Article

The International Journal of Robotics Research
2015, Vol. 34(14) 1688–1710
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364915585860
ijr.sagepub.com
SAGE

# Continuous-time batch trajectory estimation using temporal basis functions

Paul Furgale[1], Chi Hay Tong[2], Timothy D. Barfoot[3] and Gabe Sibley[4]

**Abstract**

*Roboticists often formulate estimation problems in discrete time for the practical reason of keeping the state size tractable; however, the discrete-time approach does not scale well for use with high-rate sensors, such as inertial measurement units, rolling-shutter cameras, or sweeping laser imaging sensors. The difficulty lies in the fact that a pose variable is typically included for every time at which a measurement is acquired, rendering the dimension of the state impractically large for large numbers of measurements. This issue is exacerbated for the simultaneous localization and mapping problem, which further augments the state to include landmark variables. To address this tractability issue, we propose to move the full Maximum-a-Posteriori estimation problem into continuous time and use temporal basis functions to keep the state size manageable. We present a full probabilistic derivation of the continuous-time estimation problem, derive an estimator based on the assumption that the densities and processes involved are Gaussian and show how the coefficients of a relatively small number of basis functions can form the state to be estimated, making the solution efficient. Our derivation is presented in steps of increasingly specific assumptions, opening the door to the development of other novel continuous-time estimation algorithms through the application of different assumptions at any point. We use the simultaneous localization and mapping problem as our motivation throughout the paper, although the approach is not specific to this application. Results from two experiments are provided to validate the approach: (i) self-calibration involving a camera and a high-rate inertial measurement unit, and (ii) perspective localization with a rolling-shutter camera.*

**Keywords**

Sensor fusion, sensing and perception computer vision, localization, mobile and distributed robotics SLAM, mapping

## 1. Introduction

Recent decades have seen the probabilistic approach to robotics become the dominant paradigm, particularly for estimation problems (Thrun et al., 2001); however, in all but a few specialized cases, a discrete-time approach is taken. In particular, simultaneous localization and mapping (SLAM), a canonical mobile robotics problem (Durrant-Whyte and Bailey, 2006), has been formulated in discrete time since its inception; from the early papers exploring the probabilistic fundamentals (Smith and Cheeseman, 1986; Smith et al., 1990) to recent survey and tutorial papers (Bailey and Durrant-Whyte, 2006; Durrant-Whyte and Bailey, 2006; Thrun and Leonard, 2008), SLAM is introduced in terms of a discrete sequence of robot poses. Although the continuous-time estimation theory has existed for longer than SLAM (Kalman and Bucy, 1961; Jazwinski, 1970), researchers in robotics have generally avoided the use of continuous-time models; one notable exception is the amalgamation of a sequence of inertial measurements between two robot poses using continuous-time integration (Mirzaei and Roumeliotis, 2008; Kelly and Sukhatme, 2011). While SLAM is a prevalent estimation problem in robotics, the tools derived in this paper generalize well beyond the SLAM problem. Still, we will use SLAM as the backdrop for the developments in this work.

Roboticists typically use three common estimation tools (Thrun and Leonard, 2008): (i) Gaussian filtering (here we are grouping extended Kalman filtering, unscented Kalman filtering and information filtering), (ii) particle filtering and (iii) batch nonlinear optimization. For many robot/sensor

[1]ETH Zürich, Zürich, Switzerland
[2]University of Oxford, Oxford, UK
[3]University of Toronto, Toronto, Ontario, Canada
[4]University of Colorado, Boulder, CO, USA

**Corresponding author:**
Paul Furgale, ETH Zürich, Leonhardstrasse 21, 8049 Zürich, Switzerland.
Email: paul.furgale@mavt.ethz.ch

combinations, there is no drawback to using a discrete-time formulation for SLAM and any of the three paradigms are suitable; however, the existing solutions tend to be pushed to their limits under two problematic situations:

(a) when a sensor is capturing data at a very high rate, such as an inertial measurement unit (IMU) (Hol et al., 2010; Jones and Soatto, 2011; Kelly and Sukhatme, 2011; Leutenegger et al., 2013; Mirzaei and Roumeliotis, 2008);
(b) when a ranging or imaging sensor is capturing continuously while a robot is in motion (Bosse and Zlot, 2009; Ringaby and Forssén, 2012).

The first situation, high-rate data capture, is quite naturally handled by filtering algorithms (at least from a computational complexity perspective) as they only retain an estimate of the most recent robot pose in the state vector (following the Markov assumption). Particle filtering and batch methods break down in this situation for different reasons. Particle filters are well suited for low-dimensional state vectors but become impractical for six-degrees-of-freedom pose estimation, or state vectors that include velocity, acceleration or sensor biases (Gustafsson, 2010). On the other hand, batch optimization becomes intractable in the face of high-rate measurements because the error term associated with each measurement requires an estimate of the state at the measurement time; as the number of measurement times becomes large, so does the number of state variables that the algorithm must estimate.

Examples from the literature on dealing with the second problematic situation includes the work done by Bosse and Zlot (2009), who consider a continuous-spinning planar laser rangefinder, and Ringaby and Forssén (2012), who consider a rolling-shutter camera. In both situations, the solutions are necessarily based on batch methods as a window of recent robot poses must be iteratively adjusted to correct the alignment of features extracted from laser data (Bosse and Zlot, 2009) or images (Ringaby and Forssén, 2012). Furthermore, both papers use interpolation between poses to keep the size of the state vector tractable. The adjustment of several poses and the use of interpolation both preclude the use of single-timestep filtering algorithms to solve these problems. Consequently, no single approach is able to handle both of the cases listed above.

In recent years, there have been a number of capable SLAM algorithms developed that are based on batch estimation but are also suitable for online use (Dellaert and Kaess, 2006; Konolige et al., 2010; Sibley et al., 2010) in large, unstructured, three-dimensional environments. The flexibility of these algorithms to represent large maps, to delay loop-closing decisions and to process loop closures in constant time may be attributed to the underlying graph representation. Furthermore, Strasdat et al. (2010) show that, when compared to filtering techniques, batch estimation provides greater accuracy per unit of computational effort and avoids the accumulation of linearization errors by periodically relinearizing about the current estimate.

To reconcile the interest in sensor fusion between visual and inertial sensors and to provide a theoretical framework on which to base new algorithms for mapping with continuous-capture imaging sensors, we seek an algorithm that exhibits the desirable characteristics of batch estimation, but is able to handle both problematic situations described above. We propose to do this by lifting the estimation framework into continuous time.

The contributions of this paper are as follows.

1. We introduce a novel continuous-time-state, discrete-time-measurement batch estimation methodology.
2. We provide a derivation of the full continuous time SLAM problem (CT-SLAM) and compare it directly to the well-known discrete-time derivation. To the best of our knowledge, this is the first presentation of this derivation.
3. We detail a realization of CT-SLAM using a weighted sum of known basis functions to represent the robot state.
4. We present the full derivation of how to implement CT-SLAM using B-spline basis functions (this implementation is available as part of the Kalibr calibration toolbox: `https://github.com/ethz-asl/kalibr`).
5. We evaluate the solution on two problems that have recently received a lot of attention:
   (a) the calibration of a camera and an IMU; and
   (b) perspective localization of a rolling-shutter camera.

This paper is a major extension of work first presented by Furgale et al. (2012). The details of the theoretical derivations have been greatly extended and the rolling-shutter camera experiment is entirely new.

We have intentionally divided our derivation into steps representing increasingly specific assumptions, starting with the manipulation of the full posterior in Section 3.1, assuming Gaussian densities/processes for the prior, measurement model and continuous-time motion model in Section 3.2, deriving the objective function for a Maximum-a-Posteriori (MAP) estimator in Section 3.3, describing a realization of this estimator in Section 3.4 by using a weighted sum of basis functions to represent the robot state and finally outlining our use of a matrix formulation for B-splines in Section 4. This is followed by our extension to represent six-degrees-of-freedom transformations in Section 5, which is then used within the experiments in Sections 6 and 7.

Our hope is that, by dividing the derivation into steps, we encourage other researchers to devise novel solutions to the continuous-time batch-estimation problem by making different assumptions at any point.

