PASTA: Provably Accurate Simple Transformation Alignment

Matteo Marchi, Jonathan Bunton, Yskandar Gas, Bahman Gharesifard, and Paulo Tabuada

***Abstract*— LiDAR is a widely used sensor for self-localization and SLAM algorithms. Most of these algorithms rely on solving the point cloud registration problem as a subroutine. While many registration methods exist, most are local, i.e., they only converge when presented with similar point clouds. Moreover, they offer no formal guarantees on their performance. In this work we present a low-complexity *global* point registration technique that is suitable for point clouds generated from LiDAR scans and has formal worst-case error guarantees under mild convexity assumptions on the environment. Moreover, we empirically show that the proposed method compares favorably, on real LiDAR data, to state of the art methods such as ICP.**

I. INTRODUCTION

LiDAR[1] is a popular choice of sensor for autonomous robots and vehicles [1], [2], [3]. Its accuracy, insensitivity to lighting conditions, diminishing cost, and increasing availability in recent years make LiDAR an indispensable choice in numerous contexts requiring high dimensional perceptual data as an alternative, or in addition to, camera vision. LiDAR is widely used in self-localization, mapping, and SLAM algorithms, with an increasing amount of literature devoted to its various properties [4], [5]. Closely related to the localization problem, the point cloud registration problem with LiDAR data has itself become an active area of research [6].

Informally, point cloud registration is the problem of finding the rigid transformation that maximizes the "overlap" between two point clouds. In the ideal setting with no noise and uniform data points, the problem has a classical analytical solution provided in [7]. However, this solution cannot be used in most scenarios, and additional difficulties arise when working with point clouds generated from LiDAR measurements. For example, point clouds generated from LiDAR data have a non-uniform density of data points, thus point clouds from measurements at different positions are not, in general, related by a rigid transformation.

Most LiDAR point cloud registration algorithms fall under three categories: Feature based [6], Iterative Closest Point (ICP) based [8], and Normal Distribution Transform (NDT) based [9]. Even though we do not discuss these methods in detail, we provide some remarks to contextualize our contributions.

This research was supported in part the US Army Research Laboratory (ARL) under Cooperative Agreement W911NF-17-2-0196.

M. Marchi, J. Bunton, Y. Gas, B. Gharesifard, and P. Tabuada are with the Electrical and Computer Engineering Department, University of California at Los Angeles, Los Angeles, CA 90095 USA. {matmarchi, j.bunton, yskandargas, gharesifard, tabuada}@ucla.edu

[1]Light Detection and Ranging.

Each approach has advantages and disadvantages, and may be used in conjunction with one another, as in [10], or as different steps in a coarse-to-fine localization algorithm. Feature-based registration relies on the detection of geometric elements, such as points, corners, lines, and planes, that are used to perform the alignment of the point clouds rather than working with the point data directly. Feature-based approaches can address some of the difficulties with LiDAR data, but require carefully tuning the choice of features for different environments.

Alternatively, ICP is an optimization method that attempts to repeatedly establish correspondences between points of the two clouds, and align them. The method is simple and can achieve high alignment accuracy, however, it is optimization-based and thus requires an accurate initial guess to avoid being trapped in a local optimum. NDT splits the reference point cloud into a grid, and computes a mixture of Gaussian distributions, with a Gaussian for each element of the grid based on the contained points. The target point cloud is then treated as a set of samples from this mixture, and the alignment is computed by maximizing the likelihood that the points were generated from this mixture. The method typically has lower accuracy than ICP, but a larger radius of convergence [9].

One feature that is often missing, but relevant to this work, is a *formal error guarantee* on the rigid transformation estimated by an algorithm. This guarantee is important for safety critical applications, and, to the authors' knowledge, only one point registration algorithm, introduced in [11] under the name of TEASER++, provides such results in the form of a certification guarantee for a computed transformation. TEASER++, however, is designed to work for point cloud pairs with at least a few one-to-one point correspondences which is not typically true of LiDAR data, especially with low resolution sensors.

Our main contribution is a fast and simple global method for point cloud registration, called Provably Accurate Simple Transformation Alignment (PASTA), that enjoys formal guarantees on the estimated translation vector and rotation matrix under a mild convexity assumption. Given two point clouds, PASTA computes their convex hulls and compares their first and second moments in an alignment step. Comparing the convex hulls of the point clouds, rather than the points themselves, makes the method robust to the variable point density of LiDAR measurements. We show, using real LiDAR measurements, that PASTA compares favorably against current state-of-the-art methods in terms of localization error. We then test the robustness of PASTA in non-convex environments, where our theoretical guarantees do not apply, and show that also in this case we obtain performance that is better or comparable to state of the art methods. We note that our proposed method bears minor similarities to PCA-based methods of aligning 2D images [12], [13]. In our case, however, the moments–the mean and covariance–are computed for a *set*, rather than for a point cloud or a grid of pixels.

