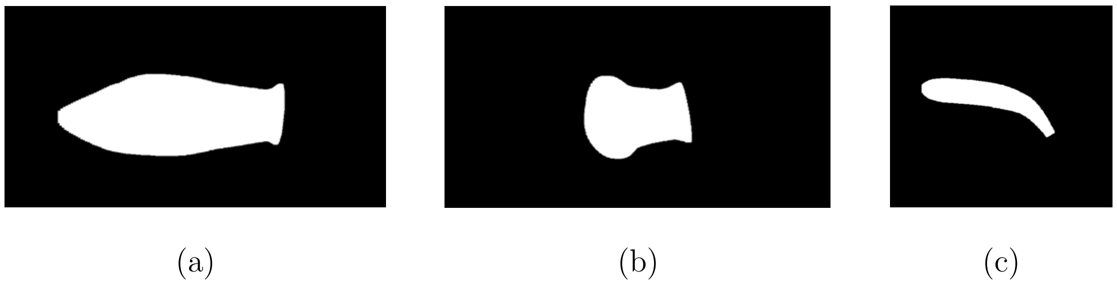


Fig. 22. Projection of the virtual fish model in Fig. 21(b) onto three orthogonal views. The projections are implemented in the same directions as captured by the camera-mirror system. (a) Front view. (b) Left view. (c) Top view.

So far, we obtain the fish area from color segmentation as shown in Fig. 17, and we build a virtual fish model as in Fig. 21(b). Projecting the virtual fish model onto three orthogonal views, we can obtain three views of the fish area as shown in Fig. 22. The projections are implemented in the same directions as captured by the camera-mirror system.

We match the projected views (Fig. 22) with the real camera images (Fig. 17) to reconstruct the fish swimming motion. We compute the overlapping area of the fish in the projected views and the real views, and we maximize the overlapping fish area to estimate the parameters of the virtual fish model. By doing this, we obtain the parameters of the virtual fish model for one camera frame. The reconstruction result is shown in Fig.23. Using multiple frames, we can recover the fish swimming motion.

H. Conclusion and Future Work

We use camera-mirror systems to capture three orthogonal views of the fish. We build two tanks: a small tank for testing purpose and a large tank for accurate experiment. Both of the tanks can completely hold water without air in them. However, the small

