Error Aware Monocular Visual Odometry using Vertical Line Pairs for Small Robots in Urban Areas [*]

Ji Zhang and **Dezhen Song**
Dept. of Computer Science and Engineering
Texas A&M University
College Station, TX, 77843
{jizhang, dzsong}@cse.tamu.edu

Abstract

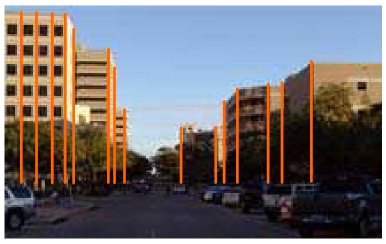
We report a new error-aware monocular visual odometry method that only uses vertical lines, such as vertical edges of buildings and poles in urban areas as landmarks. Since vertical lines are easy to extract, insensitive to lighting conditions/shadows, and sensitive to robot movements on the ground plane, they are robust features if compared with regular point features or line features. We derive a recursive visual odometry method based on the vertical line pairs. We analyze how errors are propagated and introduced 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 a method 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. The experiment results show that our method outperforms its two counterparts in robustness, accuracy, and speed. The relative errors of our method are less than 2% in experiments.

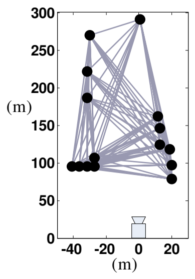
Introduction

We are interested in developing a visual odometry method for small robots in urban areas where tall buildings form a deep valley which can block GPS signals. Existing visual odometry methods are computationally challenging and cannot be used on small mobile robots with limited computation power. Employing a minimalist's approach, we only focus on the robot ego-motion estimation on the ground plane using vertical lines under a regular pinhole camera due to common requirements and configurations of small robots.

Building edges and poles are common features in urban areas (see Fig. 1(a)). These vertical lines are insensitive to lighting conditions and shadows. They are parallel to each other along the gravity direction. Extracting parallel lines using the gravity direction as a reference can be done quickly and accurately on low power computation platforms. Moreover, vertical lines are sensitive to robot motion on the ground plane.

[*]This work was supported in part by the National Science Foundation under CAREER grant IIS-0643298 and MRI program under CNS-0923203.


(a)



(b)

Figure 1: Monocular visual odometry using multiple vertical line pairs. (a) An image taken by the robot with vertical lines highlighted in orange. (b) A top view of the vertical edges (black dots in the figure) in (a) and potential choices of pairs (edges between black dots).

Utilizing the robust property of vertical lines, our new visual odometry method is error aware in landmark selection. There are often multiple vertical lines (see Fig. 1(b)) and any pair of them can provide an ego-motion estimation result with different accuracy. We analyze how errors are propagated and introduced in the continuous odometry process by deriving the recursive and closed form representation of covariance matrix. We formulate the minimum variance ego-motion estimation problem and present a method 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.

Related Work

Visual odometry (Nister, Naroditsky, and Bergen 2006; Maimone, Cheng, and Matthies 2007) utilizes images taken from on-board camera(s) to estimate the robot ego-motion. It can be viewed as a supplement when GPS signals are challenged or not available. Visual odometry is closely related to simultaneous localization and mapping (SLAM) (Thrun, Burgard, and Fox 2005) and can be viewed as a building block for visual SLAM (Davison et al. 2007;

Konolige and Agrawal 2008). A better visual odometry method will certainly increases the performance of SLAM outputs.

Visual odometry can have different sensor configurations including omnidirectional cameras and stereo vision systems. Wongphati et al. propose a fast indoor SLAM method using vertical lines from an omnidirectional camera (Wongphati, Niparnan, and Sudsang 2009). Techniques about vertical line detection and matching are developed for omnidirectional vision systems (Scaramuzza and Seigwart 2009; Caron and Mouaddib 2009). Nister et al. develop a visual odometry system to estimate the motion of a stereo head or a single camera on a ground vehicle (Nister, Naroditsky, and Bergen 2006). The stereo vision-based visual odometry on the Mars rover is a well-known example (Maimone, Cheng, and Matthies 2007). In our system, we use a regular pinhole camera due to the small form factor and low cost, which are favorable for small robots.

