Motion Model Binary Switch for MonoSLAM

Akin Tatoglu, Kishore Pochiraju
Design and Manufacturing Institute, Mechanical Engineering,
Stevens Institute of Technology
Hoboken, NJ 07030 USA
atatoglu@stevens.edu

Abstract— **Current Monocular Simultaneous Localization and Mapping (MonoSLAM) methodologies use constant velocity and smooth motion assumptions. If the motion consists of rapid accelerations, decelerations or stops, the position estimates become erroneous and unstable. Mobile robots require frequent stops due to mission dictated or safety reasons. With the objective of using MonoSLAM to localize a mobile robot, we determined the effectiveness of trajectory estimation for a typical robot moving with constant velocity and stopping to execute missions. Experiments were performed with a camera mounted on a 3-axis translational robot and several path profiles with brief stops were executed. The trajectory estimated with a MonoSLAM algorithm is compared with the known motion profile. As the stop causes significant error and drift in the position estimates, we modified the constant velocity motion model to incorporate a stop detection method. An optical flow based stop detection model was formulated and implemented in conjunction with MonoSLAM. Velocity update is modified when a stop or start is detected by optical flow. By adaptively switching between constant velocity and stop models, the trajectory estimate is seen to be more accurate and stable after an intermittent stop. Details of the adaptive switching method and the performance of the modified MonoSLAM are described in this paper.**

Keywords- MonoSLAM, localization, motion model, affine optical flow

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) generates a map of the unknown surroundings and estimates the location of a robot by detecting and tracking landmarks using a set of sensors and stochastic estimation techniques. Sensors used typically are stereo camera pairs, LIDARs supported by additional sensors like odometer or IMU [1-4]. If the objective is to estimate the robot pose, a sparse landmark map may be sufficient sufficient [5]. SLAM process is a series of motion prediction, measurement and update steps and incorporates the controller input in the prediction step. For mobile robots with controllers determining the motion, the controller input will be utilized in the prediction step. However, if motion is generated without a formal controller (e.g. camera moved by a human hand), SLAM requires a motion model.

In monocular SLAM only a single camera is used as a sensor [6]. Advantage of a single camera is that it is flexible and cost effective. MonoSLAM framework [7] requires that the motion be smooth and the velocity remains constant through out the motion profile. Therefore, the accelerations and decelerations required to start and stop the robot contradict the constant velocity assumptions.

SLAM methods also track the pose of the robot with a six degrees of freedom (DOF) robot position state. The motion of a mobile robot is typically translational or rotational, therefore one or more of the degrees of freedom may become stationary with time (or come to a stop). Therefore, in a motion model each of the components of the linear and angular velocity may have to be treated independently. Adding noise to all DOF of the system simultaneously will increase uncertainty and reduce accuracy of trajectory estimate [6].

A depth hypothesis is required for establishing a 3D position from 2D single camera image. Random Sample Consensus (RANSAC) [8] is used to establish the depth estimates for each of the observed landmarks. A landmark is expected to be detected multiple times before it is used in depth determination. This is called delayed landmark registration. In [9], a Bayesian estimation step for motion model for more efficient trajectory estimation is used. Typically the motion models assign the same linear and angular velocity assumptions to all the active DOF.

In order to improve the motion model used in MonoSLAM, a local constant velocity rather than global constant velocity was used [10]. Methods utilizing vision systems to solve localization problem with including visual odometry [11], optical flow supported by Rao-Blackwellized Particle Filter (RBPF) SLAM with a stereo camera [12] and utilizing an IMU [13] can also be found in the literature. Optical flow with a Kalman filter has also been used [14].

Most of the earlier MonoSLAM frameworks use a motion model that adds same velocity increments to all the DOF [7-9]. However, while a mobile robot is moving, there might be a dominant DOFs and stationary DOFs. Identifying and tailoring the motion models by separating the stationary DOFs from non-stationary ones is expected to increase the localization accuracy.

