Continuous-time Trajectory Estimation for Differentially Flat Systems

Jacob C. Johnson, Joshua G. Mangelson, Randal W. Beard

Abstract— **Continuous-time estimation using splines on Lie groups has been gaining traction in the literature due to the ability to incorporate high-frequency sensor data without introducing new optimization parameters. However, evaluating time derivatives and Jacobians of Lie group splines is computationally expensive, limiting their use mainly to offline applications. Motivated by the trajectory planning literature, we develop a new estimation technique that leverages the differential flatness property of many dynamic systems to define the spline in the system's flat output space, which is often Euclidean. Doing so has the benefit of providing a simple and effective way to include system inputs in the estimation process. We show an example of flatness-based estimation for the unicycle dynamic model. We then show that this new method can achieve similar performance as Lie group spline estimation with significantly less computation time, and validate its use in hardware using a differential-drive robot.**

I. Introduction

The problem of determining the trajectory traversed by a dynamic system using data acquired by available sensors is central to virtually all robotics applications. Traditionally, this is accomplished using some variant of the Extended Kalman Filter (EKF). In recent decades, however, the state estimation literature has seen a shift toward discrete batch maximum-a-posteriori (MAP) estimation techniques, such as bundle adjustment [1]. This shift has largely been driven by the finding that, in the case of visual simultaneous localization and mapping (SLAM), MAP methods offer more accuracy per unit of computation than EKF-based approaches [2]. However, unlike EKF-based methods, discrete batch MAP estimation suffers from the fact that each measurement must be tied to a parameter in the optimization, meaning that high-frequency sensors cannot easily be incorporated without introducing a high number of new optimization parameters, which can quickly cause the problem to become intractable. Methods such as inertial measurement unit (IMU) preintegration [3] can be introduced to allow for using such sensors, but these methods may not be trivial and must be carefully tailored to each individual sensor type.

To combat this issue, continuous-time estimation techniques such as spline estimation have been introduced [4]. Spline estimation techniques are able to incorporate high frequency measurement data while keeping the number of optimization variables low by optimizing a fixed number of spline control points rather than adding a new variable for each measurement. These methods have been used for estimation with a wide variety of sensors including IMUs [4], rolling shutter cameras [5], and event cameras [6].

In order to avoid rotational singularities, most of these methods use splines defined directly on Lie groups [5], [7]. However, computing time-derivatives and control point Jacobians of these Lie group splines is computationally expensive. The authors of [8] present a recursive formulation for computing the derivatives and Jacobians that is computationally linear in the order of the spline, and thus represents a significant step toward real-time applications. Unfortunately, even with the formulation in [8], Jacobian computation remains the bottleneck of Lie group spline optimization, limiting their use mainly to offline applications.

Splines are also used regularly in the path-planning literature [9], [10], and often leverage Differential Flatness (DF) [11], [12], a property shared by many robotic vehicles. If a system is Differentially Flat (DF) then all of its dynamic states and inputs can be expressed in terms of a lower-dimensional set of flat outputs and their time-derivatives (see Section III-A). This concept is useful in trajectory planning because any trajectory planned in the flat output space that has the correct degree of differentiability is dynamically feasible (provided input constraints are not violated), and the flat output space almost always has lower dimension than the state space.

To our knowledge the concept of DF has not been used in the context of continuous-time trajectory estimation in the literature, but we believe that it could offer similar benefits as those gained in trajectory planning. In this paper we propose a new continuous-time spline estimation technique for DF systems that leverages the DF property to compute residuals of measurements. The spline is defined in the low-dimensional flat output space rather than on the system's configuration manifold, often bypassing the need to use Lie group splines that have slow derivative and Jacobian evaluations.

The DF-based estimation framework also provides a simple and effective way to include the system inputs as "measurements" to improve the spline optimization without having to form motion priors through dynamics integration. This property could potentially be useful if other high frequency measurements are unavailable, or, if the model parameters are not well-known, to perform simultaneous trajectory and model parameter estimation (although we do not pursue this idea in this paper).

The contributions of the paper are as follows:

This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award No. IIP-1650547, along with significant contributions from C-UAS industry members.

J. C. Johnson is a graduate research assistant at Brigham Young University, Provo, UT, 84602 USA. (e-mail: jjohns99@byu.edu).

R. W. Beard and J. G. Mangelson are with the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84058 USA. (e-mail: beard@byu.edu, joshua_mangelson@byu.edu).

- A general framework for leveraging DF in continuous-time spline trajectory estimation,
- An example use case for the unicycle dynamic model with range, bearing, and IMU measurements, and its application to a differential-drive robot, and
- An accuracy and timing comparison against Lie group spline estimation on SE(2) with simulated data, both when inputs are included and when they are not, as well as validation of the proposed method in hardware with a differential-drive robot.

The rest of the paper is organized as follows. In Section II we present some preliminary information about Lie groups and B-splines. We introduce our general framework for DF-based spline estimation in Section III, followed by an example using the unicycle dynamic model in Section IV. We provide simulation results and comparisons in Section V and experimental results using a differential-drive robot in Section VI, and give some discussion and concluding remarks in Sections VII and VIII.

