Vision based Passive Arm Localization Approach for Underwater ROVs Using a Least Squares on $SO(3)$ Gradient Algorithm

Chandra P. Mangipudi and Perry Y. Li

Abstract—This paper proposes a vision-based alternative to the passive arm pose estimation for underwater remotely operated vehicle (ROV) performing manipulation tasks. The proposed approach attaches a fixed landmark on an underwater fixture and uses the camera images of the landmark object points to infer the pose of the ROV. A gradient descent least squares algorithm on the $SO(3)$ manifold is proposed for accurately and efficiently estimating the pose. The algorithm has been implemented on a low-cost single board computer. Numerical comparison with other existing algorithms as well as in-air and underwater experiments show the efficacy of the algorithm. Positional accuracy of the order of 1-2.5mm while the landmark is approximately 1m away has been demonstrated.

I. INTRODUCTION

For underwater robot manipulation using remotely operated vehicles (ROVs), the simplest yet robust and reliable approach for pose (i.e. position and orientation) sensing is with the so-called passive arm [6]. In this approach, the ROV is equipped with a 6-degree of freedom un-actuated robot arm whose end-effector is attached to an underwater fixture (e.g. a pipe). The pose of the ROV, relative to the fixture, is inferred from the joint angle measurements and the kinematics of the arm. The calculated pose can then be used for dynamic positioning and other control purposes.

Obvious drawbacks of the passive arm include the cost, weight and size of the arm itself, the limited range of operation, and the risk of damaging the arm when the range is violated. To overcome these drawbacks, our motivation is to replace the passive arm with a computer vision-based approach to determine the pose relative to an underwater fixture. To this end, a 3-D optical marker with known geometry will first be attached to the underwater fixture (replacing the end-effector) and one or more cameras, rigidly affixed to the ROV will be observing the marker. Because the connection between the marker and the camera(s) is by light instead of a mechanical linkage, the disadvantages of the passive arm can be avoided.

The pose estimation problem is also known as the Perspective n Point (PnP) problem where $n$ signifies the number of object points used for the estimation. Early work by Horn and co-workers [5] considered closed form solution using orthonormal matrices. Later research have investigated various iterative and closed form solutions. Several of these will be described and compared to our proposed algorithm in Section III. The main idea in all these works is that given the correct pose estimate, the estimated image locations of the marker points, based on the camera model, will be close to the image points measured by the camera(s).

In our proposed $SO(3)$ gradient algorithm, there are two distinct features. Firstly, the error function is not taken directly from the image space but instead is formulated such that it is affine in the rotation matrix and position vector to be estimated. This allows us to define an equivalent least squares minimization problem on the $SO(3)$ space of rotation matrices only instead of the $SE(3) = SO(3) \times \mathbb{R}^3$ space of both rotation matrices and position vectors. Secondly, with the formulation, we define a gradient flow algorithm on the $SO(3)$ space to minimize the least squares error. Once the rotation matrix is estimated, the position can be computed from a closed form linear expression.

The $SO(3)$ gradient algorithm is quite efficient and has been implemented on a Raspberry Pi 2B+ (an older version low-cost single board computer) at a frame-rate of 9 frames/second. To supplement pose information in-between frames, pose estimates are fused with information from an inertial measurement unit (IMU) using an event based Kalman filter [8]. This paper will however only focus on the vision-based pose-estimation aspect.

The rest of the paper is organized as follows. The proposed $SO(3)$-gradient PnP algorithm is discussed in detail in section II. Section III evaluates the performance of several different PnP algorithms. Experimental verification by trajectory tracking using $SO(3)$-gradient PnP is performed in section IV. Section V summarizes the key results and provides an insight into future research.

II. LEAST SQUARES ON $SO(3)$ FORMULATION

A. Camera Measurement Model

The camera is modeled as a pin-hole camera as shown in the schematic in Fig. 1. The position of any object point $(i)$ in the camera frame $\{C\}$ is given by $cP_i = [x_i \ y_i \ z_i]^T$, while the same point in the global frame $\{G\}$ is given by $^gP_i = [X_i \ Y_i \ Z_i]^T$. The origin of $\{G\}$ in $\{C\}$ is given by $cP_0$, while the orientation of $\{G\}$ relative to $\{C\}$ is given by the rotation matrix $^cR_G$. The object point’s positions in the camera and global frames are related by:

$$
\begin{bmatrix}
x_i \\
y_i \\
z_i \\
\end{bmatrix} = ^cP_i + ^gR_G^cP_i 
$$

(1)
The marker is assumed to have \( n \) object points \( (i = 1, \ldots, n) \) and their locations relative to \( \{G\} \) are assumed to be known.