There are two goals for this effort. The first goal is to characterize the performance of MonoSLAM with a constant velocity motion model for motion profiles with an intermittent stop. The second goal is to update the motion prediction methods by detecting a stationary DOF using image processing techniques.

II. MONOSLAM WITH SMOOTH MOTION

Constant velocity motion model for MonoSLAM assumes that observer is moving with a constant velocity with an additive linear and angular acceleration noise. Since motion is generated by an uncharacterized source, at each prediction step,

978-1-4577-1343-9/12/$26.00 ©2015 IEEE

it assumes the controller input based on the assumption that robot will move with the same velocity (both linear and angular). This velocity is the sum of estimated velocity at the previous step and motion noise defined by (8).

A. System State Definition and Predictions

Overall system state includes robots linear and angular pose, its velocity, positions of landmarks and state covariance matrices. The 6-DOF position state of the robot is given by

$$\boldsymbol{x}_v = \left[\boldsymbol{p}^{\mathrm{G}}\ \mathbf{q}^{\mathrm{GR}}\ \mathbf{v}^{\mathrm{G}}\ \mathbf{w}^{R}\right]^{\mathrm{T}} = [x\ y\ z\ q_0 q_1 q_2 q_3 \mathbf{v}^{\mathrm{G}} \mathbf{w}^{R}\ ]^T \quad (1)$$

where subscript {v} defines the current state and superscript {G} and {R} stand for global frame and robot frame. p^{G} is the position of the robot about global frame. q^{GR} defines the orientation of the robot with four quaternions. Robot's state also includes $\mathbf{v}^{\mathrm{G}}$ and $\mathbf{w}^{\mathrm{S}}$, linear and angular velocity vectors. In the first step zero initial values are assigned to these parameters, which indicates that robot frame is coinciding with global frame and the robot is idle. In the subsequent steps these values are assigned by the output of EKF update step. Landmark position vector is given by

$$\boldsymbol{y}_i = (x_i \quad y_i \quad z_i)^T \quad (2)$$

where $\boldsymbol{y}_i$ is a landmark with an index i. After a landmark is successfully tracked for a period of time, it is registered into the map and used for localization estimations. At first stage it is called semi-definite landmark and at the second stage it is called definite landmark [8]. By combining these two vectors in (1) and (2), system state is defined by

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_v \\ \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \end{bmatrix} \quad (3)$$

The state uncertainty is given by the covariance matrix:

$$\boldsymbol{P} = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \dots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \vdots\vdots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \ddots \\ \vdots & \vdots & \vdots & \end{bmatrix} \quad (4)$$

Applying a controller input processes prediction of state vector

$$\hat{\boldsymbol{x}}_{new} = \mathrm{f}(\hat{\mathrm{x}}, \mathrm{u}) \quad (5)$$

The covariance of the state uncertainty is

$$\boldsymbol{P}_{new} = \frac{\partial \boldsymbol{f}}{\partial x}\ \boldsymbol{P}\ \frac{\partial \boldsymbol{f}^T}{\partial x} + Q_V \quad (6)$$

where $\boldsymbol{P}_{new}$ is new covariance matrix after state is predicted. Calculation of Q_V will be defined in state uncertainty section.

B. State Predictions with Constant Velocity Model

Constant velocity motion model is applied during the state prediction. This model assumes that in every step, robot will move with the same velocity that was estimated on the previous step. Constant velocity motion model state prediction is as in

$$\mathrm{f}(\mathbf{x}_{\mathrm{v}}, \mathrm{u}) = \begin{bmatrix} \boldsymbol{p}^G_{new} \\ \boldsymbol{q}^{GR}_{new} \\ \boldsymbol{v}^G_{new} \\ \boldsymbol{w}^R_{new} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}^G + (\boldsymbol{v}^G + \boldsymbol{V}^G)\Delta t \\ \boldsymbol{q}^{GS} \times \boldsymbol{q}((w^R + \boldsymbol{\Omega}^R)\Delta t) \\ (\boldsymbol{v}^G + \boldsymbol{V}^G) \\ (w^R + \boldsymbol{\Omega}^R) \end{bmatrix} \quad (7)$$