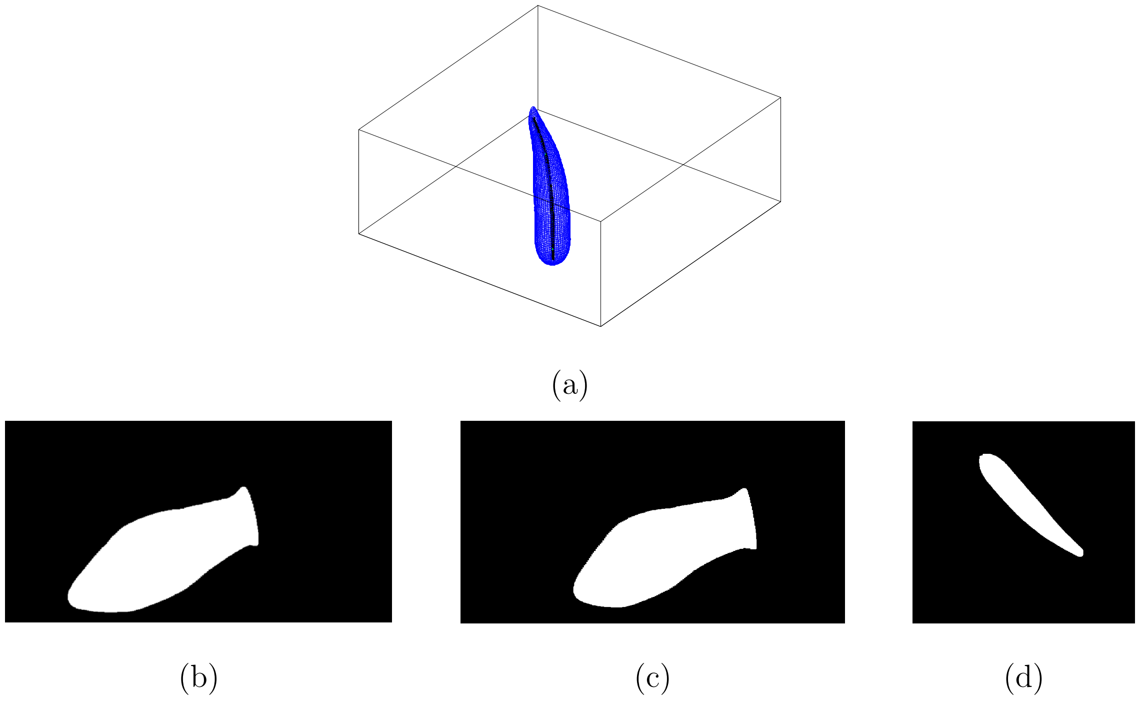


Fig. 23. (a) Recovered virtual fish model from a camera frame, corresponding to Fig. 17. (b)-(c) The three projected views of (a).

tank can only hold fish for a short amount of time (half an hour in our experiment) for the limited amount of oxygen in the tank. We extract the fish area from the camera image using color segmentation. Also, we build a virtual fish model using measurements from the real fish, the fish model is projected onto three orthogonal views. The projected views are matched with the real camera images and the fish swimming motion is then reconstructed. The algorithm works well for tracking a single fish. However, if multiple fish are put in the tank, the reconstruction becomes less actuate when two fish have an overlapping area in the camera view.

In future work, instead of using two mirrors, we will try to use only one mirror and reconstruct the fish swimming motion from two orthogonal views (the top view and the front view). This work will reduce some complexity on the experiment setup. Also, since we only adopt one species of fish in this experiment, in the future we will try to address the scenario of multiple species of fishes in the tank. Classification approaches will be adopted to recognize the fish species. Further more, since the current fish model does not have fins, we will try to add in fin models to make the reconstruction more complete and accurate.

CHAPTER IV

CONCLUSION AND FUTURE WORK

Two case studies on the vision-based measurement of moving objects are presented. In the first case, a monocular camera ego-motion estimation approach is proposed. The algorithm employs vertical line features and works particularly in urban area. Physical experiment is implemented while our algorithm outperforms its two counterparts in both speed and accuracy. In the second case, a 3D reconstruction approach for live fish swimming motion is proposed. Camera-mirror systems are built and the swimming motion is reconstructed from the fish live video. The approach works well for a single fish tracking. But if multiple fish are in the tank, the reconstruction becomes less actuate when two fish have an overlapping area in the camera view.

Both of the two case studies have their own future work. For the ego-motion estimation case, we will try to extend the approach to 3D ego-motion estimation, we will also try to employ multiple types of features (e.g. geometric features and texture features together) instead of only vertical lines. For the fish swimming motion estimation case, we will try to implement the work with two orthogonal views instead of three views. We will try to add fins onto the fish model, and we will also try to address the scenario that multiple species of fishes in the tank.

REFERENCES

[1] J. Zhang and D. Song, "On the error analysis of vertical line pair-based monocular visual odometry in urban area," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009, pp. 187–191.

[2] J. Zhang and D. Song, "Error aware monocular visual odometry using vertical line pairs for small robots in urban areas," in *Proc. of the AAAI Conference on Artifical Intelligence*, Atlanta, GA, July 2010, pp. 2531–2538.

[3] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1015–1026, 2008.

[4] O. Amidi, T. Kanade, and J. Miller, "Vision-based autonomous helicopter research at Carnegie Mellon Robotics Institute 1991-1997," in *Proc. of the American Helicopter Society International Conference*, Heli, Japan, April 1998, pp. 321–331.

[5] R. Marks, H. Wang, M. Lee, and S. Rock, "Automatic visual station keeping of an underwater robot," in *Proc. of the Oceans Engineering for Today's Technology and Tomorrow's Preservation Conference*, Brest, France, Sept. 1994, pp. 2575–2580.

[6] R. Ozawa, Y. Takaoka, and Y. Kida, "Using visual odometry to create 3D maps for online footstep planning," in *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, Big Island, HI, Oct. 2005, pp. 652–659.