II. Preliminaries

A. Lie Groups

It is often necessary to use matrix Lie groups to accurately model robotic systems. We introduce the needed notation in this section. The matrix Lie groups $SO(2)$ and $SE(2)$ can be defined as

$$SO(2) \doteq \left\{\mathbf{R} \in \mathbb{R}^{2\times 2} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I},\, \det(\mathbf{R}) = 1\right\}, \quad (1)$$

$$SE(2) \doteq \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \mid \mathbf{R} \in \mathrm{SO}(2),\, \mathbf{t} \in \mathbb{R}^2 \right\}, \quad (2)$$

with respective Lie algebras $\mathfrak{so}(2)$ and $\mathfrak{se}(2)$ given by

$$\mathfrak{so}(2) \doteq \left\{ \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2} \mid \theta \in \mathbb{R} \right\}, \quad (3)$$

$$\mathfrak{se}(2) \doteq \left\{ \begin{bmatrix} \mathbf{X} & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3} \mid \mathbf{X} \in \mathfrak{so}(2),\, \rho \in \mathbb{R}^2 \right\}. \quad (4)$$

We denote the isomorphism between Euclidean space and the Lie algebra with the hat map $\theta^\wedge \doteq \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}$, $\theta \in \mathbb{R}$ for $\mathfrak{so}(2)$ and $\begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix}^\wedge \doteq \begin{bmatrix} \theta^\wedge & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix}$, $\theta \in \mathbb{R}$, $\boldsymbol{\rho} \in \mathbb{R}^2$ for $\mathfrak{se}(2)$. The inverse of the hat map is the vee map $\vee : \mathfrak{so}(2) \to \mathbb{R}$ and $\vee : \mathfrak{se}(2) \to \mathbb{R}^3$.

The exponential $\exp : \mathfrak{g} \to G$ is a map from a Lie algebra $\mathfrak{g}$ to its corresponding Lie group G. For matrix Lie groups, $\exp(\cdot)$ is simply the matrix exponential. For convenience, we define $\mathrm{Exp} : \mathfrak{g}^\vee \to G$, where $\mathfrak{g}^\vee = \{\mathbf{X}^\vee \mid \mathbf{X} \in \mathfrak{g}\}$, as the composition of the hat and exponential maps $\mathrm{Exp} \doteq \exp \circ \wedge$. For $SO(2)$, this admits the well-known closed form

$$\mathrm{Exp}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \quad (5)$$

The inverse of Exp is the logarithmic map $\mathrm{Log} : G \to \mathfrak{g}^\vee$. Inverting (5), we see that for $SO(2)$ this has the closed form

$$\mathrm{Log}(\mathbf{R}) = \tan^{-1}\left(\frac{r_{1,0}}{r_{0,0}}\right), \quad (6)$$

where $r_{i,j}$ is the i,j-th element of $\mathbf{R}$.

B. Euclidean B-splines

A k-th order Euclidean B-spline $\mathbf{p}(t) \in \mathbb{R}^d$ (where d is the spline dimension) with control points $\{\bar{\mathbf{p}}_m \in \mathbb{R}^d\}_{m=\{0,\cdots,M\}}$ and knot points $\{t_n \in \mathbb{R}\}_{n=\{0,\cdots,M+k+1\}}$ can be evaluated at time t using

$$\mathbf{p}(t) = \sum_{m=0}^{M} B_{m,k}(t)\bar{\mathbf{p}}_m, \quad (7)$$

where $B_{m,k}(t)$ are the B-spline basis functions of order k evaluated at time t [13]. Equation (7) can be expressed in the cumulative form as

$$\mathbf{p}(t) = \tilde{B}_{0,k}(t)\bar{\mathbf{p}}_0 + \sum_{m=1}^{M} \tilde{B}_{m,k}(t)(\bar{\mathbf{p}}_m - \bar{\mathbf{p}}_{m-1}), \quad (8)$$

where $\tilde{B}_{m,k}(t) = \sum_{s=m}^{M} B_{s,k}(t)$. Using the local support property of the basis functions, the nonzero terms of (8) can be removed to get the more compact form

$$\mathbf{p}(t) = \bar{\mathbf{p}}_{n-1} + \sum_{j=1}^{k-1} b_j(u(t))(\bar{\mathbf{p}}_{n+j-1} - \bar{\mathbf{p}}_{n+j-2}), \quad (9)$$

where the index n is chosen such that $t \in (t_n, t_{n+1}]$, $u(t) = \frac{t-t_n}{t_{n+1}-t_n} \in (0,1]$, and $b_j(u)$ is the j-th element of $\mathbf{b}(u) = \mathbf{C}_k\boldsymbol{\mu}(u)$, where $\mathbf{C}_k \in \mathbb{R}^{k\times k}$ encodes the continuity constraints of the spline and

$$\boldsymbol{\mu}(u) = \begin{bmatrix} 1 & u(t) & u(t)^2 & \cdots & u(t)^{k-1} \end{bmatrix}^\top. \quad (10)$$

Finally, Equation (9) can be unraveled and stacked to obtain the matrix representation

$$\mathbf{p}(t) = \boldsymbol{\Phi}(t)\bar{\mathbf{p}}, \quad (11)$$

where

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} (b_0 - b_1)\mathbf{I} & (b_1 - b_2)\mathbf{I} & \cdots & b_{k-1}\mathbf{I} \end{bmatrix} \quad (12)$$

(the dependence of b_j on $u(t)$ has been omitted for clarity) and $\bar{\mathbf{p}} = \begin{bmatrix} \bar{\mathbf{p}}_{n-1}^\top & \bar{\mathbf{p}}_n^\top & \cdots & \bar{\mathbf{p}}_{n+k-1}^\top \end{bmatrix}^\top$. Computing time-derivatives of Euclidean B-splines is trivial with (11),

$$\dot{\mathbf{p}}(t) = \dot{\boldsymbol{\Phi}}(t)\bar{\mathbf{p}}, \quad \ddot{\mathbf{p}}(t) = \ddot{\boldsymbol{\Phi}}(t)\bar{\mathbf{p}}, \quad \cdots \quad (13)$$

To learn more about the matrix representation of Euclidean B-splines, see [14].