Constant velocity motion model works sufficiently well when the robot's motion is continuous and smooth. However, deceleration occurring before the stop might cost the algorithm to completely lose the track. Even it doesn't lose track, two cases will occur. If stop occurs at a short period of time less than landmark buffering delay, algorithm will not even be aware of the stop action. If stop occurs for a longer period of time, the detection of stop in the estimated trajectory is observed with latency. Also, if there is only single dominant DOF, this knowledge should be applied to the process. These reasons require another model that can handle these situations. Similar situation will occur during random acceleration of the robot.

C. State Uncertainty for Constant Velocity Model

MonoSLAM adds acceleration as noise vector which is given by

$$\bar{n} = \ [\mathrm{V}^{\mathrm{G}}\ \Omega^{\mathrm{S}}\ ]^{\mathrm{T}} = \ ([l_A \Delta \mathrm{t} \quad \alpha_{\mathrm{A}} \Delta \mathrm{t}\ ])^{\mathrm{T}} \quad (8)$$

where V^{G} and Ω^{R} are the same inputs as used in (7). These values are initially assigned with linear and angular acceleration sigma squares multiplied by Δt. This noise vector is also used for prediction of state transition probability:

$$\mathrm{Q_v} \ = \ \frac{\partial \hat{\mathrm{x}}_{\mathrm{new}}}{\partial \mathrm{n}}\ \mathrm{P_n}\ \frac{\partial \hat{\mathrm{x}}_{\mathrm{new}}{}^{\mathrm{T}}}{\partial \mathrm{n}} \quad (9)$$

where $\mathrm{P_n}$ is the process noise.

D. State Predictions with Optical Flow

We would like process the optical flow determined information after EKF Update step and reflect the output during the prediction step. A controller input is defined to process these outputs.

$$\hat{\boldsymbol{u}}_{OF} = \begin{bmatrix} \hat{u}_{OF_L} \\ \hat{u}_{OF_W} \end{bmatrix} = \left[V_x^G V_y^G V_z^G \Omega_x^R \Omega_y^R \Omega_z^R\right]^T \quad (10)$$

which contains output of linear and angular optical flow values. Calculations of these values are represented in Section III. The state prediction with optical flow controller input is calculated as in

$$\mathrm{f}(\mathbf{x}_{\mathrm{v}}, \hat{u}_{OF}) = \begin{bmatrix} \boldsymbol{p}^G + \left(\boldsymbol{v}^G + \hat{u}_{OF_L}\right)\Delta t \\ \boldsymbol{q}^{GS} \times \boldsymbol{q}\left(\left(w^S + \hat{u}_{OF_W}\right)\Delta t\right) \\ \left(\boldsymbol{v}^G + \hat{u}_{OF_L}\right) \\ \left(w^S + \hat{u}_{OF_W}\right) \end{bmatrix} \quad (11)$$

where $\mathrm{f}(\mathbf{x}_{\mathrm{v}}, \hat{u}_{OF})$ uses the controller input assigned by optical flow. $\boldsymbol{v}^G$ and w^S are the linear and angular velocities calculated at EKF Update step.

III. Affine Optical Flow

The affine optical flow algorithm [15] correlates two consecutive images and calculates spatial and temporal gradients of changes. Then it applies Lucas-Kanade method [16] to solve the optical flow equation with sub-pixel accuracy. Algorithm first calculates estimated velocity of a pixel by:

$$\left[\mathrm{V_{OF\_x}}\ \mathrm{V_{OF\_y}}\right] = [\mathrm{x\ y\ 1}] * \begin{bmatrix} \mathrm{d} + \mathrm{s_1} & \mathrm{s_2} + \mathrm{w_r} \\ \mathrm{s_2} - \mathrm{w_r} & \mathrm{d} - \mathrm{s_1} \\ \mathrm{V_x^0} & \mathrm{V_y^0} \end{bmatrix} \quad (12)$$

where x and y are the position of a pixel on the image. "d" is dilation rate, "$\mathrm{s_1}$&$\mathrm{s_2}$" are shear rates, "$\mathrm{w_r}$" is the rotation rate and "$\mathrm{V_x^0}$" and "$\mathrm{V_y^0}$" are the optical flow of the corner of the image. $\mathrm{V_{OF\_x}}$ and $\mathrm{V_{OF\_y}}$ are the "optical flow velocity" in pixels/s on x and y direction. Also, dilation is the change about the z axis.

