Pose-based GraphSLAM Algorithm for Robotic Fish with a Mechanical Scanning Sonar

Ling Chen, Sen Wang and Huosheng Hu

Abstract—This paper proposes a pose-based GraphSLAM algorithm for robotic fish equipped with a Mechanical Scanning Sonar (MSS) that has a low frequency of range readings. The main contribution of this paper is the construction of a pose graph as the front-end part of the normal GraphSLAM algorithm. The proposed algorithm has three stages as follows: 1) scan generation which incorporates a novel Extended Kalman Filter (EKF) based algorithm that takes the fish motion into account; 2) data association which is based on Mahalanobis distance and shape matching for determining loop closures; 3) scan matching which is for constraints calculation and pose graph construction. The constructed pose graph is then fed into a back-end optimizer - g2o for finding the optimal position of robotic fish. The viability and the accuracy of the proposed algorithm are verified by extensive simulations, compared with the dead reckoning and scan matching approaches.

I. INTRODUCTION

Over years, autonomous underwater vehicles (AUVs) have been increasingly developed for underwater searching, exploring, reconnaissance, mining, etc. To achieve these tasks, it is necessary for them to keep the track of their own positions as well as to learn how the environment looks like. This is the so-called Simultaneous Localization And Mapping (SLAM) techniques. Over last two decades, various SLAM algorithms have been successfully applied to indoor and outdoor environments. Its application to the underwater environments remains challenging.

Due to the liquid nature of the underwater environment, the perception sensors used in AUVs are normally restricted to vision and sonar. Although vision sensors provide the most rich information about the underwater environment that is shallow and bright, they suffer from the low visibility especially in deep and cloudy underwater environments. On the other hand, sonar becomes the most widely used perception sensor for AUVs. There are three types of underwater sonar sensors that have been deployed in underwater SLAM applications, namely side-scan sonar [1], [2], multi-beam sonar [3], [4], Mechanical Scanning Sonar (MSS) [5]–[7].

For a side-scan sonar, the inertial sensors such as Doppler Velocity Log (DVL) and Inertial Measurement Unit (IMU) are firstly used to provide navigation data relative to the vehicle reference frame, such as velocities, orientations and depth. Then the features extracted from side-scan sonar are applied as the landmarks for updating both the navigation data and the positions of landmarks under the EKF-SLAM. However, features are difficult to identify and have typically low spatial density [4], degrading the accuracy of SLAM.

For a multi-beam sonar, the measurement model is either the depth governed with distributed particle filter framework [3] or the displacement estimated by scan matching of different patches [4], instead of extracting features. Whereas, for a MSS sonar, its advantages of low cost, small size, low power and computation makes it very popular in AUV applications, e.g. Ictineu AUV [7] and a robotic fish [8]. However, since it has the low frequency of range readings, it is necessary to build the scan by grouping a set of discrete scanned points and taking into account vehicle motion. Then the EKF-based SLAM is adopted, which is subject to quadratic update complexity and liberalization errors [9].

GraphSLAM provides an intuitive way of formulating SLAM by using a graph whose nodes correspond to poses of the robot at different points in time and whose edges represent constraints between the poses [10]. In contrast to EKF-SLAM, it has the access to the full data when building the map so that it can revise past data association and linearize more than once [9]. GraphSLAM consists of two parts, i.e. a front-end that is in charge of graph construction and a back-end that optimally computes the best map. Although there are various state-of-art approaches in literature, no existing work has focused on the deployment of GraphSLAM in AUVs with MSS.

This paper proposes a pose-based GraphSLAM algorithm for a robotic fish. The main contribution lies in constructing a pose graph as the front-end part of the GraphSLAM algorithm. The rest of this paper is organized as follows. Section II presents the architecture of the proposed algorithm. Section III introduces how a full scan of MSS is generated. Section IV details the data association algorithm. The scan matching and pose graph construction process are described.
in Section V. The simulation results are given in Section VI to verify the effectiveness and performance of the proposed algorithm. Finally, a brief conclusion and future work are presented in Section VII.

II. THE PROPOSED POSE-BASED GRAPHSLAM

In this paper, a pose graph is constructed as part of the front-end of the traditional GraphSLAM algorithm. Then the g2o algorithm proposed by [11] can be deployed for the back-end optimization to obtain the optimized pose nodes that are maximally consistent with the measurements. Fig. 1 presents the proposed architecture of the whole algorithm. The front-end part consists of three parts, namely scan generation, data association and scan matching and pose graph construction. The detailed description of these parts are presented in Algorithm 1, 2 and 3 respectively.