III. Continuous-time Spline Estimation Using Differential Flatness

A. Differential Flatness

Definition 1: A dynamic system with states $\mathbf{x} \in \mathcal{X}$ and inputs $\mathbf{u} \in \mathcal{U}$ that evolves according to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (14)$$

is said to be DF with flat output $\mathbf{y} \in \mathcal{Y}$ if the states $\mathbf{x}$ and inputs $\mathbf{u}$ at any time t can be written as functions of the flat output $\mathbf{y}$ and its time-derivatives $\dot{\mathbf{y}}$, $\ddot{\mathbf{y}}$, $\dddot{\mathbf{y}}$, $\cdots$, i.e.

$$\mathbf{x}(t) = \mathbf{f}_x\left(\mathbf{y}(t), \dot{\mathbf{y}}(t), \ddot{\mathbf{y}}(t), \cdots\right), \quad (15)$$

$$\mathbf{u}(t) = \mathbf{f}_u\left(\mathbf{y}(t), \dot{\mathbf{y}}(t), \ddot{\mathbf{y}}(t), \cdots\right). \quad (16)$$

Note that the state space $\mathcal{X}$, input space $\mathcal{U}$, and output space $\mathcal{Y}$ need not be Euclidean.

Fig. 1. Measurement evaluation using B-splines and differential flatness.

B. Spline Estimation

If a dynamic system is DF, it is possible to represent the time-evolving state of the system as a trajectory in the flat output space $\mathcal{Y}$. This idea has often been used in the path planning literature in order to plan optimal, dynamically feasible trajectories [11], [12], but it can also be used to estimate continuous-time trajectories that fit a series of acquired measurements. In this paper we will represent trajectories as B-splines because the local property of their basis functions results in factor graphs that are relatively sparse (the trajectory at any time t is only affected by k consecutive control points). Note however that it is possible to use any form of trajectory basis functions provided the resulting trajectory meets the system's differentiability requirements.

We consider the case where a mobile robot with states $\mathbf{x} \in \mathcal{X}$ and inputs $\mathbf{u} \in \mathcal{U}$ with DF dynamics collects a set of measurements $\{\mathbf{z}_i\}_{i=\{0,\cdots,N\}}$ according to the measurement model

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}(t_i), \mathbf{u}(t_i)) + \boldsymbol{\eta}_i, \tag{17}$$

where t_i is the measurement acquisition time and $\boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i)$ is zero-mean Gaussian noise with covariance $\boldsymbol{\Sigma}_i$. Note that in this formulation it is possible for a measurement to be dependent on the input $\mathbf{u}(t_i)$.

Given a B-spline trajectory estimate $\hat{\mathbf{y}}(t) \in \mathcal{Y}$ with spline control points $\{\bar{\mathbf{y}}_m\}_{m=\{0,\cdots,M\}}$, where $\bar{\mathbf{y}}_m \in \mathcal{Y}$ [1], an expected measurement $\hat{\mathbf{z}}_i = \mathbf{h}_i(\hat{\mathbf{x}}(t_i), \hat{\mathbf{u}}(t_i))$ can be determined using the steps outlined in Figure 1. First, the spline is sampled at $t = t_i$ to determine $\hat{\mathbf{y}}(t_i)$, $\dot{\hat{\mathbf{y}}}(t_i)$, $\cdots$. These are then passed into the DF model to determine the state and input estimates $\hat{\mathbf{x}}(t_i)$, $\hat{\mathbf{u}}(t_i)$, which are then given to the measurement model to determine $\hat{\mathbf{z}}_i$. Our goal is to find a trajectory estimate $\hat{\mathbf{y}}(t),\ t \in [\min\{t_i\}, \max\{t_i\}]$ that maximizes the probability of the resulting state $\hat{\mathbf{x}}(t)$ and input $\hat{\mathbf{u}}(t)$ trajectory conditioned on the measurements $\mathbf{z}_i$. This can be done by optimally placing the spline control points $\bar{\mathbf{y}}_m$. The resulting MAP estimation problem is

$$\{\bar{\mathbf{y}}_m^*\}_{m=\{0,\cdots,M\}} = \underset{\{\bar{\mathbf{y}}_m\}}{\arg\max}\ p(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t) | \{\mathbf{z}_i\}). \tag{18}$$

Under the assumptions that no prior information is known about $\hat{\mathbf{x}}(t)$, the measurements are independent of one another, and the measurement noise is Gaussian, the problem in Equation (18) is equivalent to

$$\{\bar{\mathbf{y}}_m^*\}_{m=\{0,\cdots,M\}} = \underset{\{\bar{\mathbf{y}}_m\}}{\arg\min} \sum_{i=0}^{N} \left(\mathbf{z}_i - \mathbf{h}_i(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i)\right)^\top \boldsymbol{\Sigma}_i^{-1} \left(\mathbf{z}_i - \mathbf{h}_i(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i)\right), \tag{19}$$

[1] If $\mathcal{Y}$ is a Lie group, a Lie group spline such as that of [5] can be used.

where $\hat{\mathbf{x}}_i \doteq \hat{\mathbf{x}}(t_i)$ and similarly for $\hat{\mathbf{u}}_i$. This problem can be solved using Gauss-Newton or Levenberg-Marquardt optimization. To do so, it is necessary to linearize the measurement functions $\mathbf{h}_i(\cdot)$ about the current control point estimates $\bar{\mathbf{y}}_{m0}$ as

$$\mathbf{h}_i(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i) \approx \mathbf{h}_i(\bar{\mathbf{y}}_{m0}) + \left.\frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{y}}_m}\right|_{\bar{\mathbf{y}}_{m0}} \delta\bar{\mathbf{y}}_m, \tag{20}$$

where $\delta\bar{\mathbf{y}}_m$ is an incremental update to be applied to $\bar{\mathbf{y}}_m$ [2]. By repeatedly applying the chain rule, we find that

$$\begin{aligned}\frac{\partial \mathbf{h}_i}{\partial \bar{\mathbf{y}}_m} = &\frac{\partial \mathbf{h}_i}{\partial \hat{\mathbf{x}}}\left(\frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \bar{\mathbf{y}}_m} + \frac{\partial \hat{\mathbf{x}}}{\partial \dot{\hat{\mathbf{y}}}}\frac{\partial \dot{\hat{\mathbf{y}}}}{\partial \bar{\mathbf{y}}_m} + \cdots\right) \\ &+ \frac{\partial \mathbf{h}_i}{\partial \hat{\mathbf{u}}}\left(\frac{\partial \hat{\mathbf{u}}}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \bar{\mathbf{y}}_m} + \frac{\partial \hat{\mathbf{u}}}{\partial \dot{\hat{\mathbf{y}}}}\frac{\partial \dot{\hat{\mathbf{y}}}}{\partial \bar{\mathbf{y}}_m} + \cdots\right).\end{aligned} \tag{21}$$