A different way of classifying visual odometry is what kind of features/landmarks are used. Point features, such as Harris corners, scale-invariant feature transformation (SIFT) points (Lowe 2004), and speed up robust feature (SURF) points (Bay et al. 2008) are the most popular ones since they are readily available and well developed in computer vision literature. However, point features usually contain large amounts of noise and must be combined with filtering methods such as RANdom SAmple Consensus (RANSAC) (Fischler and Bolles 1981; Hartley and Zisserman 2004) to find the correct correspondence across frames, which usually results in high computation cost. On the other hand, humans rarely view a scene as a set of isolated points while often using line features for spatial reasoning. Lines are easy to extract (Gioi et al. 2008), inherently robust, and insensitive to lighting conditions or shadows. Therefore, many visual SLAM applications employ line features and achieve very accurate results (Lemaire and Lacroix 2007; Smith, Reid, and Davison 2006; Choi, Lee, and Oh 2008).

Vertical lines are inherently parallel to each other, which can dramatically reduce the feature extraction difficulty (Zhou and Li 2007). They are sensitive to the robot motion on the ground. These properties make vertical lines robust features for visual odometry. The number of vertical lines is usually substantially less than the number of feature points or general lines in the scene, which leads to the reduction of computation costs and makes it favorable for low power platforms. In our previous work (Zhang and Song 2009), we have shown a single pair of vertical lines can provide a minimal solution for estimating the robot ego-motion. Although using a single vertical line pair is computationally efficient, it cannot provide the most accurate ego-motion estimation. Hence we adopt multiple vertical line pairs.

Problem Definition

We want to estimate the robot motion on the horizontal plane. The robot periodically takes frames to estimate its ego-motion. To focus on the most relevant issues, we begin with assumptions. We share the same notation convention in our previous work (Zhang and Song 2009).

Assumptions

1. We assume that the robot motion of the initial step 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 follows the pinhole camera model with square pixels, a zero skew factor and the lens distortion is removed by calibration. The intrinsic parameters of the camera are known from pre-calibration.
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 matrixes (Hartley and Zisserman 2004) to rotate the image planes to satisfy the condition since camera orientations can be obtained from vanishing points (Gallagher 2005) and/or potentiometers.

Notations and Coordinate Systems

In this paper, all coordinate systems are right hand systems (RHS). For the camera coordinate system (CCS), we define z-axis as the camera optical axis, and y-axis to point upward toward the sky. The camera optical axis is always parallel to the $x - z$ plane which is perfectly 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 camera principal axis intersects ICS at its origin on the image plane. To maintain an RHS, the x-axis of CCS and its corresponding u-axis in ICS must point left (see Fig. 2).

Since the image planes are perpendicular to the horizontal plane, and parallel to each other, the CCSs are iso-oriented during the computation, the robot ego-motion on the horizontal plane in different CCSs is equivalent to the displacement of the 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) coordinates of the intersections 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 ver-

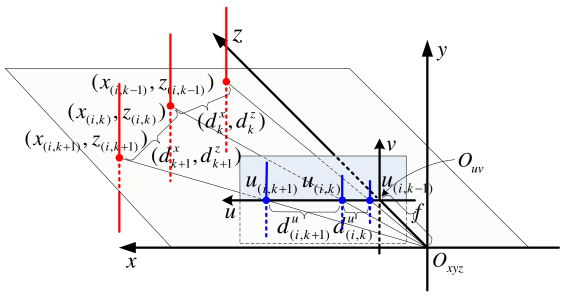


Figure 2: Superimposed CCSs $x - y - z$ and ICSs $u - v$ for the vertical line i over frames $k - 1$, k and $k + 1$.

tical lines i's displacement from frame $k-1$ to k, we have $d_k^x = x_{(i,k)} - x_{(i,k-1)}$, $d_k^z = z_{(i,k)} - z_{(i,k-1)}$.