**Table 1.** A comparison of the characteristics of various continuous-time state estimation methods, Part I: Work pre-dating our 2012 conference paper introducing the approach in this paper (Furgale et al., 2012).

| Paper(s) | State representation | Sensors used | Estimation method | Application |
|---|---|---|---|---|
| Jung and Taylor (2001) | Discrete poses and spline fit | Omnidirectional camera and IMU | Bundle adjustment | Visual odometry |
| Mirzaei and Roumeliotis (2008) | Numerical integration | Monocular camera and IMU | Iterated Extended Kalman Filter | Extrinsic calibration |
| Ait-Aider and Berry (2009) | Linear interpolation | Stereo rolling-shutter cameras | Gauss–Newton | Reconstruction and velocity estimation |
| Bosse and Zlot (2009); Zlot and Bosse (2012); Bosse et al. (2012) | Linear interpolation | Laser and IMU | Gauss–Newton | SLAM |
| Klein and Murray (2009) | Linear interpolation | Monocular rolling-shutter camera | Bundle adjustment | SLAM |
| Bibby and Reid (2010) | Cubic spline | Radar | Gauss–Newton | SLAM |
| Hol et al. (2010) | Numerical integration | Monocular camera and IMU | Extended Kalman Filter | Extrinsic calibration |
| Kelly and Sukhatme (2010) | Numerical integration | Monocular camera and IMU | Iterative Closest Point. | Time delay calibration |
| Fleps et al. (2011) | Cubic spline | Monocular camera and IMU | Sequential Quadratic Programming and Broyden-Fletcher-Goldfarb-Shanno nonlinear optimizati on algorithm | Extrinsic calibration |
| Karpenko et al. (2011) | Piecewise linear interpolation | Monocular rolling-shutter camera and gyroscope | Bundle adjustment | Video stabilization |
| Kelly and Sukhatme (2011) | Numerical integration | Monocular camera and IMU | Unscented Kalman Filter. | Extrinsic calibration and SLAM |
| Mair et al. (2011) | Numerical integration | Monocular camera and IMU | SQP | Rotational and temporal calibration |
| Ringaby and Forssén (2012); Hedborg et al. (2012) | Linear interpolation | Monocular rolling-shutter camera | Gauss–Newton | Rectification and SLAM |

IMU: inertial measurement unit; SLAM: simultaneous localization and mapping; SQP: sequential quadratic programming.

## 2. Related work

Table 1 provides a list of related papers that pre-date Furgale et al. (2012). Many prior works using continuous functions to represent robot states applied numerical integration to propagate states in a filter between measurement times. This approach was employed for calibrating monocular camera and IMU systems (Hol et al., 2010; Mair et al., 2011; Mirzaei and Roumeliotis, 2008; Kelly and Sukhatme, 2010) and SLAM (Kelly and Sukhatme, 2011). Other works employed the related idea of interpolation. Jung and Taylor (2001) presented a two-stage method of estimating a camera's trajectory with the help of an IMU by first estimating the orientation of the camera and then fitting a spline to the image and accelerometer data to recover the translation. This method was evaluated by Nützi et al. (2011), but was found to be unsuitable for online applications.

The difficult problem of estimating the trajectory of a spinning scanning laser rangefinder under continuous motion without using other sensors was addressed by Bosse and Zlot (2009). The algorithm presented estimates for a linearly interpolated trajectory by iteratively refining both the trajectory and the data associations. They impose a set of heuristic constraints to ensure the trajectory remains smooth a heuristic version of the continuous-time motion model is presented in Section 3.2. Similar approaches were applied for rolling-shutter cameras where interpolation between camera poses was used to correct for image distortion (Ait-Aider and Berry, 2009; Karpenko et al., 2011; Ringaby and Forssén, 2012) and motion estimation (Hedborg et al., 2012; Klein and Murray, 2009). Ringaby and Forssén (2012) found that the accuracy of the distortion correction increased with the addition of extra poses. This agrees with the experimental results we present in Section 6.3.2.

The work that is most closely related to ours is that of Bibby and Reid (2010). They derived a complete SLAM system capable of estimating the robot pose, building a map and tracking dynamic obstacles in the scene. Cubic splines were used to represent the trajectory of the robot and all dynamic obstacles, but unfortunately, the derivation of SLAM was left in discrete time; however, this is an important paper as it highlights some of the key benefits to using basis functions as a state representation: (i) a smooth trajectory may be represented with fewer state variables than a discrete-time formulation, and (ii) fusing information from sensors running at different rates becomes easy when the state representation is continuous. Similar work was also conducted by Fleps et al. (2011), who used a cubic spline to represent the state for extrinsic calibration of a monocular rolling-shutter camera and a gyroscope. The batch optimization in this instance was performed using a combination of the Broyden-Fletcher-Goldfarb-Shanno method and sequential quadratic programming (SQP) due to the use of quaternions to represent the rotations. We provide a more general derivation in this paper with considerably more detail.

Recent research has brought the approach of non-parametric estimation using Gaussian processes (GPs) (Rasmussen and Williams, 2006) into robotics and computer vision. Although GPs are well suited to estimating continuous functions, they are most easily adapted for estimating distributions over outputs, such as sensor and process models (Plagemann et al., 2007). Work addressing the more difficult problem of discovering a set of meaningful latent variables representing the robot poses is compelling (Ko and Fox, 2011), but introductory. These techniques are only recently being applied to standard problems in robotics (Barfoot et al., 2014; Tong et al., 2012).

Given this prior work, we believe that a clear derivation of the probabilistic form of continuous-time state estimation in robotics is a logical next step to push research forward in this area. Other than our own work, Furgale et al. (2012), we believe our approach to be novel. There have been a number of works employing continuous-time state representations since Furgale et al. (2012), several of which build directly on the ideas originally presented in that paper. A list of these papers are provided in Table 2. Our intent with this paper is to provide the strong theoretical foundations necessary for these subsequent works and to demonstrate its advantages on novel experiments.

The algorithms listed in Table 1 and 2 can be split predominantly along two axes: state representation and optimization algorithm. Methods that use higher-order continuously differentiable representations (Anderson and Barfoot, 2013b; Anderson et al., 2015b; Bibby and Reid, 2010; Fleps et al., 2011; Furgale et al., 2012; Jung and Taylor, 2001; Oth et al., 2013; Tong and Barfoot, 2013; Tong et al., 2012, 2013, 2014) are naturally able to capture more expressive motions (and avoid discontinuities) than those that use linear interpolation (Ait-Aider and Berry, 2009; Bosse and Zlot, 2009; Bosse et al., 2012; Hedborg et al., 2012; Klein and Murray, 2009; Mirzaei and Roumeliotis, 2008; Ringaby and Forssén, 2012; Zlot and Bosse, 2012). Splines over SE(3) (Lovegrove et al., 2013; Patron-Perez et al., 2015) are preferable to splines over alternative pose representations because they more closely model minimal-torque trajectories. Methods that use batch maximum-likelihood estimation solutions such as bundle adjustment (Jung and Taylor, 2001; Klein and Murray, 2009) will be more accurate but slower than filtering solutions (Hol et al., 2010; Jia and Evans, 2012; Kelly and Sukhatme, 2011; Li et al., 2013; Mirzaei and Roumeliotis, 2008). The primary differences between continuous-time derivation of SLAMs are highlighted in Table 3.

A key benefit of using B-splines is that we gain the property of *local support* where the contribution of any single basis function is local in time. With this property, the modification of a single coefficient will have only local effects. This will enable the use of these methods in *local* batch optimization where only temporally or spatially local parameters are optimized in a single batch (Konolige et al., 2010; Sibley et al., 2010).

Finally, another attractive property of B-splines is their *simple analytical derivatives and integrals*. Having access to simple analytical integrals and derivatives allows evaluating error terms in the estimation equations analytically and avoids expensive numerical schemes (Hol et al., 2010; Kelly and Sukhatme, 2010, 2011; Mair et al., 2011; Mirzaei and Roumeliotis, 2008).

## 3. Continuous-time SLAM

In this section, we derive the probabilistic formulation for the full SLAM problem with a continuous-time robot state and discrete-time measurements. We closely follow the familiar discrete-time derivation (cf. Thrun and Montemerlo (2006)) but use identities from Jazwinski (1970) when there are differences between the discrete- and continuous-time cases.

