IEIE Transactions on Smart Processing and Computing, vol. 4, no. 4, August 2015
http://dx.doi.org/10.5573/IEIESPC.2015.4.4.216

Development of Visual Odometry Estimation for an Underwater Robot Navigation System

Kandith Wongsuwan and Kanjanapan Sukvichai

Department of Electrical Engineering, Kasetsart University / Chatuchak, Bangkok, Thailand
kandithws@yahoo.com, fengkpsc@ku.ac.th

***** Corresponding Author: Kandith Wongsuwan

Received July 15, 2015; Revised August 5, 2015; Accepted August 24, 2015; Published August 31, 2015

** Regular Paper: This paper reviews the recent progress, possibly including previous works in a particular research topic, and has been accepted by the editorial board through the regular reviewing process.*

** Extended from a Conference: Preliminary results of this paper were presented at the ITC-CSCC 2015. The present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** The autonomous underwater vehicle (AUV) is being widely researched in order to achieve superior performance when working in hazardous environments. This research focuses on using image processing techniques to estimate the AUV's egomotion and the changes in orientation, based on image frames from different time frames captured from a single high-definition web camera attached to the bottom of the AUV. A visual odometry application is integrated with other sensors. An internal measurement unit (IMU) sensor is used to determine a correct set of answers corresponding to a homography motion equation. A pressure sensor is used to resolve image scale ambiguity. Uncertainty estimation is computed to correct drift that occurs in the system by using a Jacobian method, singular value decomposition, and backward and forward error propagation.

***Keywords*:** Underwater robot, Visual odometry, Monocular odometry, AUVs, Robot navigation

1. Introduction

The underwater autonomous vehicle (AUV) is still in development but aims to be effective when working in the industrial field. To create an autonomous robot, one of the important things is a strategy to autonomously navigate the robot to desired destinations. Several techniques are used to estimate its motion by using imaging sonar or Doppler velocity log (DVL). Because the cost per sensor device is extremely high, an alternative for AUV navigation is implemented in this research by using a visual odometry concept, which is normally used in mobile robots.

In our design procedure, the monocular visual odometry estimation was done by using a single high-definition camera attached to the bottom of the robot, grabbing different time sequences of images and calculating the robot's movement from changes between two images. Assume that the roll, pitch and depth of the robot in relation to the floor of the testing field is known, the monocular visual odometry concept is designed as seen in Fig. 1.

2. Odometry Estimation via Homography

The implementation is based on using a single pin-hole camera. The Shi-Tomasi [1] method and Lucas-Kanade pyramidal optical flow [2] are used in order to estimate a different time image homography. Optical flow is implemented in OpenCV for the feature matching algorithm. A random sample consensus (RANSAC) method is used to eliminate any feature outliers. Let the estimated projective homography between frames be $\boldsymbol{H}_{12}$, and let the camera intrinsic parameter be $\boldsymbol{A}_1$. Hence, the calibrated homography is shown in Eq. (1):

$$\mathbf{H}_{12}^{c} = \mathbf{A}_1^{-1}\mathbf{H}_{12}\mathbf{A}_1 = \mathbf{R}_2\left(\mathbf{I} - \frac{\mathbf{t}_2}{d}\mathbf{n}_1^{t}\right) \tag{1}$$

where

- $\boldsymbol{R}_2$ is the camera's rotation matrix
- $\boldsymbol{t}_2$ is the camera's translation vector

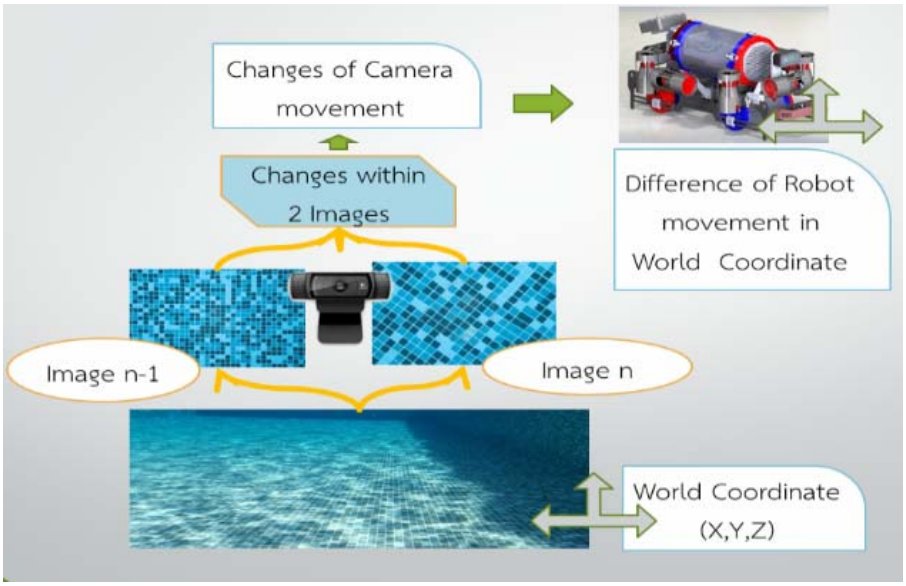

Fig. 1. The monocular visual odometry concept.

