UNSYS digital

The Journal of Instrumentation, Automation and Systems

Real Time Monocular Visual Odometry using ORB Features for Indoor Environment

Bayu Kanugrahan Luknanto[†], Adha Imam Cahyadi[†], Sunu Wibirama[†], Herianto[‡]

[†]Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia

[‡]Department of Mechanical and Industrial Engineering, Universitas Gadjah Mada, Yogyakarta, Indonesia

Abstract—Navigation is a key process in many intelligent systems. The most common way to do navigation is using Global Positioning System (GPS). However, in indoor environment, GPS is inaccurate due to multi-path problem. To solve this problem, a dead reckoning system may be implemented. Visual odometry is one of the dead reckoning methods which can help solving this problem. Among many types of visual odometry algorithm, feature-based monocular visual odometry algorithm is proposed in this research. The feature detector used in this work is ORB. The features are used for triangulation and the obtained 3D structure will be used as the basic information for poses determination. Pose determination is done by solving the PnP problem. The algorithm is validated by performing six basic motion tests while the algorithm is running. The result of the tests shows that the visual odometry algorithm can determine the position and orientation with good accuracy.

Keywords—Navigation, visual odometry, monocular, feature detection.

I. Introduction

Navigation is a process or activity of accurately ascertaining one's position and planning-and-following a route. This is a key process in many intelligent systems. In order to finish their tasks, many of these have to go through navigation process first. For example, many ITS applications and services such as route guidance, fleet management, accident and emergency response, bus arrival information, and other location-based services require location information. In aerial surveillance and monitoring using drone, navigation will be done by drone so the drone can monitor a given area. All intelligent systems will include navigation in their to-do-lists to accomplish their tasks.

In the last few years, satellite navigation systems like GPS has become the main navigation technology for providing location data for many intelligent systems [8]. However, due to signal blockage and severe multi-path in urban areas, satellite navigation systems can not satisfy most navigation requirements. Because of this deficiency, dead reckoning method is augmented into satellite navigation systems to reduce position estimation error. However, dead reckoning also has its shortage; their drift errors increase rapidly with time and frequent calibration is required [12].

Recently, visual odometry (VO) is being developed as an alternative in navigation. It is proven to be capable of determining the position of a system over long distances using only images as inputs and without prior knowledge about the environment [1]. Moreover, 3D structure can be computed from motion known as structure from motion problem [13]. Therefore, it is possible to build a 3D map using vision-based algorithms. This 3D map can be used for map-matching algorithms. When these algorithms are combined with visual odometry, drift errors from visual odometry can be corrected and longer distances can be traveled without the necessity of correction of absolute position. Based on these facts, visual odometry holds a great potential to be a robust method in navigation.

However, several visual odometry algorithms such as [14] and [15] are designed under assumption that the algorithm will be used in outdoor environment. The scene seen by the camera does not change quite often. Then, the problem is to construct a visual odometry algorithm that works in indoor environment which the scene seen by the camera changes often because the objects around the camera are close to the camera.

Among many types of visual odometry algorithm, feature-based monocular visual odometry algorithm is used in this work because usually there are many features available in indoor environment. The feature detector used is ORB feature detector because of its efficiency and SURF-like robustness. After features from images are detected, triangulation process is executed and then PnP problem is solved to determine the pose.

This work begins with Section II where we explain the processes inside our visual odometry algorithm. This includes ORB feature detection and description, feature matching, point triangulation, and solving PnP problem. Then, in Section III, we put the processes in certain order and make a flow diagram of the processes so the processes can be done continuously. After that, in section IV, we describe how the experiment is conducted to test the algorithm performance and provide the result of the experiment. Finally, in Section V, we made conclusions of this research.

Corresponding author: B.K. Luknanto (e-mail: bayu.kael@gmail.com).
This paper was submitted on September 22, 2015; revised on November 11, 2015; and accepted on December 4, 2015.

II. The Proposed Method

In this section, we will explain briefly the processes inside the proposed algorithm. There will be four processes. In each process, there will be problems to be solved and the methods to solve the problems.

A. ORB Feature Detector and Descriptor

The first problem which has to be solved in developing a visual odometry algorithm is to find correspondences between images. To find such correspondences, one of the task needs to be done is feature detection. In this work, we use ORB feature detector [11] to detect features in images.