### *3.1. Full SLAM posterior*

Consider a mobile robot navigating through an unknown environment during the time interval $T = [t_0, t_K]$. Within this interval, we define the following quantities:

$\mathbf{x}(t)$: The robot state at time $t$, defined over $T$
$\mathbf{u}(t)$: The control input to the robot at time $t$, defined over $T$
$\mathbf{m}$ : A column of parameters representing the time-invariant map
$\mathbf{z}_i$: A measurement at time $t_i$, $1 \leq i \leq N$
$\mathbf{z}_{1:N}$: The set of all measurements, $\{\mathbf{z}_1,\ldots,\mathbf{z}_N\}$

The probabilistic form of continuous-time SLAM seeks an estimate of the joint posterior density, $p(\mathbf{x}(t)\mathbf{m} \mid \mathbf{u}(t),\mathbf{z}_{1:N})$, of the robot state, $\mathbf{x}(t)$, over the interval, $T$, and the map, $\mathbf{m}$, given the control inputs, $\mathbf{u}(t)$, and measurements, $\mathbf{z}_{1:N}$. In the most general sense, $\mathbf{x}(t)$ represents time-varying states and $\mathbf{m}$ represents time-invariant states. Assigning these the labels "robot" and "map" is merely a common convention in robotics. We assume that the probability density of the robot's initial state, $p(\mathbf{x}(t_0))$, is known. Following the standard derivation, Bayes' rule is used to rewrite the posterior as

$$p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}) = \frac{p(\mathbf{z}_{1:N} \mid \mathbf{x}(t), \mathbf{m}, \mathbf{u}(t))p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t))}{p(\mathbf{z}_{1:N} \mid \mathbf{u}(t))} \tag{1}$$

**Table 2.** A comparison of the characteristics of various continuous-time state estimation methods, Part II: Work following our 2012 conference paper introducing the approach in this paper (Furgale et al., 2012).

| Paper(s) | State Representation | Sensors Used | Estimation Method | Application |
|---|---|---|---|---|
| Furgale et al. (2012) | Cubic spline | Stereo camera and IMU | Gauss–Newton | SLAM |
| Dong and Barfoot (2012) | Linear interpolation | Laser | Gauss–Newton | Motion estimation |
| Jia and Evans (2012) | Linear interpolation | Monocular rolling-shutter camera and IMU | EKF | Rectification |
| Almqvist et al. (2013) | Linear interpolation | Laser | Gauss–Newton | SLAM |
| Anderson and Barfoot (2013b) | Cubic spline | Laser | Gauss–Newton | Velocity estimation |
| Li et al. (2013) | Linear interpolation | Monocular rolling-shutter camera and IMU | EKF | Visual odometry |
| Oth et al. (2013) | Quintic spline | Monocular rolling-shutter camera | Gauss–Newton | Camera calibration |
| Furgale et al. (2013) | Quintic spline | Monocular camera and IMU | Gauss–Newton | Extrinsic/temporal calibration |
| Tong et al. (2012, 2013); Tong and Barfoot (2013); Tong et al. (2014) | Gaussian process | Laser | Gauss–Newton | Motion estimation |
| Lovegrove et al. (2013); Patron-Perez et al. (2015) | Spline over SE(3) | Monocular camera and IMU | Gauss–Newton | Extrinsic calibration and SLAM |
| Anderson and Barfoot (2013a); Anderson et al. (2015b) | Constant velocity | Laser | Gauss–Newton and One-Shot | RANSAC |
| Sheehan et al. (2013) | Catmull-Rom spline | Laser | Gradient-based | Localization against 3D model |
| Alismail et al. (2014) | Quadratic and cubic spline | Laser | ICP | Motion estimation |
| Anderson et al. (2014) | Hierarchical wavelets | Laser | Gauss–Newton | SLAM |
| Rosen et al. (2014) | Gaussian process | Range and odometry | Nonlinear with trust region | Cooperative localization and target tracking |
| Rehder et al. (2014) | Quintic spline | Monocular camera, IMU, Laser | Gauss–Newton | Extrinsic/temporal calibration |
| Barfoot et al. (2014); Anderson et al. (2015a) | Gaussian process with exactly sparse kernel functions | Laser | Gauss–Newton | SLAM |
| Zhang and Singh (2014) | Linear interpolation | Laser | Levenberg–Marquardt | Motion estimation and mapping |
| Yan et al. (2014) | Gaussian process with exactly sparse kernel functions | Range-only | Incremental Smoothing and Mapping 2.0 (Kaess et al., 2012). | SLAM |

EKF: extended Kalman filter; ICP: iterated closest point; IMU: inertial measurement unit; RANSAC: random sampling consensus; SLAM: simultaneous localization and mapping; SQP: sequential quadratic programming.

Assuming that the measurements have no dependence on the control inputs, this becomes

$$p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}) = \frac{p(\mathbf{z}_{1:N} \mid \mathbf{x}(t), \mathbf{m}) p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t))}{p(\mathbf{z}_{1:N})} \quad (2)$$

Next, we make the assumption that the measurements are independent of each other (given the robot trajectory and the map), to get

$$p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}) = \frac{p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t)) \prod_{i=1}^{N} p(\mathbf{z}_i \mid \mathbf{x}(t_i), \mathbf{m})}{p(\mathbf{z}_{1:N})} \quad (3)$$

Finally, we assume that the map, $\mathbf{m}$, is independent of the robot's trajectory, $\mathbf{x}(t)$, and control inputs, $\mathbf{u}(t)$, resulting in

$$p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}) = \frac{p(\mathbf{m}) p(\mathbf{x}(t) \mid \mathbf{u}(t)) \prod_{i=1}^{N} p(\mathbf{z}_i \mid \mathbf{x}(t_i), \mathbf{m})}{p(\mathbf{z}_{1:N})} \quad (4)$$

This expression represents the full SLAM posterior density that we would like to estimate. Building on this foundation, Table 3 highlights the differences between the discrete-time and continuous-time SLAM formulation.

**Table 3.** A comparison of the derivations of discrete-time and continuous-time SLAM.

| | Discrete time | Continuous time |
|---|---|---|
| Goal: | Estimate $p(\mathbf{x}_{0:K}, \mathrm{m} \mid \mathbf{u}_{1:K}, \mathbf{z}_{1:N})$ | Estimate $p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}), t \in [t_0, t_K]$ |
| Factor: | $\frac{p(\mathbf{m})p(\mathbf{x}_0)\prod_{k=1}^{K} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{u}_k)\prod_{i=1}^{N} p(\mathbf{z}_i \mid \mathbf{x}_{t_i}, \mathbf{m})}{p(\mathbf{z}_{1:N})}$ | $\frac{p(\mathbf{m})p(\mathbf{x}(t) \mid \mathbf{u}(t))\prod_{i=1}^{N} p(\mathbf{z}_i \mid \mathbf{x}(t_i), \mathbf{m})}{p(\mathbf{z}_{1:N})}$ |
| Observation model: | $\mathbf{z}_i \sim \mathcal{N}(\mathbf{h}(\mathbf{x}_i, \mathbf{m}), \mathbf{R}_i)$ | $\mathbf{z}_i \sim \mathcal{N}(\mathbf{h}(\mathbf{x}(t_i), \mathbf{m}), \mathbf{R}_i)$ |
| Priors: | $\mathbf{m} \sim \mathcal{N}(\hat{\mathbf{m}}, \mathbf{P}_m), gt\mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_x)$ | $\mathbf{m} \sim \mathcal{N}(\hat{\mathbf{m}}, \mathbf{P}_m), \mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{P}_x)$ |
| Motion model: | $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, gt\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ | $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\delta(t-t')\mathbf{Q})$ |
| MAP estimate: | $\left\{\mathbf{x}_{0:K}^{\star}, \mathbf{m}^{\star}\right\} = \underset{\mathbf{x}_{0:K}, \mathbf{m}}{\operatorname{argmin}}(-\log(p(\mathbf{x}_{0:K}, \mathbf{m} \mid \mathbf{u}_{1:K}, \mathbf{z}_{1:N})))$ | $\left\{\mathbf{x}(t)^{\star}, \mathbf{m}^{\star}\right\} = \underset{\mathbf{x}(t), \mathbf{m}}{\operatorname{argmin}}(-\log(p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N})))$ |

MAP: Maximum-a-Posteriori.