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-coordinates of the intersections between vertical line i and u-axis for frames $k-1$, k, and $k+1$, respectively. Let $d_{(i,k)}^u$ be vertical line i's displacement in ICS from frame $k-1$ to k, we have $d_{(i,k)}^u = u_{(i,k)} - u_{(i,k-1)}$. With the above notations and coordinate systems defined, we can describe our task.

Previous Results on a Single Vertical Line Pair

In (Zhang and Song 2009), we use a pair of vertical lines (i,j) to estimate the robot ego-motion. Define f as the focal length of the camera. Given the motion of the previous step: $\mathbf{d}_k = [d_k^x, d_k^z]^T$, we can calculate the motion of the current step: $\mathbf{d}_{k+1} = [d_{k+1}^x, d_{k+1}^z]^T$ using the positions of the vertical line pair in three images, $\mathbf{u}_i = [u_{(i,k-1)}, u_{(i,k)}, u_{(i,k+1)}]^T$ and $\mathbf{u}_j = [u_{(j,k-1)}, u_{(j,k)}, u_{(j,k+1)}]^T$, as follows,

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

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

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

In the above recursive calculation, we do not know the true values of $\mathbf{d}_k$, $\mathbf{u}_i$ and $\mathbf{u}_j$. Instead, we know their measured values $\hat{\mathbf{d}}_k$, $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}_j$ with corresponding errors $\mathbf{e}_k^{\mathbf{d}} = \hat{\mathbf{d}}_k - \mathbf{d}_k$, $\mathbf{e}_i^{\mathbf{u}} = \hat{\mathbf{u}}_i - \mathbf{u}_i$ and $\mathbf{e}_j^{\mathbf{u}} = \hat{\mathbf{u}}_j - \mathbf{u}_j$. As a convention in the paper, error value e^a of a variable a is defined as $e^a = \hat{a} - a$. Hence, (1) becomes

$$\mathbf{d}_{k+1} + \mathbf{e}_{k+1}^{\mathbf{d}} = \boldsymbol{F}(\mathbf{d}_k + \mathbf{e}_k^{\mathbf{d}}, \mathbf{u}_i + \mathbf{e}_i^{\mathbf{u}}, \mathbf{u}_j + \mathbf{e}_j^{\mathbf{u}}). \tag{2}$$

When errors are small, this error propagation process can be approximated by the linearization of function $\boldsymbol{F}(\cdot)$ with respect to $\mathbf{d}_k$, $\mathbf{u}_i$ and $\mathbf{u}_j$, respectively. Therefore, we have

$$\mathbf{e}_{k+1}^{\mathbf{d}} = \boldsymbol{P}_{(i,j)}\mathbf{e}_k^{\mathbf{d}} + \boldsymbol{Q}_{(i,j)}\mathbf{e}_i^{\mathbf{u}} + \boldsymbol{Q}_{(j,i)}\mathbf{e}_j^{\mathbf{u}}, \tag{3}$$

where $\boldsymbol{P}_{(i,j)} = \partial\boldsymbol{F}/\partial\mathbf{e}_k^{\mathbf{d}}$, $\boldsymbol{Q}_{(i,j)} = \partial\boldsymbol{F}/\partial\mathbf{e}_i^{\mathbf{u}}$ and $\boldsymbol{Q}_{(j,i)} = \partial\boldsymbol{F}/\partial\mathbf{e}_j^{\mathbf{u}}$ are Jacobian matrices based on vertical line pair (i,j). Note that $\boldsymbol{Q}_{(i,j)}$ and $\boldsymbol{Q}_{(j,i)}$ are for vertical line i and j respectively. The expressions of $\boldsymbol{P}_{(i,j)}$ and $\boldsymbol{Q}_{(i,j)}$ are presented in (4) and (5) on the next page.