ORB Corner Detector: ORB feature detector is a feature detector algorithm proposed by Rublee et al [11]. It is based on FAST corner detector [10] and BRIEF feature descriptor [2]. Actually, the features meant to be detected in FAST feature detector are corners. Therefore, it is more appropriate to call ORB feature detector as ORB corner detector. However, in the remainder of this report, ORB corner detector and descriptor will be referred simply as ORB.

In detecting corner, first, a test is performed at a pixel p by examining a circle of 16 pixels (a Bresenham circle of radius 3) surrounding p. A feature at p is said to be there if the intensities of at least 9 contiguous pixels on the circle are greater or lower than the intensity of p by some threshold t. In other words, let the intensity of pixel candidate p be I_p. Then, the pixel p is classified as a feature if there exists a set of 9 contiguous pixels on the circle which are all brighter than I_{p+t} or all darker than I_{p-t}.

Second, use Harris corner measure [3] to give a corner response to a feature. Consider taking an image patch using an image window centered at a feature p over the area (u, v) and shifting it by (x, y). Let $I(k, l)$ be the image intensity at (k, l). In this case, the image window is gaussian window i.e. $w(u, v) = e^{-(u^2+v^2)/2\sigma^2}$. Let $I_x$and $I_y$be the partial derivatives of I. Let us define A as the structure tensor as

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix} \quad (1)$$

Then, the corner response function **M** is defined as

$$\mathbf{M} = \det A - \kappa \operatorname{trace}^2 A \quad (2)$$

where κ is tunable sensitivity parameter. A feature is classified as a corner if its corner response function **M** is positive. The corner response function of every features detected in the first stage is calculated in order to take *N* most stable corner. If a feature has a positive corner response, it is accepted as a corner. Consequently, every feature with non-positive corner response is rejected.

The third stage of this corner detection is to take *N* most stable corner. All detected features are sorted by their corner response in descending order where the top feature has the greatest corner response. For a target number *N* of most stable features, the *N* top features are picked.

The next stage is to give a measure of corner orientation using intensity centroid [9]. The intensity centroid assumes that a corner's intensity is offset from its center. The vector formed from a corner center to its intensity may be used to indicate the corner's orientation. Rosin defines the moments of a patch as [9]

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y) \quad (3)$$

With this equation, the centroid of a corner's patch is given by

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \quad (4)$$

Assuming the patch coordinate frame has been set to the corner center, the corner orientation is simply written as

$$\theta = \arctan2(m_{01}, m_{10}) \quad (5)$$

where arctan2 is the quadrant-aware version of arctan. In this work, the corner type does not take into account in calculating corner orientation.

In order to be scale-invariant, a scale pyramid of the image is generated and then at each level in the pyramid, ORB feature is produced.

ORB Feature Descriptor: After corner detection has been executed properly, in order to be matched with the features in another image, each detected corner needs to be described by a feature descriptor. Here, the feature descriptor used is rotated BRIEF, a modified version of original BRIEF [2]. The BRIEF descriptor [2] is a bit string description of an image patch constructed from a set of binary intensity tests. There are several steps or stages performed by the BRIEF descriptor to describe a feature.

The first stage is to smooth the image patch. It is done in order to reduce the sensitivity of this descriptor to noise. The smoothing is achieved by using an integral image which each test point is a 5×5 sub-window of a 31×31 image patch. This smoothing method were first suggested by Rublee et al [11].

After the image patch has been smoothed, the next stage is to perform the binary tests. Consider a smoothed image patch, I, which has the center of the image patch as the origin of the patch coordinate frame. A binary test β is defined by:

$$\beta(I; \mathbf{a}, \mathbf{b}) \equiv \begin{cases} 1 \text{ for } I(\mathbf{a}) < I(\mathbf{b}) \\ 0 \text{ for } I(\mathbf{a}) \geq I(\mathbf{b}) \end{cases} \quad (6)$$

where $I(\mathbf{a})$ is the intensity of the image patch at point **a** and $I(\mathbf{b})$ is the intensity of the image patch at point **b.** Note that **a** and **b** uses patch coordinate frame. By performing n binary test over an image patch, the feature description is defined as a vector c:

$$c_n(I) \equiv \sum_{1 \leq i \leq n} 2^{i-1} \beta(I; \mathbf{a_i}, \mathbf{b_i}) \quad (7)$$