## 3.2. Gaussian assumption

Following common practice (Thrun and Montemerlo, 2006), we make the assumption that the quantities of interest in equation (4) are Gaussian. The probabilistic measurement model is thus

$$\mathbf{z}_i = \mathbf{h}(\mathbf{x}(t_i), \mathbf{m}) + \mathbf{n}_i, \quad \mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i) \tag{5}$$

where $\mathbf{h}(\cdot)$ is a deterministic measurement function and $\mathbf{n}_i$ represents measurement noise drawn from a zero-mean Gaussian density with covariance $\mathbf{R}_i$. This implies that

$$p(\mathbf{z}_i \mid \mathbf{x}(t_i), \mathbf{m}) = \mathcal{N}(\mathbf{h}(\mathbf{x}(t_i), \mathbf{m}), \mathbf{R}_i) \tag{6}$$

Similarly, we assume that the prior beliefs for the map and initial state are Gaussian-distributed

$$\mathbf{m} \sim \mathcal{N}(\widehat{\mathbf{m}}, \mathbf{P}_m), \quad \mathbf{x}(t_0) \sim \mathcal{N}(\widehat{\mathbf{x}}_0, \mathbf{P}_x) \tag{7}$$

In continuous time, the motion model, $p(\mathbf{x}(t)|\mathbf{u}(t))$, may be specified as a continuous stochastic dynamical system (Jazwinski, 1970: 143) described formally by the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t) \tag{8}$$

where the over-dot represents the time derivative, $\mathbf{f}(\cdot)$ is a deterministic function and $\mathbf{w}(t)$ is a zero-mean white GP, written $\mathcal{GP}(\mathbf{0}, \mathbf{Q}\,\delta(t-t'))$. The covariance function for this process is $\mathbf{Q}\delta(t-t')$, where $\delta(\cdot)$ is Dirac's delta function and $\mathbf{Q}$ is a power-spectral-density matrix. Hence, defining

$$\mathbf{e}_u(t) := \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{9}$$

we may write (Jazwinski, 1970: 156)

$$p(\mathbf{x}(t) \mid \mathbf{u}(t)) \propto p(\mathbf{x}(t_0)) \exp\left(-\frac{1}{2}\int_{t_0}^{t_K} \mathbf{e}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{e}_u(\tau)\, d\tau\right) \tag{10}$$

Note, this factor is a prior over all the robot motion that will be modified by the measurements. The matrix, $\mathbf{Q}$, is a hyperparameter that can be used to adjust the smoothness of the solution (i.e. how much to trust the prior versus the measurements). The prior also plays a critical role in solving problems that are under-constrained.

At this point, we have specified a form for all of the probability densities in equation (4) except $p(\mathbf{z}_i)$. This is usually left unspecified because it has no dependence on the unknown state variables.

## 3.3. MAP estimation

Using (4) together with the Gaussian densities and processes specified in Section 3.2, we may derive the objective function for a MAP estimator of the robot state and the map. We want to find $\mathbf{x}(t)^{\star}$ and $\mathbf{m}^{\star}$, the estimates that minimize the negative logarithm of the posterior likelihood

$$\left\{\mathbf{x}(t)^{\star}, \mathbf{m}^{\star}\right\} = \underset{\mathbf{x}(t), \mathbf{m}}{\operatorname{argmin}}(-\log(p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N})) \tag{11}$$

Using equations (4), (6), (7) and (10), we can expand the negative-log posterior into a quadratic form

$$-\log(p(\mathbf{x}(t), \mathbf{m} \mid \mathbf{u}(t), \mathbf{z}_{1:N}) = c + J_z + J_x + J_m + J_u \tag{12}$$

where $c$ is some constant that does not depend on $\mathbf{x}(t)$ nor $\mathbf{m}$ and

$$\mathbf{e}_{z_i} := \mathbf{z}_i - \mathbf{h}(\mathbf{x}(t_i), \mathbf{m}), \quad J_z := \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_{z_i}^T \mathbf{R}_i^{-1} \mathbf{e}_{z_i}, \tag{13a}$$

$$\mathbf{e}_m := \mathbf{m} - \hat{\mathbf{m}}, \quad J_m := \frac{1}{2}\mathbf{e}_m^T \mathbf{P}_m^{-1} \mathbf{e}_m, \tag{13b}$$

$$\mathbf{e}_x := \mathbf{x}(t_0) - \hat{\mathbf{x}}_0, \quad J_x := \frac{1}{2}\mathbf{e}_x^T \mathbf{P}_x^{-1} \mathbf{e}_x, \tag{13c}$$

$$J_u := \frac{1}{2}\int_{t_0}^{t_K} \mathbf{e}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{e}_u(\tau)\, d\tau \tag{13d}$$

Dropping $c$, we define $J(\mathbf{x}(t),\mathbf{m}) := J_z + J_x + J_m + J_u$ and note that

$$\left\{\mathbf{x}(t)^{\star}, \mathbf{m}^{\star}\right\} = \underset{\mathbf{x}(t), \mathbf{m}}{\operatorname{argmin}}(J(\mathbf{x}(t), \mathbf{m})) \tag{14}$$

has the same solution as equation (11).

## 3.4. Basis function formulation

At this point, it may be unclear what we have gained; although we have proposed an objective function, we have not demonstrated how it is possible to estimate $\mathbf{x}(t)$, an uncountably infinite number of states. One possibility is to approximate $\mathbf{x}(t)$ as the weighted sum of a finite set of known temporal basis functions,

$$\mathbf{\Phi}(t) := [\phi_1(t) \quad \cdots \quad \phi_M(t)], \quad \mathbf{x}(t) := \mathbf{\Phi}(t)\mathbf{c} \qquad (15)$$

where $\mathbf{c}$ is a $M \times 1$ column of coefficients. When $\mathbf{x}(t)$ is $D$-dimensional, each individual basis function, $\phi_j(t)$, is also $D$-dimensional and the stacked basis matrix, $\mathbf{\Phi}(t)$, is $D \times M$. Under the basis function formulation, our goal is to estimate the column of coefficients, $\mathbf{c}$.

Using basis functions to represent the state turns our goal of solving equation (14) into a parametric estimation problem. Furthermore, as $\mathbf{\Phi}(t)$ is known, we may look up the state at any time of interest simply by evaluating equation (15) and time derivatives of the state are available as $\dot{\mathbf{x}}(t) = \dot{\mathbf{\Phi}}(t)\mathbf{c}$.

Without specifying the exact basis functions we may now derive an estimator based on the Gauss–Newton algorithm, to solve (14). Defining $\boldsymbol{\theta} := [\mathbf{c}^T \; \mathbf{m}^T]^T$, a joint state vector and starting with an initial guess, $\overline{\boldsymbol{\theta}} = \begin{bmatrix} \overline{\mathbf{c}}^T & \overline{\mathbf{m}}^T \end{bmatrix}^T$, we may linearize each error term ((13a), (13b), (13c) and (9)) with respect to a perturbation, $\delta\boldsymbol{\theta} = [\delta\mathbf{c}^T \; \delta\mathbf{m}^T]^T$, representing small changes in $\boldsymbol{\theta}$ about $\overline{\boldsymbol{\theta}}$. Linearization of the error terms makes $J$ exactly quadratic in $\delta\boldsymbol{\theta}$. To find the optimal update step, $\delta\boldsymbol{\theta}^\star$, for a single iteration of Gauss–Newton, we look for the minimum of the quadratic objective function by setting $(\partial J/\partial\delta\boldsymbol{\theta})^T = \mathbf{0}$ and solving the resulting linear system of equations for $\delta\boldsymbol{\theta}^\star$.

Terms in the objective function corresponding to the priors and measurement model are very similar to the discrete-time case so we will not derive their linearized forms here; however, the motion model term, equation (9), is unique to continuous time and so we expand it fully as

$$\begin{aligned} \mathbf{e}_u(t) &= \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \dot{\mathbf{\Phi}}(t)\mathbf{c} - \mathbf{f}(\mathbf{\Phi}(t)\mathbf{c}, \mathbf{u}(t)) \\ &= \dot{\mathbf{\Phi}}(t)(\overline{\mathbf{c}} + \delta\mathbf{c}) - \mathbf{f}(\mathbf{\Phi}(t)(\overline{\mathbf{c}} + \delta\mathbf{c}), \mathbf{u}(t)) \\ &\approx \overline{\mathbf{e}}_u(t) + \mathbf{E}_u(t)\delta\boldsymbol{\theta}, \end{aligned} \qquad (16)$$

where

$$\overline{\mathbf{e}}_u(t) := \dot{\mathbf{\Phi}}(t)\overline{\mathbf{c}} - \mathbf{f}(\mathbf{\Phi}(t)\overline{\mathbf{c}}, \mathbf{u}(t)) \qquad (17)$$

and

$$\mathbf{E}_u(t) := \left[ \dot{\mathbf{\Phi}}(t) - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{\Phi}(t)\overline{\mathbf{c}}, \mathbf{u}(t)} \mathbf{\Phi}(t) \quad \mathbf{0} \right] \qquad (18)$$

Substituting equation (16) into (13d), we get

$$J_u \approx \frac{1}{2}\int_{t_0}^{t_K} (\overline{\mathbf{e}}_u(\tau) + \mathbf{E}_u(\tau)\delta\boldsymbol{\theta})^T \mathbf{Q}^{-1} (\overline{\mathbf{e}}_u(\tau) + \mathbf{E}_u(\tau)\delta\boldsymbol{\theta}) \, d\tau \qquad (19)$$

