Winning Poster Paper

Communication Constrained Trajectory Alignment For Multi-Agent Inspection via Linear Programming

Joshua G. Mangelson, Ram Vasudevan, and Ryan M. Eustice

Abstract— **This paper reports on a system for estimating the alignment between robotic trajectories under constrained communications. Multi-agent collaborative inspection and navigation tasks depend on the ability to determine an alignment between robotic trajectories or maps. The properties of the underwater environment make determining such an alignment difficult because of extreme limitations on communication and the lack of absolute position measurements such as GPS. In this paper, we propose a method that takes advantage of convex relaxation techniques to determine an alignment between robotic trajectories based on sparse observations of a low-dimensional underlying feature space. We use a linear approximation of the l2-norm to approximately enforce that the estimated transformation is an element of SO(2). Because the relaxed optimization problem is linear, we can take advantage of existing convex optimization libraries, which do not require an initial estimate of relative pose. In addition, because the proposed method does not need to perform data association, we can align trajectories using low-dimensional feature vectors and can thus decrease the amount of data that must be transferred between agents by several orders of magnitude when compared to image feature descriptors such as SIFT and SURF. We evaluate the proposed method on simulated datasets and apply it to real-world data collected during autonomous ship hull inspection field trials.**

I. Introduction

Multi-agent underwater inspection and mapping tasks depend on the ability to determine an alignment between multiple robot maps or trajectories. This is challenging in underwater environments where global positioning systems are unavailable and where acoustic positioning systems require extensive setup and calibration [1]. Moreover, in fully submersed scenarios, communication is limited to a few bits per second making data transfer a significant system constraint [2].

Existing methods for estimating this alignment rely on the matching of discrete feature points observed by multiple robotic vehicles [3]. However, performing this data association requires that feature points transfer-ed between agents be uniquely identifiable. This is usually accomplished through the transfer of high dimensional feature descriptions that often surpass the throughput available in the underwater environment.

This paper proposes a method that efficiently estimates the rigid body transformation between reference and query robot trajectories based on a sparsely sampled underlying feature space. Because we formulate the problem as a convex optimization problem, our method avoids performing data association and decreases the amount of data that needs to be transferred between robotic vehicles by several orders of magnitude. In addition, because our formulation is convex, our proposed method is not dependent on initialization and does not require a prior estimate of the relative transforma-

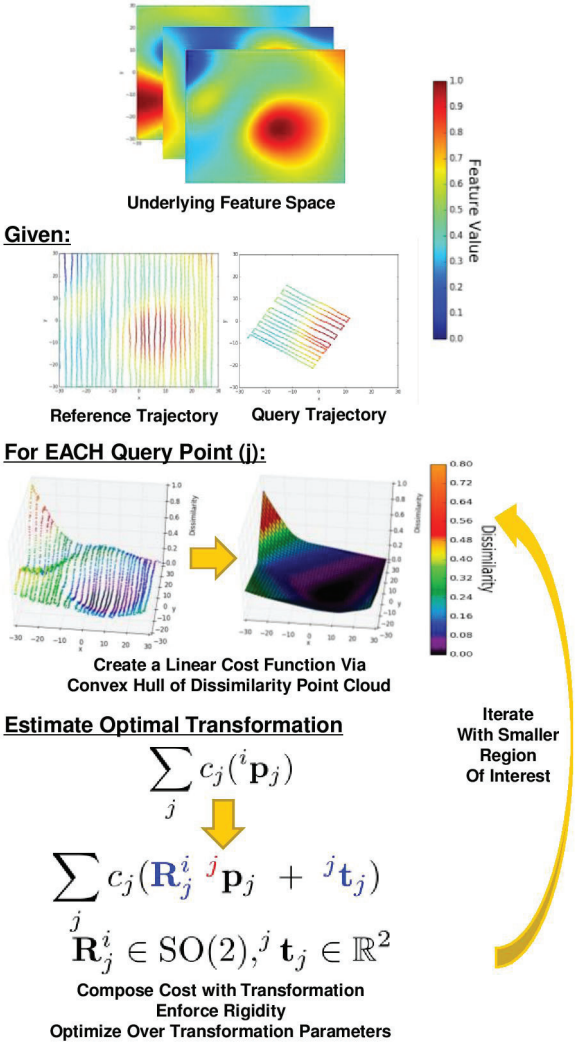


Fig. 1: An overview of the proposed trajectory alignment algorithm. Given sparse observations of an underlying feature space, we formulate a linear optimization problem that seeks to align a query trajectory with a reference trajectory. We do this by iteratively creating a linear cost that minimizes feature distance and optimizing over the transformation that minimizes that cost.

*This work was supported by the Office of Naval Research under award N00014-16-1-2102;

J. Mangelson, R. Vasudevan, and R. Eustice are with the Robotics Institute at the University of Michigan. R. Vasudevan is also with the Department of Mechanical Engineering and R. Eustice is with the Department of Naval Architecture and Marine Engineering. {mangelso, ramv, eustice}@umich.edu.

tion between trajectories. Finally, our method is parallelizable and takes advantage of existing commercial optimization libraries to increase the efficiency of the optimization process.

The contributions of this paper include the following:

1) The development of a system for alignment and localization of robot trajectories that:
   i) Relies only on low-dimensional feature observations.
   ii) Avoids performing data association.
   iii) Does not require an initial alignment estimate.
2) A novel linear approximation based method for approximate optimization over $\mathrm{SO}(2)$.
3) A parallelized implementation of the proposed method.

The remainder of this paper is organized as follows: In §II, we provide an overview of related areas of work. In §III we formalize the trajectory alignment problem. In §IV, we present a method that takes advantage of convex relaxation techniques to generate a linear cost function that is minimized when query feature points are placed near reference feature points with similar value. In §V, we present a novel method that uses linear programming to approximately optimize over $\mathrm{SO}(2)$. In §VI, we provide an outline of the full system and our released implementation. We evaluate the proposed algorithm on simulated datasets in §VII and apply it to multi-agent ship hull inspection in §VIII. Finally, we conclude in §IX.