To be rotational-aware, the third stage is to steer the image patch according to the orientation of the feature. Let L be a $4 \times n$ matrix that contains set of pair $\mathbf{a_i}$, $\mathbf{b_i}$ as the location of n binary tests over an image patch. Notice that $\mathbf{a_i}$and $\mathbf{b_i}$ are each a column-vector of 2 elements. Then, L can be written as:

$$L \equiv \begin{pmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_n \\ \mathbf{b}_1 & \dots & \mathbf{b}_n \end{pmatrix}. \quad (8)$$

Using the feature orientation θ and the corresponding 2×2 rotation matrix R_θ the steered version L_θ of L is:

$$L_\theta = [R_\theta \quad R_\theta] L. \quad (9)$$

Now, the rotated BRIEF operator becomes:

$$d_n(I, \theta) \equiv c_n(I) | (\mathbf{a_i}, \mathbf{b_i}) \in L_\theta \quad (10)$$

To simplify the computation, the orientation angle is rounded to increments of $2\pi/30$ or 12 degrees and construct a look-up table of precomputed BRIEF patterns.

B. Feature Matching

To find correspondences between two images, ORB features are extracted from these two images. The first image is assumed to be taken before the second image. Then, for each feature found in the second image, calculate the hamming distance between this feature and every feature in the first image. Two features in the first image with the lowest hamming distance to this feature are taken as 2 nearest neighbor. This feature, the feature in the second image, is considered a match with the first neighbor if the ratio of the hamming distance of the first neighbor to the hamming distance of the second neighbor is smaller than a threshold. Otherwise, it is not a match.

C. Linear Triangulation Method

In this part, linear triangulation method will be used. To be precise, the linear triangulation method used is Linear-Eigen method. Consider a projection equation $\mathbf{x} = P\mathbf{X}$ where $\mathbf{x} = z[u, v, 1]^T$, $\mathbf{X} = [x, y, z, 1]^{\mathrm{T}}$ and $[u, v]$ is the measured point in the image coordinates. By denoting $\mathbf{p}_i^T$ as the i-th row of projection matrix P one may get an equation system:

$$u\mathbf{p}_3^T\mathbf{X} - \mathbf{p}_1^T\mathbf{X} = 0 \quad (11)$$

$$v\mathbf{p}_3^T\mathbf{X} - \mathbf{p}_2^T\mathbf{X} = 0 \quad (12)$$

Given a pair of point measurement from two cameras viewing a same scene $\mathbf{x} = z[u, v, 1]^T$, $\mathbf{x}' = z'[u', v', 1]^T$ and the second camera projection matrix P' , with the same procedure as above the second view, an equation in matrix form $G\mathbf{X} = \mathbf{0}$ can be obtained where:

$$G = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}'^T_3 - \mathbf{p}'^T_1 \\ v'\mathbf{p}'^T_3 - \mathbf{p}'^T_2 \end{bmatrix} \quad (13)$$

To solve for $\mathbf{X}$, in Linear-Eigen method, $\|G\mathbf{X}\|$ is minimized subject to condition $\|\mathbf{X}\| = 1$. According to Hartley and Zisserman [4], let $G = UDV^T$ be the singular value decomposition of G.The triangulated point $\mathbf{X}$ is simply the last column of V.

D. Solving Perspective-n-Point Problem

Problem Formulation: Given n 3D points in world coordinate frame represented in-homogenous 3D vector $\widetilde{\boldsymbol{X}}_i$ with i denotes the i-th point. Each 3D point $\widetilde{\boldsymbol{X}}_i$ corresponds to a normalized point $\widehat{\boldsymbol{x}}_i$ in camera coordinate frame on an image plane represented in homogenous 3D vector. This point $\widehat{\boldsymbol{x}}_i$ can also be considered as a 3D point in camera coordinate frame represented in in-homogenous 3D vector. Let $\bar{\boldsymbol{x}}_i$ be the direction of $\widehat{\boldsymbol{x}}_i$ so

$$\bar{\boldsymbol{x}}_i = \frac{\widehat{\boldsymbol{x}}_i}{\|\widehat{\boldsymbol{x}}_i\|} \quad (14)$$