The optimized poses are then applied to render the map using the transformation history \( \mathbf{T}_{\text{cur}} \) stored when scan is built. For each scan \( S_j \), let \( \mathbf{T}_{W,i}^{\text{cur}}(i) \) be the corresponding optimized pose, we can calculate the corrected global position of each scanned point \( p_j \) by:

\[
p_o^{W}(i,j) = \mathbf{T}_{W,i}^{\text{cur}}(i) \oplus \mathbf{T}_{\text{cur}} \oplus \mathbf{T}_{S}^{R} \oplus \Delta \mathbf{C}(\bar{\mathbf{p}}(j))
\]

where \( p_o^{W}(i,j) \) is the corrected global position of scanned point \( p_j \) for scan \( S_j \). All the corrected global positions will be used to build a map by occupancy grid mapping algorithm [12].

The experimental platform in consideration is our robotic fish, as shown in Fig. 2. The sonar in our robotic fish is Tritech Micron (MSS) whose head rotates at a fixed angular speed and one complete scan lasts a few seconds. While the robotic fish is moving, the scanned image of the environment will be deformed, and can not be used directly for scan matching. Therefore, it is necessary to use robotic fish motion information to make a correction. Here, the robot pose data from dead reckoning is deployed for the scan correction.

III. SCAN GENERATION

A. Dead Reckoning

It is well known that the mean-squared navigation errors of an IMU increase with time quadratically or cubically without a boundary [13]. Therefore, instead of adopting the integration-based dead reckoning scheme, an EKF is used to estimate the robotic fish’s pose while the sonar is scanning. A simple 7 DoF \( ((x, y, z) \) and quaternion \( (q0, q1, q2, q3) \) constant velocity kinematics model is used in the prediction process of the EKF. A Mongoose 9DoF IMU is then used for providing the orientation measurement which updates orientation in the prediction. The state vector is represented as:

\[
X = [p^W, v^R]^T
\]

with:

\[
p^W = [x, y, z, q0, q1, q2, q3]^T;
\]

\[
v^R = [u_x, u_y, u_z, \omega_x, \omega_y, \omega_z]^T
\]

where \( p^W \) is the position and attitude vector represented by a quaternion referenced to a world frame \( W \), and \( v^R \) is the linear and angular velocity vector referenced to the vehicle’s coordinate frame \( R \). The reason for why quaternion is adopted as the orientation vector is that quaternion representation does not have a Gimbal lock problem and facilitates transformation and calculation in a simple form.

1) Prediction: The estimate of the state vector can be achieved by

\[
\hat{X}_k = f(\hat{X}_{k-1})
\]

Specifically,

\[
r_k = r_{k-1} + Q_{k-1} * U_{k-1} \ast Q_{k-1}^T
\]

where * is the quaternion multiplication, \( T \) is the cyclic updating time. The quaternion can be updated using the 4-order Runge-Kutta formula:

\[
Q_k = Q_{k-1} + (K_1 + 2K_2 + 2K_3 + K_4)/6
\]

where

\[
\omega_b = \begin{bmatrix}
0 & -\omega_z & -\omega_y & -\omega_x \\
\omega_x & 0 & -\omega_z & -\omega_y \\
\omega_y & -\omega_x & 0 & -\omega_z \\
\omega_z & \omega_y & \omega_x & 0
\end{bmatrix}
\]

\[
K_1 = 0.5T[\bar{\omega}_b, 0]Q_{k-1};
\]

\[
K_2 = 0.5T[\bar{\omega}_b, 0]Q_{k-1} + K_1/2;
\]

\[
K_3 = 0.5T[\bar{\omega}_b, 0]Q_{k-1} + K_2/2;
\]

\[
K_4 = 0.5T[\bar{\omega}_b, 0](Q_{k-1} + K_3);
\]

The slow movement of the robotic fish allows us to consider its linear and angular velocities as “constant”. An acceleration noise \( s_k \) is introduced for slight changes to them, which
is subject to a Gaussian distribution with zero mean and covariance \( C \) respectively.

\[
U_k = U_{k-1} + s_k T \\
E[s_k] = 0; E[s_k s_k^T] = C
\]

The covariance of the predicted state is obtained as follows:

\[
P_k = F_k P_{k-1} F_k^T + G_k C_k G_k^T
\]

where \( F_k \) and \( G_k \) are the Jacobian matrices of the partial derivatives of the non-linear model function \( f \) with respect to the state \( X \) and the noise \( s_k \), respectively.