Problem Description

Eqs. (1) and (3) provide the recursive computation of the robot ego-motion and its error propagation for a single vertical line pair. However, there are often multiple vertical lines. For n vertical lines, there are $n(n-1)/2$ pairs. Each pair is capable of providing a solution for the robot ego-motion estimation. We are interested in providing an estimation strategy with the minimum estimation error variance.

To achieve this, we first define the ego-motion estimation as a weighted sum of the solutions from each pair. Plugging (1) in, the recursive ego-motion estimation for multiple vertical line pairs is

$$\mathbf{d}_{k+1} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{(i,j)}\boldsymbol{F}(\mathbf{d}_k, \mathbf{u}_i, \mathbf{u}_j), \tag{6}$$

where $w_{(i,j)}$ is the weight of vertical line pair (i,j). Define $I = \{1, 2, ..., n\}$ as the index set of all vertical lines. $w_{(i,j)}$'s are standardized,

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{(i,j)} = 1, \tag{7}$$

$$w_{(i,j)} = w_{(j,i)} \geq 0,\ i \in I,\ j \in I, \text{ and } i \neq j. \tag{8}$$

We want to choose a set of $w_{(i,j)}$ to minimize the ego-motion estimation error variance. Define $\Sigma_k^{\mathbf{d}}$ and $\Sigma_{k+1}^{\mathbf{d}}$ as the covariance matrices for estimation errors $\mathbf{e}_k^{\mathbf{d}}$ and $\mathbf{e}_{k+1}^{\mathbf{d}}$, respectively. Define Σ_i^u as the covariance matrix for measurement error $\mathbf{e}_i^{\mathbf{u}}$. At time k, $\Sigma_k^{\mathbf{d}}$ is known from the previous step. $\Sigma_{k+1}^{\mathbf{d}}$ is influenced by the estimation error of the previous step, $\Sigma_k^{\mathbf{d}}$, and the newly introduced error of the current step, Σ_i^u. To measure how $\Sigma_{k+1}^{\mathbf{d}}$ changes, we use its trace $\sigma_{k+1}^2 = \mathrm{Tr}(\Sigma_{k+1}^{\mathbf{d}})$ as a metric to measure the variance of $\mathbf{e}_{k+1}^{\mathbf{d}}$. Hence our problem is defined as,

Definition 1 *Given $\mathbf{d}_k$, $\mathbf{u}_i$, Σ_i^u, $i \in I$, and $\Sigma_k^{\mathbf{d}}$, derive $\Sigma_{k+1}^{\mathbf{d}}$, and compute*

$$\{w_{(i,j)}, \forall i,j \in I, i \neq j\} = \arg\min_{w_{(i,j)}} \sigma_{k+1}^2, \tag{9}$$

subject to the constraints in (7) and (8).

With $w_{(i,j)}$'s obtained, the robot ego-motion estimation can be obtained using (6).

Minimum Variance Ego-motion Estimation

In this section, we first derive the expression of σ_{k+1}^2, then we convert the minimization of σ_{k+1}^2 to a quadratic convex optimization problem. We name this method the minimum variance ego-motion estimation (MVEE) method.

Derive the Estimation Error Variance

We begin the modeling with deriving $\Sigma_{k+1}^{\mathbf{d}}$. Recall that $\Sigma_{k+1}^{\mathbf{d}}$ is the variance matrix of $\mathbf{e}_{k+1}^{\mathbf{d}}$. We know that $\mathbf{e}_{k+1}^{\mathbf{d}}$ has two parts,

$$\mathbf{e}_{k+1}^{\mathbf{d}} = \mathbf{e}_{k+1}^p + \mathbf{e}_{k+1}^m, \tag{10}$$

where $\mathbf{e}_{k+1}^p$ is the estimation error propagated from the previous step $\mathbf{e}_k^{\mathbf{d}}$, and $\mathbf{e}_{k+1}^m$ is introduced from the measurement errors of the current step $\mathbf{e}_i^{\mathbf{u}}$, $i \in I$. From (3) and (6), we