II. Preliminaries

We first introduce some mathematical preliminaries. Note that depending on the LiDAR sensor, the point cloud dimension n can be either 2 or 3.

1) A pose of the LiDAR sensor is represented by a pair $(\mathbf{p}, \mathbf{R}) \in \mathbb{R}^n \times \mathbb{R}^{n\times n}$, where $\mathbf{p}$ denotes a translation vector and $\mathbf{R}$ denotes a rotation matrix between some fixed frame of reference and a frame fixed to the LiDAR. The origin of the LiDAR frame is taken to be the origin of the rays used by the sensor to perform the distance measurements.
2) We denote the raw data provided by the LiDAR sensor by a set of m pairs $(\mathbf{u}^{(i)}, d^{(i)})$, where $i \in \{1, 2, \dots, m\}$, with $d^{(i)} \in \mathbb{R}_{\geq 0}$ the depth measured by the i-th ray, and $\mathbf{u}^{(i)} \in \mathbb{R}^n$ the unit vector pointing in the direction of this ray.
3) Instead of using the raw measurement pairs, we transform them into a point cloud in the local LiDAR coordinate frame. The resulting points are denoted by $\mathbf{r}^{(i)} \in \mathbb{R}^n$, where $j \in \{1, 2, \dots, m\}$, and are given by
$$\mathbf{r}^{(i)} = d^{(i)}\mathbf{u}^{(i)}. \tag{1}$$
4) Given a cloud of points $\{\mathbf{r}^{(i)}\}_{i=1}^m$, their convex hull is given by:
$$H = \left\{ x = \sum_{i=1}^{m} \lambda_i \mathbf{r}^{(i)} \,\middle|\, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0 \right\}. \tag{2}$$

III. Problem Statement

We consider the point cloud registration problem with LiDAR data applied to a self-localization task in a convex environment. Concretely, we are given a point cloud $\{\mathbf{r}_1^{(i)}\}_{i=1}^{m_1}$ resulting from a LiDAR measurement with another pose $(\mathbf{p}_1, \mathbf{R}_1)$, and a second point cloud $\{\mathbf{r}_2^{(i)}\}_{i=1}^{m_2}$ from some pose $(\mathbf{p}_2, \mathbf{R}_2)$. Our goal is to estimate the relative translation vector $\hat{\mathbf{p}}$ and rotation matrix $\hat{\mathbf{R}}$ relating the two poses, i.e., such that $\hat{\mathbf{R}}\mathbf{R}_1 = \mathbf{R}_2$ and $\hat{\mathbf{p}} + \mathbf{p}_1 = \mathbf{p}_2$.

IV. Approach and Guarantees

First we review some mathematical preliminaries, then we detail our approach, termed Provably Accurate Simple Transformation Alignment (Pasta).

A. First and Second Moments of a Set

Pasta uses the first and second moment of a set, which we define here for clarity. Given a bounded set $H \subseteq \mathbb{R}^n$ and a distribution μ on $\mathbb{R}^n$, the *first moment* of H is:
$$\mathbf{c} = \frac{\int_{\mathbf{x}\in H} \mathbf{x} \, \mathrm{d}\mu}{\int_{\mathbf{x}\in H} \mathrm{d}\mu}. \tag{3}$$
In addition, its second moment relative to some point $\mathbf{q}$ is:
$$\mathbf{\Sigma} = \frac{\int_{\mathbf{x}\in H} (\mathbf{x}-\mathbf{q})(\mathbf{x}-\mathbf{q})^T \, \mathrm{d}\mu}{\int_{\mathbf{x}\in H} \mathrm{d}\mu}. \tag{4}$$
In the rest of this work, second moments will always be computed with respect to the first moment, i.e., with $\mathbf{q} = \mathbf{c}$.

We can interpret the definitions (3) and (4) as the mean and covariance matrix, respectively, of a uniform probability distribution over the set H. In this work, we may use these terms interchangeably.

For a convex hull H constructed from (non-empty) point clouds, these quantities are well-defined. In fact, any convex hull constructed according to the definition (2) is a polytope, and hence H can be partitioned into a set of disjoint simplices[2]. Algorithms such as `Qhull` [14] partition a given polytope H precisely this manner. Partitioning the set H into simplices lets us compute the first and second moment of H as a weighted sum of the moments of the simplices. The latter can be efficiently computed using the coordinates of the vertices of the polytope (see [15]).

