# An Incremental Robust Underwater Navigation with Expectation-Maximisation

**Sajad Hassan, Jonghyuk Kim and Shoudong Huang**
Robotics Institute
University of Technology Sydney
{sajad.hassan@student.uts.edu.au} {jonghyuk.kim, shoudong.huang}@uts.edu.au

## Abstract

This paper presents a robust navigation solution using low-cost visual-inertial sensors in a 6-Degree of Freedom (DoF) environment. That is an incremental/online navigation solution using the nonlinear least-squares optimisation with classification expectation-maximisation (EM). In this problem, weights are assigned to each measurement observation using the Cauchy function that are iteratively computed from the errors between predicted robot poses and the observed robot measurement. However, the computational cost is quite high in solving the full-batch estimation via Gauss-Newton. By implementing the sliding window filter (SWF), we introduce an incremental EM based robust navigation where the computational cost is shown a significant reduction compared to the full robust batch estimation. The impact of window size on the navigation performance is studied given the dataset is unknown to predict the optimum window gating. This allows a robust constant-time estimation of the robot pose. Such a capability is desirable in underwater navigation applications such as intervention missions. We verify this work using the experimental dataset collected by the UTS submersible pile inspection robot (SPIR).

## 1 Introduction

The focus of many robust solutions for the navigation tracking problem have been on improving the robustness of the Kalman filter or extended Kalman filter (EKF) [Agamennoni *et al.*, 2011][Kautz and Eskofier, 2015] [Sjanic *et al.*, 2011]. Often the proposed solutions are fairly involved with computational burden and require tuning the process and measurement noise covariance matrices. Many do not consider optimisation techniques such as the full least-squares for the navigation problem.

In the 1960's, the nonlinear least-squares was in active use at NASA JPL. Optimisation has several advantages to filtering methods such as EKF. The EKF is suboptimal compared to full batch estimation [Sibley *et al.*, 2008]. [Sjanic *et al.*, 2011] present a detailed optimisation based solution to the simultaneous localisation and mapping (SLAM) problem using the nonlinear least-squares. One main benefit of this optimisation technique to filtering is the smoothing. However, outlier measurements influence the smoothing estimates and degrades the navigation accuracy.

Previously in [Hassan *et al.*, 2021], we introduced an iterative smoothing and outlier detection approach using EKF. The Biswas-Mahalanobis fixed lag smoother (BMFLS) was applied to detect outliers and smooth out the navigation trajectory at constant-time. But, this approach only works well for moderate set of outliers.

A work by [Lee *et al.*, 2013] has illustrated a promising result by proposing a robust pose-graph SLAM problem based on the classification EM algorithm that can detect wrong loop closure constraint. From the navigation's perspective this is a full-batch estimation problem. However, solving full-batch estimation via Gauss-Newton runs offline and is not a constant time algorithm [Barfoot, 2017]. [Sibley *et al.*, 2008] introduces sliding-window filter by iterating over a window of timesteps and sliding this window forward for online/constant-time implementation. We apply the concept in [Lee *et al.*, 2013] to the navigation problem and extend it to introduce an incremental robust estimation with EM. In this work, for each measurement observation a weight is assigned using the Cauchy function, which are iteratively computed from the errors between the predicted robot poses and observed measurements. As a result, outlier observation measurements are assigned low weights and removed.

Similarly [Cheng *et al.*, 2015] extended the concept from [Lee *et al.*, 2013] and proposed a robust efficient solution to the Linear SLAM problem by introducing a delayed optimisation approach. Both approaches in [Sibley *et al.*, 2008] and [Cheng *et al.*, 2015] apply marginalisation using Schur complement to maintain the prior information. Other approaches such as variable state dimension filter (VSDF) ignores conditional dependencies when marginalising out old parameters. Neglecting the conditional dependencies my lead to divergence [Sibley *et al.*, 2010]

In summary, low-cost IMU/vision based navigation in a difficult environment is a challenging task. The main contributions of this paper are as follows:

- By combining various techniques, an incremental robust navigation strategy is introduced to provide constant-time estimation after outlier rejection.
- Investigating the effect of various window sizes on the navigation performance for constant-time applications.