2) Update using IMU measurements: The Mongoose 9DoF IMU is able to provide the calculated quaternion directly which then can be used for updating the prediction model. The measurement model is

\[
\hat{Z}_k = H_k \hat{X}_k + n_k
\]

where \( \hat{Z}_k \) is the predicted measurement, \( H_k \) is the measurement matrix, and \( n_k \) is the measurement noise with covariance matrix \( R_k = E(n_k n_k^T) \), and the measurement matrix is:

\[
H_k = [0_{4 \times 3} \ 1_{4 \times 4} \ 0_{4 \times 6}]
\]

The real measurement is:

\[
Z_{IMU,k} = [q_0, q_1, q_2, q_3]^T
\]

Then the standard Kalman Filter equations are used for updating the model prediction.

B. Scan Generation

The purpose of scan generation is to represent each separately read scan point with respect to a specific coordinate frame to form a full scan with the angle range of 360° as if all scan points are read instantaneously like a laser scanner. Theoretically, when each scanned point is sampled, any local coordinate frame of robotic fish could be chosen as a reference frame. For simplicity, we choose the coordinate frame of the robotic fish at the first scan (the scanning angle is -180°) as the reference frame, namely \( \{I\} \). This means all the scan points whose scanning angle is between -180° and 180° will be represented in frame \( \{I\} \), as shown in Fig. 3.

Suppose that MSS has a scanned point \( p_i \) in the sensor frame \( \{S\} \) whose coordinate is \( \tau(r, \theta) \) represented in a polar form. For convenience of subsequent calculation, \( \tau(r, \theta) \) is converted into cartesian coordinates:

\[
p^S = P2C(\tau) \Rightarrow p^S = N(P2C(\tau), J_S p_s J_s^T)
\]

where \( P2C(\tau) \) is a function of converting polar coordinates into cartesian coordinates and \( p^S = (x_s, y_s)^T = (r \cos(\theta), r \sin(\theta))^T \), \( J_s = \frac{\partial P2C(\tau)}{\partial \tau} \)

\[
p^S \text{ is then transformed to the body frame of the robotic fish } \{R\} \text{ by } T^R_S \text{ which is a constant vector that describes the position and attitude of the sensor frame } S \text{ with respect to frame } R.
\]

\[
T^R_S = (x_d, y_d, \alpha)
\]

where \( \alpha \) is the angle of frame \( S \) with respect to frame \( R \), and \((x_d, y_d) \) is the offset of the origin of frame \( S \) with respect to \( R \). The position (and its uncertainty) of scanned point \( p_i \) represented in \( \{R\} \) is calculated as:

\[
p^R = T^R_S \oplus p^S \Rightarrow p^R = N(T^R_S \oplus p^S, J_R p_s J_s^T)
\]

where \( \oplus \) is the compounding calculation proposed in [14], and \( J_R = \frac{\partial T^R_S \oplus p^S}{\partial p^S} \)

The position of \( p_i \) represented in the global frame \( \{W\} \) is calculated as:

\[
p^W = T^W_R \oplus p^R \Rightarrow p^W = N(T^W_R \oplus p^R, J_W p_s J_s^T)
\]

where \( T^W_R \propto N(T^W_R \oplus p_{WR}) \) is a random gaussian variable (r.g.v) representing the uncertain position of the robotic fish when the scanned point \( \tau \) is sampled, and

\[
J_W = \frac{\partial T^W_R \oplus p^R}{\partial p^R} \mid_{p^R}
\]

Then \( T^W_R \) can be obtained from the dead reckoning. The position of \( p_i \) represented in frame \( \{I\} \) can be calculated using the tail-to-tail combination [14] as follows:

\[
p^I = \ominus T^W_I \oplus p^W \Rightarrow p^I = N(T^W_I \oplus p^W, J_{11} p_{I1} p_{I1}^T + J_{12} p_{W1} p_{W1}^T)
\]

where \( T^W_I \propto N(T^W_I \oplus p_{WI}) \) is a r.g.v representing the initial uncertain position of the robotic fish in the global frame \( \{W\} \) when the scan forming starts at \( t = 0 \). This value
can also be obtained from dead reckoning. The Jacobian matrices are calculated as:

\[ J_{1\beta} = \frac{\partial T_{W}^{0}}{\partial T_{W}^{1\beta}} \]  \hspace{1cm} (21)

\[ [J_{11\beta} J_{12\beta}] = \frac{\partial \theta T_{W}^{1\beta} + \partial p_{W}^{1\beta}}{\partial T_{W}^{0} \partial p_{W}^{0}} \]  \hspace{1cm} (22)

The scan generation process is described in Algorithm 1. As a returned value, the generated scan is denoted as \( S_{\text{cur}} \).