The noise-free perspective projection due to a camera lens in standard form is shown in (2):

\[
\begin{bmatrix}
u_i - \frac{w_{img}}{2} \\ v_i - \frac{h_{img}}{2}
\end{bmatrix} = \frac{1}{z_i} \begin{bmatrix}
x_i \\ y_i
\end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ -z_i & 0 & 1
\end{bmatrix} \begin{bmatrix} C \end{bmatrix} \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix}
\]

\( \alpha \in \mathbb{R}^{2 \times 3} \)

where \((u_i, v_i)\) are measured from the camera image and the extrinsic matrix is the unknown to be estimated. The camera intrinsic matrix is represented by the camera focal length \((f_x, f_y)\) parameters and the image dimensions \( (H_{img}, W_{img}) \).

Because of the dependence on \( z_i \), (2) is nonlinear w.r.t. to the position \( C \), the rotation matrix \( G \). The perspective projection equation can, however, be re-arranged as:

\[
0 = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ -(u_i - \frac{w_{img}}{2}) & 0 & -z_i & 0 & 1
\end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i
\end{bmatrix} = \begin{bmatrix} \alpha_1 \end{bmatrix} \begin{bmatrix} C \end{bmatrix} \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix}
\]

\( \alpha \in \mathbb{R}^{2 \times 3} \)

In real world, due to the presence of noise in the measured pixels, modeling errors and environmental disturbances, the equation errors for each object point \( \in \mathbb{R}^3 \) can be modeled as:

\[
e_i = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n\end{bmatrix} = \begin{bmatrix} \alpha_1 \end{bmatrix} \begin{bmatrix} \gamma(G) \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n\end{bmatrix} \begin{bmatrix} C \end{bmatrix} \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix}
\]

or

\[
e = \gamma(G) e_i - \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n\end{bmatrix} \begin{bmatrix} C \end{bmatrix} \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix}
\]

Note that \( e \) is affine in both \( \gamma(G) \) and \( C \).

The pose estimation problem is therefore to find \( \gamma(G) \) and \( C \) that minimize the least squares error:

\[
\text{Objective: } \begin{bmatrix} \gamma(G) \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n\end{bmatrix} \begin{bmatrix} C \end{bmatrix}
\]

Due to the affine dependence on \( \gamma(G) \), if \( \gamma(G) \) is known, the corresponding optimal \( C \) for the inner minimization can be easily obtained using linear least squares formula. To wit, for each estimate of \( \gamma(G) \), the corresponding estimate of \( C \) is:

\[
C = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n\end{bmatrix} \begin{bmatrix} G \end{bmatrix} \begin{bmatrix} P_i \end{bmatrix}
\]

\( \alpha \in \mathbb{R}^{2 \times 3} \)

C. Gradient Descent Algorithm

Although \( e_i^*(\gamma(G)) \) in (8) appears to be affine in \( \gamma(G) \), it cannot be solved linearly because \( \gamma(G) \) is restricted to the space of rotation matrices - \( SO(3) \). We compute it iteratively via the negative gradient flow of the cost function \( J^* \), for simplicity \( J^* \). Using the fact that \( e_i^* = \beta(\alpha) \gamma(G) \), the derivative of \( J^* \) is:

\[
\partial J^* = e_i^T \gamma(G) \partial \gamma(G) = \sum_{i=1}^{n} e_i^T \partial \gamma(G)
\]

The differential of the cost function can be explicitly expressed in terms of the rotation matrix:

\[
\partial J^* = \sum_{i=1}^{n} e_i^T \partial \gamma(G) \partial \gamma(G) = \sum_{i=1}^{n} e_i^T \partial \gamma(G) \partial \gamma(G)
\]

\( \beta(\alpha) \) is a projection so that \( \beta^2(\alpha) = \beta(\alpha) \) and \( \beta(\alpha) = \beta^-1(\alpha) \). Using this property, the least squares objective is now given by:

\[
\min_{\gamma(G) \in SO(3)} \left\{ J^* \gamma(G) = \frac{1}{2} e_i^T e_i^* = \frac{1}{2} \gamma^T (G) \beta(\alpha) \gamma(G)
\right\}
\]

where \( e_i^* \) and \( \alpha_i \) are known or measured, and \( e_i^* (G) \) is given by (8). Once \( \gamma(G) \) is arg \( \min \) \( J^* \) is found, \( C \) can be obtained from (7).