Thus we need to evaluate the Jacobians of the DF model $\frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{y}}}$, $\frac{\partial \hat{\mathbf{x}}}{\partial \dot{\hat{\mathbf{y}}}}$, $\cdots$ and $\frac{\partial \hat{\mathbf{u}}}{\partial \hat{\mathbf{y}}}$, $\frac{\partial \hat{\mathbf{u}}}{\partial \dot{\hat{\mathbf{y}}}}$, $\cdots$, as well as the structural Jacobians of the spline $\frac{\partial \hat{\mathbf{y}}}{\partial \bar{\mathbf{y}}_m}$, $\frac{\partial \dot{\hat{\mathbf{y}}}}{\partial \bar{\mathbf{y}}_m}$, $\cdots$, at each iteration in order to minimize (19).

One of the major benefits of using DF to evaluate the measurement models (17) is that the spline is defined on the low-dimensional flat output space $\mathcal{Y}$, rather than on the full configuration manifold of the system. For many systems the output space is Euclidean, whereas the configuration manifold is a Lie group such as $SE(2)$ or $SE(3)$. Evaluating time-derivatives and structural Jacobians of splines on Lie groups is computationally expensive, even when using the recursive form derived in [8]. On the other hand, evaluating derivatives and structural Jacobians of Euclidean B-splines is trivial and can be done in constant time.

IV. Unicycle Model Example

A. Model

In this paper we will use a unicycle model to demonstrate continuous-time spline estimation using DF. The states of the unicycle model are described by the tuple $\mathbf{x} = \{\mathbf{T}_{\mathcal{B}}^{\mathcal{I}}, v, \omega\} \in SE(2) \times \mathbb{R} \times \mathbb{R}$, where $\mathcal{B}$ and $\mathcal{I}$ denote the body and inertial frames of the vehicle respectively,

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{I}} = \begin{bmatrix} \mathbf{R}_{\mathcal{B}}^{\mathcal{I}} & \mathbf{t}_{\mathcal{B}/\mathcal{I}}^{\mathcal{I}} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(2) \tag{22}$$

is the pose of the model represented by the transformation from frame $\mathcal{B}$ to frame $\mathcal{I}$, with $\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} \in SO(2)$ representing the rotation from $\mathcal{B}$ to $\mathcal{I}$ and $\mathbf{t}_{\mathcal{B}/\mathcal{I}}^{\mathcal{I}}$ representing the position of frame $\mathcal{B}$ with respect to frame $\mathcal{I}$ expressed in frame $\mathcal{I}$, $v \in \mathbb{R}$ is the forward speed of the vehicle, and ω is the angular speed (see Figure 2). If the inputs to the model are $\mathbf{u} = \begin{bmatrix} F & \tau \end{bmatrix}^\top \in \mathbb{R}^2$, where F is a forward directional force and τ is a moment applied about the vehicle's center of mass,

[2] In the case that $\mathcal{Y}$ is Euclidean, this is done with normal vector addition. If $\mathcal{Y}$ is a Lie group, this is done with $\bar{\mathbf{y}}_m \leftarrow \text{Exp}(\delta\bar{\mathbf{y}}_m)\bar{\mathbf{y}}_m$ (assuming left trivialization).

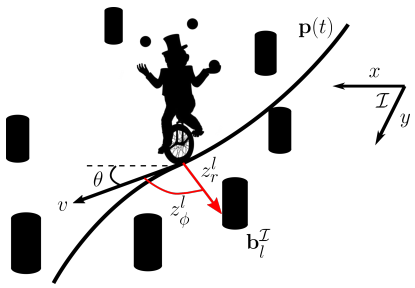

Fig. 2. Illustration of key variables in the unicycle model and the range and bearing measurements to known landmarks.

then the states $\mathbf{x}$ evolve according to

$$\dot{\mathbf{T}}_{\mathcal{B}}^{\mathcal{I}} = \mathbf{T}_{\mathcal{B}}^{\mathcal{I}} \begin{pmatrix} \begin{bmatrix} v & 0 & \omega \end{bmatrix}^\top \end{pmatrix}^\wedge \tag{23a}$$

$$\dot{v} = \frac{1}{m}(F - d_v v + \eta_F), \tag{23b}$$

$$\dot{\omega} = \frac{1}{J}(\tau - d_\omega \omega + \eta_\tau), \tag{23c}$$

where m and J are the mass and inertia of the vehicle, d_v and d_ω are linear and rotational drag coefficients, and $\eta_F \sim \mathcal{N}(0, \sigma_F^2)$ and $\eta_\tau \sim \mathcal{N}(0, \sigma_\tau^2)$ are Gaussian input noise. We will refer to the set $\Gamma = \{m, J, d_v, d_\omega\}$ as the parameters of the model. Note that the model in Equation (23) requires that the lateral speed always be zero.