Our approach draws inspiration from methods for 2D image alignment based on principal component analysis [12], [13]. The intuition behind these methods is as follows: the eigenvectors of the covariance matrix of a cloud of points span the principal axes of the data set. If the point clouds are related by a simple rotation and translation, these axes undergo the same rotation. As a consequence, it is possible to reconstruct the rotation between two point clouds by simply matching the eigenvectors of their covariance matrices. Once the rotation is known, we can easily find the translation by matching the first moments of the data sets.

This idea does not directly apply to point clouds generated by a LiDAR sensor, as even without occlusions the points are not related by a pure rotation and translation. This issue arises because the point clouds from LiDAR measurements are not uniformly distributed over the environment when the sensor changes pose, shown more clearly in Figure 1. To avoid this problem, Pasta first generates the *convex hull* of the point cloud, and then computes *its* first and second moment, rather than using the points alone.

B. Environment

Our method involves an alignment step, where we match the second moment of the *convex hulls* of point clouds. To guarantee this step produces the correct result, we require the operating environment to be approximately visible from any location. In this way, we guarantee that the convex hulls reconstructed from measurements taken at different positions are comparable. This requirement is easily satisfied when the volume of the environment is convex. While small obstacles in the environment can change the point cloud, if they only block the view of a limited portion of the environment, the point cloud's convex hull can still closely resemble the unobstructed environment. We further explore this "robustness" to non-convexity property in Section V.

[2]Triangles are simplices in $\mathbb{R}^2$, and tetrahedrons are simplices in $\mathbb{R}^3$.

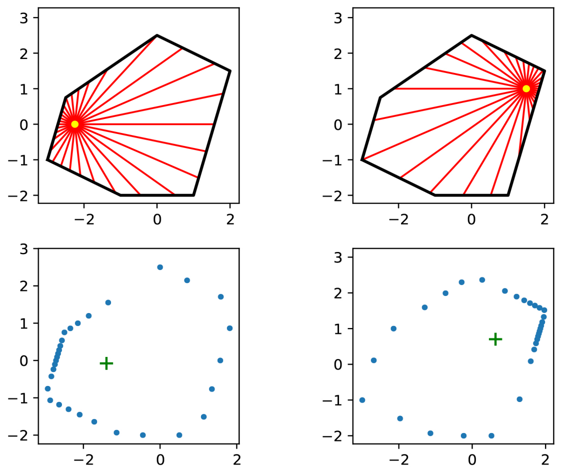

Fig. 1. Top row: LiDAR rays from different positions in a 2D environment. Bottom row: corresponding distance measurements converted into a point cloud. The average of the points (green cross) differs between the two measurements.

Our approach matches the eigenvectors of the computed covariance matrices of each convex hull, and to do so without any ambiguity the associated eigenvalues need to be sufficiently separated. This separation means that the environment must be sufficiently *asymmetric*. This is not a strong assumption, since symmetries in the environment *always* create ambiguities that are impossible to avoid without additional knowledge or features.

C. Relative Pose Estimate

Following (1), for the remainder of this paper, we assume that we are given two point clouds $\{\mathbf{r}_1^{(i)}\}_{i=1}^{m_1}$ and $\{\mathbf{r}_2^{(i)}\}_{i=1}^{m_2}$, corresponding to measurements at two sensor poses. Using these point clouds, we compute the corresponding convex hulls H_1 and H_2 using (2), and then compute the first and second moments, $\mathbf{c}_1, \mathbf{\Sigma}_1$ and $\mathbf{c}_2, \mathbf{\Sigma}_2$, respectively.

The estimated rotation matrix and translation vector relating the second pose of the sensor to the first, denoted $(\hat{\mathbf{p}}, \hat{\mathbf{R}})$, is computed as follows: We first construct the matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ whose columns are the unit eigenvectors of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$, respectively, ordered according to their corresponding eigenvalues[3]. For example, if $n = 3$ and the eigenvalues of some $\mathbf{\Sigma}$ are $\lambda_1 > \lambda_2 > \lambda_3$, the (increasingly ordered) matrix $\mathbf{V}$ will be:

$$\mathbf{V} = \begin{bmatrix}\mathbf{v}^1 & \mathbf{v}^2 & \mathbf{v}^3\end{bmatrix}, \tag{5}$$

where $\mathbf{v}^1, \mathbf{v}^2, \mathbf{v}^3$ are the corresponding unit eigenvectors.