The remainder of this paper is organised as follows: Section 2 covers the problem formulation, dynamics and measurement models; Section 3 manifests the proposed solution methodology; Section 4 outlines the SWF approach; Section 5 are the experimental data collected from the UTS water tank facility and the solution results; and Section 6 will conclude with future direction.

## 2 Problem Formulation

We assume that the nonlinear motion and observations model are in the following form:

$$x_k = f(x_{k-1}, u_k) + g(x_{k-1}, w_k) \tag{1a}$$

$$z_k = h(x_k) + v_k, \tag{1b}$$

where $x_k$ and $u_k$ are the vehicle state and control input at time $k$ respectively, $w_k$ is the process noise, and $z_k$ is the observation vector with $v_k$ being the observation noise.

The Jacobian matrices (note $k$-subscript denotes step $k$) for the nonlinear model $f(\cdot)$ and $g(\cdot)$ are as follows:

$$F_{k-1} \approx \left.\frac{\partial f(x,u)}{\partial x}\right|_{(x,u)=(\hat{x}_{k-1},u_k)} \tag{2a}$$

$$G_{k-1} \approx \left.\frac{\partial g(x,w)}{\partial x}\right|_{x=\hat{x}_{k-1}} \tag{2b}$$

The motion model can be written in the form:

$$x_k = f(x_{k-1}, u_k) + \underbrace{G_{k-1} w_k}_{\tilde{w}_k}, \qquad w_k \sim N(0, Q_k) \tag{3}$$

where

$$E\langle \tilde{w}_k \tilde{w}_k^T \rangle = \underbrace{G_{k-1} Q_k G_{k-1}^T}_{\tilde{Q}_k} \tag{4}$$

### 2.1 Coordinate Frames

The definition of coordinates frames used to derive the dynamic model are:

- Body frame ($b$), moving with sensor where its origin is fixed in the IMU's centre of mass.
- Camera frame ($c$), moving with its origin fixed in the camera's optical centre.
- Navigation frame ($n$), fixed point in space with origin arbitrarily located.

### 2.2 Dynamics

The vehicle state prediction model in (1a), $x = \begin{bmatrix} p^n & v^n & \Psi^n \end{bmatrix}^T$contains 9 states expressed in the navigation frame, where

- vehicle position $p^n = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$,
- vehicle velocity $v^n = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$,
- vehicle attitude $\Psi^n = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ (Euler angles in roll, pitch and yaw).

The IMU data are treated as the control inputs $u = \begin{bmatrix} f^b & \omega^b \end{bmatrix}^T$ which include the vehicle body acceleration $f^b = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$ in $m/s^2$ and angular velocity $\omega^b = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ in $rad/s$.

Then, the state prediction model are as follows:

$$\underbrace{\begin{bmatrix} p_k^n \\ v_k^n \\ \Psi_k^n \end{bmatrix}}_{x_k} = \underbrace{\begin{bmatrix} p_{k-1}^n + v_{k-1}^n \Delta t \\ v_{k-1}^n + (C_b^n f_k^b + g)\Delta t \\ \Psi_{k-1}^n + (E_b^n \omega_k^b)\Delta t \end{bmatrix}}_{f(x_{k-1},u_k)} + \underbrace{\begin{bmatrix} 0_{3,3} & 0_{3,3} \\ C_b^n & 0_{3,3} \\ 0_{3,3} & E_b^n \end{bmatrix}}_{G_{k-1}} \Delta t \underbrace{\begin{bmatrix} w_{a,k}^b \\ w_{\omega,k}^b \end{bmatrix}}_{w_k} \tag{5}$$