We assume that the vehicle receives planar gyroscope and accelerometer measurements from an onboard inertial measurement unit (IMU) as well as range and bearing measurements to known inertial landmark positions $\mathbf{b}_l^{\mathcal{I}} \in \mathbb{R}^2$ (where l denotes the l-th landmark) as they enter a field of view defined by a maximum range $r_{\max}$ and bearing $\phi_{\max}$. The gyroscope and accelerometer measurement models are given by

$$z_g = \omega + \eta_g, \quad \mathbf{z}_a = \begin{bmatrix} \dot{v} \\ 0 \end{bmatrix} + \boldsymbol{\eta}_a, \tag{24}$$

where $\eta_g \sim \mathcal{N}(0, \sigma_g^2)$ and $\boldsymbol{\eta}_a \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 \mathbf{I})$ are zero-mean Gaussian measurement noise. The range and bearing measurement models are

$$z_r^l = \|\mathbf{b}_l^{\mathcal{B}}\| + \eta_r, \tag{25a}$$

$$z_\phi^l = \tan^{-1}\left(\frac{b_{ly}}{b_{lx}}\right) + \eta_\phi, \tag{25b}$$

where $\mathbf{b}_l^{\mathcal{B}} = \mathbf{T}_{\mathcal{B}}^{\mathcal{I}\,-1} \mathbf{b}_l^{\mathcal{I}}$ is the l-th landmark position transformed into the $\mathcal{B}$ frame, b_{lx} and b_{ly} are its components, and $\eta_r \sim \mathcal{N}(0, \sigma_r^2)$ and $\eta_\phi \sim \mathcal{N}(0, \sigma_\phi^2)$ are zero-mean Gaussian measurement noise.

If we have access to the inputs $\mathbf{u}$ that drive the system, then it is possible to use them as "measurements" to improve the trajectory estimate. The resulting input measurements are

$$z_F = F + \eta_F, \quad z_\tau = \tau + \eta_\tau. \tag{26}$$

B. Differential Flatness

Theorem 1: The unicycle model in Equation (23) is DF with flat output $\mathbf{y}(t) = \mathbf{p}(t) \in \mathbb{R}^2$, where $\mathbf{p}$ is the position of frame $\mathcal{B}$ w.r.t. frame $\mathcal{I}$ expressed in frame $\mathcal{I}$. The DF equations (15), (16) are given by

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{I}}(\mathbf{p}, \dot{\mathbf{p}}) = \begin{bmatrix} \mathrm{Exp}\left(\tan^{-1}\left(\frac{\dot{p}_y}{\dot{p}_x}\right)\right) & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{27}$$

$$v(\dot{\mathbf{p}}) = \|\dot{\mathbf{p}}\|, \quad \omega(\dot{\mathbf{p}}, \ddot{\mathbf{p}}) = -\frac{1}{\|\dot{\mathbf{p}}\|^2} \dot{\mathbf{p}}^\top 1^\wedge \ddot{\mathbf{p}}, \tag{28}$$

$$\dot{v}(\dot{\mathbf{p}}, \ddot{\mathbf{p}}) = \frac{1}{\|\dot{\mathbf{p}}\|} \dot{\mathbf{p}}^\top \ddot{\mathbf{p}}, \tag{29}$$

$$\dot{\omega}(\dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}) = \frac{2}{\|\dot{\mathbf{p}}\|^4} \left(\dot{\mathbf{p}}^\top \ddot{\mathbf{p}}\right) \left(\dot{\mathbf{p}}^\top 1^\wedge \ddot{\mathbf{p}}\right) - \frac{1}{\|\dot{\mathbf{p}}\|^2} \dot{\mathbf{p}}^\top 1^\wedge \dddot{\mathbf{p}}, \tag{30}$$

$$F(\dot{\mathbf{p}}, \ddot{\mathbf{p}}) = m\dot{v}(\dot{\mathbf{p}}, \ddot{\mathbf{p}}) + d_v v(\dot{\mathbf{p}}) \tag{31}$$

$$\tau(\dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}) = J\dot{\omega}(\dot{\mathbf{p}}, \ddot{\mathbf{p}}, \dddot{\mathbf{p}}) + d_\omega \omega(\dot{\mathbf{p}}, \ddot{\mathbf{p}}). \tag{32}$$

Because the unicycle model is DF, we can use a B-spline in the output space $\mathbb{R}^2$ to represent its state and input trajectory, and we can use the measurement models (24), (25), (26) and DF equations to optimally fit the spline to the acquired measurements. We note that if the input measurements (26) are not used, knowledge of the model parameters Γ is not required, and the only assumption made by using the DF model is that the lateral speed of the vehicle is always zero. However, if the input measurements are used, the additional information will improve the trajectory estimate.

C. A note on Jacobians

As was noted in Section III-B, the Jacobians of the DF equations as well as the spline structural Jacobians are required to perform optimization on the spline control points. The spline structural Jacobians are straightforward to compute from the matrix representation (11),

$$\frac{\partial \hat{\mathbf{p}}(t)}{\partial \bar{\mathbf{p}}_m} = \boldsymbol{\Phi}(t) \begin{bmatrix} \vdots \\ \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \\ \vdots \end{bmatrix}, \quad \frac{\partial \dot{\hat{\mathbf{p}}}(t)}{\partial \bar{\mathbf{p}}_m} = \dot{\boldsymbol{\Phi}}(t) \begin{bmatrix} \vdots \\ \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \\ \vdots \end{bmatrix}, \quad \cdots \tag{33}$$

(where $\mathbf{I}$ goes in the $(m - i + 1)$-th position) if $m \in [i - 1, i + k - 1]$ and are zero otherwise. The Jacobians of the DF equations and measurement models can also be computed analytically. However, we do not present them here due to space constraints.