Determination of pixel velocity varies by the pixel location. The center pixel is chosen since the image has a radial distortion on the sides. However selecting the center point itself will miss a fixed point rotation about $\mathrm{Z^R}$ axis. In this case, $\mathrm{w_r}$ rotation rate must be calculated at another point on image. We have used center point for optical flow velocity calculation and $\mathrm{w_r}$ is calculated on left top corner.

Affine Optical Flow then assigns the controller input $\hat{u}_{OF}$ as shown below:

$$\hat{u}_{OF_L} = \begin{pmatrix} V_x^G = \begin{cases} l_{AX}\Delta\mathrm{t\ if\ V_{OF\_x}} > Th \\ -\boldsymbol{v}_x^G\ if\ \mathrm{V_{OF\_x}} \le Th \end{cases} \\ V_y^G = \begin{cases} l_{AY}\Delta\mathrm{t\ if\ V_{OF\_y}} > Th \\ -\boldsymbol{v}_y^G\ if\ \mathrm{V_{OF\_y}} \le Th \end{cases} \\ V_z^G = \begin{cases} l_{AZ}\Delta\mathrm{t\ if\ d} > Th \\ -\boldsymbol{v}_z^G\ if\ \mathrm{d} \le Th \end{cases} \end{pmatrix} \quad (13)$$

$$\hat{u}_{OF_W} = \begin{pmatrix} \Omega_x^R = \begin{cases} \alpha_{AX}\Delta\mathrm{t\ if\ s_1} > Th \\ -w_x^G\ if\ \mathrm{s_1} \le Th \end{cases} \\ \Omega_y^R = \begin{cases} \alpha_{AY}\Delta\mathrm{t\ if\ s_2} > Th \\ -w_y^G\ if\ \mathrm{s_2} \le Th \end{cases} \\ \Omega_z^R = \begin{cases} \alpha_{AZ}\Delta\mathrm{t\ if\ w_r} > Th \\ -w_z^G\ if\ \mathrm{w_r} \le Th \end{cases} \end{pmatrix} \quad (14)$$

where {Th} stands for threshold. If optical flow calculated $\mathrm{V_{OF}}$ is greater than the threshold, then V^G component is set to predefined linear acceleration noise value l_A multiplied by Δt. If the velocity detected is below a threshold, the controller input is modified to nullify the velocity component.

A. Algorithm for Stop Detection with Affine Optical Flow

The MonoSLAM and optical flow algorithms are executed concurrently. Affine optical flow determines displacement of the image in x and y direction as well as dilation rate, shear values and rotation rates. By using all these parameters it calculates $\hat{\boldsymbol{u}}_{OF}$ as given in (13) and (14). This controller input is used in the next prediction step.

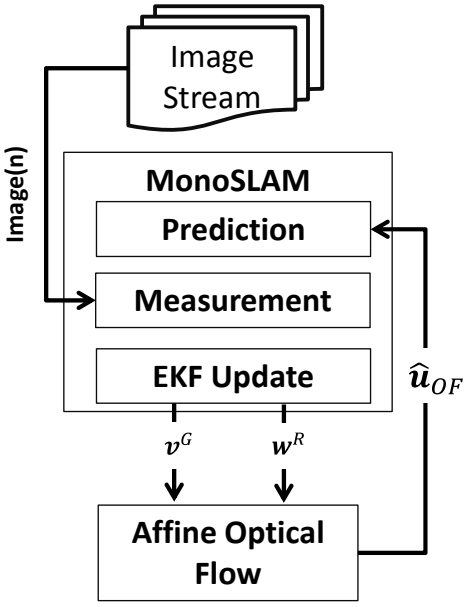

Figure 1. Optical Flow (OF) binary switch and MonoSLAM processes

Fig. 1 shows the data flow in the developed method. The affine optical flow modifies the controller input used in the prediction step.