- $\boldsymbol{n}_1$ is a normal vector to the object (ground) plane, and
- d is the distance to the object (ground) plane in meters

In advanced, every single image frame is adjusted for its rotation about the x and y axes, Roll (φ) and Pitch (θ) are known by applying a compensated homography to the image. If $\mathbf{t}_2 = [000]^T$ from Eq. (1), the compensated homography can be rewritten as Eq. (2):

$$\mathbf{H} = \mathbf{A}_1 \mathbf{R}_\varphi \mathbf{R}_\theta \mathbf{A}_1^{-1} \tag{2}$$

From Eq. (1), two sets of solutions can be obtained: $S_1 = \{\mathbf{R}_{12}^1, \mathbf{t}_2^1, \mathbf{n}_1^1\}$ and $S_2 = \{\mathbf{R}_{12}^2, \mathbf{t}_2^2, \mathbf{n}_1^2\}$. The criteria for choosing a correct solution is, as both frames are compensated on the same plane (plane normal vector direction toward camera), Roll (φ) and Pitch (θ) of the correct rotation matrix (either $\mathbf{R}_{12}^1$ or $\mathbf{R}_{12}^2$) must be set to about zero, which must correspond to the correct set of answers, as shown in Eq. (3):

$$S_i = \underset{i}{\operatorname{argmin}}(\sqrt{\varphi_i^2 + \theta_i^2}),\ i = 1,2 \tag{3}$$

3. Covariance Matrix Estimation

An odometry estimation of the sensors' covariance matrix is needed in order to determine the uncertainty occurring in the system. To estimate the uncertainty of rotation about the z-axis, or yaw (ψ), and horizontal translation (x, y), there are two steps as shown in sections 3.1 and 3.2.

3.1 Homography Covariance Matrix

First, backward error propagation is used to find a homography error covariance matrix [3]:

$$\Sigma_{p'} = \left(\frac{\partial \boldsymbol{p'}}{\partial h_{ij}, \mathbf{x}_i}^T \Sigma_{x'}^{-1} \frac{\partial \boldsymbol{p'}}{\partial h_{ij}, \mathbf{x}_i} \right)^\dagger \tag{4}$$

where

- $\boldsymbol{p'}$ is the vector of the matched feature point in the second image
- $\frac{\partial \boldsymbol{p'}}{\partial h_{ij}, \mathbf{x}_i}$ is a Jacobian matrix of $\boldsymbol{p'}$
- † is a pseudo inverse.
- Σ is a covariance matrix

For $p_i^{'}$ and each element of $p_i^{'}$ in the Jacobian $\frac{\partial \boldsymbol{p'}}{\partial h_{ij}, \mathbf{x}_i}$, if the estimate point is normalized in the image plane, the estimate homography is affine in all cases ($h_{31} = h_{32} = 0$ and $h_{33} = 1$). So, we have

$$\begin{aligned} \hat{x}_i^{'} &= h_{11}\hat{x}_i + h_{12}\hat{y}_i + h_{13} \\ \hat{y}_i^{'} &= h_{21}\hat{x}_i + h_{22}\hat{y}_i + h_{23} \end{aligned} \tag{5}$$

By taking a partial derivative of Eq. (5), the Jacobian element is:

$$\begin{aligned} \frac{\partial \hat{x}_i^{'}}{\partial \boldsymbol{h}_{11}} &= \hat{x}_i, \frac{\partial \hat{x}_i^{'}}{\partial \boldsymbol{h}_{12}} = \hat{y}_i, \frac{\partial \hat{x}_i^{'}}{\partial \boldsymbol{h}_{13}} = 1 \\ \frac{\partial \hat{y}_i^{'}}{\partial \boldsymbol{h}_{21}} &= \hat{x}_i, \frac{\partial \hat{y}_i^{'}}{\partial \boldsymbol{h}_{22}} = \hat{y}_i, \frac{\partial \hat{y}_i^{'}}{\partial \boldsymbol{h}_{23}} = 1 \\ \frac{\partial \hat{x}_i^{'}}{\partial \hat{x}_i} &= h_{11}, \frac{\partial \hat{x}_i^{'}}{\partial \hat{y}_i} = h_{12}, \frac{\partial \hat{y}_i^{'}}{\partial \hat{x}_i} = h_{21}, \frac{\partial \hat{y}_i^{'}}{\partial \hat{y}_i} = h_{22} \end{aligned} \tag{6}$$

And the other element of the Jacobian is 0, since we assume that each pixel feature is independent.

In this case, variance of error that occurs in the matching algorithm is assumed to be less than 1 pixel, and there is no error in first-image acquisition. $6\boldsymbol{\sigma}$ is used to determine the variance in every single pixel when all of them are independent. Then, the Jacobian of the SVD is computed by using Eq. (7) in order to solve another layer of back error propagation:

$$\frac{\partial \mathbf{H}_{12}^{c}}{\partial h_{ij}} = \frac{\partial \mathbf{U}}{\partial h_{ij}}\mathbf{D}\mathbf{V}^{T} + \mathbf{U}\frac{\partial \mathbf{D}}{\partial h_{ij}}\mathbf{V}^{T} + \mathbf{U}\mathbf{D}\frac{\partial \mathbf{V}^{T}}{\partial h_{ij}} \tag{7}$$