$$\boldsymbol{P}_{(i,j)} = \frac{1}{u_{(j,k+1)} - u_{(i,k+1)}} \begin{bmatrix} \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(j,k+1)} - \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} u_{(i,k+1)} & -\frac{d^u_{(i,k+1)}}{d^u_{(i,k)} f} u_{(j,k+1)} u_{(i,k-1)} + \frac{d^u_{(j,k+1)}}{d^u_{(j,k)} f} u_{(i,k+1)} u_{(j,k-1)} \\ \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} f - \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} f & -\frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(i,k-1)} + \frac{d^u_{(j,k+1)}}{d^u_{(j,k)}} u_{(j,k-1)} \end{bmatrix}, \tag{4}$$

$$\boldsymbol{Q}_{(i,j)} = \frac{1}{u_{(j,k+1)} - u_{(i,k+1)}} \begin{bmatrix} \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} u_{(j,k+1)} z_{(i,k-1)} & -(1 + \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}) u_{(j,k+1)} z_{(i,k)} & u_{(j,k+1)} z_{(i,k+1)} \\ \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}} f z_{(i,k-1)} & -(1 + \frac{d^u_{(i,k+1)}}{d^u_{(i,k)}}) f z_{(i,k)} & f z_{(i,k+1)} \end{bmatrix}. \tag{5}$$

have the expressions of $\mathbf{e}^p_{k+1}$ and $\mathbf{e}^m_{k+1}$ as,

$$\mathbf{e}^p_{k+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} \boldsymbol{P}_{(i,j)} \mathbf{e}^{\mathbf{d}}_k = \boldsymbol{R}\mathbf{e}^{\mathbf{d}}_k,$$

$$\boldsymbol{R} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} \boldsymbol{P}_{(i,j)}, \tag{11}$$

$$\mathbf{e}^m_{k+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{(i,j)} (\boldsymbol{Q}_{(i,j)} \mathbf{e}^{\mathbf{u}}_i + \boldsymbol{Q}_{(j,i)} \mathbf{e}^{\mathbf{u}}_j) = \sum_{i=1}^{n} \boldsymbol{S}_i \mathbf{e}^{\mathbf{u}}_i,$$

$$\boldsymbol{S}_i = \sum_{j=1, j\neq i}^{n} w_{(i,j)} \boldsymbol{Q}_{(i,j)}. \tag{12}$$

In the above equations, $\boldsymbol{R}$ 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 (10), 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,

$$\Sigma^{\mathbf{d}}_{k+1} = \Sigma^p_{k+1} + \Sigma^m_{k+1}, \tag{13}$$

where Σ^p_{k+1} and Σ^m_{k+1} are corresponding to $\mathbf{e}^p_{k+1}$ and $\mathbf{e}^m_{k+1}$, respectively.

Recall that the covariance matrix $\Sigma^{\mathbf{d}}_k$ of $\mathbf{e}^{\mathbf{d}}_k$ in (11) is known from the previous step. Define $e^u_{(i,k)}$ as the measurement error of line position $u_{(i,k)}$. Assume that they are independently identically distributed (i.i.d.) Gaussian with zero mean and a variance of σ^2_u. The covariance matrix Σ^u_i of $\mathbf{e}^{\mathbf{u}}_i$ in (12) is a diagonal matrix, $\Sigma^u_i = \mathrm{diag}(\sigma^2_u, \sigma^2_u, \sigma^2_u)$.