IV. Experimental Setup

In order to generate the ground truth we assembled a camera on a 3DOF linear actuator robot. It runs on a worm gear driven by a stepper motor which results with highly accurate velocity control. A 30fps off-the-shelf camera with 320x240 pixels is used. The images are converted to grayscale. Purpose

Figure 2. Experimental Scene Sample for Affine Optical Flow and MonoSLAM

of the experiment is to analyze the behavior of MonoSLAM with constant velocity model. In order to reduce the landmark identification issues a texture rich environment is used. Fig. 2 shows a sample scene and Fig. 3 shows the system used for experiments. The camera is mounted to the interface on the machine.

Motions generated have three sections starting with STOP [4 s.], followed by constant velocity [10 s.] and then STOP for varying durations [2, 4, 8 s.]. Fig. 4 shows generic structure of the motion profiles.

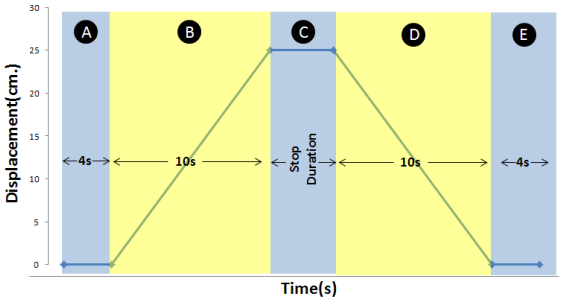

Figure 4. Experimental motion profiles with five periods

V. Experimental Results

A. Affine Optical Flow Output

As shown in Fig.1, affine optical flow calculates $\hat{\boldsymbol{u}}_{OF}$.These calculations are conducted by using V_{OF_x} , V_{OF_y} and dilation. Fig. 5 and 6 show linear and angular velocity calculations by affine optical flow in pixel velocities (px/s) for experiment with 8 second stop. Fig. 5 clearly shows the durations of stops, constant velocity and acceleration/decelerations for three different DOFs.