From Eq. (7), the equation is solved as referred to by Papadopoulo and Lourakis [4], as follows:

$$\frac{\partial \mathbf{D}}{\partial h_{ij}} = \frac{\partial \lambda_k}{\partial h_{ij}} = u_{ik}v_{jk}, k = 1,2,3 \tag{8}$$

$$\left.\begin{aligned} \lambda_l \Omega_{U,kl}^{ij} + \lambda_k \Omega_{V,kl}^{ij} &= u_{ik}v_{jl} \\ \lambda_k \Omega_{U,kl}^{ij} + \lambda_l \Omega_{V,kl}^{ij} &= -u_{il}v_{jk} \end{aligned}\right\} \tag{9}$$

where the index ranges are $k = 1,2,3$ and $l = i+1, i+2$. Since $\lambda_k \neq \lambda_l$, the 2x2 equation system (9) has a unique solution that is practically solved by using Cramer's Rule:

$$\Omega_{U,kl}^{ij} = \frac{\begin{vmatrix} u_{ik}v_{jl} & \lambda_k \\ -u_{il}v_{jk} & \lambda_l \end{vmatrix}}{\sqrt{\lambda_l^2 - \lambda_k^2}} \tag{10}$$

$$\Omega_{V,kl}^{ij} = \frac{\begin{vmatrix} \lambda_k & u_{ik}v_{jl} \\ \lambda_l & -u_{il}v_{jk} \end{vmatrix}}{\sqrt{\lambda_l^2 - \lambda_k^2}} \tag{11}$$

And finally, we obtain the Jacobian of $\mathbf{U}$ and $\mathbf{V}$ from

$$\frac{\partial \mathbf{U}}{\partial h_{ij}} = \mathbf{U}\Omega_U^{ij} \tag{12}$$

$$\frac{\partial \mathbf{V}}{\partial h_{ij}} = -\mathbf{V}\Omega_V^{ij} \tag{13}$$

From Eqs. (8), (12), and (13), $\frac{\partial \mathbf{U}}{\partial h_{ij}}$, $\frac{\partial \mathbf{D}}{\partial h_{ij}}$ and $\frac{\partial \mathbf{V}^{T}}{\partial h_{ij}}$ can be obtained, where $\mathbf{U},\mathbf{D},\mathbf{V}$ are results of applying SVD to $\mathbf{H}_{12}^{c}$. Their covariance matrix $\sum_U, \sum_D, \sum_V$ can be computed via a forward error propagation method as follows:

$$\left.\begin{aligned} \sum_U &= \mathbf{J}_U \sum_H \mathbf{J}_U^T \\ \sum_D &= \mathbf{J}_D \sum_H \mathbf{J}_D^T \\ \sum_V &= \mathbf{J}_V \sum_H \mathbf{J}_V^T \end{aligned}\right\} \tag{14}$$

where $\mathbf{J}_U$, $\mathbf{J}_D$, $\mathbf{J}_V$ is the Jacobian of $\mathbf{U},\mathbf{D},\mathbf{V}$, respectively. From Eq. (14), the covariance matrix of $\mathbf{U},\mathbf{V},\mathbf{D}$ is $\sum_{\mathbf{U},\mathbf{V},\mathbf{D}}$ concatenated together as

$$\mathbf{diag}(\sum_{\mathbf{U},\mathbf{V},\mathbf{D}}) = \left[\sigma_{u_{11}}^2 \cdots \sigma_{u_{33}}^2 \; \sigma_{v_{11}}^2 \cdots \sigma_{v_{33}}^2 \; \sigma_{\lambda_1}^2 \; \sigma_{\lambda_2}^2 \; \sigma_{\lambda_3}^2\right] \tag{15}$$

In this case, we assume that each parameter in $\mathbf{U},\mathbf{V},\mathbf{D}$ matrices are independent, so as we propagate the value from Eq. (14), we force the other element to be 0.

3.2 Rotation Matrix and Translation Vector Covariance Matrix Estimation

For the rotation matrix, recall that, in this paper, we are only interested in yaw (ψ), which could be determined from the camera's frame rotation matrix ($\mathbf{R}_2$) which can be computed from

$$\mathbf{R}_2 = \mathbf{U}\mathbf{M}\mathbf{V} = \mathbf{U}\begin{bmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -s\beta & 0 & s\beta \end{bmatrix}\mathbf{V} \tag{15}$$

where

- $s = \det(\mathbf{U})\det(\mathbf{V})$
- $\alpha = \frac{\lambda_1 + s\lambda_3\delta^2}{\lambda_2(1+\delta^2)}$
- $\delta = \pm\sqrt{\frac{\lambda_1^2 - \lambda_2^2}{\lambda_2^2 - \lambda_3^2}}$
- $\beta = \pm\sqrt{1-\alpha^2}$

with criteria such that $sgn(\beta) = -sgn(\delta)$. And the yaw of $\mathbf{R}_2$ can be obtained as:

$$\psi = \arctan\left(\frac{r_{21}}{r_{11}}\right) \tag{16}$$

where the rotation matrix elements r_{11}, r_{21} can be derived from Eq. (17):

$$\left.\begin{aligned} r_{11} &= v_{11}(\alpha u_{11} - s\beta u_{13}) + v_{12}u_{12} + v_{13}(\beta u_{11} + s\alpha u_{13}) \\ r_{21} &= v_{21}(\alpha u_{11} - s\beta u_{13}) + v_{22}u_{12} + v_{23}(\beta u_{11} + s\alpha u_{13}) \end{aligned}\right\} \tag{17}$$