Because of a measurement noise ϵ_i , the measurement equation of the measured point can be written as:

$$\bar{\mathbf{r}}_i = \bar{\mathbf{x}}_i + \epsilon_i. \quad (15)$$

The 3D point $\widetilde{\mathbf{X}}_i$ is seen from camera coordinate frame as $\widetilde{\mathbf{X}}_{i_{cam}}$ so

$$\widetilde{\mathbf{X}}_{i_{cam}} = R\widetilde{\mathbf{X}}_i + \mathbf{t} \quad (16)$$

where R is a world-to-camera rotation matrix and $\mathbf{t}$ is a world-to-camera translation matrix. Define α_i as the distance of the point $\widetilde{\boldsymbol{X}}_i$ to the camera center $\widetilde{\mathbf{C}}$:

$$\alpha_i = \left\| R\widetilde{\mathbf{X}}_i + \mathbf{t} \right\| \quad (17)$$

so the 3D point $\widetilde{\mathbf{X}}_{i_{cam}}$ can be written in terms of $\bar{\mathbf{x}}_i$ as

$$\widetilde{\mathbf{X}}_{i_{cam}} = \alpha_i \bar{\mathbf{x}}_i \quad (18)$$

Perspective-n-Point (PnP) problem can be formulated as the following constrained non-linear least-squares problem:

$$\begin{aligned} \underset{\alpha_i, R, \mathbf{t}}{\text{minimize}} \quad & J \\ \text{subject to} \quad & R^T R = I, \\ & \det R = 1, \\ & \alpha_i = \|R\widetilde{\mathbf{X}}_i + \mathbf{t}\| \end{aligned}$$

where the cost function J is the sum of the squared measurement errors, i.e.,

$$J = \sum_{i=1}^{n} \|\bar{\mathbf{r}}_i - \bar{\mathbf{x}}_i\|^2 = \sum_{i=1}^{n} \left\| \bar{\mathbf{r}}_i - \frac{1}{\alpha_i}\left(R\widetilde{\mathbf{X}}_i + \mathbf{t}\right) \right\|^2 \quad (19)$$

Unfortunately, J is non-linear in unknown quantities. Computing all the local minimum can be quite challenging. Therefore, Hesch and Roumeliotis [5] suggests to relax the optimization problem and manipulate the measurement equation to reduce the number of unknowns.

Optimization Problem Relaxation: The first two constraints in the optimization problem above can be satisfied using the Rodrigues formula [6] of the rotation matrix. In the Rodrigues formula, the rotation matrix is parametrized to only 3 parameters, called rodrigues vector. Let these parameters be written as $\omega = [w_1, w_2, w_3]^T$. According to the Rodrigues formula, the rotation matrix R in terms of ω is:

$$R = I + \frac{\omega_\times}{\|\omega\|} sin\|\omega\|\theta + \frac{\omega_\times^2}{\|\omega\|^2}(1 - cos\|\omega\|\theta) \quad (20)$$

where ω is the rotation axis or rodrigues vector, $\omega_\times$ is the rotation axis in the form of a skew-symmetric matrix so

$$\omega_\times = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \quad (21)$$

and $\|\omega\|\theta$ is the amount of rotation. To simplify the equation, θ is assumed to be 1 and hence, the amount of rotation is simply $\|\omega\|$. Using the Rodrigues formula, any real-valued ω will satisfy the constraint $R^T R = I$ and $\det R = I$. Now, because the rotation matrix R is a function of ω, the rotation matrix will be denoted as $R(\omega)$.