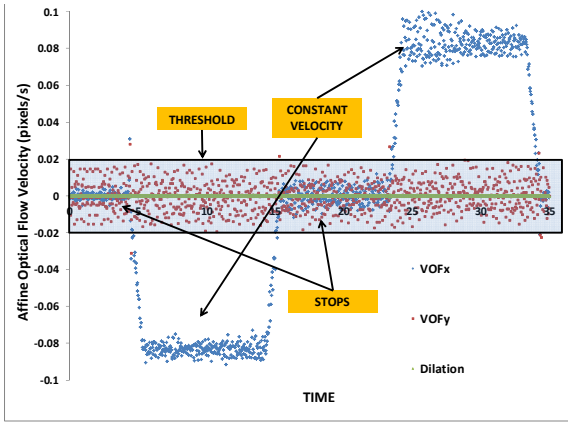

Figure 5. Affine Optical Flow Output Translational Output

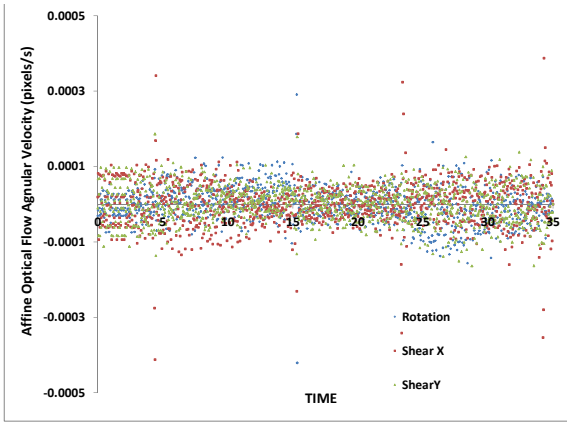

Figure 6. Affine Optical Flow Rotational Output

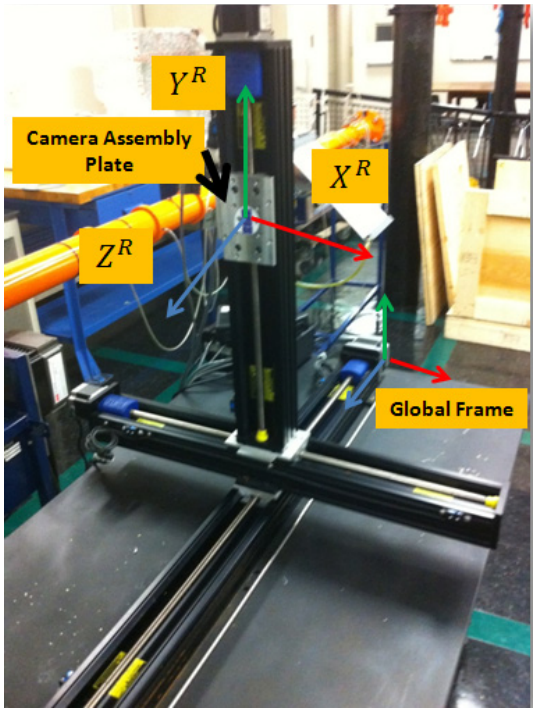

Figure 3. 3-DOF translational robot to apply several path profiles

B. Constant Velocity Model behavior from Period-A to B

MonoSLAM requires warming time for stabilization due to delayed landmark initialization. In all experiments, there is initial stop duration for 4 seconds. Fig.7 shows the behavior of MonoSLAM with constant velocity motion model while motion is switching between periods A to B. {GT} and {CVM} stand for ground truth and constant velocity motion(single model system) respectively. The 2, 4 and 8 defines duration of period-C in Fig.4 for each experiment processed with single model. During the warming time, it is expected to see a detection of stop with zero velocity. Even the velocity is zero; there is a fixed shift between ground truth and estimated trajectory. After acceleration occurs and motion starts, even output could catch up with the motion, estimated velocity (slope) is slower than ground truth.

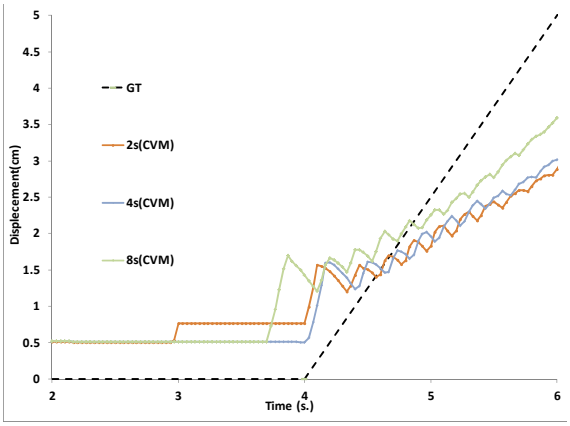

Figure 7. MonoSLAM contant velocity motion model behavior while switching between Period A and B.

C. *Constant Velocity Model behavior from Period-B to C*

After constant velocity occurs for 10 seconds, robot stops for three different time durations. Fig. 8 shows the behavior of estimated trajectory while switching between periods B and C. It is observed that MonoSLAM requires some time to detect stop motion. However, even after stop is detected, it starts climbing again.