To achieve the target to determine yaw uncertainty, we apply forward propagation to Eq. (17) where its Jacobian can be retrieved by doing a partial derivative function of yaw with respect to the $\mathbf{U},\mathbf{V},\mathbf{D}$ parameters. Therefore,

$$\mathbf{J}_{\psi} = \frac{\partial \psi}{\partial \mathbf{U},\mathbf{V},\mathbf{D}} = \frac{1}{1+a^2}\left(\frac{\partial a}{\partial \mathbf{U},\mathbf{V},\mathbf{D}}\right) \tag{18}$$

where

- $a = \frac{r_{21}}{r_{11}}$

And finally, the covariance of yaw can be obtained from forward error propagation:

$$\sigma_{\Psi}^2 = \mathbf{J}_{\psi}\Sigma_{\mathbf{U,V,D}}\mathbf{J}_{\psi}^T \tag{19}$$

For translation covariance, from Eq. (1), the translation vector can be obtained by

$$\mathbf{t}_2 = \frac{1}{\omega}\left(-\beta\mathbf{u}_1 + \left(\frac{\lambda_3}{\lambda_2} - s\alpha\right)\mathbf{u}_3\right) \tag{20}$$

where

- $\mathbf{t}_2 = [x\, y\, z]^T$
- ω is a scale factor of normal vector

ω is a factor that scales the normal vector to $\|\mathbf{n}_1\| = 1$, so we have

$$\omega = \frac{1}{\sqrt{n_x^2 + n_y^2 + n_z^2}} \tag{21}$$

By substituting Eq. (21) into Eq. (20), we get a full camera translation vector:

$$\mathbf{t}_2 = \sqrt{n_x^2 + n_y^2 + n_z^2}\left(-\beta\mathbf{u}_1 + \left(\frac{\lambda_3}{\lambda_2} - s\alpha\right)\mathbf{u}_3\right) \tag{22}$$

The translation vector Jacobian matrix $\mathbf{J}_t$ can be derived from a partial derivative to all 21 parameters $\left(all\ element\ of\ \mathbf{U}, \text{diag}(\mathbf{D}), \mathbf{V}\right)$ that have dependency in Eq. (22). Finally, we apply forward propagation and we have

$$\Sigma_t = \mathbf{J}_t\Sigma_{\mathbf{U,V,D}}\mathbf{J}_t^T \tag{23}$$

4. Experimental Procedure and Results

In order to implement our visual odometry algorithm, the OpenCV library (in C++) for image processing is used. Now, the visual odometry component is tested on a prototype frame to which the selected camera (Logitech C920) is attached, as shown in Fig. 2. A Vectornav VN-100 (Fig. 3) internal measurement unit was chosen to measure the prototype frame orientation. The IMU specification is described in Table 1, in which we used fused data from a gyroscope, an accelerometer and a magnetometer. Leaves and rocks were selected as the objects in order to simulate a real underwater environment. They are a good for a real-time feature tracking with good tracking results. The features of leaves and rocks are displayed in Fig. 4.

The conclusion of the implementation procedure for the monocular visual odometry is described in Fig. 5.

In system integration, all of the software components are run on the robot operating system (ROS). ROS is a middleware or a framework for robot software development. Instead of programming every single module in one project or process, ROS provides tools and libraries to do inter-process communication using a publish-and-subscribe mechanism to the socket servers that handle all

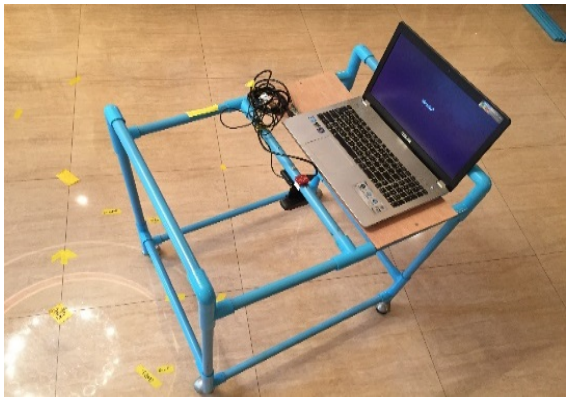

Fig. 2. Prototype frame for testing visual odometry.

Fig. 3. Vectornav VN-100 IMU.

Table 1. Vectornav VN-100 IMU Specification.

Specification	
Range: Heading, Roll	±180°
Range: Pitch	±90°
Static Accuracy (Heading, Magnetic)	2.0° RMS
Static Accuracy (Pitch/Roll)	0.5° RMS
Dynamic Accuracy (Heading, Magnetic)	2.0° RMS
Dynamic Accuracy (Pitch/Roll)	1.0° RMS
Angular Resolution	< 0.05°
Repeatability	< 0.2°

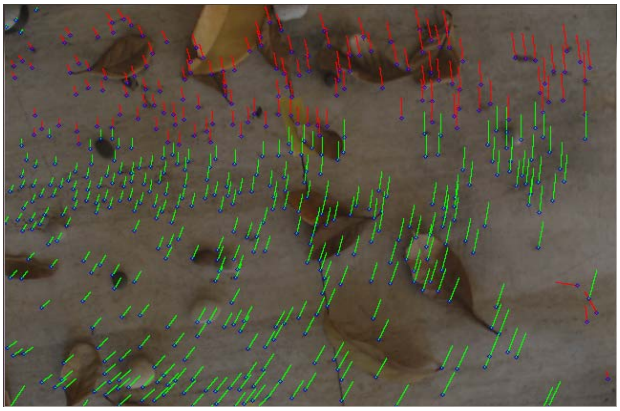

Fig. 4. Real-time feature tracking using Lucas-Kanade pyramidal optical flow on a prototype frame.