where

$$w_{a,k}^b \sim N(0, Q_a), \qquad Q_a = \sigma_a \mathbf{I}_3 \tag{6a}$$

$$w_{\omega,k}^b \sim N(0, Q_\omega), \qquad Q_\omega = \sigma_\omega \mathbf{I}_3 \tag{6b}$$

$$C_b^n = \begin{bmatrix} c_\theta c_\psi & c_\phi s_\psi + s_\phi s_\theta c_\psi & s_\phi s_\psi - c_\phi s_\theta c_\psi \\ -c_\theta s_\psi & c_\phi c_\psi - s_\phi s_\theta s_\psi & s_\phi c_\psi + c_\phi s_\theta s_\psi \\ s_\theta & -s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{6c}$$

$$E_b^n = \begin{bmatrix} 1 & s_\phi t_\theta & c_\psi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{6d}$$

where $s_{(\cdot)}$, $c_{(\cdot)}$ and $t_{(\cdot)}$ are shorthand notations for $sin(\cdot)$, $cos(\cdot)$ and $tan(\cdot)$ respectively, $E_b^n$ is the body rotation rate to Euler rate transformation matrix, $C_b^n$ is the body to navigation frame rotation matrix and $g = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ accounts for the earth's gravitational field.

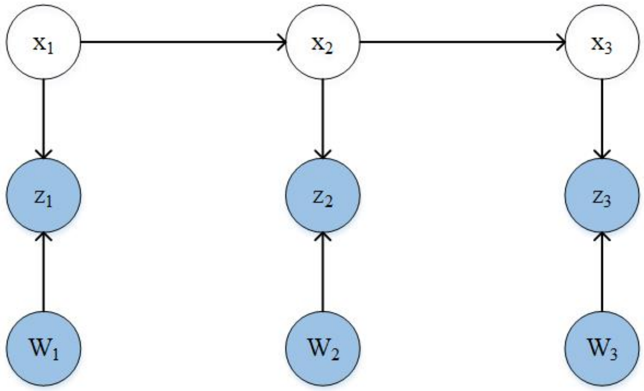

Figure 1: Bayesian network representing the navigation problem. $W_k$ is the weight for the measurement $z_k$ in the optimisation

### 2.3 Environment Observation Setup

The robot operates in a 6-DoF environment. Visual ARtag fiducial markers are installed on floor of the water tank to provide the position and orientation measurements. As a result, the environment observation setup provides the direct pose measurements. Hence, the observation model in (1b) is linear, that is:

$$z_k = Hx_k + v_k \tag{7}$$

where

$$v_k \sim N(0, R_k), \qquad R_k = \sigma_{meas}\mathbf{I}_6 \tag{8}$$

$$H = \begin{bmatrix} I_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 \end{bmatrix} \tag{9}$$

## 3 Solution

The proposed robust navigation method is illustrated by the Bayesian network shown in Figure 1. As shown, $X = [x_1, x_2, x_3, ..., x_n]$ are the predicted robot poses using IMU input and $Z = [z_1, z_2, z_3, ..., z_n]$ are the observations. The variables $W = [W_1, W_2, W_3, ..., W_n]$ $W \in [0, 1]$ are the weights assigned to each observation. From the expectation maximisation formulation in [Lee *et al.*, 2013], the weight $W_k$ is found to be the Cauchy function:

$$W_k = \frac{C^2}{C^2 + ||z_k - f(x_k)||^2_{R_k}} \tag{10}$$

where $C$ is a constant and $||z_k - f(x_k)||^2_{R_k}$ is the Mahalanobis distance. It can be observed the Cauchy weight function varies between 1 and 0 with increasing error (Mahalanobis distance dominates).

### 3.1 Initialisation

The nonlinear least-squares approach requires an initial estimates of the states which are obtained using the EKF algorithm. Every time a measurement is available, the position and orientation states are corrected.