[7] P. Corke, D. Strelow, and S. Singh, “Omnidirectional visual odometry for a planetary rover,” in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sept. 2004, pp. 149–171.

[8] M. Agrawal and K. Konolige, “Rough terrain visual odometry,” in *Proc. of the International Conference on Advanced Robotics*, Jeju, South Korea, Aug. 2007, pp. 3721–3726.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, Cambridge, MA, The MIT Press, 2005.

[10] A. Davison, L. Reid, N. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[11] T. Lemaire and S. Lacroix, “Monocular-vision based SLAM using line segments,” in *Proc. of the IEEE International Conference on Robotics and Automation*, Roma, Italy, May 2007, pp. 2791–2796.

[12] B. Steder, G. Grisetti, and C. Stachniss, “Visual SLAM for flying vehicles,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1088–1093, 2008.

[13] T. Marks, “Gamma-SLAM: Visual SLAM in unstructured environments using variane grid maps,” *Journal of Field Robotics*, vol. 26, no. 1, pp. 26–51, 2009.

[14] K. Konolige and M. Agrawal, “FrameSLAM: From bondle adjustment to real-time visual mapping,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[15] J. Civera, A. Davison, and J. Montiel, “Inverse depth parametrization for monocular SLAM,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[16] J. Civera, D. Bueno, A. Davison, and J. Montiel, "Camera self-calibraction for sequential bayesian structure form motion," in *Proc. of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 130–134.

[17] R. Sim, P. Elinas, M. Griffin, and J. Little, "Vision-based SLAM using the rao-blackwellised particle filter," in *Proc. of the International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, July 2005, pp. 303–318.

[18] G. Klein and D. Murray, "Parallel tracking amd mapping for small AR workspaces," in *Proc. of the International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 2007, pp. 1–10.

[19] T. Lemaire and S. Lacroix, "SLAM with panoramic vision," *Journal of Field Robotics*, vol. 24, no. 1/2, pp. 91–111, 2007.

[20] D. Scaramuzza and R. Siegwart, "Monocular omnidirectional visual odometry for outdoor ground vehicles," *Computer Vision Systems*, vol. 5008, pp. 5206–215, 2008.

[21] M. Wongphati, N. Niparnan, and A. Sudsang, "Bearing only fast SLAM using vertical line information from an omnidirectional camera," in *Proc. of the IEEE International Conference on Robotics and Biomimetics*, Bangkok, Thailand, Feb. 2009, pp. 494–501.

[22] D. Scaramuzza and R. Seigwart, "A robust descriptor for tracking vertical lines in omnidirectional images and its use in mobile robotics," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 149–171, 2009.

[23] G. Caron and E. Mouaddib, "Vertical line matching for omnidirectional stereo-vision images," in *Proc. of the IEEE International Conference on Robotics and*

Automation, Kobe, Japan, May 2009, pp. 149–171.

[24] J. Sola, A. Monin, M. Devy, and T. Vidal-Calleja, "Fusing monocular information in multicamera SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 958–968, 2008.

[25] L. Paz, P. Pinies, and J. Tardos, "Large-scale 6-DOF SLAM with stereo-in-hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[26] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vechicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[27] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 2004, pp. 652–659.

[28] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 2, pp. 169–186, 2007.

[29] Y. Cheng, M. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," in *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, Big Island, HI, Oct. 2005, pp. 903–910.

[30] D. Helmick, Y. Cheng, and D. Clouse, "Path following using visual odometry for a mars rover in high-slip environments," in *Proc. of the IEEE Aerospace Conference*, Big Sky, MT, March 2004, pp. 255–264.

[31] D. Cobzas, H. Zhang, and M. Jagersand, "Image-based localization with depth-enhanced image map," in *Proc. of the IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, Sept. 2003, pp. 243–282.

[32] W. Zhou, J. Miro, and G. Dissanayake, "Information-efficient 3-D visual SLAM for unstructured demains," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1078–1087, 2008.

[33] S. Li, T. Kanbara, and A. Hayashi, "Making a local map of indoor environments by swiveling a camera and a sonar," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, South Korea, Oct. 1999, pp. 239–246.