II. Related Work

Multi-agent collaborative mapping has been heavily researched and a variety of methods have been developed for estimating the relative transformation between robot coordinate frames. Early methods assumed that vehicles were able to observe one another directly [4–6]. These methods relied on a single direct observation to determine the relative pose (position and orientation) of the two agents. Later, maximum likelihood based simultaneous localization and mapping (SLAM) methods enabled the use of multiple observations by estimating the most likely alignment and map given all the observed measurements [7–11]. These methods often relax the assumption that vehicles must be able to observe one another directly. Instead, features present in the environment observed by both vehicles are co-registered and used to relate the pose of the two vehicles. Most recent methods for determining multi-agent alignment are based on co-registering data in this way [3, 12–14]. While co-registering observed data works well in many cases, it depends on the ability to transmit large amounts of data between vehicles. This work focuses on developing methods that minimize the amount of data that must be transmitted between agents.

Our work is also related to research in the area of terrain based navigation [15–17]. In the underwater environment, this refers the use of a known bathymetric map of the seafloor to improve navigation estimates. These methods, however, are more focused on improving the estimate of a single robot's trajectory than on determining the alignment between multiple trajectories. In addition, these methods require an a prioi map of the environment, while our ultimate goal is to directly align the trajectories of two robotic vehicles that are simultaneously performing an inspection/mapping task in a potentially unknown environment.

There has also been significant recent interest in the SLAM and computer vision communities in developing estimation algorithms that leverage convex optimization techniques to avoid the need for an initial guess [14, 18–21]. This is especially useful in the underwater environment where global positioning system (GPS) is not available and global measurements of position can be hard to come by. In 2014, Li et al. [21] proposed a method that uses the convex hull of a set of dissimilarity points to estimate the affine transformation that must have occurred to transform a set of points observed in one image to a similar set of points observed in another. Their proposed method results in a linear (convex) cost function that can take advantage of existing optimization libraries and does not require an initialization. However, to apply their method to the trajectory alignment problem, we need to ensure that the estimated trajectory is rigid as opposed to affine.

Optimization over the group of rigid body transformations is generally non-convex making it hard to guarantee the true optimum. Recent works have investigated convex relaxation based methods for performing optimization over the group of rigid body transformations (the special euclidean group $\mathrm{SE}(d)$) [18, 19]. Specifically, these methods relax optimization over the set of valid rotation matrices $\mathrm{SO}(d)$ to optimization over the convex hull of $\mathrm{SO}(d)$. These methods work well in many cases. However, they fail to enforce that the estimated transformation be a valid rigid body transformation. The optimization problem used in our method also takes advantage of convex relaxation techniques, however, we use a linear approximation of the $\ell 2$-norm to add an additional set of constraints to the optimization that collectively enforce that the estimated transformation be approximately rigid. This results in more accurate transformation estimates than optimization over the convex hull of $\mathrm{SO}(d)$.

III. Problem Formulation

In this section, we outline the need for communication constrained trajectory alignment in underwater inspection and then formalize the trajectory alignment problem.

A. Trajectory Alignment w/o Data Association

In multi-agent inspection tasks, multiple vehicles navigate through the environment collecting information and estimating a map of the structure or scene they are inspecting. Some form of these local maps are then transmitted between vehicles allowing the agents to use the data collected by other vehicles for navigation, path planning, or global map generation. However, before an agent can use the data collected by another agent, it must first determine an alignment between its own local trajectory/map and the trajectory/map received from the other agent. Traditionally, this alignment is determined by matching locations in the environment observed by both agents [4–6].

However, communicating large amounts of data such as point clouds or high-dimensional image feature vectors between vehicles is often not practical in the underwater domain. While one approach is to limit data transfer by prioritizing data that is most likely to be useful, our approach is to cut out the transfer of this high-dimensional data completely. Instead, we take a descretized version of the robot trajectory and summarize information observed near each individual robot position using a small low-dimensional feature vector. We then align the robot trajectories by trying to find a rigid body transformation that places poses with similar descriptions near one another, without trying to match individual features. Taking this approach allows us to limit the data that must be transferred between vehicles to the discretized set of positions and the associated set of low-dimensional feature vectors.

B. Convex Trajectory Alignment

Formally, our goal is to align a query trajectory with a reference trajectory based only on low dimensional feature vectors describing the environment at each position visited by the two trajectories.

We denote the positions visited by the reference trajectory by $\{\mathbf{p}_1^a, \cdots, \mathbf{p}_{n_a}^a\}$ and the associated feature vectors by $\{\xi_1^a, \cdots, \xi_{n_a}^a\}$, indexed by i. Similarly, the query trajectory positions and feature vectors are denoted by $\{\mathbf{p}_1^b, \cdots, \mathbf{p}_{n_b}^b\}$ and $\{\xi_1^b, \cdots, \xi_{n_b}^b\}$, respectively, indexed by j. We then frame the trajectory alignment problem as an optimization that seeks to find a transformation that transforms positions in the coordinate frame of the query trajectory into the coordinate frame of the reference trajectory, such that when the points $\{\mathbf{p}_1^b, \cdots, \mathbf{p}_{n_b}^b\}$ are transformed they lie nearby points in $\{\mathbf{p}_1^a, \cdots, \mathbf{p}_{n_a}^a\}$ with similar feature values. Thus, the feature vectors ξ only need to describe the local environment as opposed to uniquely identify a specific point and can, as a result, have a much lower dimension.