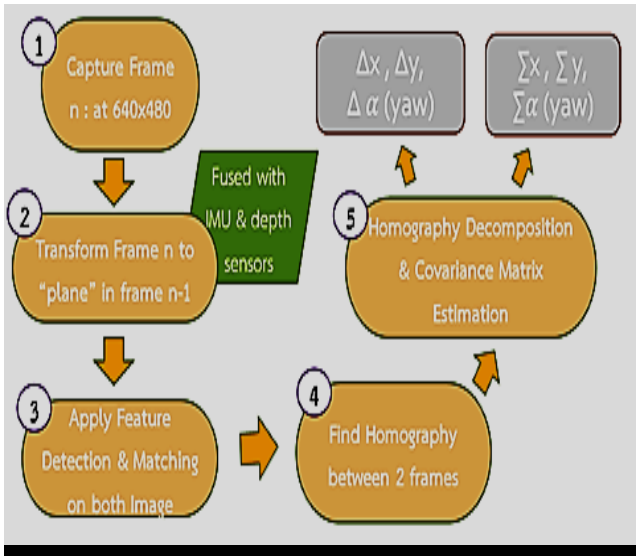

Fig. 5. Implementation procedure for visual odometry.

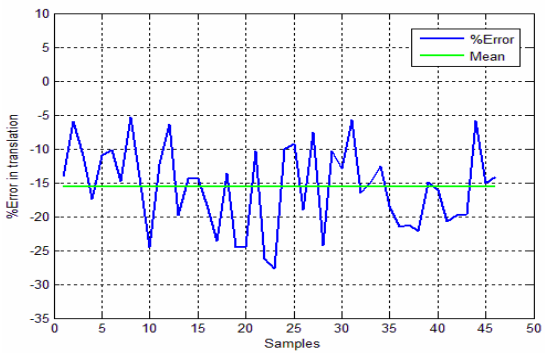

Fig. 6. Visual odometry translation error.

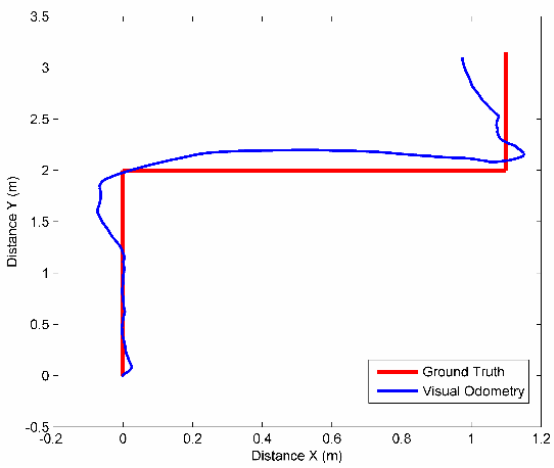

Fig. 7. Visual odometry experimental results (trajectory).

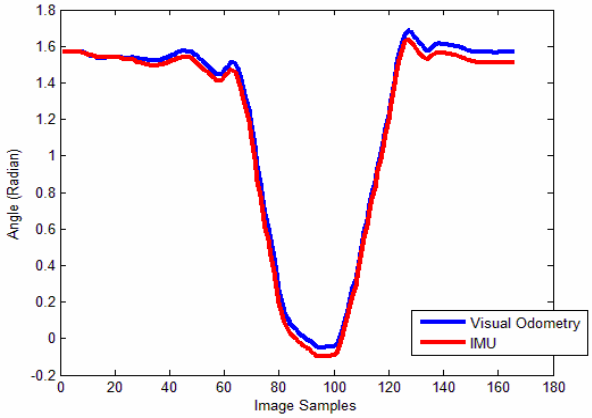

Fig. 8. Visual odometry experimental result (ψ) .

messages, parameters and services that occur in the systems, which support python and C++. Moreover, ROS also links many useful libraries for robotics programming, such as OpenCV and OpenNI, and it provides some hardware driver packages (e.g., Dynamixel Servo) and a visualization package to use with ROS messages. Furthermore, ROS handles sensors, images and data flow in the system, which makes system integration easier.

4.1 Error Evaluation

The system was tested over several iterations. In our experiment, bias of 15.6344% of translation was added to the system in order to compensate for the translation error. Percent error of translation, by using the LK optical flow algorithm, in the experiments is packed enough to compensate, as shown in Fig. 6.

4.2 Results

The visual odometry algorithm was subjected to experimentation. The prototype frame was driven along the ground in order to create translation motion as a fixed trajectory. In the experiment, the prototype frame was turned clockwise 90 degrees, and then sent straight for a while; after that, we turned it back by 90 degrees. The real experimental results compared with the ground truth are shown in Fig. 7.

The experimental results show that the estimated trajectory is close to the real translation trajectory. The second experiment was conducted in order to obtain the estimated yaw angle by using the proposed algorithm. The Vectornav VN-100 internal measurement unit (IMU) was used in the second experiment in order to obtain the yaw angle when the prototype frame is rotated. With the same trajectory as in Fig. 7, the results from the visual odometry estimation algorithm and from the real yaw angle from the IMU were compared and are shown in Fig. 8. The experimental results show that the estimated trajectory is also close to the real translation trajectory, even when the frame is rotated.

The covariance matrix estimation of the visual odometry algorithm was obtained from a real experiment in real time. As we calculated them frame by frame, each output parameter variance is shown compared with x, y, yaw output from visual odometry estimation in Fig. 9, Figs. 10 and 11, respectively. Note that we scale the value of yaw so that it can be seen in relation to the value transition and its covariance value.