Note that there is some ambiguity in the sign chosen for each of the eigenvectors, and that the choice of sign pattern needs to be consistent between the matrices $\mathbf{V}_1$ and $\mathbf{V}_2$ in order to match the vectors. We can achieve this consistency by exploiting the asymmetry in the environment the same way for both sets of measurements.

[3]The choice between increasing and decreasing order is arbitrary. Note also that covariance matrices are symmetric and positive semidefinite, and therefore always have real and nonnegative eigenvalues.

Explicitly, let $\text{sign}_H(\mathbf{v})$ denote the sign choice for the vector $\mathbf{v}$ given the convex hull H. Then we could compare the maximum and minimum values in the eigenvector directions:

$$\text{sign}_H(\mathbf{v}) = \begin{cases} +1, & \text{if } |\max_i \mathbf{v}^T\tilde{\mathbf{r}}^{(i)}| > |\min_i \mathbf{v}^T\tilde{\mathbf{r}}^{(i)}| \\ -1, & \text{else,} \end{cases} \tag{6}$$

$$\text{where } \tilde{\mathbf{r}}^{(i)} := \mathbf{r}^{(i)} - \mathbf{c}.$$

We could also compare the direction of positive skewness (i.e., the third moment):

$$\text{sign}_H(\mathbf{v}) = \begin{cases} +1, & \text{if } \frac{\int_{\mathbf{x}\in H} \left(\mathbf{v}^T(\mathbf{x}-\mathbf{c})\right)^3 \mathrm{d}\mu}{\int_{\mathbf{x}\in H} \mathrm{d}\mu} > 0 \\ -1, & \text{else.} \end{cases} \tag{7}$$

If we have a reference $\mathbf{\Sigma}_1$ from a nearby measurement, we could use the same signs:

$$\text{sign}_{H_1}(\mathbf{v}) = \begin{cases} +1, & \text{if } (\mathbf{v}_2^i)^T\mathbf{v}_1^i > 0 \\ -1, & \text{else.} \end{cases} \tag{8}$$

Once we construct $\mathbf{c}_1, \mathbf{c}_2, \mathbf{V}_1$, and $\mathbf{V}_2$, the pair $(\hat{\mathbf{p}}, \hat{\mathbf{R}})$ can be easily computed:

$$\begin{aligned} \hat{\mathbf{R}} &= \mathbf{V}_1\mathbf{V}_2^T \\ \hat{\mathbf{p}} &= \mathbf{c}_1 - \hat{\mathbf{R}}\mathbf{c}_2. \end{aligned} \tag{9}$$

D. Algorithm

Algorithm 1 summarizes the entire PASTA procedure that computes the translation vector and rotation matrix $(\hat{\mathbf{p}}, \hat{\mathbf{R}})$ relating two sensor poses from the respective point clouds generated from LiDAR measurements. We assume that we have available some basic functions:

- `hull`: takes a list of points, and returns a list of simplices forming a partition of the convex hull of the points.
- `vol`: computes the volume of a simplex.
- `mean`: computes the first moment of a simplex.
- `cov`: computes the second moment of a simplex with respect to a point.
- `eig`: computes the eigenvectors of a symmetric matrix, returning them in increasing eigenvalue order.
- `dir`: given an eigenvector and point cloud data, returns the eigenvector with the sign given by one of the conventions above.

Note that the functions `vol`, `mean`, `cov`, and `dir` can be implemented efficiently for simplices using only their vertex information (see [15]).

E. Guarantees

A crucial aspect of our algorithm is that it enjoys worst-case guarantees on the estimated translation and rotation between poses. To the best of the authors' knowledge, *no other method provides guarantees of this form.* As noted above, TEASER++ has a form of post-hoc guarantee, but relies on point-to-point correspondences that do not exist in LiDAR data, causing the method to fail.

Consider two point clouds H_1 and H_2 resulting from different measurements of the same environment related by the relative pose $(\mathbf{R}, \mathbf{p})$. Since the two scans may sample different points in the environment, the relationship between the convex hulls of the two scans is:

$$H_2 = \mathbf{R}\tilde{H}_1 + \mathbf{p}, \tag{10}$$

where $\tilde{H}_1$ is a perturbed version of H_1 induced by the difference in sampled points or noise. We measure the size of this perturbation with the parameter $\delta \in [0, 1]$, defined as:

$$\delta = \frac{\text{vol}(H_1 \cap \tilde{H}_1)}{\max\{\text{vol}(H_1), \text{vol}(\tilde{H}_1)\}}. \tag{11}$$

In words, a larger value of δ implies that the convex hull of the measured points H_1 greatly overlaps with the perfectly transformed convex hull $\tilde{H}_1$. In a convex environment, any mismatch is purely a result of noise and nonuniform LiDAR resolution. Hence, a noiseless LiDAR with infinite resolution would have $\delta = 1$. We measure the diameter $\rho(H)$ of a set H as follows:

$$\rho(H) = \min_{\mathbf{q}\in\mathbb{R}^n} \max_{\mathbf{x}\in H} \|\mathbf{x} - \mathbf{q}\|. \tag{12}$$

The next result gives an error guarantee for Alg. 1, PASTA.