Using a formulation similar to [21], we define a transformation function $T_j^{ab}(\mathbf{\Theta}) : \mathbb{R}^n \mapsto \mathbb{R}^d$ that maps the j-th query point, $\mathbf{p}_j^b$, to a position represented with respect to the reference trajectory coordinate frame. Specifically, we define $T_j^{ab}(\mathbf{\Theta})$ as

$$T_j^{ab}(\mathbf{\Theta}) = \mathbf{R}^{ab}\mathbf{p}_j^b + \mathbf{t}^{ab}, \tag{1}$$

where $\mathbf{\Theta} = (\mathbf{R}^{ab}, \mathbf{t}^{ab})$ are the parameters of the function. Together, $\mathbf{R}^{ab} \in \mathrm{SO}(d)$ and $\mathbf{t}^{ab} \in \mathbb{R}^d$ parameterize a global rigid body transformation relating the local coordinate frames of the two vehicles. The points $\mathbf{p}_j^b$ are fixed in the function T_j^{ab} and we thus define n_b transformation functions, one for each feature in the query trajectory.

We also define a function $c_j : \mathbb{R}^d \mapsto \mathbb{R}$ that takes a position represented in the reference trajectory coordinate frame, $\mathbf{p}^a \in \mathbb{R}^d$, and calculates the feature dissimilarity between ξ_j^b and the given position. As before, because there are n_b query feature points, we define n_b different dissimilarity functions $c_j, j = 1, \cdots, n_b$. §IV provides more detail on how these functions are defined.

With these definitions, we can formulate the final overall objective function as follows:

$$\underset{\substack{\mathbf{R}^{ab}\in \mathrm{SO}(d)\\ \mathbf{t}^{ab}\in\mathbb{R}^d}}{\text{minimize}} \sum_{j=1}^{n_b} c_j(T_j^{ab}(\mathbf{\Theta})) \tag{2}$$

where $c_j(T_j^{ab}(\mathbf{\Theta}))$ is the dissimilarity between the feature vector ξ_j^b and its new transformed position in the reference trajectory coordinate frame. Solving this optimization problem would allow us to find the transformation that minimizes the dissimilarity between query feature points and their associated positions in the reference trajectory coordinate frame.

Note, that while our formulation is based on that of [21], their method assumes the transformation is affine, while we restrict it to be an isometric (or rigid body [22]) transformation. This assumption on their part, results in the associated terms of their cost function being affine and the resulting optimization problem being convex. However, when dealing with physical transformations between coordinate frames, an affine transformation does not represent reality and a rigid body transformation must be used. This makes our resulting optimization problem non-convex and more difficult to solve. In §V we explain a novel method to approximately optimize over $\mathrm{SO}(2)$.

IV. Convex Transformation Estimation via the Lower Convex Hull

In this section, we discuss the formulation of the dissimilarity functions $c_j,\ j = 1, \cdots, n_b$ that measure the feature dissimilarity of the feature vector ξ_j^b and a position in the reference trajectory coordinate frame. We then compose the dissimilarity functions c_j with the transformation function (1) so that we can optimize over the transformation parameters as opposed to individual pose positions. There are two cases that we cover:

1) The 2D case where both maps lie in a 2D world($d = 2$)
2) The known depth case where both maps lie in the 3D world($d = 3$), but one coordinate is known.

We first cover the 2D case which follows directly from [21]. We then generalize this to the case with known depth.

A. Convex Dissimilarity Function Definition in 2D

Following [21], we denote the feature dissimilarity between ξ_i^a and ξ_j^b by C_{ij}^{ab}. If we use every possibly pairing, there are $n_a \times n_b$ of these values and they can be calculated before performing registration via an arbitrary dissimilarity function. However, because we want to find a transformation as opposed to a discrete matching, we create n_b continuous and convex feature dissimilarity functions $c_j,\ j = 1, \cdots, n_b$. Each c_j takes a position represented with respect to the reference trajectory coordinate frame and returns a predicted lower bound on the dissimilarity of that position with respect to the feature vector ξ_j^b. We derive this function by taking

the lower convex hull of the following point cloud:

$$\begin{bmatrix} x_1^a & y_1^a & C_{1j}^{ab} \\ x_2^a & y_2^a & C_{2j}^{ab} \\ \vdots & \vdots & \vdots \\ x_{n_a}^a & y_{n_a}^a & C_{n_a j}^{ab} \end{bmatrix}. \tag{3}$$

For a given feature vector ξ_j^b with $d = 2$, the point cloud (3) defines discrete points in a space that relates 2D positions in the reference trajectory coordinate frame to their associated dissimilarity value with respect to ξ_j^b. Because this space is 3D, we can calculate the convex hull of these points using an algorithm like [23]. The lower convex hull with respect to the dissimilarity value dimension is made up of a set of planes (facets) that represent lower bounds on the discrete dissimilarity values C_{ij}^{ab} for a given fixed j. These planes can be found by selecting the planes that have normal vectors with a negative component in the dimension corresponding to dissimilarity. Assuming there are M_j planes in the lower convex hull for feature ξ_j^b, we can define these planes using the equations $a_m x + b_m y + c_m C_m + d_m = 0$, for $m = 1, \cdots, M_j$, where x and y correspond to the position dimensions and C_m corresponds to the dissimilarity dimension in the point cloud (3). We then rearrange these equations to arrive at the plane functions $C_m = r_m x + s_m y + t_m$, for $m = 1, \cdots, M_j$ that calculate the predicted dissimilarity given a specified feature position. Finally, we can now define the feature dissimilarity function c_j as

$$c_j([x, y]^\top) = \max_m (r_m x + s_m y + t_m), \quad m = 1, \cdots, M_j. \tag{4}$$

This function is both continuous and convex and its minimization can be transformed into an equivalent linear program [24]:

$$\begin{aligned} \underset{u_j, x, y}{\text{minimize}} \quad & u_j \\ \text{subject to} \quad & r_m x + s_m y + t_m \leq u_j, \\ & m = 1, \cdots, M_j. \end{aligned} \tag{5}$$

Using the dissimilarity function (4) and the linear program (5) enables us to efficiently solve for an arbitrary 2D position $[x, y]^\top \in \mathbb{R}^2$ that minimizes the dissimilarity with respect to ξ_j^b for a given value of j. The next section describes how to compose $c_j(\cdot)$ with the transformation $T_j^{ab}(\mathbf{\Theta})$ which enables us to optimize over the parameters $\mathbf{\Theta} = (\mathbf{R}^{ab}, \mathbf{t}^{ab})$.