### 3.2 Nonlinear Least-Squares Smoothing

The nonlinear least-squares problem is linearised during each iteration and then solved using the Gauss-Newton approach. The nonlinear full least-squares smoothing requires minimising the prediction and measurement error. This can be represented as:

$$[\delta x_k^*] = \arg\min_{\delta x_k} \sum_{k=0}^{K} (L_{v,k}(x) + L_{z,k}(x)) \tag{11}$$

where $L_{v,k}(x)$ and $L_{z,k}(x)$ are the weighted squared errors based on the process and observation models.

$$L_{v,k}(x) = \frac{1}{2} e_{v,k}(x)^T \tilde{Q}_{v,k}^{-1} e_{v,k}(x) \tag{12a}$$

$$L_{z,k}(x) = \frac{1}{2} W_k e_{z,k}(x)^T R_{z,k}^{-1} e_{z,k}(x) \tag{12b}$$

The motion model prediction error is:

$$e_{v,k} = \begin{cases} \hat{x}_0 - x_0, \ k = 0 \\ f(x_{k-1}, u_k) - x_k, \ k = 1, ...K \end{cases} \tag{13a}$$

and the observation model measurement error is:

$$e_{z,k} = z_k - h(x_k). \tag{13b}$$

The linearised motion model and prediction model error are:

$$e_{v,k}(x_{op} + \delta x) \approx \begin{cases} e_{v,0}(x_{op}) + \delta x_0, \ k = 0 \\ e_{v,k}(x_{op}) + F_{k-1}\delta x_{k-1} - \delta x_k, \ k = 1, ...K \end{cases} \tag{14a}$$

$$e_{z,k}(x_{op} + \delta x) \approx e_{z,k}(x_{op}) - H_k \delta x_k, \tag{14b}$$

where $x_{op}$ denotes the operating point. We solve for $\delta x$ at each iteration until the algorithm converges.

$$\delta x = \begin{bmatrix} \delta x_0 \\ \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_K \end{bmatrix} \tag{15}$$

The combined prediction and observation error is $e(x_{op})$.

$$e(x_{op}) = \left[\begin{array}{c} \delta_{v,0}(x_{op}) \\ \delta_{v,1}(x_{op}) \\ \vdots \\ e_{v,K}(x_{op}) \\ \hline e_{z,0}(x_{op}) \\ e_{z,1}(x_{op}) \\ \vdots \\ e_{y,K}(x_{op}) \end{array}\right] \tag{16}$$

The stacked version of the problem can be written and solved iteratively. The structure of the combined motion and observation model Jacobians $J$ matrix can be represented as:

$$J = \left[\begin{array}{cccccc} I & & & & & \\ -F_0 & I & & & & \\ & -F_1 & \ddots & & & \\ & & \ddots & & & \\ & & & I & & \\ & & & -F_{K-1} & I & \\ \hline H_0 & & & & & \\ & H_1 & & & & \\ & & H_2 & & & \\ & & & \ddots & & \\ & & & & H_K & \end{array}\right] = \left[\begin{array}{c} J_v \\ \hline J_z \end{array}\right] \tag{17}$$

$$\Lambda = diag(P_0^{-1}, \tilde{Q}_1^{-1}, \cdots \tilde{Q}_K^{-1}, W_0 R_0^{-1}, \cdots W_K R_K^{-1}) \tag{18}$$

$$(J^T \Lambda J)\delta x^* = J^T \Lambda e(x_{op}) \tag{19}$$

Notice the covariance matrix of the process noise $\tilde{Q} = G_{k-1} Q_k G_{k-1}^T$, is singular and not invertible. We can add a small diagonal matrix $\Delta I$, to allow the covariance matrix become invertible.

# 4 Sliding Window Filter

For the non-linear full least-squares optimisation to operate efficiently, the size of parameter vector cannot grow without bound. One simple approach to reduce the computational complexity is by removing old poses from the state vector. However by removing parameters directly from the system equation could result in information loss of how parameters interact. The correct way to remove parameters from a multidimensional distribution is to marginalise them out [Sibley *et al.*, 2008]. This section highlights the marginalisation using Schur complement and the steps taken in the SWF algorithm.