Theorem 1: Let H_1 and H_2 be the convex hulls of point clouds from two measurement locations connected by the relative pose $(\mathbf{R}, \mathbf{p})$. Let $\tilde{H}_1$ be such that $H_2 = \mathbf{R}\tilde{H}_1 + \mathbf{p}$, and $(\hat{\mathbf{R}}, \hat{\mathbf{p}})$ be PASTA's output. Then:

$$\|\hat{\mathbf{p}} - \mathbf{p}\| \le \|\mathbf{c}_2\| \frac{e_R\sqrt{n}}{\min_{i,j} |\lambda_i - \lambda_j| - 2e_R} + e_p, \tag{13}$$

$$\|\hat{\mathbf{R}} - \mathbf{R}\| \le \frac{e_R\sqrt{n}}{\min_{i,j} |\lambda_i - \lambda_j| - 2e_R}, \tag{14}$$

where $\mathbf{c}_i$ is the first moment of H_i and:

$$e_p = 3(1-\delta)\rho(H_1 \cup \tilde{H}_1)$$
$$e_R = (25(1-\delta)^2 + 8(1-\delta))\rho^2(H_1 \cup \tilde{H}_1).$$

The proof of this result is in [15]. Theorem 1 states PASTA's errors are bounded by continuous functions of δ that both go to zero as δ approaches one. As discussed above, δ approaches one as the LiDAR increases its sample density and measurement accuracy. The bounds in (13) and (14) also depend monotonically on the size of the operating region, $\rho(H_1 \cup \tilde{H}_1)$. This dependence is natural, since increasing the size of the environment while keeping the LiDAR resolution fixed will cause more drastic variations in measurement density between poses, and therefore a larger potential mismatch between H_1 and $\tilde{H}_1$.

V. EXPERIMENTAL RESULTS

In this section we show that our alignment algorithm is effective in a real environment, using actual LiDAR measurements. In each experiment, we place an LiDAR sensor at different positions/orientations in an approximately convex environment, then estimate the relative poses between point clouds produced by the sensor measurements. For measuring the ground truth, we use an OptiTrack camera positioning system. In all experiments, the testing environment is approximately 6.0m$\times$3.5m, with an outline shown in Figure 2.

Algorithm 1 PASTA

```
    Input: ${\mathbf{r}_1^{(i)}}_{i=1}^{m_1}$, ${\mathbf{r}_2^{(i)}}_{i=1}^{m_2}$.
    Dependencies: hull, vol, mean, cov, eig, dir
    Output: $\hat{\mathbf{p}}, \hat{\mathbf{R}}$
 1: procedure HULLMEAN($H$)
 2:     $V \leftarrow 0$
 3:     $\mu \leftarrow 0$
 4:     for $simplex$ in $H$ do
 5:         $V \leftarrow V + \texttt{vol}(simplex)$
 6:         $\mu \leftarrow \mu + \texttt{mean}(simplex) \times \texttt{vol}(simplex)$
 7:     end for
 8:     return $\mu/V$
 9: end procedure
10:
11: procedure HULLCOV($H, \mathbf{c}$)
12:     $V \leftarrow 0$
13:     $\mathbf{\Sigma} \leftarrow 0$
14:     for $simplex$ in $H$ do
15:         $V \leftarrow V + \texttt{vol}(simplex)$
16:         $\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \texttt{cov}(simplex, \mathbf{c}) \times \texttt{vol}(simplex)$
17:     end for
18:     return $\mathbf{\Sigma}/V$
19: end procedure
20:
21: procedure RELATIVEPOSE(${\mathbf{r}_1^{(i)}}_{i=1}^{m_1}$, ${\mathbf{r}_2^{(j)}}_{j=1}^{m_2}$)
22:     $H_1, H_2 \leftarrow \texttt{hull}({\mathbf{r}_1^{(i)}}_{i=1}^{m_1}), \texttt{hull}({\mathbf{r}_2^{(j)}}_{j=1}^{m_2})$
23:     $\mathbf{c}_1, \mathbf{c}_2 \leftarrow \texttt{HullMean}(H_1), \texttt{HullMean}(H_2)$
24:     $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 \leftarrow \texttt{HullCov}(H_1, \mathbf{c}_1), \texttt{HullCov}(H_2, \mathbf{c}_2)$
25:     $\mathbf{V}_1, \mathbf{V}_2 \leftarrow \texttt{eig}(\mathbf{\Sigma}_1), \texttt{eig}(\mathbf{\Sigma}_2)$
26:     $\mathbf{V}_1, \mathbf{V}_2 \leftarrow \texttt{dir}(\mathbf{V}_1, {\mathbf{r}_1^{(i)}}_{i=1}^{m_1}), \texttt{dir}(\mathbf{V}_2, {\mathbf{r}_1^{(j)}}_{j=1}^{m_2})$
27:     $\hat{\mathbf{R}} \leftarrow \mathbf{V}_1\mathbf{V}_2^T$
28:     $\hat{p} \leftarrow \mathbf{c}_1 - \hat{\mathbf{R}}\mathbf{c}_2$
29:     return $\hat{\mathbf{p}}, \hat{\mathbf{R}}$
30: end procedure
```