Taking the derivative with respect to our perturbation gives us

$$\frac{\partial J_u}{\partial \delta\boldsymbol{\theta}}^T = \int_{t_0}^{t_K} \mathbf{E}_u(\tau)^T \mathbf{Q}^{-1} (\overline{\mathbf{e}}_u(\tau) + \mathbf{E}_u(\tau)\delta\boldsymbol{\theta}) \, d\tau \qquad (20)$$

which splits into two parts

$$\begin{aligned} \frac{\partial J_u}{\partial \delta\boldsymbol{\theta}}^T &= \underbrace{\int_{t_0}^{t_K} \mathbf{E}_u(\tau)^T \mathbf{Q}^{-1} \mathbf{E}_u(\tau) \, d\tau}_{=:\mathbf{A}_u} \, \delta\boldsymbol{\theta} \\ &\quad + \underbrace{\int_{t_0}^{t_K} \mathbf{E}_u(\tau)^T \mathbf{Q}^{-1} \overline{\mathbf{e}}_u(\tau) \, d\tau}_{=:\mathbf{b}_u} \end{aligned} \qquad (21)$$

where we have defined $\mathbf{A}_u$ and $\mathbf{b}_u$ for compactness of notation. The difficulty of evaluating these integrals depends on the specific form of the functions, $\mathbf{\Phi}(t)$, $\mathbf{u}(t)$ and $\mathbf{f}(\cdot)$.

Following a similar process for $J_z$, $J_m$ and $J_x$ results in the matrices $\mathbf{A}_z$, $\mathbf{A}_m$ and $\mathbf{A}_x$ and the corresponding columns $\mathbf{b}_z$, $\mathbf{b}_m$ and $\mathbf{b}_x$. Using these definitions, the Gauss–Newton update step equation for a single iteration becomes

$$\underbrace{[\mathbf{A}_z + \mathbf{A}_m + \mathbf{A}_x + \mathbf{A}_u]}_{=:\mathbf{A}} \delta\boldsymbol{\theta}^\star = -\underbrace{[\mathbf{b}_z + \mathbf{b}_m + \mathbf{b}_x + \mathbf{b}_u]}_{=:\mathbf{b}} \qquad (22)$$

Starting from an initial guess, $\overline{\boldsymbol{\theta}}$, Gauss–Newton proceeds by repeatedly (i) evaluating and solving equation (22) and (ii) applying the update step, $\overline{\boldsymbol{\theta}} \leftarrow \overline{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}^\star$, until convergence (Nocedal and Wright, 2006).

After Gauss–Newton has converged, we may recover $\boldsymbol{\theta}^\star$, the optimal point (MAP) estimate, and the covariance of the estimate

$$\boldsymbol{\theta}^\star = \begin{bmatrix} \mathbf{c}^\star \\ \mathbf{m}^\star \end{bmatrix}, \quad \begin{bmatrix} \mathbf{\Sigma}_{cc} & \mathbf{\Sigma}_{cm} \\ \mathbf{\Sigma}_{cm}^T & \mathbf{\Sigma}_{mm} \end{bmatrix} := \mathbf{A}^{-1} \qquad (23)$$

where $\mathbf{\Sigma}_{cc}$ and $\mathbf{\Sigma}_{mm}$ are the covariance matrices representing uncertainty of the estimates of, respectively, the basis function coefficients, $\mathbf{c}$, and the map, $\mathbf{m}$, and $\mathbf{\Sigma}_{cm}$ encodes correlations between these two.

If we write our uncertainty about the coefficients as

$$\mathbf{c} := \mathbf{c}^\star + \delta\mathbf{c}, \delta\mathbf{c} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{cc}) \qquad (24)$$

it is clear that the resulting estimate for $\mathbf{x}(t)$ is a GP with mean $E[\mathbf{\Phi}(t)(\mathbf{c}^\star + \delta\mathbf{c})] = \mathbf{\Phi}(t)\mathbf{c}^\star$, and covariance function $E[(\mathbf{\Phi}(t)\delta\mathbf{c})(\mathbf{\Phi}(t')\delta\mathbf{c})^T] = \mathbf{\Phi}(t)\mathbf{\Sigma}_{cc}\mathbf{\Phi}(t')^T$, where $E[\cdot]$ is the expectation operator. Therefore, our estimate for the robot state may be written in the notation common for GPs (Rasmussen and Williams, 2006) as

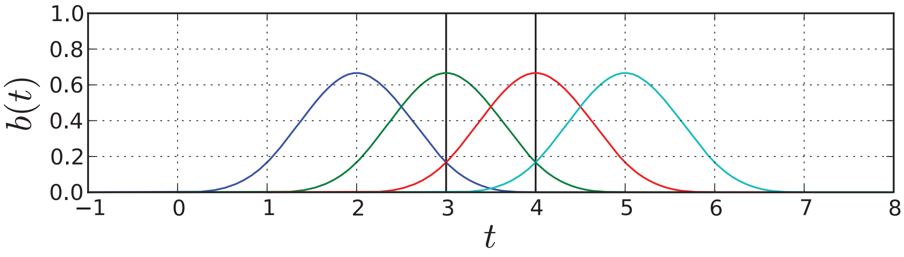

**Fig. 1.** When cubic B-spline basis functions are aligned with a non-decreasing knot sequence, only four basis functions are non-zero for any value of $t$.

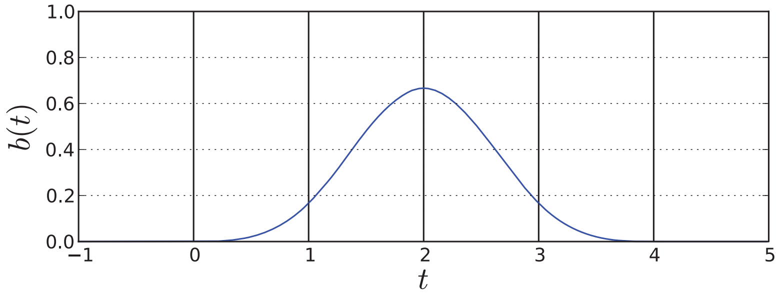

**Fig. 2.** A cubic B-spline basis function is defined as a piecewise cubic polynomial over four segments. The segment transition points are known as knots, shown here as vertical lines. In between each pair of knots, the curve is defined by a cubic polynomial known as the *segment basis polynomial*. By construction, two segment basis polynomials meeting at a knot share the same value, as well as the first and second derivatives. This plot shows a single cubic B-spline basis function on the knot sequence $\{0, 1, 2, 3, 4\}$.

$$\mathbf{x}(t) \sim \mathcal{GP}\left(\mathbf{\Phi}(t)\mathbf{c}^{\star}, \mathbf{\Phi}(t)\mathbf{\Sigma}_{cc}\mathbf{\Phi}(t')^T\right) \tag{25}$$

Up to this point, we have not specified the form of our basis functions.

## 3.5. B-spline state representation

The above derivation could be used with any set of basis functions, but for our particular problem of state estimation in robotics, we would like the basis functions to satisfy some general properties.

1. *Local support*. We would like the contribution of any single basis function to be local in time. With this property the modification of a single coefficient will have only local effects. This will enable the use of these methods in *local* batch optimization where only temporally or spatially local parameters are optimized in a single batch (Konolige et al., 2010; Sibley et al., 2010).
2. *Simple analytical derivatives and integrals*. The relationship between sensory inputs and the robot state is often based on derivatives or integrals of the state equations. Having access a to simple analytical integrals and derivatives will allow us to evaluate error terms in the estimation equations analytically and avoid expensive numerical schemes.

Under these criteria, B-spline functions (Figure 1) become an attractive choice; for any time a B-spline function is a simple polynomial that uses only a subset of temporally local coefficients.

## 4. Matrix formulation for B-splines

A B-spline basis function of order $K$ is a piecewise polynomial function of degree $K-1$. For simplicity of exposition, we will present our derivations using cubic (fourth-order) B-splines. The derivations may be easily extended to work for any order. Furthermore, rather than using the recurrence relation approach to B-splines popularized by De Boor (2002) and others, we will use a matrix formulation that is more amenable to the typical robotics estimation framework. For an introduction to B-splines in general, see Bartels et al. (1987).