\[
\beta(\alpha) = \frac{1}{2} \gamma^T (\gamma(G) \beta(\alpha))
\]

\( \beta(\alpha) \) is a projection so that \( \beta^2(\alpha) = \beta(\alpha) \) and \( \beta(\alpha) = \beta^-1(\alpha) \). Using this property, the least squares objective is now given by:

\[
\min_{\gamma(G) \in SO(3)} \left\{ J^* \gamma(G) = \frac{1}{2} e_i^T e_i^* = \frac{1}{2} \gamma^T (G) \beta(\alpha) \gamma(G)
\right\}
\]

where \( e_i^* \) and \( \alpha_i \) are known or measured, and \( e_i^* (G) \) is given by (8). Once \( \gamma(G) \) is arg \( \min \) \( J^* \) is found, \( C \) can be obtained from (7).
Any differential change in rotation can be represented as:

\[
\partial_c^* \mathbf{R} = \Omega \mathbf{c} \in \text{SO}(3), \quad \Omega = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}
\]

for some skew-symmetric matrix \( \Omega \) with arbitrary \((p, q, r)\) representing a direction of differential rotation. Using the property of trace that \( \text{tr}(AB) = \text{tr}(BA) \), the differential of the cost function can be simplified as:

\[
\partial J^* = \sum_{i=1}^{n} e_i^T \alpha_i \mathbf{c}^T \mathbf{R}^T G P_i e_i = \sum_{i=1}^{n} \text{tr}(\mathbf{c}^T \mathbf{R}^T G P_i e_i^T \alpha_i \Omega)
\]

\[
= \text{tr} \left( \sum_{i=1}^{n} \mathbf{c}^T \mathbf{R}^T G P_i e_i^T \alpha_i \right) \Omega = \text{tr}(N(\mathbf{c}^T \mathbf{R}) \Omega)
\]

where \(N(\mathbf{c}^T \mathbf{R}) = \sum_{i=1}^{n} \mathbf{c}^T \mathbf{R}^T G P_i e_i^T \alpha_i \). The gradient flow for \( \mathbf{c}^T \mathbf{R} \) is obtained by choosing a direction for \( \Omega \) that minimizes \( \partial J^* \). In this case, \( \Omega \) should be the negative skew symmetric part of \(N(\mathbf{c}^T \mathbf{R})\): \( \Omega = (\mathbf{N} - \mathbf{N}^T)/2 \). The axis of rotation \( \tilde{k} \) is determined by the normalized angular velocity vector:

\[
\tilde{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = -\frac{1}{2} \begin{bmatrix} (n_{23} - n_{32}) \\ (n_{31} - n_{13}) \\ (n_{12} - n_{21}) \end{bmatrix}; \quad \tilde{k} = \frac{\tilde{\omega}}{||\tilde{\omega}||}
\]

D. Discrete implementation

A continuous implementation of the gradient flow would require continuous adjustment and re-computation of the rotation angle which would be computationally prohibitive. Instead, a discrete approximation is used in which the direction of rotation as determined by \( \Omega \) or \( \tilde{k} \) is fixed. \( \mathbf{c}^T \mathbf{R} \) is updated along this direction until \( J^* \) cannot decrease:

\[
\mathbf{c}^T \mathbf{R}^{i+1} = e^{[\tilde{k} \times]} \theta \mathbf{c}^T \mathbf{R}^i
\]

where \( e^{[\tilde{k} \times]} \theta \) is the matrix representing the rotation about the axis \( \tilde{k} \) for the angle \( \theta \). Bisection search is used for this line search (for rotation \( \theta \) in the direction of \( \tilde{k} \)). When the minimum cost along this direction is found, a new gradient direction is computed and the process repeats. The algorithm terminates when a terminal threshold for the step size is reached.