B. Composition with the Transformation Function

Our goal is to estimate the rigid body transformation between respective trajectories. As such, rather than estimate the individual locations of points, we estimate the parameters of the transformation, $\mathbf{\Theta}$, or more specifically the parameters $\mathbf{R}^{ab}$ and $\mathbf{t}^{ab}$. We do this by minimizing $c_j(T_j^{ab}(\mathbf{\Theta}))$, for all $j = 1, \cdots, n_b$, as opposed to $c_j([x, y]^\top)$ directly.

Remembering that $T_j^{ab}(\mathbf{\Theta}) \in \mathbb{R}^d$ represents the transformed coordinates of $\mathbf{p}_j^b$, we represent the function that calculates the first coordinate of $T_j^{ab}(\mathbf{\Theta})$ by $f_j(\mathbf{\Theta})$ and the second coordinate by $g_j(\mathbf{\Theta})$. Specifically, define

$$f_j(\mathbf{\Theta}) = \mathbf{R}^{ab(1)} \mathbf{p}_j^b + \mathbf{t}^{ab(1)} \tag{6}$$

and

$$g_j(\mathbf{\Theta}) = \mathbf{R}^{ab(2)} \mathbf{p}_j^b + \mathbf{t}^{ab(2)}, \tag{7}$$

where the notation $^{(i)}$ denotes the i-th row of the given matrix or vector and again $\mathbf{p}_j^b$ is fixed for both f_j and g_j.

Using this, we can rewrite (5) to be a minimization of $c_j(T_j^{ab}(\mathbf{\Theta}))$ over $\mathbf{\Theta}$ as

$$\begin{aligned} \underset{\substack{u_j \in \mathbb{R} \\ \mathbf{R}^{ab} \in \mathrm{SO}(2) \\ \mathbf{t}^{ab} \in \mathbb{R}^2}}{\text{minimize}} \quad & u_j \\ \text{subject to} \quad & r_m f_j(\mathbf{\Theta}) + s_m g_j(\mathbf{\Theta}) + t_m - u_j \leq 0, \\ & m = 1, \cdots, M_j. \end{aligned} \tag{8}$$

If the elements of the matrix $\mathbf{R}^{ab}$ are treated as individual elements in $\mathbf{\Theta}$, then both $f_j(\mathbf{\Theta})$ and $g_j(\mathbf{\Theta})$ are affine functions of $\mathbf{\Theta}$ and $r_m f_j(\mathbf{\Theta}) + s_m g_j(\mathbf{\Theta}) + t_m - u_j$ is also an affine function of $\mathbf{\Theta}$ and u_j. This was noted in [21]. However, in our case the resulting problem (8) is non-convex because of the implicit constraint that $\mathbf{R}^{ab} \in \mathrm{SO}(2)$. §V proposes a solution to this problem.

The next section explains how this can be generalized to three dimensions when depth is known.

C. The Known Depth Case

The prior section is defined with a planar world in mind. However, the real world exists in three dimensions. Although the prior section can be generalized to three dimensions, in the underwater domain we can take advantage of the fact that we can accurately measure depth and just estimate translation in the xy-plane and rotation about the z (vertical) axis. Under these assumptions, we can restrict T_j^{ab} to the following:

$$T_j^{ab}(\mathbf{\Theta}) = \begin{bmatrix} \mathbf{R}_z^{ab} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{p}_j^b + \begin{bmatrix} \mathbf{t}_z^{ab} \\ 0 \end{bmatrix}, \tag{9}$$

with $\mathbf{R}_z^{ab} \in \mathrm{SO}(2)$, $\mathbf{t}_z^{ab} \in \mathbb{R}^2$, and $\mathbf{p}_j^b \in \mathbb{R}^3$. This restriction simplifies the problem and enables us to work on $\mathrm{SO}(2)$ as opposed to $\mathrm{SO}(3)$ when relaxing the optimization problem.

In addition, if the feature value varies with depth, then we can limit the reference trajectory feature points that need to be paired with each query feature ξ_j^b to those with a depth within a threshold γ of the estimated depth of $\mathbf{p}_j^b$. This allows us to decrease the size of the point cloud (3) and tighten the lower bounds defined by the planes in the lower convex hull.

V. Ensuring the Transformation is Rigid

The ultimate optimization problem we want to solve, (2), is non-convex because of the implicit constraint that $\mathbf{R}^{ab}$ (or equivalently $\mathbf{R}_z^{ab}$) $\in \mathrm{SO}(d)$. This makes it difficult to ensure that the solution obtained is globally optimal without a good initialization. However, this constraint is essential because it ensures that the estimated transformation is rigid as opposed to affine. A general affine transformation allows scaling and distortions of the object in addition to rotation

and translation. Physical rigid body transformations on the other hand preserve distance between points and thus do not allow scaling or distortions [22].

A. *Definition of* $SO(2)$ *and* $\operatorname{conv} \mathrm{SO}(2)$

Formally, $\mathrm{SO}(d)$ is defined as follows:

$$\mathrm{SO}(d) = \left\{ \mathbf{R} \in \mathbb{R}^{d \times d} : \mathbf{R}^\top \mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \det \mathbf{R} = 1 \right\}. \tag{10}$$

In the two dimensional case, an alternative definition for (10) is:

$$\mathrm{SO}(2) = \left\{ \mathbf{R} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \in \mathbb{R}^{2\times 2} : c^2 + s^2 = 1 \right\}. \tag{11}$$

A variety of recent papers have investigated the use of the convex hull of $\mathrm{SO}(d)$ [18, 19]:

$$\operatorname{conv} \mathrm{SO}(2) = \left\{ \tilde{\mathbf{R}} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \in \mathbb{R}^{2\times 2} : \begin{bmatrix} 1+c & s \\ s & 1-c \end{bmatrix} \succeq \mathbf{0} \right\}. \tag{12}$$

These methods relax optimization over $\mathrm{SO}(d)$ to optimization over the smallest convex set of matrices containing it. In this case the set of valid rigid body transformations lies on the border of the convex hull. However, if the minimum of the cost function lies within that convex hull as opposed to outside it or on its border, then these methods still return a transformation that scales and/or distorts the transformed trajectory. While rounding to the nearest rigid body transformation is possible, if the estimated transformation is no where near valid then the rounded solution tends to be inaccurate.