![Algorithm 1 Scan Generation](image)

IV. DATA ASSOCIATION

In the pose-based GraphSLAM, the front-end is in charge of constructing the graph from raw measurements. While the nodes are determined as the robot positions at the 1st scan point of a whole scan, the constraints between nodes come from two scenarios: constraints between consecutive nodes and constraints resulting from loop closures. Constraints between consecutive nodes can be easily obtained by aligning generated two consecutive scans using scan matching algorithms. The determination of constraints from loop closures has to start with correct data association which tells whether the robot is revisiting previously mapped areas.

As proposed in [5], the data association is achieved by choosing those scans whose corresponding poses fall in the neighboring range of the pose of current scans as the loop closing candidates. We refer this strategy as pose-threshold based association. This strategy does not take into account the uncertainty of poses. So instead of using pose-threshold based association, the Mahalanobis distances between poses and shape matching are combined together to give more reliable associations. The Mahalanobis distance between two poses with uncertainty can be calculated as:

\[ D_{M} = \left( X(i) - X(j) \right) \left( \frac{P_X(i) + P_X(j)}{2} \right)^{-1} \left( X(i) - X(j) \right)^{T} \]  \hspace{1cm} (23)

where \( D_{M} \) is the Mahalanobis distance between pose \( X(i) \) and \( X(j) \) with covariances \( P_X(i) \) and \( P_X(j) \) respectively.

In addition to the Mahalanobis distance, the shape matching method is also introduced here to determine whether two scans are associated. Originally designed for shape matching in image processing, Angle Histogram of Vectors in [15] can directly be used for shape matching between two sonar scans as we expected. Two scans, \( S_{i} \) and \( S_{j} \), can be represented by \( S_{i} = \{ h_{1}, \cdots , h_{n} \} \) and \( S_{j} = \{ h_{1}, \cdots , h_{n} \} \), where \( h_{j} \) is the angle histogram from each row of the angle matrix of \( S_{i} \) and \( S_{j} \). The cost \( C_{ij} \) for matching two points is defined as:

\[ C_{ij} = C(p_{i}, q_{j}) = \frac{1}{2} \sum_{k=1}^{M} \left[ h_{i}(k) - h_{j}(k) \right]^{2} \]  \hspace{1cm} (24)

where \( p_{i} \) and \( q_{j} \) are the points on each segments, \( M \) is the number of the bins of the corresponding angle histograms. Then the distance between these two segments is calculated as:

\[ D(I_{1}, I_{2}) = \frac{1}{K_{1}} \sum_{p \in I_{1}} \arg \min_{q \in I_{2}} C(p, q) + \frac{1}{K_{2}} \sum_{q \in I_{2}} \arg \min_{p \in I_{1}} C(p, q) \]  \hspace{1cm} (25)

This \( D(I_{1}, I_{2}) \) is then assigned to \( D_{S} \) representing the shape matching distance between scans \( S_{i} \) and \( S_{j} \). When scans \( D_{M} \) and \( D_{S} \) are less than some predefined thresholds (\( M_{\text{thd}} \) for Mahalanobis distance, \( S_{\text{thd}} \) for shape matching), they are considered as associated. Algorithm 2 presents the data association strategy. The inputs of the algorithm include all the scan sets \( S \), robot pose sets from dead reckoning \( X \) and their corresponding covariance \( P_X \). The returns of the algorithm are the indices (represented as \( A_{r} \) and \( A_{s} \)) of the associated scans.

![Algorithm 2 Data Association](image)

V. SCAN MATCHING AND POSE GRAPH CONSTRUCTION

Scan matching is to determine both the constraints between consecutive scans and the constraints resulting from loop closures. The most popular technique for scan matching...
is the Iterative Closest Point (ICP) algorithm and its variants such as pIC [16], MSISpIC [17] and uspIC [18]. Instead of using Euclidean distance as ICP does, they are all based on Mahalanobis distance to find correspondences between two scans. As claimed in [18], uspIC outperforms pIC and MSISpIC in terms of accuracy, so uspIC is adopted here as the scan matching algorithm.