Using the covariance matrices with (11) and (12), we have

$$\Sigma^p_{k+1} = \boldsymbol{R}\Sigma^{\mathbf{d}}_k \boldsymbol{R}^T \text{ and } \Sigma^m_{k+1} = \sigma^2_u \sum_{i=1}^{n} \boldsymbol{S}_i \boldsymbol{S}_i^T. \tag{14}$$

Therefore, Σ^p_{k+1} and its trace σ^2_{k+1} can be obtained,

$$\Sigma^{\mathbf{d}}_{k+1} = \Sigma^p_{k+1} + \Sigma^m_{k+1} = \boldsymbol{R}\Sigma^{\mathbf{d}}_k \boldsymbol{R}^T + \sigma^2_u \sum_{i=1}^{n} \boldsymbol{S}_i \boldsymbol{S}_i^T, \tag{15}$$

$$\sigma^2_{k+1} = \mathrm{Tr}(\boldsymbol{R}\Sigma^{\mathbf{d}}_k \boldsymbol{R}^T) + \sigma^2_u \sum_{i=1}^{n} \mathrm{Tr}(\boldsymbol{S}_i \boldsymbol{S}_i^T). \tag{16}$$

Formulate in a Convex Optimization Problem

With the closed-form of σ^2_{k+1} derived, we can formulate the problem defined in (9) into a convex optimization problem. Let us define vector $\boldsymbol{w} = [w_1, ..., w_{n(n-1)/2}]^T$ with its a-th entry obtained as follows,

$$w_a = w_{(i,j)}, \text{ where} \begin{cases} i = 1, ..., n-1, \ j = i+1, ..., n, \\ a = (i-1)(n-i/2) + j - i. \end{cases} \tag{17}$$

Vector $\boldsymbol{w}$ is our decision vector for the optimization problem in (9), which can be rewritten as,

$$\min_{\boldsymbol{w}} \sigma^2_{k+1} = \boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w}, \text{ subject to: } -\boldsymbol{w} \leq \boldsymbol{0}, \ \boldsymbol{c}^T \boldsymbol{w} = 1, \tag{18}$$

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 from (16).

Now we 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 the newly introduced in the current step. Define $A^p_{a,b}$ as the (a,b)-th entry of matrix $\boldsymbol{A}^p$. Similarly, $A^m_{a,b}$ is the (a,b)-th entry of matrix $\boldsymbol{A}^m$. $\boldsymbol{A}^p$ and $\boldsymbol{A}^m$ are obtained as follows,

$$\begin{cases} A^p_{a,a} = \mathrm{Tr}(\boldsymbol{P}_{(i,j)} \Sigma^{\mathbf{d}}_k \boldsymbol{P}^T_{(i,j)}), & i = r, \ j = l, \\ A^p_{a,b} = A^p_{b,a} = \mathrm{Tr}(\boldsymbol{P}_{(i,j)} \Sigma^{\mathbf{d}}_k \boldsymbol{P}^T_{(r,l)}), & \text{otherwise}, \end{cases} \tag{19}$$

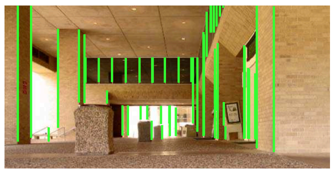
$$\begin{cases} A^m_{a,a} = \sigma^2_u (\mathrm{Tr}(\boldsymbol{Q}_{(i,j)} \boldsymbol{Q}^T_{(i,j)}) + \mathrm{Tr}(\boldsymbol{Q}_{(j,i)} \boldsymbol{Q}^T_{(j,i)}), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = r, \ j = l, \\ A^m_{a,b} = A^m_{b,a} = \sigma^2_u \mathrm{Tr}(\boldsymbol{Q}_{(i,j)} \boldsymbol{Q}^T_{(r,l)}), & i = r, \ j \neq l, \\ A^m_{a,b} = A^m_{b,a} = \sigma^2_u \mathrm{Tr}(\boldsymbol{Q}_{(j,i)} \boldsymbol{Q}^T_{(l,r)}), & i \neq r, \ j = l, \\ A^m_{a,b} = A^m_{b,a} = \sigma^2_u \mathrm{Tr}(\boldsymbol{Q}_{(j,i)} \boldsymbol{Q}^T_{(r,l)}), & j = r, \ j \neq l, \\ A^m_{a,b} = A^m_{b,a} = 0, & \text{otherwise}, \end{cases} \tag{20}$$