V. Simulation Results

We tested DF-based spline estimation on the unicycle model of Section IV by simulating the dynamics (23) and measurements over a period of 20 seconds using the following parameters: $m = 1$ kg, $J = 1$ kg m^2, $d_v = 1$ kg s^{-1}, $d_\omega = 0.3$ kg m^2 s^{-1}, $\sigma_F = 0.01$, $\sigma_\tau = 0.01$, $\sigma_g = 0.005$, and $\sigma_a = 0.01$. Additionally, 500 landmarks were randomly placed in a $20\text{m} \times 20\text{m}$ area, and range and bearing measurements were simulated with $\sigma_r = 0.1$, $\sigma_\phi = 0.01$, $r_{\max} = 3$ m, and $\phi_{\max} = \pi/3$ rad. Input and IMU measurements were collected at 100 Hz and, when available, range and bearing measurements were collected at 10 Hz.

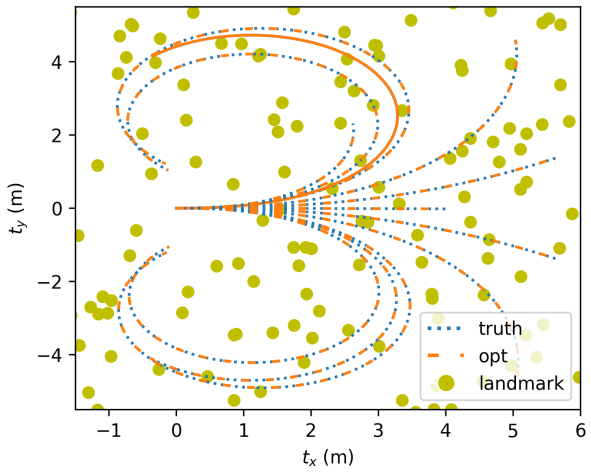

Fig. 3. Optimized splines compared to truth for several example trajectories. The states and inputs for the solid trajectory are shown in Figure 4.

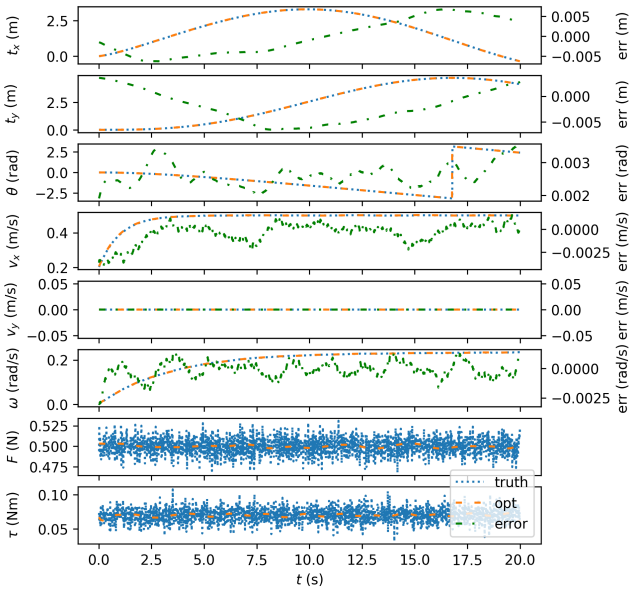

Fig. 4. Optimized trajectory states and inputs for the solid trajectory of Figure 3, compared to truth. Error is shown in green using the error scale on the right side of the plots.

The control points for a uniform B-spline in $\mathbb{R}^2$ with order $k = 6$ (a quintic spline)[3] and knot spacing of 1 second were optimized for several example trajectories using the Ceres Solver [15]. The input measurements were not used. In all cases the control points were initialized to lie at uniformly-spaced points along the x-axis. The optimized splines are shown compared to truth trajectories in Figure 3 (see the dotted lines), and the states and inputs for the solid line trajectory are shown in Figure 4. As can be seen from the plots, the optimized states fit the true states very well. The optimized inputs also match the true inputs well while smoothing out the noise, even when inputs were not included in the estimation.

[3]We found that $k = 6$ resulted in the lowest RMSE values while also allowing for low solve times. Lower spline orders could also be used.

Next, we compared the performance of DF-based spline estimation against spline estimation on $SE(2)$ using $k = 6$ and a knot spacing of 1 s. For $SE(2)$ spline estimation, we used a recursive method to analytically compute spline derivatives and structural Jacobians, similar to the method in [8]. We compared with splines directly on $SE(2)$ as well as splines on $SO(2) \times \mathbb{R}^2$ to see if decoupling position and rotation gives better performance [16]. Range and bearing measurements were used in all experiments, and we compared the difference between using IMU only, input only, and both IMU and input measurements for the DF-based method. Input measurements were not used for the $SE(2)$ and $SO(2) \times \mathbb{R}^2$ spline estimation, as these methods do not constrain the estimated lateral velocity, thus there is no direct way to evaluate the inputs. We seeded the optimization for both DF-based and $SE(2)$ estimation by perturbing the optimal $SE(2)$ spline control points with Gaussian noise, with a position standard deviation of 0.3 m and a rotation standard deviation of $\pi/6$ rad. Because rotation is determined by the velocity of the spline, the DF-based experiments were not affected by the rotation perturbation. All runs were done for a similar trajectory as that of Figure 4.

For each run, we computed pose, velocity (including ω), and input (only for DF-based estimation) root-mean-squared-error (RMSE) by sampling the optimized spline at 2000 evenly spaced times. The RMSE and run time values, averaged over 50 runs, are shown in Table I. In all cases optimization was done on a single thread. All methods had similar pose RMSE, but spline estimation on $SE(2)$ and $SO(2) \times \mathbb{R}^2$ had slightly higher velocity RMSE due to the fact that the lateral speed was not constrained to zero as it was in the DF-based methods. The $SE(2)$ method had a significantly longer average solve time than the DF methods, even though the number of required iterations was much lower. This is because computing time-derivatives and Jacobians of the $SE(2)$ spline is computationally expensive, even with the efficient recursive formulation of [8]. Computing the spline on $SO(2) \times \mathbb{R}^2$ rather than $SE(2)$ sped up the computation (while increasing the required number of iterations), but still took longer to solve than the DF-based methods. Additionally, we note that the DF method with both IMU and input measurements had lower pose, velocity, and input RMSE than the other DF-based methods, although it had a higher solve time due to the increased number of measurements. This suggests that incorporating input measurements could be worthwhile if the added computation time is acceptable. Inputs can also be used in place of IMU measurements if an IMU is unavailable and model parameters are well-known.