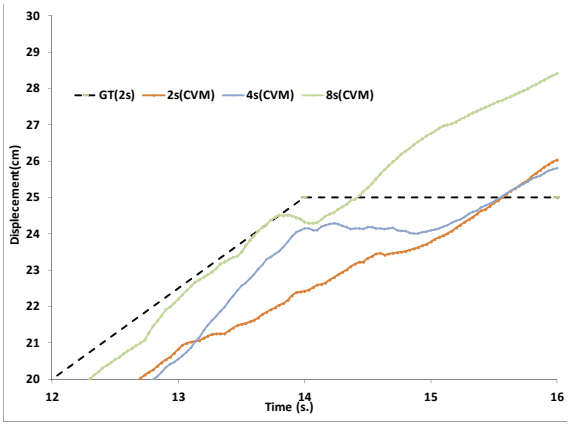

Figure 8. MonoSLAM contant velocity motion model behavior while switching between Period B and C.

D. *Optical Flow Model behavior from Period-A to B*

Fig. 9 shows the behavior of estimated trajectory with optical flow motion model. The 2, 4 and 8 defines duration of period-C in Fig.4 for each experiment processed with model switch. Optical flow identifies both the stop and start of the motion.

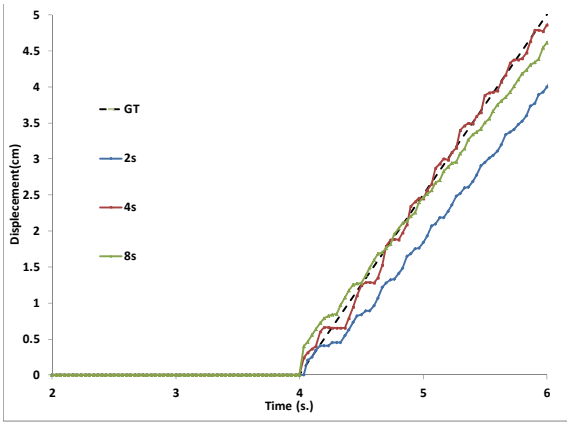

Figure 9. MonoSLAM with optical flow generated controller input behavior while switching between Period A and B.

E. *Optical Flow Model behavior from Period-B to C*

Fig. 10 shows the behavior while motion is switching from period B to C. Velocity at period B seems to be constant. Figure shows that optical flow detects the stop accurately and modifies the controller input correctly.

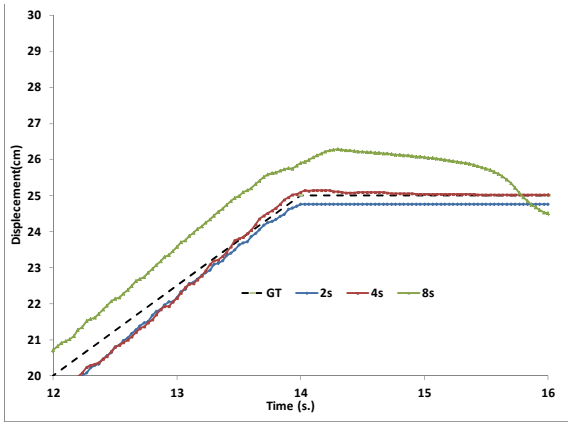

Figure 10. MonoSLAM with optical flow generated controller input behavior while switching between Period B and C.

F. *Full Trajectory*

Fig. 11 shows the overall comparison of 8 seconds stop experiment for all five periods. MonoSLAM with constant velocity motion model is seen to be inaccurate after the stop. On the other hand, after an overshoot, optical flow aided motion model is setting its speed to zero. Also, after motion is restarted by changing the direction, optical flow method is seen to produce appropriate controller inputs to accurately estimate the trajectory.

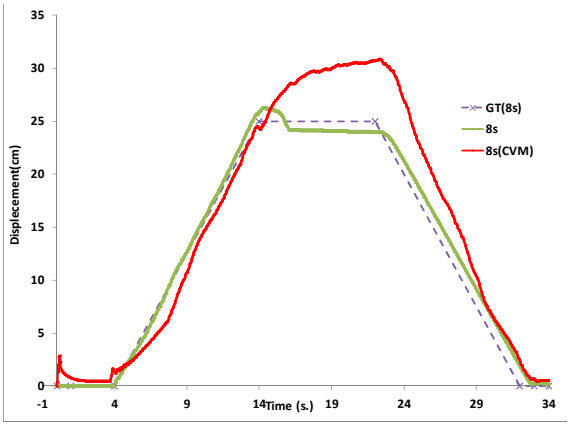

Figure 11. Overall displacement plot for five periods. GT: Ground Truth, 8s(CVM): Single model with 8 s. period-C duration, 8s: Swithing model with 8 s. period-C duration.