Let us consider equation (16). By substituting (18) into equation (16), there will be geometric constraint:

$$\alpha_i \bar{\mathbf{x}}_i = R\tilde{\mathbf{X}}_i + \mathbf{t} \quad (22)$$

For n points, this equation can be written in matrix form $A\mathbf{a} = W\mathbf{b}$ where:

$$A = \begin{bmatrix} \bar{\mathbf{x}}_1 & & -I \\ & \ddots & \vdots \\ & \bar{\mathbf{x}}_n & -I \end{bmatrix} \quad (23)$$

$$\mathbf{a} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \mathbf{t} \end{bmatrix} \quad (24)$$

$$W = \begin{bmatrix} R(\omega) & & \\ & \ddots & \\ & & R(\omega) \end{bmatrix} \quad (25)$$

$$\mathbf{b} = \begin{bmatrix} \tilde{\mathbf{X}}_1 \\ \vdots \\ \tilde{\mathbf{X}}_n \end{bmatrix}. \quad (26)$$

From this, $\mathbf{t}$ and α_i can be expressed in terms of other quantities:

$$\mathbf{a} = (A^T A)^{-1} A^T W\mathbf{b} = \begin{bmatrix} \mathrm{U} \\ \mathrm{V} \end{bmatrix} W\mathbf{b} \quad (27)$$

Where

$$U = \begin{bmatrix} \bar{\mathbf{x}}_1^T & & \\ & \ddots & \\ & & \bar{\mathbf{x}}_n^T \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{x}}_1^T \\ \vdots \\ \bar{\mathbf{x}}_n^T \end{bmatrix} V \quad (28)$$

Now, the pose-determination problem can be reformulated as the following unconstrained least-squares minimization problem:

$$\underset{\omega}{\text{minimize}} \quad J'$$

Where the new cost function J' is

$$J' = \sum_{i=1}^{n} \left\| \mathbf{u}_i^T W\mathbf{b}\bar{\mathbf{r}}_i - R(\omega)\tilde{\mathbf{X}}_i - VW\mathbf{b} \right\|^2 \quad (29)$$

III. Algorithm Design

This section explains about the developed monocular VO algorithm. There will be three parts: the initialization, the main algorithm, and the full algorithm. In the initialization part, procedure to start the algorithm will be explained. The flowchart of initialization will be described also. Meanwhile, in the main algorithm part, every steps performed in this algorithm after initialization will be explained. The last, the full algorithm part, will explain the whole algorithm. In this part, the initialization procedure and the main algorithm will be combined into a single flowchart.

A. Initialization

An initialization procedure needs to be executed first in order this algorithm to work properly because there is not enough information to process. In order to use P*n*P solver, the 3D structure needs to be known first. The 3D structure can be known from triangulation of points viewed from two different viewpoints. However, the triangulation can only be done if two camera poses is known first. The first camera poses can be assumed to be located at the origin of world coordinate frame and having an identity matrix as its rotation matrix while the second camera cannot. Hence, an initialization procedure is needed to obtain the second camera pose.

To obtain the second camera pose without 3D structure, the essential matrix generated from first two images can be used. To do this, first we compute the essential matrix using five-point algorithm proposed by Nister [7]. After the essential matrix has been computed, the essential matrix is decomposed using singular value decomposition (SVD). After that, four solutions given by SVD of the essential matrix are checked using cheirality check to find the true solution.

The five-point algorithm needs minimum five corresponding points. To give enough corresponding points, features are extracted from two images using ORB feature detector. Next, the features found in the second image are matched with the features found in the first image using 2 nearest-neighbor and ratio test.

Wrapping it all up, in initialization mode, the machine will:

1) Capture a new image.
2) Extract features from the image using ORB feature detector.
3) If the captured image is the first image, set this image, its features, and its external camera parameters assumed as [I | **0**] as the first frame and go back to 1. Otherwise, set this image and its features as the second frame and continue.
4) Match the features in the second frame to the features in the first frame using 2 nearest-neighbor and ratio test algorithm.
5) Compute essential matrix between the second frame and the first frame.
6) Recover the camera pose from the essential matrix.
7) Check if there is user interaction. If there is user interaction, exit initialization mode. Otherwise, back to 1.

B. Main Algorithm

After initialization procedure has been initiated, the main algorithm is executed. In main algorithm part, the camera pose will be estimated using the 3D structure of the scene. This 3D structure can be obtained through the triangulation of points from two previous camera poses. Since initialization procedure has been done, when this algorithm is started in the first time, there will be no problem in triangulation as the two previous camera poses has been found.

The triangulation from two previous camera poses is done first. With this triangulation, the 3D structure of the scene (only the features, though) can be obtained. As stated in chapter II, the most common one, the Linear-Eigen triangulation method, will be used to obtain the 3D structure.