In addition, we applied our monocular visual odometry algorithm to a video data log of a real underwater robot.

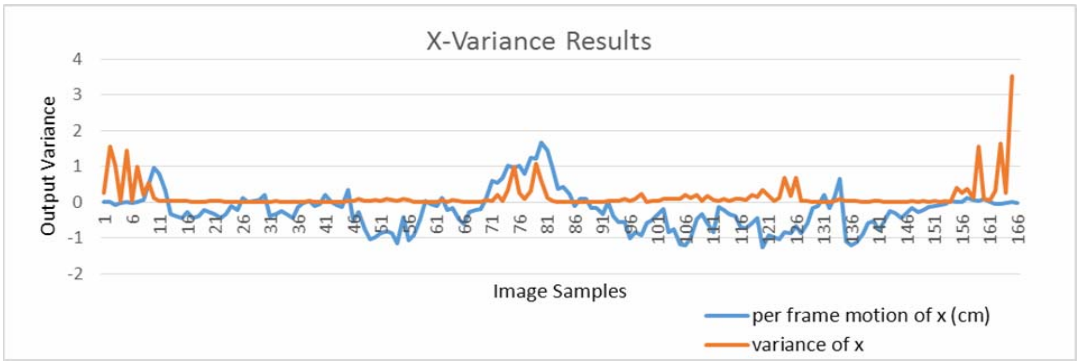

Fig. 9. X variance estimation from visual odometry results.

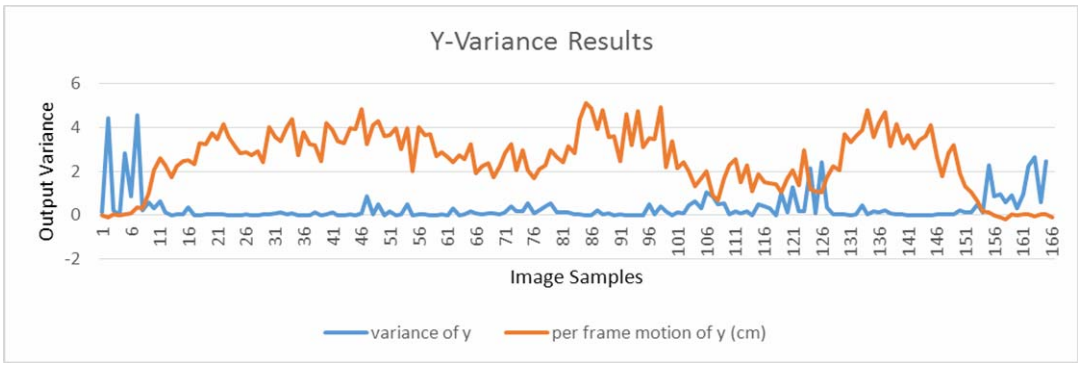

Fig. 10. Y variance estimation from visual odometry results.

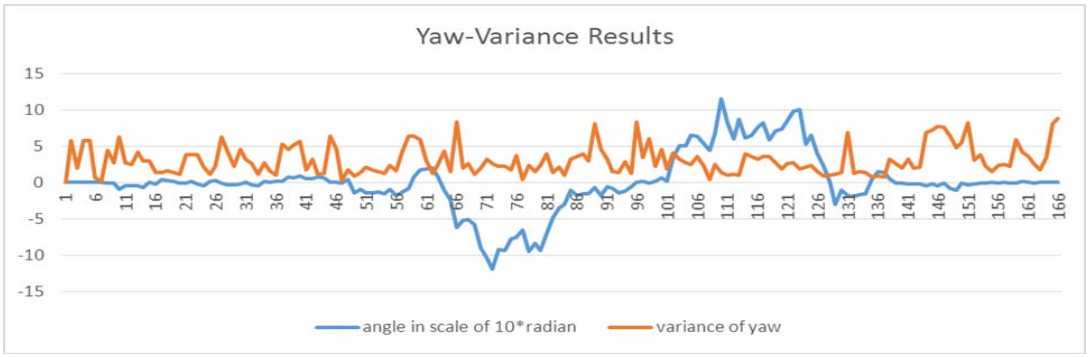

Fig. 11. Yaw variance estimation from visual odometry results.

The data log was collected from a Robosub 2014 competition organized by the Association for Unmanned Vehicle Systems International (AUVSI), which was an international competition. By using the ROS, IMU data and barometer data was also collected with time synchronization with the video data log so that we could apply our algorithm. Results are shown in Fig. 12, which demonstrates that our algorithm can be used in a hazardous underwater environment with good performance.

Despite unavailability of the real ground truth of the competition field, we could still estimate the robot displacement using a GPS and the competition field plan given by AUVSI. The displacement from the AUV deployment point to where it stopped is about 8.5 meters, which is shown in Fig. 12, such that our algorithm could estimate AUV trajectory.

All of the experiments were well tested in a prototype

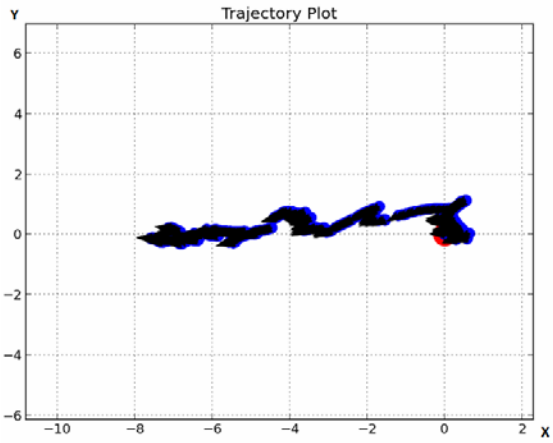

Fig. 12. Final trajectory result of the data log.