Finally, we note that the DF-based methods all required a higher number of iterations to converge than the $SE(2)$ and $SO(2) \times \mathbb{R}^2$ methods. We believe that this is because the DF equations in Section IV are highly nonlinear, causing the problem to be less convex. In rare occasions (fewer than 5%) this would even cause the optimization to converge to false minima (these occasions are not represented in Table I).

Spline Est. Type	Pose RMSE	Velocity RMSE	Input RMSE	Solve Time (s)	Iterations	Avg. Time per Iteration (s)
DF with IMU	0.07914	0.03440	0.11363	**0.03950**	10.20	**0.00418**
DF with input	0.07934	0.03292	0.11366	0.06195	14.90	0.00433
DF with IMU and input	**0.07801**	**0.02774**	**0.11283**	0.07557	13.44	0.00597
$SE(2)$ with IMU	0.07945	0.03852	NA	0.12425	**5.98**	0.02100
$SO(2) \times \mathbb{R}^2$ with IMU	0.08090	0.03839	NA	0.07891	8.06	0.01009

TABLE I

COMPARISON OF POSE, VELOCITY, AND INPUT RMSE, SOLVE TIME, AND ITERATIONS TO CONVERGENCE FOR EACH SPLINE ESTIMATION TECHNIQUE, AVERAGED OVER 50 RUNS.

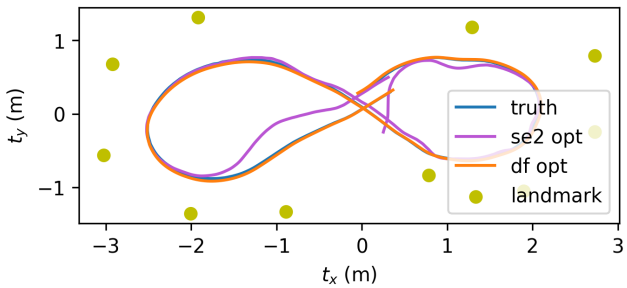

Fig. 5. Optimized DF spline (orange) and $SE(2)$ spline (purple) compared to data from a motion capture system (blue).

The $SE(2)$ and $SO(2) \times \mathbb{R}^2$ methods seem to be better conditioned and are able to converge to the true optimum for a wider variety of initial conditions than the DF-based methods can.

VI. EXPERIMENTAL RESULTS

In addition to the simulations presented in Section V, we validated DF-based spline estimation with experimental data collected from a differential-drive robot. Differential-drive robot dynamics follow a unicycle model similar to that presented in Section IV, but with right and left motor throttles δ_r, $\delta_l \in [-1, 1]$ as inputs rather than force and torque. Motor throttles can be mapped directly to forward and angular velocity using the linear relationship

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} rk/2 & rk/2 \\ rk/b & -rk/b \end{bmatrix} \begin{bmatrix} \delta_r \\ \delta_l \end{bmatrix}, \tag{34}$$

where r is the wheel radius, b is the distance between the wheels, and k is a coefficient that maps motor throttles to wheel angular rates. Using motor throttles as inputs, we replace the input measurement model (26) with

$$\mathbf{z}_{mt} = \begin{bmatrix} \delta_r \\ \delta_l \end{bmatrix} + \boldsymbol{\eta}_{mt}, \tag{35}$$

where $\boldsymbol{\eta}_{mt} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{mt})$ is zero mean Gaussian noise with covariance $\boldsymbol{\Sigma}_{mt} \in \mathbb{R}^{2\times 2}$.

We used the Kobuki Turtlebot2[4] differential-drive platform. The Turtlebot2 has a wheel radius of $r = 0.035\text{m}$, wheel base $b = 0.23\text{m}$, and coefficient $k = 23.81\text{rad/s}$. Input measurements were collected at a rate of 20Hz. The robot was equipped with an Intel RealSense D435i camera that was used to detect ArUco [17] fiducial landmarks in the environment and extract range and bearing measurements at 30Hz. In addition, planar inertial measurements were acquired from the RealSense IMU at a rate of 400Hz, where the non-planar components were ignored. The robot was driven in a room with an Optitrack[5] motion capture system to determine its true trajectory. For simplicity, we assumed that the camera, IMU, wheel, and motion capture coordinate frames all coincide. We additionally estimated constant IMU bias terms for both the accelerometer and the gyroscope.

The robot was driven in a figure-eight pattern while observing 10 different landmarks over a 28 second period. We tested DF-based spline estimation using range, bearing, IMU, and input measurements, as well as $SE(2)$ spline estimation using range, bearing, and IMU measurements. In both cases we used $k = 6$ with knot spacing 0.1 seconds. The DF spline control points were initialized in a line on the x-axis and the $SE(2)$ spline control points were initialized at identity. The estimation results are plotted against motion capture data in Figures 5 and 6. As can be seen, the DF-based spline was able to track the true motion of the robot very well, with pose, velocity, and input data matching the true values very closely despite only viewing one or two landmarks at a time. The pose RMSE for the trajectory was 0.261, with a solve time of 0.424 s. Notice that the estimator was not noticeably affected by the small non-zero lateral velocities of the robot. The $SE(2)$ spline estimation did not perform as well, with a pose RMSE of 0.317 and solve time of 1.386 s. We believe that the difference in performance is because the $SE(2)$ estimator did not have access to the inputs. There were several sections of the trajectory where one or fewer landmarks were visible, so global pose estimation was not possible and odometry had to be performed. Odometry using the low-noise inputs had lower drift than odometry using only the IMU.