In order to estimate the current camera pose using PnP solver, the next step is to find 3D-to-2D point correspondences after the 3D structure of the scene has been computed. This is done by matching the features in the current frame to the features in previous frame and eliminate the matched feature that does not correspond to the available 3D structure. In other words, the 3D structure are assigned with the feature descriptors from the previous frame.

Given 3D-to-2D point correspondences, PnP solver will now be used to find the real camera pose of current frame. The PnP solver used in this work will be DLS PnP solver proposed by Hesch and Roumeliotis [5].

As the camera moves away from the previous camera pose, the matched features between the current frame and the previous frame are getting fewer. This leads to inaccurate camera pose estimation. If the camera keeps moving away, the DLS PnP solver will run out of 3D-to-2D correspondences and thus failing. To overcome this problem, a new frame needs to be inserted and a new triangulation is done using point correspondences found between the two previous frames. Then, the 3D-to-2D correspondences is found using the new 3D structure.

When to insert a new frame is now being a question. In order to triangulate properly, the new frame has to be taken at different viewpoint. It should not be too far to the previous frame as the point correspondences is getting fewer but also not too close to the previous frame. A simple rule is then applied; when the current frame is one unit away from the previous frame, a new frame is inserted.

In summary, after initialization procedure, when the machine enters the main algorithm, the machine will:

1) Capture a new image.
2) Extract features from the image using ORB feature detector.
3) If this image is the first image after initialization procedure, add a new frame as current frame then continue. Otherwise, continue.
4) Set this image and its features as the current frame f_n.
5) If a new frame has been added, triangulate the point correspondences between the first previous frame f_{n-1} and the second previous frame f_{n-2}. Otherwise, continue.
6) Find 3D-to-2D correspondences between the latest 3D structure and the current frame f_n.
7) Compute the current camera pose using the DLS PnP solver.
8) If the current camera pose is one unit away from the precious frame, add a new frame $n = n + 1$. Otherwise, continue.
9) Go back to 1.

IV. Results and Discussion

To test the performance of the algorithm, six tests are conducted, i.e. sideways motion test, forward and backward motion test, upward and downward motion test, L-shaped motion test 1, L-shaped motion test 2, rotational motion test. All of these tests are performed in our laboratory during the day.

The algorithm is coded using OpenCV 3.0 library which has many features such as Rublee's ORB feature detection [11], feature matching, linear triangulation, PnP solver, and the implementation of Nister's five-point algorithm [7].

For test number 1, i.e. sideways motion test, for each meter, the algorithm produced drift error by 48 mm on x-axis, 28 mm on y-axis, and 72 mm on z-axis. The traversed path is shown on **Figure 1**.

For test number 2, i.e. forward and backward motion test, for each meter, the algorithm produced drift error by 26 mm on x-axis, 93 mm on y-axis, and 74 mm on z-axis. The traversed path is shown on **Figure 2**.

For test number 3, i.e. upward and downward motion test, for each meter, the algorithm produced drift error by 55 mm on x-axis, 93 mm on y-axis, and 186 mm on z-axis. The traversed path is shown on **Figure 3**.

For test number 4, i.e. first L-shaped motion test, for each meter, the algorithm produced drift error by 44 mm on x-axis, 41 mm on y-axis, and 103 mm on z-axis. The traversed path is shown on **Figure 4**.

For test number 5, i.e. second L-shaped motion test, for each meter, the algorithm produced drift error by 18 mm on x-axis, 33 mm on y-axis, and 89 mm on z-axis. The traversed path is shown on **Figure 5**.

For test number 6, i.e. rotational motion test, for each meter, the algorithm produced drift error by 54.75 mm on x-axis, 77.67 mm on y-axis, and 31.83 mm on z-axis. The traversed path is shown on **Figure 6**.

Putting all these results into **Table I**, we get the average drift errors are 40.958 mm on x-axis, 60.945 mm on y-axis, and 92.638 mm on z-axis. Also, we define the accuracy η as:

$$\eta = \left(1 - \frac{\sqrt{\bar{x}_{drift}^2 + \bar{y}_{drift}^2 + \bar{z}_{drift}^2}}{\sqrt{\bar{x}_{drift}^2 + \bar{y}_{drift}^2 + \bar{z}_{drift}^2} + 1000}\right) 100\%$$

Then, the accuracy is $\eta = 89.43\%$.