## 4.1 Marginalisation

Marginalising out parameters is equivalent to applying the Schur complement to the linear equation. For example, given the system:

$$\underbrace{\begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \delta x_1^* \\ \delta x_2^* \end{bmatrix}}_{\delta x^*} = \underbrace{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}_{b} \tag{20}$$

Reducing the parameter $x_1$ (marginalised state) into $x_2$ gives:

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} - A_{12}^T A_{11}^{-1} A_{12} \end{bmatrix} \begin{bmatrix} \delta x_1^* \\ \delta x_2^* \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 - A_{12}^T A_{11}^{-1} b_1 \end{bmatrix} \tag{21}$$

where the matrix $A$ represents the $(J^T \Lambda J)$ and $b$ represents $(J^T \Lambda e(x_{op}))$ from equation (19).

## 4.2 SWF Algorithm

Here, we provide a summary of the SWF algorithm.

- Add the new pose parameters: after completing $k-1$ steps, we apply the process model using $x_k = f(x_{k-1}, u_k)$ and also update the information matrix.
- Remove parameters: if there are more than $k$ poses, we marginalise out old poses using the Schur complement.
- Update parameters: we now solve using Gauss Newton and apply the outlier detection method.

Algorithm 1 outlines the pseudo code for the incremental robust navigation. Lines 12-34 repeats the EM process for the current window where lines 18-24 is the expectation step where $W_k$ is calculated based on the current pose $x_k$. Lines 26-33 is maximisation step where the pose update $x_k$ is computed from the maximisation step. The EM process is repeated until the weights converges in line 7. Lines 36-41 goes through entire measurement in the current window and recalculates the weight $W_k$ based on the current smoothed trajectory. The removal of measurement with low weight and the repeated EM process help in preventing local minima. Further, the current window is then slided forward (line 6) with oldest pose discarded.

# 5 Experiment

The robot collects the measurement data using a monocular camera. The *ArUco* ROS library is used to compute the pose measurements from the markers. The robot was manually controlled to maintain a hovering position under a current disturbance source. Due to the lower visibility and low illumination, the markers were frequently

**Algorithm 1** Incremental Robust Navigation Pseudo Code

```
Window size $ws$
Input $x_{k(-)}$ : states from previous state, observation $z$, all measurements I
Output $x_s$ : smoothed trajectory
N = # IMU measurement
$W' = 0$; $W = I$; $I' = I$;
for $i = 0 : N$ do
     //Weighting convergence
    while $|W - W'| > \nu$ do
        $W' = W$
        // Classification EM iterations
        $k = 0$; $X^k = X$
        while $|\delta| > \eta$ do
     //SWF
            for $k = i : i + ws$ do
                predict state: $x_{k(-)} = f(x_{k-1})$
                calculate $J_v$ and
                $\Lambda_Q$ = blkdiag($\Lambda_Q, Q^{-1}$)
                calculate $e_{v,k}(x_{op})$ = f($x_{k-1}$) - $\hat{x}_k$
                set $e_v(x_{op}) = [e_v(x_{op}),\ e_{v,k}(x_{op})]$
     //Expectation step
                if measurement $k$ in $\in I'$ then
                    compute $W_k$ with equation (10)
                    calculate $J_z$ and
                    $\Lambda_R$ = blkdiag($\Lambda_R, W_k R^{-1}$)
                    calculate $e_{z,k}(x_{op}) = z_k$ - $H\hat{x}_k$
                    set $e_z(x_{op}) = [e_z(x_{op}),\ e_{z,k}(x_{op})]$
                end if
            end for
     //Maximisation step
            Assemble
            J = $[J_v;\ J_z]$
            $\Lambda = [\Lambda_Q;\ \Lambda_R]$
            $e(x_{op}) = [e_v(x_{op}); e_z(x_{op})]$
            Set up Schur complement (Equation 21)
            and solve for $\delta x^*$
            $J^T \Lambda J\ \delta x^* = J^T \Lambda e(x_{op})$
            calculate $[x_s]^T = [x_k]^T + \delta x^*$
        end while
        //Remove outliers
        for all measurements in $I$ do
            compute $W_k$ with equation 10
            if $W_k < \omega$ then
                Remove measurement $k$ from $I'$
            end if
        end for
        $X = X_k,\ W = W_k$
    end while
    Store states and remove oldest pose
end for
return $X, W$;
```