A. Convex Environment

We produced our first data set by placing the LiDAR sensor at different positions and angles in a convex environment (see Fig. 2), recording LiDAR measurements for each, and then converting these measurements into a point cloud. In this experiment, we used an RPLIDAR-A2 sensor set to produce a distance measurement for each 1° angle increment over a full 360° circle. We discarded any failed distance measurement (corresponding to an `inf` distance) before generating the point cloud. Given the set of point clouds resulting from k different poses, we estimate the relative pose between every possible pair of point clouds $i, j \in \{1, 2, \ldots, k\}$, and record the corresponding error.

We compare PASTA with a state of the art ICP implementation from the Point Cloud Library [16]. Unlike PASTA, standard ICP is primarily used for local point cloud registration, and is typically unable to perform global alignment with low pose estimation error (see Fig. 3). With this understanding, we also investigate improving the output of PASTA by using

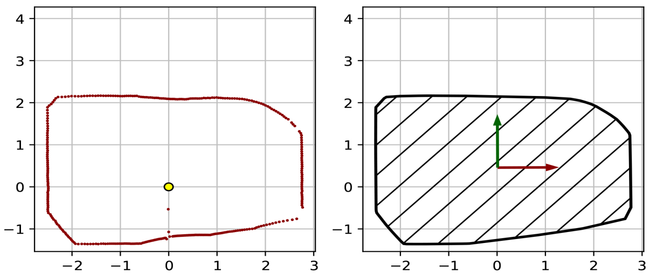

Fig. 2. Left: example of LiDAR point cloud in an approximately convex room with no obstacles. Right: convex hull of the point cloud, with the computed centroid, and the directions corresponding to the eigenvectors of its covariance matrix.

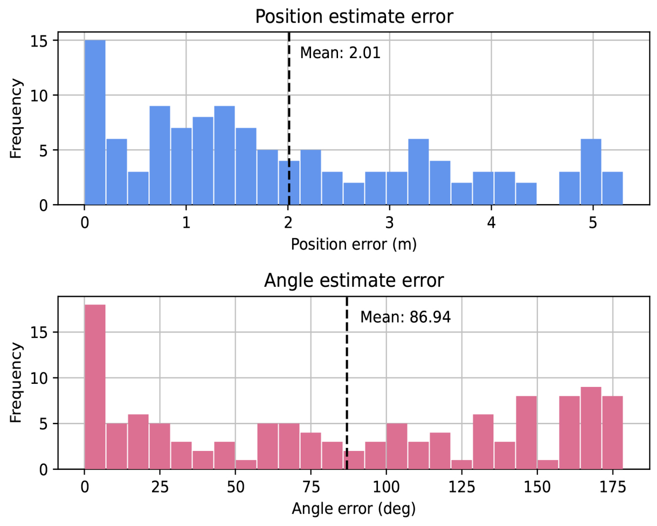

Fig. 3. The error distribution for ICP's global pose estimate.

its output as an initial guess for the ICP algorithm, denoted by PASTA-ICP in figures.

The results of this experiment can be seen in Fig. 4, with the dashed lines indicating the means of the errors. Clearly, running the ICP algorithm after PASTA does not bring a major improvement in estimation accuracy–in fact it causes slightly higher error in position, and only slightly lower angular error. The mean position and angle errors for PASTA are 0.07m, and 0.84° respectively, and the vast majority of errors lie below ~ 0.15m and $\sim 3^{\circ}$, with just a few outliers.