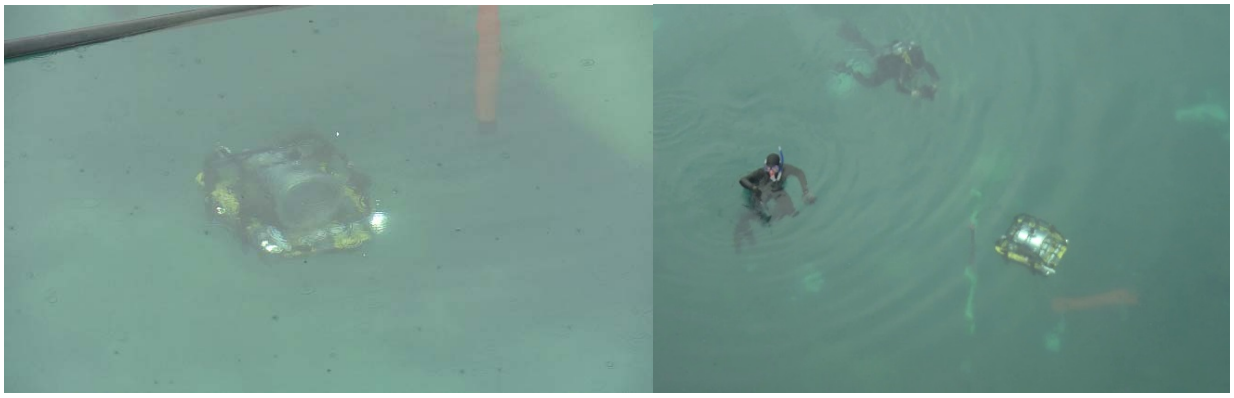

Fig. 13. Bird's eye view of testing the AUV in the Robosub 2014 competition.

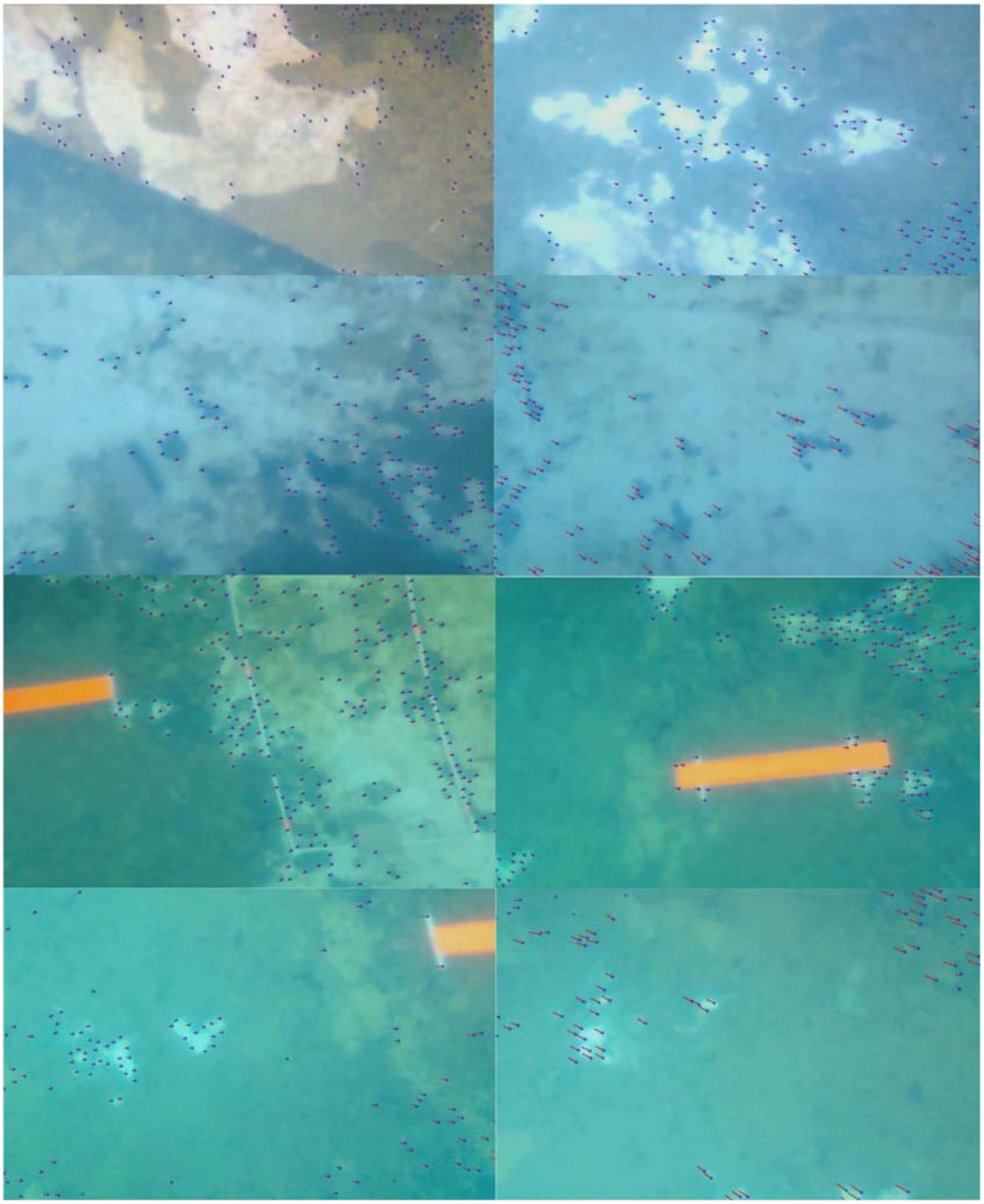

Fig. 14. Visual odometry from the Robosub 2014 competition data log.