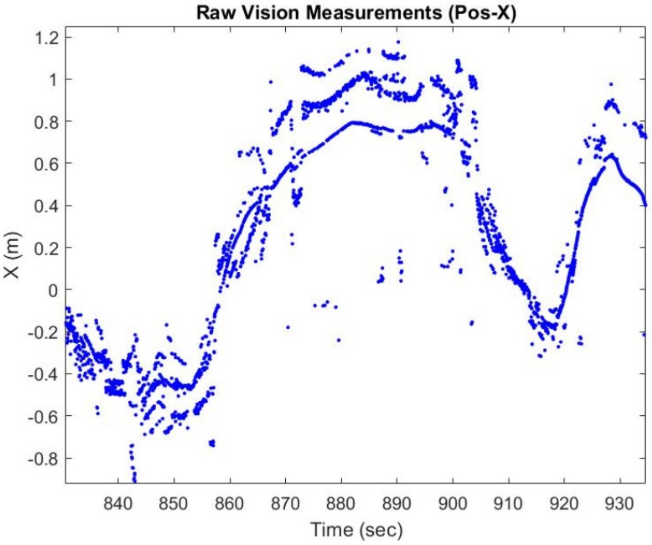

Figure 2: An example of outliers from the raw vision and markers measurements. It can be seen that the noises do not follow the standard Gaussian statistics but an offset-like outlier pattern, which stems from the confusion in recognising the markers.

confused with other markers, causing frequent outliers in the measurements as shown in Figure 2. The sampling rate of the IMU and the camera are 252Hz and 26Hz respectively.

### 5.1 Results

The resulting trajectory obtained from the experimental data after outlier rejection are presented in Figures 3 and 4. It can be observed that the proposed solution is capable of detecting wrong measurements in the interval (0-6s) where there are high number of outliers. Figure 5 illustrates the 3D raw vision position measurements of the underwater robot. The 3D classification of outliers and inliers for the position are shown in Figure 6. We can observe the influence of outliers on smoothed trajectory are minimised.

### 5.2 Window Size

The effect of different window sizes on the navigation performance are presented in Figure 7. The root mean-square error (RMSE) is calculated using the inlier measurement to compare the navigation accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(z_{k_{inlier}} - \hat{x}_k)^2} \tag{22}$$

A summary of the different window sizes performance are shown in Tables 1 and 2. We implemented and tested the algorithm in MATLAB using Intel Core i7 (4 Core) 3 GHz processor. It can be observed that the accuracy

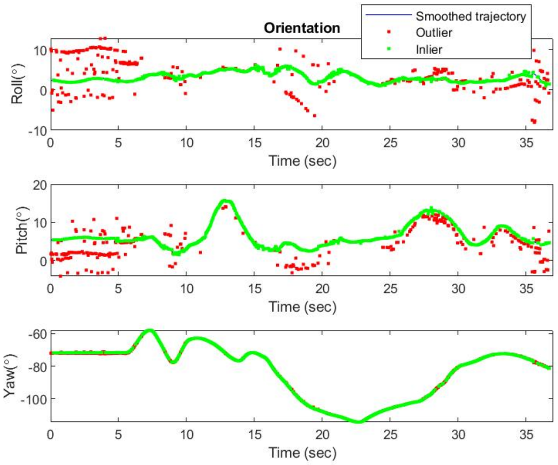

Figure 3: The smoothed orientation trajectory shown in black after classification of inliers(green) and outliers(red)