A summary of the algorithm is presented below:

**Algorithm 1:** Summary of the gradient \( \text{SO}(3) - \text{PnP} \)

1) Take image and determine pixel locations of markers;
2) Initial guess of \( \mathbf{c}^T \mathbf{R}^{0} \) (e.g. from last image)
3) Find gradient of \( J^* \) at \( \mathbf{c}^T \mathbf{R} \) with rotation axis \( \tilde{k} \) in (13)
4) Determine rotation angle \( \theta \) in direction of \( \tilde{k} \) with bisection search to minimize \( J^* \) in this direction.
5) Set \( \mathbf{c}^T \mathbf{R}^{t+1} \) according to (14)
6) Goto 3 if \( \theta \) is above threshold
7) \( \mathbf{c}^T \mathbf{R}^t \) has converged. Calculate \( \mathbf{c}^T \mathbf{P}_0 \) according to (7)

III. PERFORMANCE COMPARISON

In this section, we compare the accuracy and computation cost of the proposed \( \text{SO}(3) \) gradient algorithm (\( \text{SO}(3) \)-PnP) against those of several popular state-of-the-art PnP algorithms:

1) The POSIT algorithm [1] uses scaled orthographic projections of the object points are superposed with the actual image points and an approximate pose is computed with the assumption that the projection model is scaled orthographic. The object points are shifted from their approximate position to positions on their line of sight which alters the object properties and computed image points are obtained using a scaled orthographic projection. The iterations are repeated with the scaled orthographic projection of the computed image points till the computed projection points converge with the image points.

2) P-PnP [2] is a specific implementation of POSIT. It formulates the problem as an anisotropic orthogonal Procrustes problem. The depth matrix (\( \mathbf{x}' \) of each object point) is solved iteratively. For each iteration, the rotation matrix and the position vector are obtained from the guess value of the depth matrix using Singular Value Decomposition (SVD) [3].

3) Orthogonal Iteration (LHM) [7] algorithm minimizes an object space co-linearity error. The position vector is formulated as a function of the rotation matrix and is assumed to be a constant for that particular iteration. This reduces the problem to an absolute orientation problem (i.e. find an orientation \( R \) s.t. \( R = \text{argmin} \Sigma (R P_i - F P_i) \)). The iterations are repeated till convergence and the LHM method ensures global convergence.

4) E-PnP[9] reduces the problem by choosing 4 optimal object points (obtained from all the object points) referred to as the control points. The 4 optimal control points are used to estimate the pose using a closed form linear constrained solution. E-PnP + GN (Efficient PnP with Gauss Newton iterations) utilize the E-PnP to obtain the initial pose estimate and perform iterations using GN algorithm around the initial pose to reduce the image space error.

5) The DLS-PnP[4] formulates the position vector as a function of the rotation matrix using Least Squares. The cost function is the non-linear pixel error norm. The rotation matrix is expressed in terms of the Cayley-Gibbs-Rodriguez (CGR) parameters. The scalar cost function is optimized w.r.t. the scalar CGR parameters (in polynomial form) using SVD.

A summary of the cost functions and solution methods is presented in table I.

The different algorithms were ran in MATLAB. The object points were chosen using a random distribution in the Field of View (FOV) of the camera. The image size was chosen as \((640 \times 480 \text{ pixels})\) with a camera focal length of \((f = 800)\). The noise in the image was assumed to be Gaussian with
TABLE I: Cost functions and solution methods for the various algorithms. \( S = \) Object Points, \( P = \) Image Points, \( Z = \) Depth matrix, \( \mathbf{c} = \) Translation Vector

<table>
<thead>
<tr>
<th>Algo</th>
<th>Type</th>
<th>Cost Function ( J )</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSIT</td>
<td>Iterative</td>
<td>( Ax = b )</td>
<td>( x = (A^TA)^{-1}A^T )</td>
</tr>
<tr>
<td>LHM</td>
<td>Iterative</td>
<td>( \sum_{i=0}^{n}(</td>
<td>RPr + t(R) - Fr</td>
</tr>
<tr>
<td>P-PnP</td>
<td>Iterative</td>
<td>( S = ZPR + 1c )</td>
<td>( R^{k+1} = Udiag(1,1,</td>
</tr>
<tr>
<td>E-PnP</td>
<td>Closed-Form</td>
<td>( Mx = 0 )</td>
<td>( x = \frac{\sum_{i=0}^{n} \beta}{\sqrt{n}} ), ( \beta = [\beta_0, \beta_1, ..., \beta_n] )</td>
</tr>
<tr>
<td>DLS-PnP</td>
<td>Closed-Form</td>
<td>( f_{poly}(s_0^\gamma, s_1^\gamma, s_2^\gamma), 0 \leq \gamma \leq 4 )</td>
<td>SVD of Macaulay Matrix</td>
</tr>
</tbody>
</table>

variance \( \sigma = 0.5 \) and mean \( \mu = 0 \).