TABLE I THE DRIFT ERROR PRODUCED IN SIX TESTS

Test No.	(in $\frac{mm}{m}$)		
	$\bar{x}_{drift}$	$\bar{y}_{drift}$	$\bar{z}_{drift}$
1	48	28	72
2	26	93	74
3	55	93	186
4	44	41	103
5	18	33	89
6	54.75	77.67	31.83
mean	40.958	60.945	92.638

We also measure the computation time of the algorithm and we get the result:

Feature detection	:	4.4197 ms
Finding 3D-2D correspondences	:	5.4905 ms
Solving PnP problem	:	14.2656 ms
Total Time	:	24.5586 ms.

So, the maximum frame the algorithm can handle is $f = \frac{1}{24.5586 \times 10^{-3}} = 40.7189$ frames per second.

V. CONCLUSIONS

By looking at the result, we can see that this algorithm achieve 89.43% of accuracy and is able to handle up to 40.7189 frames per second. Considering the good accuracy and real-time performance of this algorithm, we call our output as visual odometry.

As an addition, we successfully built this algorithm using the well-known OpenCV library. This opens the possibility to build an embedded system that relies on OpenCV library for real-time visual odometry.

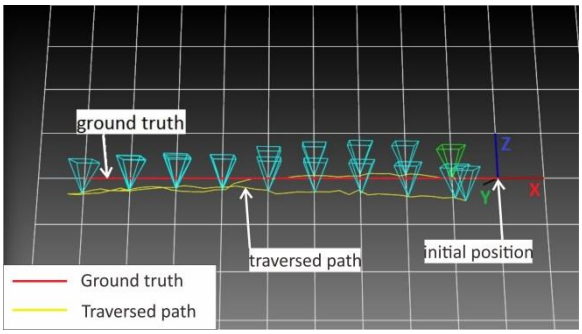

Figure 1 The traversed path in the sideways motion test

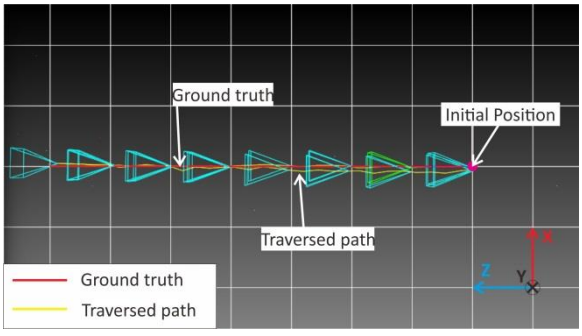

Figure 2 The traversed path in the forward and backward motion test

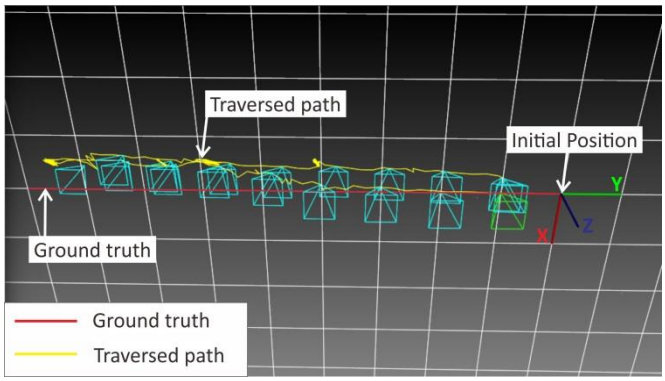

Figure 3 The traversed path in the upward and downward motion test

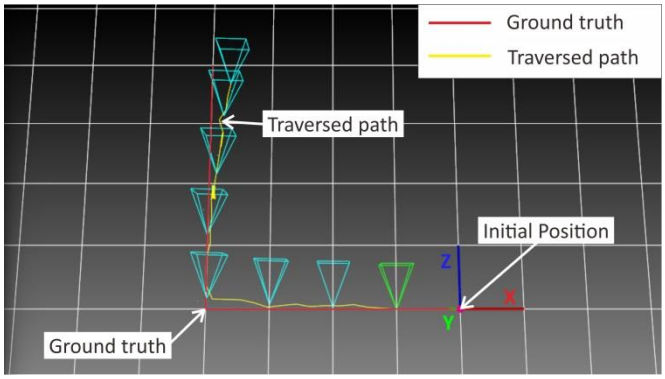

Figure 4 The traversed path in the L-shaped motion test 1

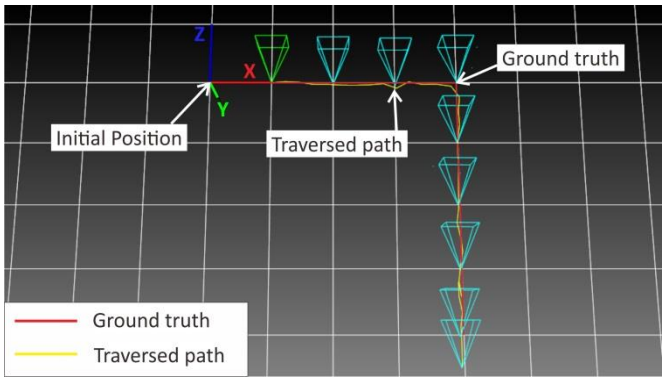

Figure 5 The traversed path in the L-shaped motion test 2

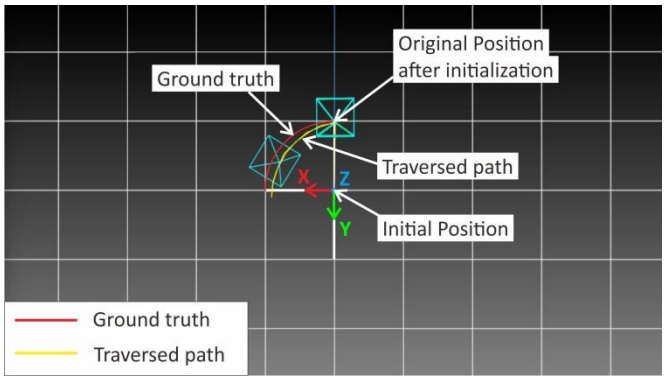

Figure 6 The traversed path in the rotational motion test

REFERENCES

[1] M. Agrawal and K. Konolige, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS," in Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. IEEE, 2006, pp. 1063–1068. CrossRef