VII. DISCUSSION

These results highlight two major advantages of the proposed method over splines on Lie groups. The first is that modeling the trajectory on the flat output space allows for more efficient estimation. This is mainly due to the fact that computing time-derivatives and Jacobians of splines on Lie groups requires many recursive matrix computations [8], whereas for Euclidean splines it only requires computing $\boldsymbol{\Phi}(t)$, $\dot{\boldsymbol{\Phi}}(t)$, etc. The second is the ability to use the system inputs directly in the estimation. It is possible to include system inputs indirectly with other trajectory representations through the use of dynamic motion priors. However, this requires integrating the dynamics between discrete points on

[4] turtlebot.com

[5] optitrack.com

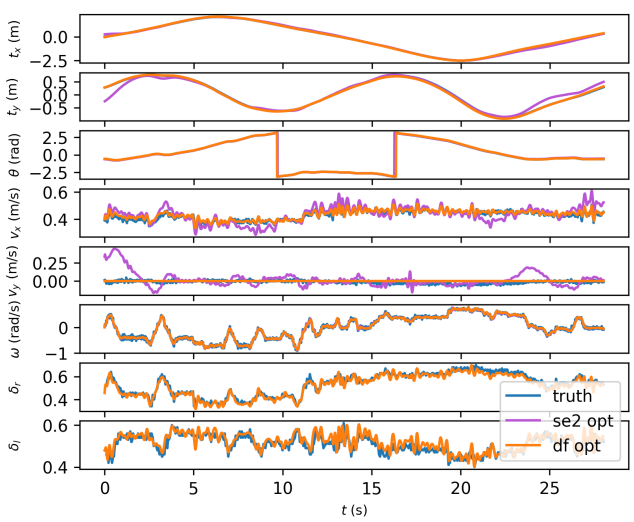

Fig. 6. Optimized DF and $SE(2)$ states and inputs for the trajectory shown in Figure 5, compared to data from a motion capture system. **Note**: the blue, purple, and orange traces are on top of one another in the θ and ω plots. **Note**: inputs are not shown for the $SE(2)$ spline because the DF constraint was not enforced, so inputs could not be computed.

the trajectory and does not ensure that the nonholonomic constraints of the system are enforced. An experimental comparison between DF-based estimation and the use of motion priors is beyond the scope of this paper.

Our results also highlight some disadvantages of DF-based estimation. First, the use of the nonlinear DF equations (15), (16) causes the problem to be less convex and more sensitive to initial conditions. This fact is evident in Table I, where a higher number of iterations were required for convergence. We also note that we did not include DF-based estimation without inputs in the experimental results because it required a large number of iterations to converge to a good solution. Convergence and avoidance of local minima can be improved by providing better initial conditions than were given in this experiment. Also, using splines in the flat output space requires that the motion of the true system agree with the DF model. While we see in Figure 6 that the DF-based estimator was not significantly affected by the small lateral velocities of the robot, it is not clear at this point how much disturbance can be handled reliably. We leave this question open for further investigation.

VIII. Conclusion

In this paper we presented a general framework for leveraging DF in continuous-time spline-based trajectory estimation, similar to how DF has been used in the trajectory planning literature. This method models the trajectory of the robot in the flat output space rather than on its configuration manifold. We showed how it can be used for unicycle model dynamics that receive planar IMU measurements and range and bearing measurements to known landmarks. Additionally, we showed that with this method, it is possible to include the inputs to the system in the estimation process without forming dynamic motion priors. We showed through simulations that DF-based spline estimation can provide similar accuracy as Lie group spline estimation with significantly less computation time, implying that our method may be better suited for real-time applications, and we validated the proposed method in hardware using a differential-drive robot.

We believe that this new estimation technique can be used for a wide variety of applications in robotics. The framework presented in Section III is general enough to be used for any DF system. In the future we plan to use it with more complicated systems, including quadrotors [11] and winged eVTOL aircraft [18] in offline scenarios, such as calibration and system identification, as well as in online scenarios using sliding window batch optimization.

References

[1] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[2] H. Strasdat, J. Montiel, and A. J. Davison, "Visual SLAM: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[3] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[4] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2088–2095.

[5] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras." in *BMVC*, vol. 2, no. 5, 2013, p. 8.

[6] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1425–1440, 2018.

[7] H. Sommer, J. R. Forbes, R. Siegwart, and P. Furgale, "Continuous-time estimation of attitude using B-splines on Lie groups," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 242–261, 2016.

[8] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative B-splines on Lie groups," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 145–11 153.

[9] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 167–172, 2010.

[10] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[11] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[12] C. P. Tang, P. T. Miller, V. N. Krovi, J.-C. Ryu, and S. K. Agrawal, "Differential-flatness-based planning and control of a wheeled mobile manipulator—theory and experiment," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 4, pp. 768–773, 2011.

[13] M. G. Cox, "The numerical evaluation of B-splines," *IMA Journal of Applied mathematics*, vol. 10, no. 2, pp. 134–149, 1972.

[14] K. Qin, "General matrix representations for B-splines," in *Proceedings Pacific Graphics' 98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No. 98EX208)*. IEEE, 1998, pp. 37–43.

[15] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[16] H. Ovren and P.-E. Forssen, "Trajectory representation and landmark projection for continuous-time structure from motion," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 686–701, 2019. [Online]. Available: https://doi.org/10.1177/0278364919839765

[17] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[18] J. B. Willis and R. W. Beard, "Nonlinear trajectory tracking control for winged eVTOL UAVs," in *2021 American Control Conference (ACC)*, 2021, pp. 1687–1692.