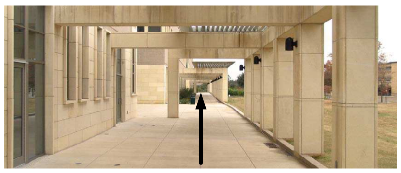
where

$$\begin{cases} i = 1, ..., n-1, \ j = i+1, ..., n, \\ r = i, ..., n-1, \ l = j, ..., n, \\ a = (i-1)(n-i/2) + j - i, \\ b = (r-1)(n-r/2) + l - r. \end{cases}$$

Matrix $\boldsymbol{A}$ is positive definite, which can be easily proved by comparing (16) with (18). Since the feasible set of the optimization problem in (18) is also convex, the problem is a quadratic convex optimization problem, which is a well studied problem in operations research. Here we use the well known interior-point method (Boyd and Vandenberghe 2006) to solve it. With the optimized weights defined in (9) obtained, we can estimate the robot ego-motion according to (6). Hence we complete our MVEE method.

(a)


(b)


(c)

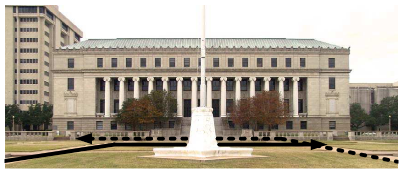

(d)

Figure 3: (a) The camera and the robot used in physical experiments. (b) Experiment site 1 from the robot view with vertical edges highlighted in green. (c,d) Experiment sites 2 and 3 with robot trajectories highlighted in black.

Experiments

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

- Nister (Nister, Naroditsky, and Bergen 2006): 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 (Lemaire and Lacroix 2007): 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 define a relative error metric ε for the comparison purpose. Let d_k^x and d_k^z be the true displacements of the robot in x- and z-directions at step k, respectively, which are measured by a tape measure in the experiments. The corresponding outputs of visual odometry are defined as $\hat{d}_k^x$ and $\hat{d}_k^z$, and ε is defined as

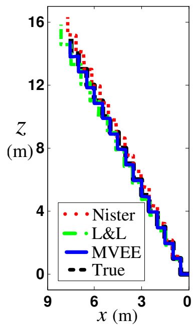
$$\varepsilon = \frac{\sqrt{(\sum_k \hat{d}_k^x - \sum_k d_k^x)^2 + (\sum_k \hat{d}_k^z - \sum_k d_k^z)^2}}{\sum_k \sqrt{(d_k^x)^2 + (d_k^z)^2}}. \quad (21)$$

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

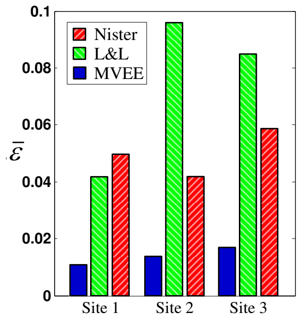
We use a Sony DSC-F828 camera mounted on a robot in the experiments (Fig. 3(a)). The camera has a $50°$ horizontal field of view and a resolution of 640×480 pixels. The robot is custom made in our lab, which measures $50 \times 47 \times 50\,\text{cm}^3$ in size. The visual odometry algorithms run on a Compaq V3000 laptop PC with an 1.6GHz dual core CPU and 1.0G RAM. We implement Nister, L&L, and MVEE on the laptop PC using MatLab 2006a.

We run tests at three experiment sites (Fig. 3(b-d)) for all three methods. At each site, the robot moves along a planed trajectory for a certain number of steps. The robot takes one image at the end of each step. The robot displacement of the first step is given as a reference. The details about each site are described below:

- Site 1: 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. 3(b) and Fig. 4(a)).



(a)



(b)

Figure 4: 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.

- Site 2: The robot moves toward the depth direction for 51 steps with a step length of 1 m (Fig. 3(c)).
- 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. 3(d)).

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.