Instead of using the convex hull of $\mathrm{SO}(2)$, we break the problem up into linear sub-problems, within which we can use a linear approximation of the $\ell 2$ norm [25] to enforce that $c^2+s^2 \approx 1$ and thus that the transformation be approximately rigid.

B. *Linear Approximation of* $\ell 2$

Celebi et al. [26] evaluate several potential linear approximations of the $\ell 2$-norm. According to their evaluation, the approximation presented by Barni et al. [25] has the lowest maximum error. This approximation states that given a vector $\mathbf{x} = [x_1, x_2, \cdots, x_n]^\top \in \mathbb{R}^n$,

$$||\mathbf{x}||_2 \approx \delta^* \sum_{i=1}^{n} \alpha_i^* x_{(i)}, \tag{13}$$

where $(x_{(1)}, x_{(2)}, \cdots, x_{(n)})$ is a permutation of $(|x_1|, |x_2|, \cdots, |x_n|)$ such that $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$, and δ, α_i are parameters optimally given by (See (20) and (21) in [25]):

$$\alpha_i^* = \sqrt{i} - \sqrt{i-1}, \quad \delta^* = \frac{2}{1 + \sqrt{\sum_{i=1}^{n} \alpha_i^{*2}}}. \tag{14}$$

In the two dimensional case, $\alpha_1^* = 1$, $\alpha_2^* = \sqrt{2} - 1$, and $\delta^* = \frac{2}{1+\sqrt{1+(\sqrt{2}-1)^2}} \approx 0.96044$. In addition, in the two dimensional case, the sorting of $(|x_1|, |x_2|)$ can be implemented via a maximization term, and (13) can be rewritten as

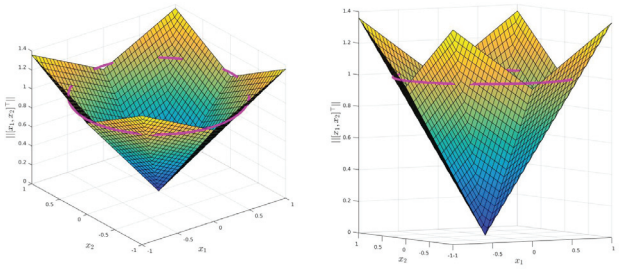


Fig. 2: The planes in these plots represent a piece-wise linear approximation of the $\ell 2$-norm of $\mathbf{x} = [x_1, x_2]^\top$ (16). The pink circle represents where the $\ell 2$-norm is one, $||\mathbf{x}||_2 = x_1^2 + x_2^2 = 1$.

$$\begin{aligned} ||\mathbf{x}||_2 \approx \delta\alpha_1 \max(|x_1|, |x_2|) \; + \\ \delta\alpha_2(|x_1| + |x_2| - \max(|x_1|, |x_2|). \end{aligned} \tag{15}$$

Implementing $\max$ and absolute value in convex optimization problems is not always possible. Instead of actually evaluating the $\max$ and absolute values, we can enumerate the possible values and rewrite (15) as:

$$\begin{aligned} ||\mathbf{x}||_2 \approx \max(&\delta(\alpha_1 x_1 + \alpha_2 x_2), \\ &\delta(\alpha_1(-x_1) + \alpha_2 x_2), \\ &\delta(\alpha_1(-x_1) + \alpha_2(-x_2)), \\ &\delta(\alpha_1 x_1 + \alpha_2(-x_2)), \\ &\delta(\alpha_1 x_2 + \alpha_2 x_1), \\ &\delta(\alpha_1(-x_2) + \alpha_2 x_1), \\ &\delta(\alpha_1(-x_2) + \alpha_2(-x_1)), \\ &\delta(\alpha_1 x_2 + \alpha_2(-x_1))) \end{aligned} \tag{16}$$

This function is shown plotted in Fig. 2.

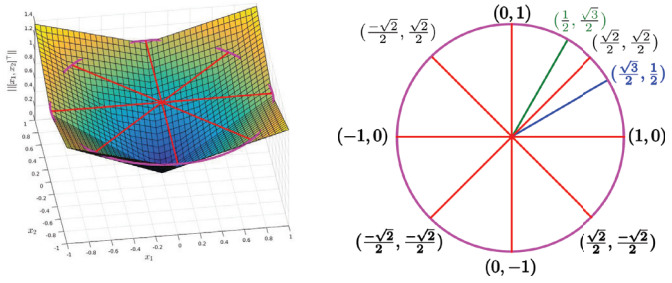
C. *Breaking the Optimization into Linear Sub-Problems*

Using (16) to enforce unit norm is still difficult because enforcing the $\max$ operation requires simultaneous minimization and maximization, instead, we split the problem into eight sub-problems in a way that enables us to ignore the $\max$ operation. Each of the eight terms in (16) correspond to a single plane in Fig. 2. Additionally, each term is maximal only when the signs and relative values of x_1 and x_2 meet certain conditions. These conditions correspond to sections of the unit circle (Fig. 3(b)) and can be enforced with linear constraints on x_1 and x_2.