[34] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] H. Bay, A. Ess, T. Tuytelaars, and L. Gool, "SURF: Speeded up robust features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[36] M. Agrawal, K. Konolige, and M. Blas, "SenSurE: Center surround extremas for realtime feature detection and matching," in *Proc. of the European Conference on Computer Vision*, Marseille, France, Oct. 2008, pp. 79–116.

[37] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[38] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, New York, Cambridge University Press, 2004.

[39] D. Ziou and S. Tabbone, "Edge detection techniques-an overview," *International Journal of Pattern Recognition and Image Analysis*, vol. 8, no. 4, pp. 537–559, 1998.

[40] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *Proc. of the British Machine Vision Conference*, Edinburgh, UK, Sept. 2006, pp. 3015–3020.

[41] Y. Choi, T. Lee, and S. Oh, "A line feature based SLAM with low grad range sensors using geometric constrains and active exploration for mobile robot," *Autonomous Robot*, vol. 24, pp. 13–27, 2008.

[42] M. Dailey and M. Parnichkun, "Landmark-based simultaneous localization and mapping with stereo vision," in *Proc. of the Asian Conference on Industrial Automation and Robotics*, Bangkok, Thailand, May 2005, pp. 847–853.

[43] A. Gee and W. Mayol-Cuevas, "Real-time model-based SLAM using line segments," *Advances in Visual Computing*, vol. 4292, pp. 354–363, 2006.

[44] A. Martignoni and W. Smart, "Localizing while mapping: A segment approach," in *Proc. of the AAAI Conference on Artificial Intelligence*, Edmonton, Canada, July 2002, pp. 45–61.

[45] C. Taylor and D. Kriegman, "Structure and motion from line segments in multiple images," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1021–1032, 1995.

[46] J. Montiel, J. TardoH, and L. Montano, "Structure and motion from straight line segments," *Pattern Recognition*, vol. 32, pp. 1295–1307, 2000.

[47] M. Kim, S. Lee, and K. Lee, "Self-localization of mobile robot with single camera in corridor environment," in *Proc. of the IEEE International Symposium on Industrial Electronics*, Pusan, South Korea, June 2001, pp. 4100–4105.

[48] Y. Sakamoto and M. Aoki, "Street model with multiple movable panels for pedestrian environment analysis," in *Proc. of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004, pp. 61–85.

[49] A. Gallagher, "Using vanishing points to correct camera rotation in images," in *Proc. of the Canadian Conference on Computer and Robot Vision*, Victoria, Canada, May 2005, pp. 127–140.

[50] J. Guerrero, R. Martinez-Cantin, and C. Sagues, "Visual map-less navigation based on homographies," *Journal of Field Robotics*, vol. 22, no. 10, pp. 569–581, 2005.

[51] J. Zhou and B. Li, "Exploiting vertical lines in vision-based navigation for mobile robot platforms," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, HI, April 2007, pp. 465–468.

[52] J. Nocedal and S. Wright, *Numerical Optimization*, New York, Springer-Verlag, 2006.

[53] S. Boyd and L. Vandenberghe, *Convex Optimization*, New York, Cambridge University Press, 2006.

[54] R. Gioi, J. Jakubowicz, J. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 772–732, 2010.

[55] R. Oliveira, G. Rosenthal, and I. Schlupp, "Considerations on the use of video playbacks as visual stimuli," *Acta Ethologica*, vol. 3, no. 1, pp. 61–65, 2000.

[56] T. Ord and C. Evans, "Interactive video playback and opponent assessment in Lizards," *Behaviural Processes*, vol. 59, no. 2, pp. 55–65, 2002.

[57] G. Rosenthal, "Design considerations and techniques for constructing video stimuli," *Acta Ethologica*, vol. 3, no. 1, pp. 49–54, 2000.

[58] G. Rosenthal and C. Evans, "Femal preference for swords in Xiphophorus Helleri reflects a bias for large apparent size," *National Academy of Science USA*, vol. 95, pp. 4431–4436, 1998.