The translation error percentage is the percentage ratio of the magnitude of the error displacement vector \( \delta = T' - T \) with respect to the magnitude of the original translation vector \( T \). The rotation error in degrees is computed as the inverse cosine \( \text{acos}(x'x + y'y + z'z) \) of the sum of the dot products of each unit vector between the estimated rotation matrix \( (R' = [x', y', z']) \) and the corresponding actual rotation matrix \( \mathbf{R} = [x, y, z] \).

The performances of the various algorithms are shown in Fig. 4. The proposed \( SO(3) \) gradient algorithm \( (SO(3)-\text{PnP}) \) gives the smallest translational and rotational errors. The computation cost of the \( SO(3) \)-gradient algorithm increases with the number of points and is second highest of the algorithms tested. However, for small number landmark points \( (4 \leq n \leq 20) \), the computational time is competitive. Moreover, in actual application, where continuous motion is expected, the gradient algorithm can use the previous result as initial guess. This will significantly reduce the number of iterations and computation time.

IV. EXPERIMENTS

A Raspberry Pi 2B+ single board computer and camera module were used to implement the proposed algorithm. The camera was calibrated using a standard checker board pattern to estimate the camera intrinsic matrix.

![Image](a) LAB image

![Image](b) Color Threshold image

![Image](c) Identified contours

Fig. 3: Sample sequence of image processing

The landmark comprised of 4 differently colored spherical ping pong balls as shown in Fig. 2. It consists in first translating the color image into the LAB uniform color space. This allows the color balls to be identified with less sensitivity to lightness. The colors are then given independent thresholds to identify the ball contours. The centroids of the ball images are then obtained and used in the \( SO(3) \) gradient algorithm.

![Figure 2](Configuration of landmarks. The board is approximate 15cm x 20cm. The pink ball protrudes out-of-the-plane by 10cm.)
 Portions of the image processing steps are performed on the graphics processing unit (GPU) of the Raspberry Pi to increase computational efficiency. Fig. 3 shows a sample image at different stages of the processing sequence (Fig. 5). From any viewing angle, the center point of any circular contour yields the center point of the corresponding spherical marker. A frame rate of approximately 9 frames per second is achieved with this low cost hardware.

Two sets of experiments have been performed, in air and underwater.

![Sample Image](image)

**Fig. 6: In-air experimental setup in which the camera moves around a rectangle on a table.**

### A. In-air experiment

For the in-air experiment, the Raspberry-Pi camera on a stand was manually moved around a 40cm × 30cm rectangular trajectory on a table as shown in Fig. 6. The landmark is approximately 1m away. This trajectory constitutes the absolute measurement and the sensor estimate is compared with the absolute measurement as a reference. The pose estimation is performed in $\{C\}$ and then transformed into $\{G\}$ to compare the estimated trajectory with the real world trajectory. The results of the trajectory tracking are shown in fig. 7. The accuracy of the position estimate is $\pm 0.1cm$ in x,y-axis and $\pm 0.25cm$ in z-axis. This experiment validates the SO(3)-gradient algorithm.

### B. Underwater Experiments

The camera was calibrated underwater using a standard checkerboard pattern to account for the change in focal length of the lens w.r.t. the medium of light propagation. The experimental setup is shown in Fig. 8. The submersed camera sensor was mounted on a carriage that can translate radially or rotate about the vertical axis. Linear and rotary encoders are available for measuring these motions to compare with the pose estimation obtained using the camera. The pose
estimate from the sensor compared with the encoder data is shown in Fig. 9.

Fig. 7: In-air experimental results with the camera moving around a square

(a) 3D-View

(b) Horizontal Planar View

Fig. 8: Underwater experimental setup

Fig. 9: Underwater experimental results.

V. CONCLUSIONS

This paper proposes to use a vision based localization approach to replace the passive arm approach for estimating the pose of an underwater ROV. The proposed pose estimation algorithm is based on gradient flow of the least squares error function on the SO(3) manifold. The proposed SO(3)-gradient algorithm is the most accurate of the several existing algorithms and is competitive as far as computation burden is concerned. The formulation of the position in terms of rotation matrix leads to the possibility of a closed form solution similar to the DLS approach in SO(3) space. The limitations of using the SO(3) gradient PnP algorithm arise from the Field of View (FOV) limitations on the monocular camera. An extension to the algorithm with stereo cameras would compensate for the FOV limitation. In addition, the vision based pose estimation can be augmented and fused with IMU information to provide higher frequency (between camera frames) updates [8].

REFERENCES