Specifically, we can enforce that $\mathbf{x}$ lie within any given section by adding the corresponding choice of the following constraints to the optimization problem:

$$\begin{aligned} x_1 \leq x_2 \quad &\text{or} \quad x_2 \leq x_1 \\ x_1 \leq 0 \quad &\text{or} \quad x_1 \geq 0 \\ x_2 \leq 0 \quad &\text{or} \quad x_2 \geq 0. \end{aligned} \tag{17}$$

Within a given section, only a single plane is maximal and the approximation of $\ell 2$ becomes linear (Fig. 3(a)). Thus, we



(a) Division into Sections **(b)** Unit Circle

Fig. 3: By dividing the optimization into eight sub-problems, we no longer need to execute the max operation in (16) and each sub-problem then becomes convex. The red lines in (a) demonstrate where this division takes place. These divisions correspond to the unit circle (b) and the enumerated sign/max combinations in (16).

can enforce that our solution be close to unit norm by adding the following linear constraint to the problem:

$$\delta(\alpha_1 x_p + \alpha_2 x_q) = 1 \tag{18}$$

where x_p and x_q represent the appropriately selected elements of $\{x_1, -x_1, x_2, -x_2\}$, such that they match the respective maximal term in (16). The maximum error of this approximation (18) is related to, but likely slightly higher than ϵ_{max} as defined in (4) and Table 3 of [26].

Formulating the problem in this way makes each sub-problem a linear program, and so we can take advantage of existing efficient optimization libraries to solve each sub-problem to a global minimum [27]. In addition, because the sub-problems divide the space and optimize over the same linear cost function, the solution to the sub-problem with lowest optimal value is identical to what the solution would be if we were able to constrain (16) to be equal to 1. Finally, the sub-problems are independent and thus can be parallelized.

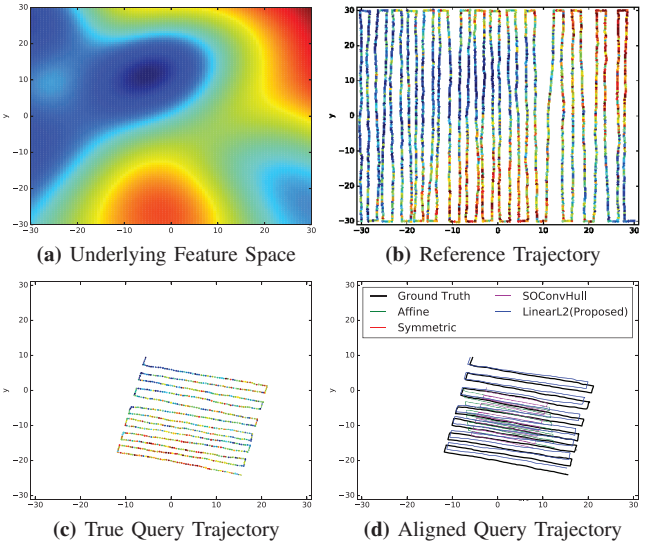
VI. The Full System

We are now able to create a general system for aligning robot trajectories.

A. The Final Optimization Problem

We can rewrite the optimization problem (2), by combining (8), (17), and (18).

$$\begin{aligned}
\underset{\substack{u_j \in \mathbb{R} \\ c,s \in \mathbb{R} \\ \mathbf{t}^{ab} \in \mathbb{R}^2}}{\text{minimize}} \quad & \sum_{j=1}^{n_b} u_j \\
\text{subject to} \quad & r_{m_j} f_j(\mathbf{\Theta}) + s_{m_j} g_j(\mathbf{\Theta}) + t_{m_j} - u_j \leq 0, \\
& \qquad m_j = 1, \cdots, M_j, \quad j = 1, \cdots, n_b \\
& \mathbf{R}^{ab} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \\
& c \leq s \quad \text{or} \quad s \leq c \\
& c \leq 0 \quad \text{or} \quad c \geq 0 \\
& s \leq 0 \quad \text{or} \quad s \geq 0 \\
& \delta(\alpha_1 x_p + \alpha_2 x_q) = 1,
\end{aligned} \tag{19}$$



(a) Underlying Feature Space **(b)** Reference Trajectory

(c) True Query Trajectory **(d)** Aligned Query Trajectory

Fig. 4: The feature space for a sample simulated trial is shown in (a). The reference and query trajectories are then generated as shown in (b) and (c). To simulate the fact that the relative transformation between trajectories is unknown, we then randomly rotate and translate the query trajectory and use the proposed and evaluated methods to estimate the inverse of the random transformation. The resulting trajectories are shown in (d).

where the inequality constraints and x_p and x_q are chosen according to the specific sub-problem.

Enumerating the possible combinations of these constraints results in eight sub-problems. Solving all eight sub-problems and then selecting the solution with lowest cost enables us to estimate a transformation to align the two trajectories without an initial estimate of alignment. In addition, this method enables us to avoid performing data association and thus perform alignment based on relatively indistinct, low-dimensional, feature observations. We then iterate over this process with consecutively smaller regions of interest, as explained in Section 4 of [21], to decrease the number of reference trajectory points used to create the cost function and thus increase accuracy.

B. Parallel and Feature Agnostic Implementation

We implemented the proposed system in c++. The released implementation uses MOSEK [27] and multiple threads to efficiently solve the linear sub-problems in parallel. The released code can be found at the following link: https://bitbucket.org/jmangelson/cte.

Our implementation and method are agnostic to the underlying feature space. When specifying a problem to be solved, the user provides feature point positions and a function that calculates the feature dissimilarity of a given pair of points. As such, the specifics of the feature space being used and the dissimilarity function are free to be chosen by the user.

In addition to our own proposed method for approximate optimization over $\mathrm{SO}(2)$, we also implemented functionality for estimating affine and symmetric transformations [21], as well as rigid body transformations via $\mathrm{conv}\,\mathrm{SO}(2)$ [19] for comparison.

TABLE I: Comparison of the proposed linear programming based method with other convex transformation estimation algorithms. Standard deviation shown is after removing outliers outside 1.5*IQR, IQR=Inter-Quartile-Range.