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in Proceedings of the 11th European Conference on Computer Vision: Part IV, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 778–792. CrossRef

[3] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147–151. CrossRef

[4] R. Hartley and A. Zisserman, Multiple view geometry in computer vision, 2nd ed. Cambridge, UK ; New York: Cambridge University Press, 2003.

[5] J. A. Hesch and S. I. Roumeliotis, "A Direct Least-Squares (DLS) method for PnP," in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, Nov. 2011, pp. 383–390. CrossRef

[6] R. M. Murray, Z. Li, and S. Sastry, A mathematical introduction to robotic manipulation. Boca Raton: CRC Press, 1994.

[7] D. Nister, "An efficient solution to the five-point relative pose problem," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 756–770, Jun. 2004. CrossRef

[8] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," Transportation Research Part C: Emerging Technologies, vol. 15, no. 5, pp. 312–328, Oct. 2007. CrossRef

[9] P. L. Rosin, "Measuring Corner Properties," Computer Vision and Image Understanding, vol. 73, no. 2, pp. 291–307, Feb. 1999. CrossRef

[10] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. IEEE, 2005, pp. 1508–1515 Vol. 2. CrossRef

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, Nov. 2011, pp. 2564–2571. CrossRef

[12] C. Wu, Y. Meng, L. Zhi-lin, C. Yong-qi, and J. Chao, "Tight integration of digital map and in-vehicle positioning unit for car navigation in urban areas," Wuhan University Journal of Natural Sciences, vol. 8, no. 2, pp. 551–556, Jun. 2003. CrossRef

[13] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," IEEE Robotics & Automation Magazine, vol. 18, no. 4, pp. 80–92, Dec. 2011. CrossRef

[14] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in Proc. Int. Conf. Computer Vision and Pattern Recognition, 2004, pp. 652–659. CrossRef

[15] P. I. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2005, pp. 4007–4012.