[59] W. Hu, T. Tan, L. Wang, and S. Manybank, "A survey on visual surveillance of ojbect motion and behaviors," *IEEE Transactions on System, Man, and Cybernetics*, vol. 34, no. 3, pp. 334–352, 2004.

[60] P. Perez, J. Vermaak, and A. Blake, "Data fusion for fisual tracking with particles," *Proc. of the IEEE*, vol. 92, no. 3, pp. 495–513, 2004.

[61] C. Smith, C. Richards, S. Brandt, and N. Papanikolopoulos, "Visual tracking for intelligent vechicle-hightway systems," *IEEE Transactions on Vehicular Technoloty*, vol. 45, no. 4, pp. 744–759, 2002.

[62] S. Atev, H. Arumugam, O. Masoud, R. Janardan, and N. Papanikolopoulos, "A vision-based apporach to collision prediction at traffic intersections," *IEEE Transactions on Intelligent Transportation*, vol. 6, no. 4, pp. 416–423, 2005.

[63] M. Betke, J. Gips, and P. Fleming, "The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, 2002.

[64] R. Collins, "Mean-shift blob tracking through scale space," in *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, WI, July 2003, pp. 234–240.

[65] M. Isard and J. MacCormick, "BraMBLe: A bayssian multiple-blob tracker," in *Proc. of the International Conference on Computer Vision*, Vancouver, Canada, July 2001, pp. 285–295.

[66] A. Baumberg and D. Hogg, "An efficient method for coutour tracking using active shape models," in *Proc. of the IEEE Workshop on Motion for Non-rigid and Articulated Objects*, Austin, TX, Nov. 1994, pp. 194–199.

[67] L. Peihua, T. Zhang, and E. Arthur, "Visual contour tracking based on particle filters," *Image and Vision Computing*, vol. 21, no. 1, pp. 111–123, 2003.

[68] S. Se, D. Lowe, and J. Little, "Vision-based mobile robot localization and mapping using scale-invariant features," in *Proc. of the IEEE Internatioanl Conference on Robotics and Automation*, Taipei, Taiwan, July 2003, pp. 2051–2058.

[69] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. of the IEEE Internatioanl Conference on Computer Vision*, Beijing, China, Oct. 2005, pp. 1508–1515.

[70] A. Sundaresan and R. Chellappa, "Markerless motion capture using multiple cameras," in *Proc. of the Computer Vision for Interactive and Intelligent Environment Conference*, Lexington, KY, Nov. 2005, pp. 1–12.

[71] E. Aguiar, C. Stoll, C. Theobale, N. Ahmed, H. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 1–10, 2008.

[72] Y. Chen, J Min, and J Chai, "Flexible registration of human motion data with perameterized motion models," in *Proc. of the Symposium on Interactive 3D Graphics and Games*, New York, Nov. 2009, pp. 183–190.

[73] A. Sharf, D. Alcantara, T. Lewiner, C. Greif, A. Sheffer, N. Amenta, and D. Cohen-Or, "Space-time surface reconstruction using incompressible flow," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 110:1–110:10, 2008.

[74] V. Kraevoy and A. Sheffer, "Mean-value geometry encoding," *International Journal of Shape Modeling*, vol. 12, no. 41, pp. 29–46, 2006.

[75] K. Bates, P. Manning, D. Hodgetts, and W. Sellers, "Estimating mass properties of dinosaurs using laser imaging and 3D computer modelling," *PLoS ONE*, vol. 4, no. 2, pp. 4532–4552, 2009.

[76] K. Wolf, "Geometry and dynamics in refracting systems," *European Journal of Physics*, vol. 16, pp. 4–12, 1995.

[77] J. Yu and L. Wang, "Parameter optimization of simplified propulsive model for biomimetic robot fish," in *Proc. of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 179–193.

VITA

Name: Ji Zhang

Address: 316 H.R. Bright Building, Texas A&M University,

Department of Computer Science and Engineering,

College Station, TX 77843-3112

Email Address: jizhang@cse.tamu.edu

Education: B.E., Automation, Tsinghua University, 2008

M.S., Computer Engineering, Texas A&M University, 2011