In our experiments we also observed that PASTA's accuracy is very high when comparing poses close to the center of the environment, and degrades with poses closer to the sides and corners. This observation matches the intuition that LiDAR clouds generated from such poses would have a drastically varying point density along the perimeter of the environment, and even computing the convex hull cannot recover from these effects.

B. Breaking Convexity

In the previous experiment, we showed that PASTA performs well in a convex environment where the theoretical guarantees hold. In many real environments, however, this assumption is not satisfied. Since PASTA relies on the *convex hull* of the measured point clouds, we expect that small obstacles blocking only a few LiDAR rays will only have minor impact on the resulting convex hull (see Figure 5). Moreover, the value of δ determining the tightness of PASTA's error bound in Theorem 1 only depends on the mismatch between these convex hulls, which may still be small. In these experiments, we verify this intuition and further investigate refining PASTA's estimates with ICP in this more challenging scenario.

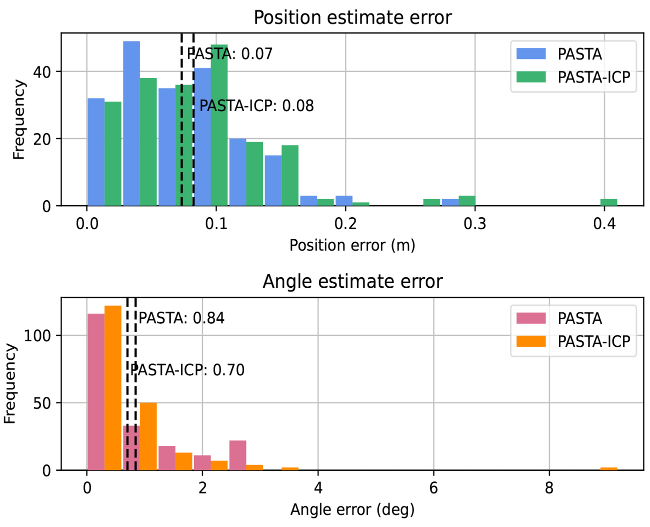

Fig. 4. Error distribution for the estimate of the relative poses between randomly picked LiDAR positions and angles. PASTA's error and ICP's error when using PASTA for its initial guess are almost identical.

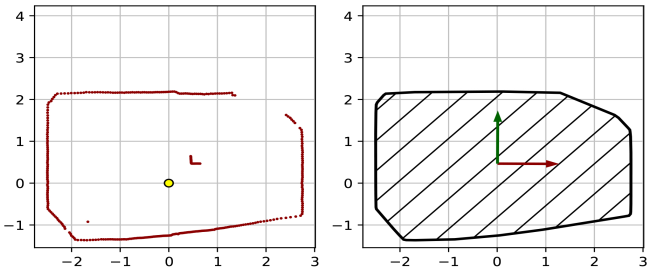

Fig. 5. Left: example of LiDAR point cloud in an approximately convex room with an obstacle. Notice that the top right part of the room is shadowed by the obstacle. Right: convex hull of the point cloud. The hull computation allows us to fill in the data missing because of the obstacle.

We record RPLIDAR-A2 measurements in k poses exactly as above, but we add an approximately 0.5m square obstacle to the center of the environment. The results of the experiment are in Fig. 6. As expected, the estimation accuracy gracefully degrades, with means of 0.11m and 1.64°. As before, the ICP algorithm does not refine the estimate given by PASTA significantly, and in fact the position error is worse than in the convex environment.

C. Incremental Localization

In the previous experiments we tested the accuracy of the pose estimation in a *global setting*, comparing point clouds generated from arbitrarily different poses. In many

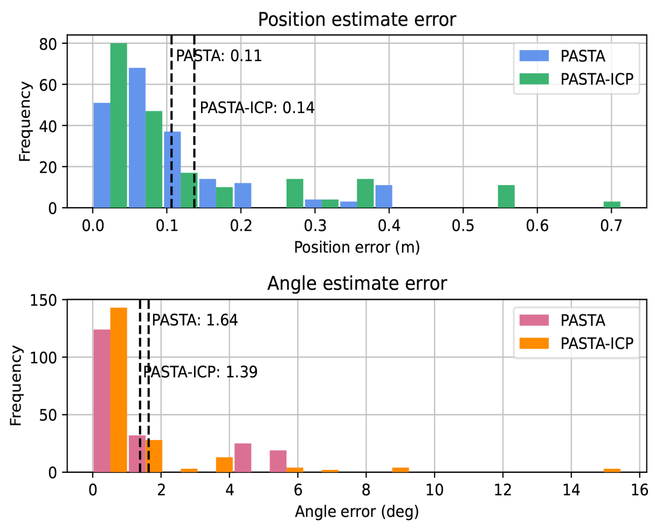

Fig. 6. Error distribution for the estimate of the relative poses between random LiDAR poses with an obstacle in the environment. PASTA's error and ICP's error using PASTA as initial guess are almost identical.