### 4.1. B-Spline basis functions

A cubic B-spline basis function is defined as a piecewise cubic polynomial that is non-zero over four consecutive segments of the real line and zero everywhere else, as shown in Figures 1 and 2. The segment transition points, $\{t_i\}$, are known as knots and, in between each pair of knots, the curve is defined by a cubic polynomial known as the *segment basis polynomial*. By construction, two segment

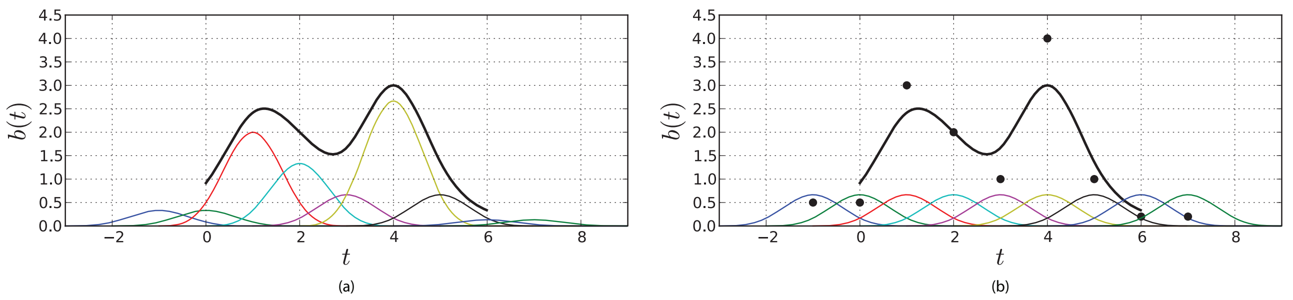

**Fig. 3.** The curve defined by a B-spline (shown in bold black) is the weighted sum of a series of B-spline basis functions. The plot on the left-hand side shows the scaled basis functions. The plot on the right-hand side shows the unscaled basis functions and the spline coefficients plotted as black dots. Note that the B-spline curve does not generally interpolate the coefficients.

basis polynomials meeting at a knot share the same value, as well as the first and second derivatives at that knot. Segment basis polynomial $i$ is defined on the interval $[t_i, t_{i+1})$ and has the form

$$a_i + b_i u_i(t) + c_i u_i(t)^2 + d_i u_i(t)^3 \tag{26}$$

where

$$u_i(t) := \frac{t - t_i}{t_{i+1} - t_i} \tag{27}$$

The coefficients of this polynomial, $a_i$, $b_i$, $c_i$ and $d_i$, are determined by the spacing of the surrounding knots and a single basis function is defined by 16 coefficients (four for each segment). We will define $b_i(t)$ to be the basis function whose first non-zero segment starts at knot $t_i$.

## 4.2. B-spline functions

A cubic B-spline function is a weighted sum of a sequence of B-spline basis functions. When cubic basis functions are aligned with a non-decreasing knot sequence, $\{t_i\}$, only four basis functions are non-zero for any value of $t$, as shown in Figure 1. By convention, we say that the curve is undefined where less than four basis functions are non-zero. Therefore, to have a curve defined over $M$ segments, we need $M + 7$ knots and $M + 3$ basis functions.

Given a strictly non-decreasing knot sequence $\{t_i | i = -3, \ldots, M + 3, t_i \le t_{i+1}\}$, basis functions $\{b_i(t) | i = -3, \ldots, M - 1\}$ and a set of coefficients $\{c_i | i = -3, \ldots, M - 1\}$, a B-spline function, $b(t)$ may be written as

$$b(t) := \begin{bmatrix} b_{-3}(t) & \cdots & b_{M-1}(t) \end{bmatrix} \begin{bmatrix} c_{-3} \\ \vdots \\ c_{M-1} \end{bmatrix} \tag{28}$$

This function is well defined in the interval $[t_0, t_M)$. Because only four basis functions are non-zero for any value of $t$, for $t_i \le t < t_{i+1}$, equation (28) reduces to

$$b(t) := \begin{bmatrix} b_{i-3}(t) & b_{i-2}(t) & b_{i-1}(t) & b_i(t) \end{bmatrix} \underbrace{\begin{bmatrix} c_{i-3} \\ c_{i-2} \\ c_{i-1} \\ c_i \end{bmatrix}}_{=:\mathbf{c}_i} \tag{29}$$

which may be written as

$$b(t) = \mathbf{u}_i(t)^T \mathbf{B}_i \mathbf{c}_i \tag{30}$$

where $\mathbf{B}_i$ is a $4 \times 4$ matrix of basis polynomial coefficients and

$$\mathbf{u}_i(t) := \begin{bmatrix} 1 \\ u_i(t) \\ u_i(t)^2 \\ u_i(t)^3 \end{bmatrix} \tag{31}$$

A sample curve together with the basis functions and spline coefficients is shown in Figure 3. The entries of $\mathbf{B}_i$ are dependent on the knot spacing. The general formula for $\mathbf{B}_i$ is given by Qin (2000) for B-splines of any order.

## 4.3. Derivatives of a B-spline function

In matrix form analytical derivatives of the cubic B-spline function become easy to compute. In fact, we may define a matrix, $\mathbf{D}$, which applies the derivative through matrix multiplication. For a given $t$ on the interval $[t_i, t_{i+1})$, the derivatives are

$$\frac{db(t)}{dt} = \frac{1}{t_{i+1} - t_i} \begin{bmatrix} 0 & 1 & 2u_i(t) & 3u_i(t)^2 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i \tag{32}$$

$$\frac{d^2 b(t)}{dt^2} = \left(\frac{1}{t_{i+1} - t_i}\right)^2 \begin{bmatrix} 0 & 0 & 2 & 6u_i(t) \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i \tag{33}$$

$$\frac{d^3 b(t)}{dt^3} = \left(\frac{1}{t_{i+1} - t_i}\right)^3 \begin{bmatrix} 0 & 0 & 0 & 6 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i = \mathbf{u}_i(t)^T \mathbf{D}_i \mathbf{D}_i \mathbf{D}_i \mathbf{B}_i \mathbf{c}_i \tag{34}$$

$$\frac{d^4 b(t)}{dt^4} = 0 \tag{35}$$

where we have defined the matrix

$$\mathbf{D}_i := \frac{1}{t_{i+1} - t_i} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{36}$$

## 4.4. Definite integrals of a B-spline function

The definite integral of a B-spline function is similarly easy to compute. To evaluate a definite integral, we may compute the integral of each segment separately. Let us consider the integral within a single segment from $t = s_1$ to $t = s_2$ where $t_i \leq s_1 < s_2 < t_{i+1}$

$$\int_{s_1}^{s_2} b(t)\, dt = \int_{s_1}^{s_2} \mathbf{u}_i(t)^T \mathbf{B}_i \mathbf{c}_i \, dt \tag{37}$$

We may simplify the integral with a variable substitution,

$$\begin{aligned} v = \frac{t - t_i}{t_{i+1} - t_i}, v_1 = \frac{s_1 - t_i}{t_{i+1} - t_i}, \\ v_2 = \frac{s_2 - t_i}{t_{i+1} - t_i}, dt = (t_{i+1} - t_i)\, d \end{aligned} \tag{38}$$

which becomes

$$\int_{s_1}^{s_2} b(t)\, dt = (t_{i+1} - t_i) \int_{v_1}^{v_2} \underbrace{\begin{bmatrix} 1 & v & v^2 & v^3 \end{bmatrix} \mathbf{B}_i \mathbf{c}_i}_{=: v_i(v)} \, dv \tag{39}$$

The anti-derivative of $v_i(v)$ is

$$B_i(v) := \begin{bmatrix} v & \frac{v^2}{2} & \frac{v^3}{3} & \frac{v^4}{4} \end{bmatrix} \mathbf{B}_i \mathbf{c}_i \tag{40}$$

and the integral on this interval is

$$\begin{aligned} \int_{s_1}^{s_2} b(t)\, dt = (t_{i+1} - t_i) \\ \int_{v_1}^{v_2} v_i(v)\, dv = (t_{i+1} - t_i)(B_i(v_2) - B_i(v_1)) \end{aligned} \tag{41}$$

When $s_1$ is equal to $t_i$, $v_1 = 0$ and hence $B(v_1) = 0$. In the general case, integrating the B-spline function with arbitrary limits $a$ and $b$ where $t_i \leq a < t_{i+1} < \cdots < t_j \leq b < t_{j+1}$, breaks down into the following form

$$\begin{aligned} \int_a^b b(t)\, dt = & \underbrace{-(t_{i+1} - t_i) B_i\left(\frac{a - t_i}{t_{i+1} - t_i}\right)}_{\text{LHS remainder}} \\ & + \underbrace{\sum_{s=i}^{j-1} (t_{s+1} - t_s) B_s(1)}_{\text{full segments}} + \underbrace{(t_{j+1} - t_j) B_j\left(\frac{b - t_j}{t_{j+1} - t_j}\right)}_{\text{RHS remainder}} \end{aligned} \tag{42}$$