During the experiments, we employ Gioi et al.'s method to extract the line segments from the images (Gioi et al. 2008). The vertical lines are found using an inclination angle threshold (Zhou and Li 2007) and vanishing points. Then, we employ the vanishing point method (Gallagher 2005) 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. Finally, MVEE is applied.

The experiment results of the three methods are shown in Fig. 4. Fig. 4(a) presents a representative sample trial of estimated trajectory comparison at site 1. Fig. 4(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: 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

Table 1 compares feature quality and computation speed for the three methods in each step. 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.

Conclusion and Future Work

We reported our development of a monocular visual odometry method that utilizes vertical lines in urban areas. We derived how to estimate the robot ego-motion using multiple vertical line pairs. To improve the accuracy, we analyzed how errors are propagated and introduced in the continuous odometry process by deriving the recursive and closed form representation of the error covariance matrix. We minimize the ego-motion estimation error variance by solving a convex optimization problem. The resulting visual odometry method is tested in physical experiments and compared with two existing methods, where the results show that our method is better in robustness, accuracy, and speed.

In the future, we will extend our method to 3D visual odometry by exploring different combinations of geometric features such as horizontal lines, vertical planes, and points with geometric meanings (e.g. intersections between lines and planes). We will also look into methods using texture features in combination with geometric features.

Acknowledgement

We thank Dr. J. Yi for his insightful inputs and discussions. Thanks Y. Xu, C. Kim, A. Aghamohammadi, S. Hu, and W. Li for their contributions to NetBot Laboratory in Texas A&M University.

References

Bay, H.; Ess, A.; Tuytelaars, T.; and Gool, L. 2008. Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)* 110(3):346–359.

Boyd, S., and Vandenberghe, L. 2006. *Convex Optimization*. Cambridge University Press.

Caron, G., and Mouaddib, E. 2009. Vertical line matching for omnidirectional stereovision images. In *IEEE International Conference on Robotics and Automation (ICRA)*.

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

Davison, A.; Reid, L.; Molton, N.; and Stasse, O. 2007. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(6):1052–1067.

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

Gallagher, A. 2005. Using vanishing points to correct camera rotation in images. In *The 2nd Canadian Conference on Computer and Robot Vision*.

Gioi, R.; Jakubowicz, J.; Morel, J.; and Randall, G. 2008. Lsd: A line segment detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hartley, R., and Zisserman, A. 2004. *Multiple View Geometry in Computer Vision, 2nd Edition*. Cambridge University Press.

Konolige, K., and Agrawal, M. 2008. FrameSLAM: From bondle adjustment to real-time visual mapping. *IEEE Transactions on Robotics* 24(5):1066–1077.

Lemaire, T., and Lacroix, S. 2007. Monocular-vision based SLAM using line segments. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(4):91–110.

Maimone, M.; Cheng, Y.; and Matthies, L. 2007. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics* 24(2):169–186.

Nister, D.; Naroditsky, O.; and Bergen, J. 2006. Visual odometry for ground vechicle applications. *Journal of Field Robotics* 23(1):3–20.

Scaramuzza, D., and Seigwart, R. 2009. A robust descriptor for tracking vertical lines in omnidirectional images and its use in mobile robotics. *The International Journal of Robotics Research* 28(2):149–171.

Smith, P.; Reid, I.; and Davison, A. 2006. Real-time monocular SLAM with straight lines. In *The 17th British Machine Vision Conference*.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts.

Wongphati, M.; Niparnan, N.; and Sudsang, A. 2009. Bearing only FastSLAM using vertical line information from an omnidirectional camera. In *IEEE International Conference on Robotics and Biomimetics*.

Zhang, J., and Song, D. 2009. On the error analysis of vertical line pair-based monocular visual odometry in urban area. In *International Conference on Intelligent Robots and Systems (IROS)*.

Zhou, J., and Li, B. 2007. Exploiting vertical lines in vision-based navigation for mobile robot platforms. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, I–465–I–468.