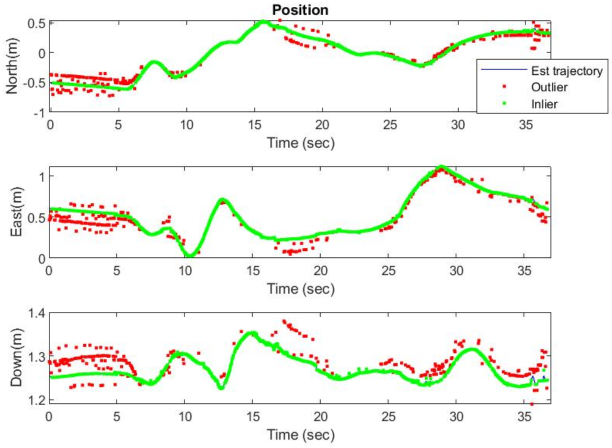

Figure 4: The smoothed position trajectory shown in black after classification of inliers(green) and outliers(red)

reduces as window size decreases while the full batch estimation yields the lowest RMSE after outlier rejection. On the other hand, as shown in Table 3, the full batch estimation has the highest performance run-time and the computational cost improves with reducing window size.

### 5.3 Discussion

The incremental batch estimation over a window of $n$ time steps presented in this work is equivalent to the fixed-lag smoother (BMFLS) in [Hassan *et al.*, 2021]. The results shown in Figures 3 and 4 for the 10-35 seconds periods are quite similar to results achieved in [Hassan *et al.*, 2021].

Although the proposed approach or SWF reduces the computational run-time, but it is still not very efficient. As a result SWF are still an active area of research [Barfoot, 2017]. However, the outlier detection loop in the algorithm also contributes in increasing the computational cost.

Table 1: Average transational RMSE comparison for different window sizes after outlier rejection

| Method | Window size | Average RMSE(m) |
|---|---|---|
| Full Batch Est | N/A | 3.5108e-04 |
| SWF | 60 | 1.6e-03 |
| | 80 | 1.5e-03 |
| | 100 | 1.3e-03 |

Table 2: Average rotational RMSE comparison for different window sizes after outlier rejection

| Method | Window size | Average RMSE(rad) |
|---|---|---|
| Full Batch Est | N/A | 5.1916e-04 |
| SWF | 60 | 9.5622e-04 |
| | 80 | 7.0716e-04 |
| | 100 | 7.2035e-04 |

Table 3: Performance run-time for different window sizes after outlier rejection

| Method | Window size | Run-time(s) |
|---|---|---|
| Full Batch Est | N/A | 1180.6 |
| SWF | 60 | 487.1 |
| | 80 | 917.9 |