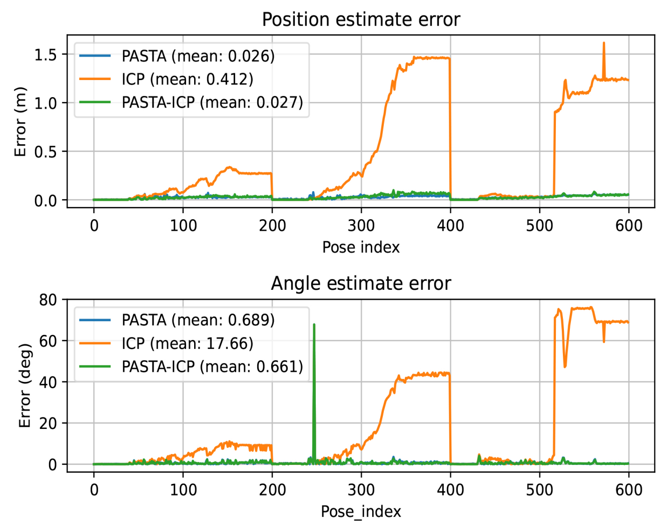

Fig. 7. Errors in the pose estimate between the current pose and first measured point cloud. As the current point cloud grows further from the reference, ICP's error increases while PASTA remains low.

applications, however, the point clouds being compared are from very nearby poses (typically arising from sequential measurements on a moving platform), which can imply better results from optimization-based methods like ICP.

To create this scenario, we slowly moved the LiDAR sensor in the vacant environment, while continuously taking measurements. In this experiment, we instead used an RPLIDAR A1 set to take one distance measurement at each $0.31°$ angle increment. We repeated this process three times, taking $\sim$200 measurements in each experiment. We first compared the results of ICP and PASTA when using the first point cloud as the reference for *all* subsequent measurements, shown in Fig. 7. Starting at zero, each set of 200 poses in Fig. 7 correspond to a single experiment. As expected, once the reference point cloud is sufficiently far from the current one, ICP's error rapidly increases.

We performed the same comparison using the *most recently measured* point cloud as a reference for both ICP and PASTA, and both methods' error reduced by an order of magnitude. However, in this comparison ICP and PASTA have average error within 0.005 m and 0.01 degrees of each other, meaning the difference between both methods is negligible even when conditions are more favorable to ICP.

VI. Conclusions

In this paper, we presented a novel point cloud registration method, termed PASTA, that is ideally suited for LiDAR measurements. The estimated rotation and translation from PASTA comes with a hard theoretical worst-case error guarantee, standing in stark contrast to existing methods. Finally, we showed that PASTA outperforms current state-of-the-art methods for point cloud alignment with real LiDAR data.

References

[1] R. W. Wolcott and R. M. Eustice, "Visual localization within LiDAR maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 176–183.

[2] ——, "Fast LiDAR localization using multiresolution gaussian mixture maps," in *2015 IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 2814–2821.

[3] M. Labbé and F. Michaud, "Rtab-map as an open-source LiDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[4] C. Debeunne and D. Vivet, "A review of visual-LiDAR fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, 2020.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, p. 1309–1332, 2016.

[6] L. Cheng, S. Chen, X. Liu, H. Xu, Y. Wu, M. Li, and Y. Chen, "Registration of Laser scanning point clouds: A review," *Sensors*, vol. 18, no. 5, p. 1641, 2018.

[7] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Josa a*, vol. 4, no. 4, pp. 629–642, 1987.

[8] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[9] P. Biber and W. Straßer, "The normal distributions transform: A new approach to Laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2003, pp. 2743–2748.

[10] Y. He, B. Liang, J. Yang, S. Li, and J. He, "An iterative closest points algorithm for registration of 3d Laser scanner point clouds with geometric features," *Sensors*, vol. 17, no. 8, p. 1862, 2017.

[11] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.

[12] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011, pp. 247–262.

[13] H. Z. U. Rehman and S. Lee, "Automatic image alignment using principal component analysis," *IEEE Access*, vol. 6, pp. 72 063–72 072, 2018.

[14] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[15] M. Marchi, J. Bunton, B. Gharesifard, and P. Tabuada. (2022, Feb) Technical note in support of the paper: PASTA. UCLA. [Online]. Available: https://gharesifard.github.io/pdfs/IROS_2022_NOTE.pdf

[16] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.