VI. CONCLUSION

Detecting a stop is significant for accurately localizing mobile robots. After a stop, MonoSLAM loses the track. Also, if we assume a robot is mostly moving forward and no motion is occurring in an individual axis, removing the acceleration noise improves the estimation accuracy.

MonoSLAM with constant velocity motion model drifts during stops due to the motion model assumptions. In SLAM, this is typically corrected by encoder or accelerometer inputs. We investigated the use of optical flow that can detect and manage stop signs during the motion of the robot without additional sensor requirements. In addition to that detecting single axis idle behavior increases the accuracy. Use of total image processing techniques such as the optical flow to adaptively change the motion model is seen as an effective way to estimate robot trajectory with MonoSLAM.

REFERENCES

[1] Zhou X. S., and Roumeliotis S. I., "Determining the robot-to-robot 3D relative pose using combinations of range and bearing measurements: 14 minimal problems and closed-form solutions to three of them," Intelligent Robots and Systems, pp. 2983–2990, 2010.

[2] Paz, L. M., Piniés, P., Tardós, J. D., Neira, J., "Large-scale 6-DOF SLAM with stereo-in-hand," Transactions on Robotics, vol. 24, no. 5, pp. 946–957, 2008.

[3] D. Craciun, N. Paparoditis, F. Schmitt, "Multi-view scans alignment for 3D spherical mosaicing in large-scale unstructured environments", Computer Vision and Image Understanding, vol. 114, no. 11, pp. 1248–1263, November 2010.

[4] Mourikis A. I., Roumeliotis S. I., "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," International Conference on Robotics and Automation, pp. 3565–3572, 2007.

[5] Klein, G., and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces," 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225 –234, 2007.

[6] Davison A. J., Reid I. D., Molton N. D., Stasse O., "MonoSLAM: Real-time single camera SLAM," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052–1067, June 2007.

[7] Davison A., "Real-time simultaneous localisation and mapping with a single camera," International Conference on Computer Vision, pp. 1403-1410, 2003.

[8] Civera J., Grasa O., "1 Point RANSAC for extended Kalman filtering: Application to real time structure from motion and visual odometry," Journal of Field Robotics, vol. 27, no. 5, pp. 609-631, 2010.

[9] J. Civera, A. J. Davison, and J. M. M. Montiel, "Interacting multiple model monocular SLAM," 2008 IEEE Int. Conf. Robot. Autom., pp. 3704–3709, May 2008.

[10] Hesch J. A., Roumeliotis S. I., "Consistency analysis and improvement for single-camera localization," Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 15-22, 2012.

[11] D. Nistér, O. Naroditsky, J. Bergen, "Visual odometry," in Computer Vision and Pattern Recognition, 2004.

[12] P. Elinas, R. Sim, and J. J. Little, "σSLAM: stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution," Proc. 2006 IEEE Int. Conf. Robot. Autom. 2006. ICRA 2006., no. May, pp. 1564–1570, 2006.

[13] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles," Rob. Auton. Syst., vol. 57, no. 6–7, pp. 591–602, Jun. 2009.

[14] X. Song, L. D. Seneviratne, and K. Althoefer, "A Kalman Filter-Integrated Optical Flow Method for Velocity Sensing of Mobile Robots," IEEE/ASME Trans. Mechatronics, vol. 16, no. 3, pp. 551–563, Jun. 2011.

[15] Young D. S., "First-order optic flow and the control of action," European Conference on Visual Perception, 2000.

[16] Lucas B., Kanade T., "An iterative image registration technique with an application to stereo vision," International Joint Conferences on Artificial Intelligence, vol. 81, pp. 674-679, 1981.