Algorithm 3 shows the detailed process of both scan matching and pose graph construction. For each pair of associated scans whose indices are stored as \( A_i \) and \( A_j \) from the returns of Algorithm 2, the uspIC is utilized to calculate the relative displacement \( q_0 \) and its covariance \( P_0 \) which then can be added as a constraint or edge to the pose graph \( G \). The initial relative displacement estimation \( q_0 \) and its covariance \( P_0 \) for uspIC are computed by tail-to-tail combination.

**Algorithm 3 Scan Matching and Pose Graph Construction**

1: \([G] = SMPCG(S, A_i, A_j, X, P_X, G)\{\)
2: len\_associated = length\((A_i)\);
3: for \( k = 1 \) to (len\_associated) do
4: \( i = A_i(k)\);
5: \( j = A_j(k)\);
6: \( q_0 = \circ X(i) \oplus X(j)\);
7: \( P_0 = J_{1G}P_X(i)J^T_{1G} + J_{2G}P_X(j)J^T_{2G}\)
8: \([q_0, P_0] = uspIC(S(i), S(j), q_0, P_0)\);
9: Add constraint \((q_0, P_0)\) to \( G \)
10: end for\}

VI. SIMULATION RESULTS

Simulation based on Robot Operation System (ROS) [19] is conducted to verify the effectiveness of proposed pose-based GraphSLAM algorithm. The simulated environment in Fig. 4 is imported to the 3D simulator Gazebo [20]. In the simulation, the robotic fish is tele-operated to swim around the middle object for two rounds with a 174m trajectory at an average speed of 0.1m/s corrupted by a gaussian noise with standard variance of 0.05 m/s.

The MSS sensor obtains 360° scans of the environment by rotating a sonar beam through 240 angular steps in 12 seconds. During the experiment, IMU readings, sonar readings and the ground truth of the robotic fish pose are stored for off-line processing and performance analysis. The proposed algorithm is then applied to optimally estimate the positions of the robotic fish. For comparison, the positions estimated by dead reckoning and uspIC are also provided.

Fig. 5 shows trajectories generated by dead reckoning, uspIC, pose-based GraphSLAM and the ground truth respectively. It can be clearly seen that among all the trajectories, the trajectory calculated from the pose-based GraphSLAM is the closest one to the ground truth. Trajectory from dead reckoning is the furthest from the ground truth because of its unbounded accumulated errors from IMU. The trajectory generated by uspIC is better than that of dead reckoning, but also suffers from unbounded accumulated errors which can be seen in the magnified part in Fig. 5.

Fig. 6 presents the distance errors between positions estimated by each algorithm and ground truth. It is obvious that dead reckoning has the largest distance errors, and uspIC is better and the pose-based GraphSLAM is the best. The average distance errors for dead reckoning, uspIC and pose-based GraphSLAM are 4.43 (m), 1.05 (m) and 0.46
(m) respectively. Fig. 7 is given to show maps rendered by using the positions estimated by dead reckoning, uspIC and pose-based Graph SLAM respectively compared with ground truth. As expected, the map built by pose-based GraphSLAM matches well with the ground truth without the ghosting effect. The map built by dead reckoning, however, is seriously distorted and has much accumulated errors. Although the map built by uspIC outperforms the map built by dead reckoning, it is still far from the ground truth.

VII. CONCLUSION

This paper focuses on a pose-based GraphSLAM algorithm for a robotic fish to operate in an underwater environment. The main contribution is the construction of a pose graph as the front-end part before the application of back-end GraphSLAM algorithm g2o. An EKF-based algorithm is firstly designed for scan generation, which takes the motion of the robotic fish into account. Then the data association algorithm based on Mahalanobis distance and shape matching is deployed to determine loop closures, leading to associated scan pairs used for calculating constraints of the pose graph. After the pose graph has been constructed, the GraphSLAM algorithm g2o is applied for optimally estimating the poses. The proposed pose-based GraphSLAM algorithm is tested in a simulated environment that is based on ROS. The trajectories and maps obtained from experiments are presented. It is clear that the proposed algorithm outperforms other two traditional algorithms such as dead reckoning and uspIC in terms of both localization and mapping accuracy. Our future work will focus on testing the algorithm in the real underwater environment with the real robotic fish.

ACKNOWLEDGMENTS

This research is financially supported by the REO Development Fund of Essex University, with a grant number of FIO3005, and the EPSRC Global Engagements funding, with a grant number of EP/K004638/1. Our thanks also go to Ian Dukes and Robin Dowling for their technical support.

REFERENCES