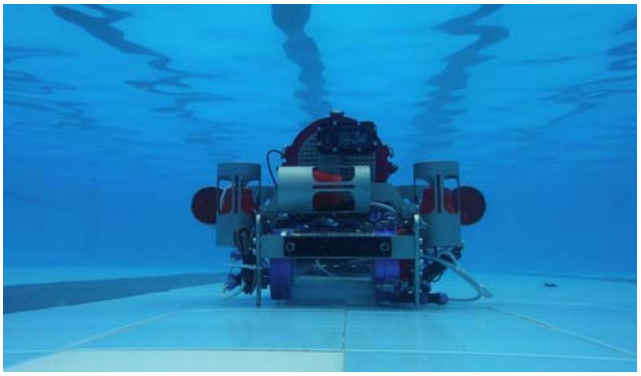

Fig. 15. Autonomous underwater robot.

frame. In practice, it will be implemented and tested "in real time" in our designed autonomous underwater robot, as shown in Fig. 15.

5. Conclusion

A monocular visual odometry estimation was implemented and tested in a prototype frame by looking downward to the ground plane and compensating every input image using pitch and roll from an IMU to guarantee that the input features were not distorted by the camera's direction. With Lucas-Kanade pyramidal optical flow, the tracked Shi-Tomasi method on the ground can be calculated between frame homography. After the homography is decomposed and the two sets of answers are obtained, the criteria for choosing them was explained. In order to control robot navigation, covariance of the visual odometry output, x y and yaw, is computed by using back and forward error propagation and Jacobian matrix and all mathematical derivatives are explained. The experimental visual odometry algorithm was tested, showed good results, and will later be implemented on a real underwater robot.

Acknowledgement

This research was supported by the Department of Electrical Engineering in the Faculty of Engineering, Kasetsart University, as part of courses 01205491 and 01205499, Electrical Engineering Project I & II. Finally, the authors want to express sincere gratitude to Mr. Somphop Limsoonthrakul, a doctoral student from the Asian Institute of Technology, Prathum-Thani, Thailand, who has always been a supporter of this research.

References

[1] Shi, Jianbo, and Carlo Tomasi. "Good features to track." Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE, 1994 Article (CrossRef Link)

[2] B. D. Lucas, T. Kanade. An iterative image registration technique with an application to stereo vision. IJCAI, 1981. Article (CrossRef Link)

[3] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision Second Edition". Cambridge University Press, March 2004. Article (CrossRef Link)

[4] Papadopoulo, T., Lourakis, M.I.A., "Estimating the jacobian of the singular value decomposition:theory and applications". In: Proceedings of the 2000 European Conference on Computer Vision,vol. 1, pp. 554–570 (2000) Article (CrossRef Link)

[5] F. Caballero, L. Merino, J. Ferruz, A. Ollero. "Vision-Based Odometry and SLAM for Medium and High Altitude Flying UAVs". J Intell Robot Syst, June 2008. Article (CrossRef Link)

[6] P. Drews, G.L. Oliveira and M. da Silva Figueiredo. Visual odometry and mapping for Underwater Autonomous Vehicles In Robotics Symposium (LARS), 2009 6th Latin American Article (CrossRef Link)

[7] D. Scaramuzzaand and F. Fraundorfer. Visual Odometry [Tutorial] In Robotics & Automation Magazine, IEEE (Volume:18, Issue: 4) Article (CrossRef Link)

[8] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. [Research Report] RR-6303, 2007, pp.90.

[9] Article (CrossRef Link)

Kandith Wongsuwan received his BSc in Electrical Engineering from Kasetsart University (KU), Bangkok, Thailand, in 2015. He is now a team leader of the SKUBA robot team, which was a five-year world champion in world robocup small-size robot competition. Furthermore, he is experienced in autonomous robot competition, as well, such as the @Home robot, which is an autonomous servant robot, and the autonomous underwater vehicle (AUV). His current research interests include computer vision, robotics, signal processing, image processing and embedded systems application.

Kanjanapan Sukvichai received a BSc with first class honours in electrical engineering from Kasetsart University, Thailand, an MSc in electrical and computer engineering from University of New Haven, CT, U.S.A., in 2007 and a DEng in Mechatronics from the Asian Institute of Technology, Thailand, in 2014. He worked as a Professor in the Department of Electrical Engineering, Faculty of Engineering, Kasetsart University, Thailand. He is the advisor to the SKUBA robot team. He served on the Executive Committee, Organizing Committee and Technical Committee of the Small Size Robot Soccer League, which is one division in the RoboCup organization, from 2009 to 2014. His current research interests include multi-agent autonomous mobile robot cooperation, underwater robots, robot AI, machine vision, machine learning, robot system design, robot system integration and control of an unstable robot system.