## 4.5. Multidimensional B-spline functions

Many problems in robotics require a time-varying multidimensional state vector. Consequently, we must lift these equations into multiple dimensions. To achieve this, we first arrange our coefficients so that temporally consecutive parameters are adjacent to each other. In this case, the full column of coefficients, $\mathbf{c}$, is assembled as

$$\mathbf{c} := \begin{bmatrix} \mathbf{d}_{-3} \\ \mathbf{d}_{-2} \\ \vdots \\ \mathbf{d}_{M-2} \\ \mathbf{d}_{M-1} \end{bmatrix} \tag{43}$$

where each $\mathbf{d}_i$ is a $D \times 1$ coefficient column associated with basis function $i$

$$\mathbf{d}_i =: \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,D} \end{bmatrix} \tag{44}$$

For a particular $t$ with $t_i \leq t < t_{i+1}$, we may define the columns of the B-spline basis matrix to be

$$\mathbf{B}_i =: \begin{bmatrix} \mathbf{b}_{i1} & \mathbf{b}_{i2} & \mathbf{b}_{i3} & \mathbf{b}_{i4} \end{bmatrix} \tag{45}$$

which allows us to write the general $D$-dimensional version of equation (30) as

$$\mathbf{b}(t) := \underbrace{\begin{bmatrix} \mathbf{u}(t)^T \mathbf{b}_{i1} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i2} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i3} \mathbf{1} & \mathbf{u}(t)^T \mathbf{b}_{i4} \mathbf{1} \end{bmatrix}}_{=: \boldsymbol{\Phi}_i(t)} \mathbf{c}_i \tag{46}$$

where $\mathbf{1}$ is the $D \times D$ identity matrix and

$$\mathbf{c}_i := \begin{bmatrix} \mathbf{d}_{i-3} \\ \mathbf{d}_{i-2} \\ \mathbf{d}_{i-1} \\ \mathbf{d}_i \end{bmatrix} \tag{47}$$

Equation (46) may be rewritten in blocks. Defining

$$\mathbf{U}(t) := \begin{bmatrix} \mathbf{u}(t) & & \\ & \ddots & \\ & & \mathbf{u}(t) \end{bmatrix}, \mathbf{B}_{ij} := \begin{bmatrix} \mathbf{b}_{ij} & & \\ & \ddots & \\ & & \mathbf{b}_{ij} \end{bmatrix} \tag{48}$$

where both $\mathbf{U}(t)$ and $\mathbf{B}_{ij}$ are $4D \times D$ matrices, equation (46) becomes

$$\mathbf{b}(t) = \boldsymbol{\Phi}_i(t) \mathbf{c}_i \tag{49a}$$

$$= \begin{bmatrix} \mathbf{U}(t)^T \mathbf{B}_{i1} & \mathbf{U}(t)^T \mathbf{B}_{i2} & \mathbf{U}(t)^T \mathbf{B}_{i3} & \mathbf{U}(t)^T \mathbf{B}_{i4} \end{bmatrix} \mathbf{c}_i \tag{49b}$$

$$= \mathbf{U}(t)^T \underbrace{\begin{bmatrix} \mathbf{B}_{i1} & \mathbf{B}_{i2} & \mathbf{B}_{i3} & \mathbf{B}_{i4} \end{bmatrix}}_{=: \mathbf{M}_i} \mathbf{c}_i \tag{49c}$$

where we have defined $\mathbf{M}_i$, the multidimensional block version of the B-spline basis matrix $\mathbf{B}_i$.

## 4.6. B-Spline function quadratic integrals

Finally, we examine quadratic forms which arise in motion-constraint terms, such as the minimum acceleration model used in Furgale et al. (2012). The term required in that paper was quadratic in the second time derivative of the basis functions, $\ddot{\mathbf{\Phi}}(t)$; however, in Section 4.3 we showed that the derivatives of $\mathbf{\Phi}(t)$ may be expressed through multiplication by a constant matrix, $\mathbf{D}_i$. Hence, it is sufficient to derive the equations for the most general form of quadratic integral for a B-spline of dimension $D$,

$$\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \tag{50}$$

where $\mathbf{W}$ is some $D \times D$ weighting matrix

$$\mathbf{W} =: \begin{bmatrix} w_{11} & \cdots & w_{1D} \\ \vdots & \ddots & \vdots \\ w_{D1} & \cdots & w_{DD} \end{bmatrix} \tag{51}$$

Just as in Section 4.4, the integral in equation (50) may be split up into a sum of integrals over the segments

$$\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t) dt = \sum_{i=0}^{K-1} \int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \tag{52}$$

For a single segment, substituting equation (49) into (52) gives us

$$\begin{aligned} &\int_{t_i}^{t_{i+1}} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \\ &= \int_{t_i}^{t_{i+1}} \mathbf{M}_i^T \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T \mathbf{M}_i\, dt \\ &= \mathbf{M}_i^T \int_{t_i}^{t_{i+1}} \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T\, dt \mathbf{M}_i \end{aligned} \tag{53}$$

The integral on the right-hand side may be expanded to

$$\begin{aligned} &\int_{t_i}^{t_{i+1}} \mathbf{U}(t) \mathbf{W} \mathbf{U}(t)^T\, dt \\ &= \int_{t_i}^{t_{i+1}} \begin{bmatrix} w_{11}\mathbf{u}(t)\mathbf{u}(t)^T & \cdots & w_{1D}\mathbf{u}(t)\mathbf{u}(t)^T \\ \vdots & \ddots & \vdots \\ w_{D1}\mathbf{u}(t)\mathbf{u}(t)^T & \cdots & w_{DD}\mathbf{u}(t)\mathbf{u}(t)^T \end{bmatrix} dt \end{aligned} \tag{54}$$

which shows that the final subproblem we must solve is

$$\int_{t_i}^{t_{i+1}} \mathbf{u}(t)\mathbf{u}(t)^T\, dt \tag{55}$$

Applying the same change of variables we used in Section 4.4, equation (55) becomes

$$\begin{aligned} &(t_{i+1} - t_i) \int_0^1 \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \begin{bmatrix} 1 & v & v^2 & v^3 \end{bmatrix} dt = (t_{i+1} - t_i) \\ &\int_0^1 \begin{bmatrix} 1 & v & v^2 & v^3 \\ v & v^2 & v^3 & v^4 \\ v^2 & v^3 & v^4 & v^5 \\ v^3 & v^4 & v^5 & v^6 \end{bmatrix} dt \end{aligned} \tag{56}$$

with the solution

$$\mathbf{V}_i := (t_{i+1} - t_i) \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix} \tag{57}$$

which we have designated $\mathbf{V}_i$. Putting it all together, the solution to the initial problem is

$$\begin{aligned} &\int_{t_0}^{t_K} \mathbf{\Phi}(t)^T \mathbf{W} \mathbf{\Phi}(t)\, dt \\ &= \sum_{i=0}^{K-1} \mathbf{L}_i^T \mathbf{M}_i^T \begin{bmatrix} w_{11}\mathbf{V}_i & \cdots & w_{1D}\mathbf{V}_i \\ \vdots & \ddots & \vdots \\ w_{D1}\mathbf{V}_i & \cdots & w_{DD}\mathbf{V}_i \end{bmatrix} \mathbf{M}_i \mathbf{L}_i \end{aligned} \tag{58}$$

where we define the large sparse matrix

$$\mathbf{L}_i := \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{1} & \cdots & \mathbf{0} \end{bmatrix} \tag{59}$$

such that $\mathbf{c}_i = \mathbf{L}_i \mathbf{c}$.

## 5. Continuous-time representation for vehicles in three-dimensional space

If we assume a stationary world frame, $\underrightarrow{\mathcal{F}}_w$, we can define a spline that encodes the parameters of the time-varying transformation, $\boldsymbol{T}_{wb}(t)$, between the inertial frame and the body frame, a frame attached to the moving robot, $\underrightarrow{\mathcal{F}}_b(t)$. The most straightforward way to do this is to choose a rotation parametrization and have three dimensions of the spline encode rotation and the other three encode translation. The drawback to this method is the same faced any time one must choose a rotation parametrization, rotation parameterizations are subject to either singularities or constraints; however, for a particular problem, it is often possible to choose a parametrization with singularities in improbable or impossible positions with respect to the domain. Hence, we will proceed with a general exposition that covers all three-parameter rotation representations. We will use the notation from Barfoot and Furgale (2014).