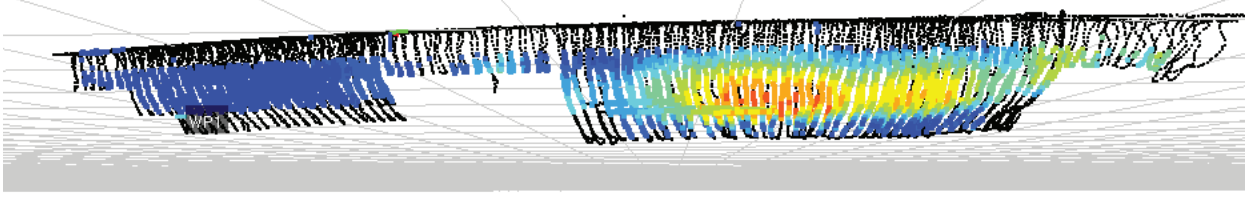
	Proposed	SOConvHull [19]	Symmetric [21]	Affine [21]
Rotation Median SE	**0.011**	0.039	0.042	0.338
Rotation Stddev SE	**0.017**	0.096	0.095	0.498
Trans. Median SE (m^2)	**4.796**	10.974	11.601	32.582
Trans. Stddev SE (m^2)	**6.770**	18.219	20.318	57.370
% Approx. Valid Rotations	**100.0**	37.0	25.5	1.0
Avg Runtime (s)	41.444	29.527	25.416	**23.926**

VII. Comparison With Existing Convex Optimization Methods

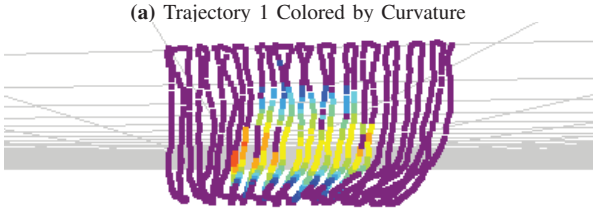
To provide quantitative results, we generated 200 synthetic worlds with smoothly varying feature spaces. We then simulated both a reference and a query lawn-mower trajectory within that space and randomly rotated and translated the query trajectory. Finally, we compared the alignment results from a variety of convex alignment methods formulated as in §IV but with different types of transformation constraints. Specifically, we compared our proposed linear programming-based approach to methods based on [21] that allowed either affine or symmetric transformations as well as a method that enforced that the estimated transformation lie within the convex hull of $SO(2)$ [19]. Fig. 4 shows a sample simulated trajectory and associated alignment results. The nominal trackline width for this dataset was two meters and the feature vector dimension was three.

Table I provides a summary of the comparison results. Our proposed method outperforms other methods in all metrics except for runtime. Note that the median translation squared error for the proposed method is 4.796 meters squared while the resolution of the tracklines simulated was two meters, meaning that the median error is only slightly higher than the resolution of the input data.

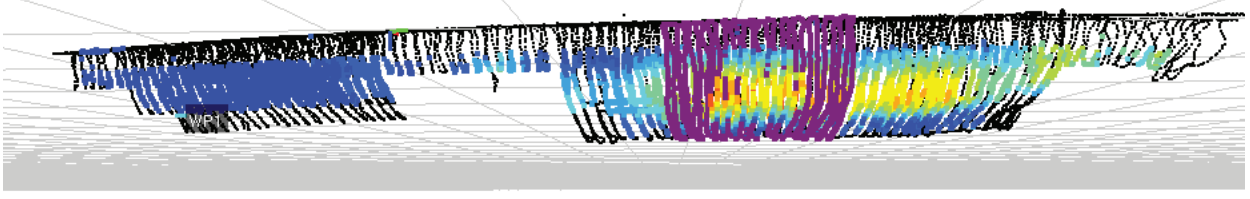
The current implementation uses four threads that independently solve two linear programs each via the optimization library Mosek [27]. However, the speed of the algorithm can be additionally improved by increasing the number of cores or by using a faster convex optimization library.



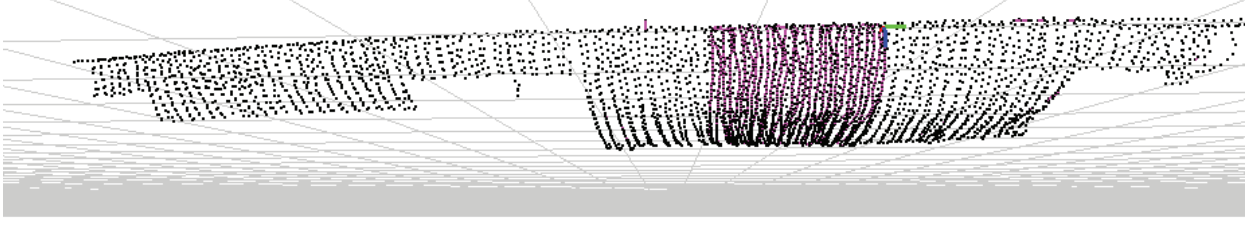
(a) Trajectory 1 Colored by Curvature



(b) Trajectory 2 Colored by Curvature



(c) Alignment Based on Hull Curvature Alone



(d) Ground Truth

Fig. 5: Two trajectories colored by estimated curvature are shown in (a) and (b). The estimated alignment of these two trajectories is shown in (c). The groundtruth alignment is shown in (d). This data was collected during field trials inspecting the hull of the USCGC Spencer in Boston, MA.

VIII. Application to Multi-Agent Autonomous Ship Hull Inspection

We also tested this method on real-world field data collected using a Hovering Autonomous Underwater Vehicle (HAUV) performing autonomous ship hull inspection. We used the sparse range returns of the Doppler velocity log (DVL) to estimate the curvature of the hull. Then, by treating the local curvature of the ship hull as a feature vector, we are able to re-localize to an earlier trajectory using only the DVL. Fig. 5 shows an alignment using this method.