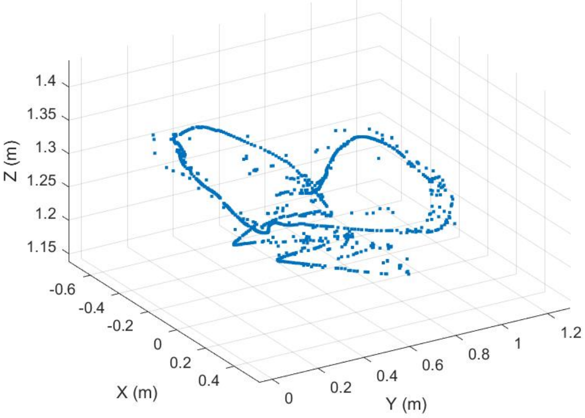

Figure 5: A proportion of the 3D raw vision position measurements

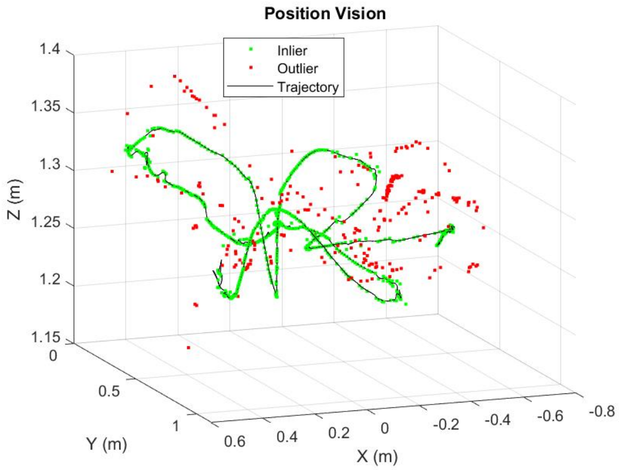

Figure 6: The smoothed trajectory shown in black line after classification of inliers (green) and outliers (red)

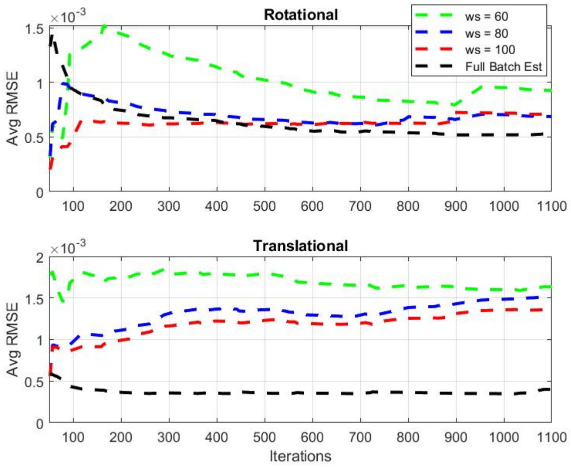

Figure 7: Average RMSE of the experimental dataset after outlier rejection for rotational and translational motions

## 6 Conclusions

In this work, we have presented an incremental robust navigation solution with EM using an experimental underwater dataset. The proposed solution has improved the computational efficiency compared to full bach estimation method while being capable of eliminating wrong measurements even when there are high intensity of outliers. However, other ideas such as smart marginalisation mentioned in [Sibley *et al.*, 2010] could be explored to increase the computational efficiency.

Modifying current work to an adaptive solution such that when there are less outliers, small window size is selected is one possible idea to further improve the computational performance.

## 7 Acknowledgment*

We acknowledge Dr Andrew To and Dr Khoa Le in collecting the dataset from the robot.

## References

[Agamennoni *et al.*, 2011] Gabriel Agamennoni, Juan I Nieto, and Eduardo M Nebot. An outlier-robust kalman filter. In *2011 IEEE International Conference on Robotics and Automation*, pages 1551–1558. IEEE, 2011.

[Barfoot, 2017] Timothy D. Barfoot. *Nonlinear Non-Gaussian Estimation*, page 88–144. Cambridge University Press, 2017.

[Cheng *et al.*, 2015] Jiantong Cheng, Jonghyuk Kim, Jinliang Shao, and Weihua Zhang. Robust linear pose graph-based slam. *Robotics and autonomous systems*, 72:71–82, 2015.

[Hassan *et al.*, 2021] Sajad Hassan, Hongkyoon Byun, and Jonghyuk Kim. Iterative smoothing and outlier detection for underwater navigation. *ACRA*, 2021.

[Kautz and Eskofier, 2015] Thomas Kautz and Bjoern M Eskofier. A robust kalman framework with resampling and optimal smoothing. *Sensors (Basel, Switzerland)*, 15(3):4975–4995, 2015.

[Lee *et al.*, 2013] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–563, 2013.

[Sibley *et al.*, 2008] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. *A Sliding Window Filter for Incremental SLAM*, pages 103–112. Springer US, 2008.

[Sibley *et al.*, 2010] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of field robotics*, 27(5):587–608, 2010.

[Sjanic *et al.*, 2011] Zoran Sjanic, Martin A. Skoglund, Thomas B. Schön, and Fredrik Gustafsson. A nonlinear least-squares approach to the slam problem*. *IFAC Proceedings Volumes*, 44(1):4759–4764, 2011. 18th IFAC World Congress.