Using the proposed alignment method limits the information that needs to be passed between vehicles to six fixed or floating-point values for each position visited by the agent (including three position coordinates and three curvature feature values), while the throughput needed to transfer image features between agents would be on the order of 1000-10000 fixed or floating-point values per position. Thus, the proposed method results in a decrease in required throughput of approximately 3-4 orders of magnitude.

The final accuracy of the alignment is dependent on the distinctiveness of the features being observed as well as the sampling resolution inherited from the reference trajectory. However, as formulated, the proposed method is independent of the specific feature and thus can be applied to whatever feature set is most distinctive in a given environment, subject to the communication bandwidth available. In addition, alignment can be further refined if desired using higher dimensional data such as imagery once an initial estimate of alignment has been obtained, thus minimizing the amount of image data that must be transferred between agents.

IX. Conclusion

In this paper, we propose a method for aligning robot trajectories that is linear and thus does not require initialization. In addition, the proposed method aligns trajectories without performing data association which decreases the amount of information that must be transferred between agents. We compared the existing method to similar convex methods that fail to enforce that the estimated transformation be rigid. We also applied the proposed algorithm to localization in the context of multi-agent autonomous ship hull inspection.

Future work would include: extending the proposed ideas to three dimensions and relaxing the assumption that the query trajectory be contained within the convex hull of the reference trajectory.

References

[1] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced perception, navigation and planning for autonomous in-water ship hull inspection," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1445–1464, 2012.

[2] J. M. Walls, A. G. Cunningham, and R. M. Eustice, "Cooperative localization by factor composition over a faulty low-bandwidth communication channel," in *Proc. IEEE Int. Conf. Robot. and Automation*, Seattle, WA, USA, May 2015, pp. 401–408.

[3] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert, "Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference," *IEEE Control Syst. Mag.*, vol. 36, no. 2, pp. 41–74, 2016.

[4] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri, "Rao-blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication," in *Proc. IEEE Int. Conf. Robot. and Automation*, Anchorage, Alaska, USA, May 2010, pp. 243–249.

[5] A. Howard, M. J. Matark, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, vol. 1. Lausanne, Switzerland: IEEE, Oct 2002, pp. 434–439.

[6] X. S. Zhou and S. I. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.* Beijing, China: IEEE, Oct 2006.

[7] L. A. Andersson and J. Nygards, "C-sam: Multi-robot slam using square root information smoothing," in *Proc. IEEE Int. Conf. Robot. and Automation*, Pasadena, CA, USA, May 2008, pp. 2798–2805.

[8] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *Proc. IEEE Int. Conf. Robot. and Automation*, Anchorage, Alaska, USA, May 2010, pp. 3185–3192.

[9] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, "Fully distributed scalable smoothing and mapping with robust multi-robot data association," in *Proc. IEEE Int. Conf. Robot. and Automation*, St. Paul, Minnesota, USA, May 2012, pp. 1093–1100.

[10] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based slam," *IEEE Trans. on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[11] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.

[12] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert, "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach," in *Proc. IEEE Int. Conf. Robot. and Automation*, Seattle, Washington, USA, May 2015, pp. 5807–5814.

[13] T. M. Bonanni, B. Della Corte, and G. Grisetti, "3-d map merging on pose graphs," *IEEE Robot. Autom. Letters*, vol. 2, no. 2, pp. 1031–1038, 2017.

[14] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan, "Pairwise consistent measurement set maximization for robust multi-robot map merging," in *Proc. IEEE Int. Conf. Robot. and Automation*, Brisbane, Australia, May 2018, pp. 1–8.

[15] J. Melo and A. Matos, "Survey on advances on terrain based navigation for autonomous underwater vehicles," *Ocean Engineering*, vol. 139, pp. 250–264, 2017.

[16] S. Carreno, P. Wilson, P. Ridao, and Y. Petillot, "A survey on terrain based navigation for AUVs," in *Proc. IEEE/MTS OCEANS Conf. Exhib.*, Sydney, Australia, May 2010, pp. 1–7.

[17] G. M. Reis, M. Fitzpatrick, J. Anderson, L. Bobadilla, and R. N. Smith, "Augmented terrain-based navigation to enable persistent autonomy for underwater vehicles," in *Proc. IEEE Int. Conf. Robot. Comput.*, Taichung, Taiwan, Apr 2017, pp. 292–298.

[18] D. M. Rosen, C. DuHadway, and J. J. Leonard, "A convex relaxation for approximate global optimization in simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robot. and Automation*, Seattle, Washington, USA, May 2015, pp. 5822–5829.

[19] J. Saunderson, P. A. Parrilo, and A. S. Willsky, "Semidefinite relaxations for optimization problems over rotation matrices," in *Proc. IEEE Conf. Decision Control*, Los Angeles, California, Dec 2014, pp. 160–166.

[20] L. Carlone, A. Censi, and F. Dellaert, "Selecting good measurements via 1 relaxation: A convex approach for robust estimation over graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, Chicago, Illinois, USA, September 2014, pp. 2667–2674.

[21] H. Li, X. Huang, J. Huang, and S. Zhang, "Feature matching with affine-function transformation models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 12, pp. 2407–2422, 2014.

[22] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[23] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996.

[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[25] M. Barni, F. Buti, F. Bartolini, and V. Cappellini, "A quasi-euclidean norm to speed up vector median filtering," *IEEE Trans. Image Process.*, vol. 9, no. 10, pp. 1704–1709, 2000.

[26] M. E. Celebi, F. Celiker, and H. A. Kingravi, "On euclidean norm approximations," *Pattern Recognition*, vol. 44, no. 2, pp. 278–283, 2011.

[27] M. ApS, *The MOSEK Fusion API for C++ 8.1.0.38*, 2018. [Online]. Available: https://docs.mosek.com/8.1/cxxfusion/index.html