## 5.1. General rotation parametrization

Our transformation matrix is constructed as

$$\boldsymbol{T}_{wb}(t) := \begin{bmatrix} \mathbf{C}(\boldsymbol{\theta}(t)) & \mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{60}$$

where $\boldsymbol{\theta}(t)$ is a continuous function representing our rotation parameters, $\mathbf{C}(\cdot)$ is a function that builds a rotation matrix and $\mathbf{t}(t)$ is a continuous function representing our translation parameters. Following Section 3.4, these functions may be written as

$$\boldsymbol{\theta} := \boldsymbol{\Phi}(t)\mathbf{c}_\theta, \quad \mathbf{t}(t) := \boldsymbol{\Phi}(t)\mathbf{c}_t \tag{61}$$

and in the notation of Section 4.5, on the interval $t_i < t < t_{i+1}$, they may be written as

$$\boldsymbol{\theta} := \mathbf{U}_i(t)\mathbf{B}_i\mathbf{c}_{\theta_i}, \quad \mathbf{t}(t) := \mathbf{U}_i(t)\mathbf{B}_i\mathbf{c}_{t_i} \tag{62}$$

For the representation to be useful for state estimation, we must answer two questions:

1. What is the relationship between (60) and other useful quantities, namely the angular velocity, $\boldsymbol{\omega}(t)$, linear velocity, $\mathbf{v}(t)$ and linear acceleration of the platform $\mathbf{a}(t)$ ?
2. How do we linearize equations involving (60) with respect to small changes in $\mathbf{c} := \begin{bmatrix} \mathbf{c}_t^T & \mathbf{c}_\theta^T \end{bmatrix}^T$ ?

The answer to the first question is straightforward; because we are encoding the transformation matrix that takes points from the world frame to the body frame, the linear velocity and acceleration are

$$\mathbf{v}(t) = \dot{\mathbf{t}}(t), \quad \mathbf{a}(t) = \ddot{\mathbf{t}}(t) \tag{63}$$

Linearizing these terms with respect to small changes in $\mathbf{c}_v$ is straightforward. For a given rotation parametrization, the relationship to angular velocity, $\boldsymbol{\omega}(t)$, is of the form

$$\boldsymbol{\omega}(t) = \mathbf{S}(\boldsymbol{\theta}(t))\dot{\boldsymbol{\theta}}(t) \tag{64}$$

where $\mathbf{S}(\cdot)$ for many common rotation parameterizations may be found in Table 2.3 in Hughes (1986: 31). The linearization of this term with respect to small changes in $\mathbf{c}_\theta$ must be computed for each choice of parameters. We will present an example in the next section.

The second question is how to linearize expressions involving $\mathbf{C}(\boldsymbol{\theta}(t))$ with respect to small changes in $\mathbf{c}_\theta$. Considering an interval $t_i < t < t_{i+1}$, we define

$$\mathbf{c}_{\theta_i} = \bar{\mathbf{c}}_{\theta_i} + \delta\mathbf{c}_{\theta_i} \tag{65}$$

where $\bar{\mathbf{c}}_{\theta_i}$ is the nominal value of our spline coefficients and $\delta\mathbf{c}_{\theta_i}$ is a perturbation. This gives us

$$\boldsymbol{\theta}(t) := \underbrace{\mathbf{U}_i(t)\mathbf{B}_i\bar{\mathbf{c}}_{\theta_i}}_{\bar{\boldsymbol{\theta}}(t)} + \underbrace{\mathbf{U}_i(t)\mathbf{B}_i\delta\mathbf{c}_{\theta_i}}_{\delta\boldsymbol{\theta}(t)} \tag{66}$$

These definitions may be substituted into the derivations in Barfoot et al. (2011) and Barfoot and Furgale (2014) such that expressions involving rotation matrices may be linearized using

$$\mathbf{C}(\boldsymbol{\theta}(t)) = \left(\mathbf{1} + \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\wedge\right)\mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right) \tag{67}$$

where $\mathbf{1}$ is the identity matrix, $(\cdot)^\wedge$ is the skew-symmetric operator that may be used to implement the cross product

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^\wedge := \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{68}$$

and, again, the form of $\mathbf{S}(\cdot)$ is available in Hughes (1986: 31). Using equation (67) we may write the equations that describe how small changes in our coefficients become changes in an arbitrary homogeneous vector, $\boldsymbol{v} := [\mathbf{v}^T \ s]^T$, which represents a point being transformed

$$\boldsymbol{T}_{wb}(t)\boldsymbol{v} = \begin{bmatrix} \left(\mathbf{1} + \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\wedge\right)\mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t) + \delta\mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{v} \\ s \end{bmatrix} \tag{69a}$$

$$= \begin{bmatrix} \mathbf{1} + \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\wedge & \delta\mathbf{t}(t) - \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\wedge\bar{\mathbf{t}}(t) \\ \mathbf{0}^T & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{v} \\ s \end{bmatrix} \tag{69b}$$

$$= \begin{bmatrix} \mathbf{1} + \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t)\right)^\wedge & \delta\mathbf{t}(t) + \bar{\mathbf{t}}^\wedge(t)\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t) \\ \mathbf{0}^T & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{v} \\ s \end{bmatrix} \tag{69c}$$

$$= \left(\mathbf{1} + \underbrace{\begin{bmatrix} \left(\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\,\boldsymbol{\theta}(t)\right)^\wedge & \bar{\mathbf{t}}^\wedge(t)\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right)\delta\boldsymbol{\theta}(t) + \delta\mathbf{t}(t) \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix}}_{=:\delta\mathbf{d}^\wedge(t)}\right) \begin{bmatrix} \mathbf{C}\left(\bar{\boldsymbol{\theta}}(t)\right) & \bar{\mathbf{t}}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} \mathbf{v} \\ s \end{bmatrix} \tag{69d}$$

where we have defined

$$\delta\mathbf{d}(t) := \begin{bmatrix} \mathbf{1} & \bar{\mathbf{t}}^\wedge(t)\mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right) \\ \mathbf{0} & \mathbf{S}\left(\bar{\boldsymbol{\theta}}(t)\right) \end{bmatrix}\begin{bmatrix} \delta\mathbf{t}(t) \\ \delta\,\boldsymbol{\theta}(t) \end{bmatrix} \tag{70}$$

and have used the (overloaded) $(\cdot)^\wedge$ operator defined on $6 \times 1$ vectors as

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}^\wedge := \begin{bmatrix} \mathbf{b}^\wedge & \mathbf{a} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{71}$$

where **a** and **b** are both 3 × 1 columns. Relating this back to the perturbations of our spline coefficients, at a particular time, $\mathbf{t}_k$, in the interval $[t_i, t_{i+1})$ this gives us

$$\delta\mathbf{d}(t_k) := \underbrace{\begin{bmatrix} \mathbf{1} & \bar{\mathbf{t}}^\wedge(t_k)\mathbf{S}(\bar{\boldsymbol{\theta}}(t_k)) \\ \mathbf{0} & \mathbf{S}(\bar{\boldsymbol{\theta}}(t_k)) \end{bmatrix} \begin{bmatrix} \mathbf{U}_i(t_k)\mathbf{B}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_i(t_k)\mathbf{B}_i \end{bmatrix}}_{=:\mathbf{D}_i(t_k)} \underbrace{\begin{bmatrix} \delta\mathbf{c}_{t_i} \\ \delta\mathbf{c}_{\theta_i} \end{bmatrix}}_{=:\delta\mathbf{c}_i} \tag{72}$$

Next, we will choose a specific three-parameter representation for rotations.

### 5.2. Specific rotation parametrization

For all experiments in this paper, we used the Cayley–Gibbs–Rodrigues parametrization (Bauchau and Trainelli, 2003) in which the column of parameters, $\boldsymbol{\varphi} = \mathbf{a}\tan(1/2)\varphi$, defines rotation about the axis **a** by an angle $\varphi$. This parametrization has simple algebraic forms for $\mathbf{C}(\boldsymbol{\varphi})$ and $\mathbf{S}(\boldsymbol{\varphi})$ (Hughes, 1986)

$$\mathbf{C}(\boldsymbol{\varphi}) := \mathbf{1} + \frac{2}{1+\boldsymbol{\varphi}^T\boldsymbol{\varphi}}(\boldsymbol{\varphi}^\wedge\boldsymbol{\varphi}^\wedge - \boldsymbol{\varphi}^\wedge) \tag{73a}$$

$$\mathbf{S}(\boldsymbol{\varphi}) := \frac{2}{1+\boldsymbol{\varphi}^T\boldsymbol{\varphi}}(\mathbf{1} - \boldsymbol{\varphi}^\wedge) \tag{73b}$$

where $\mathbf{1}$ is the identity matrix and $(\cdot)^\wedge$ is the skew-symmetric operator that acts on 3 × 1 vectors that may be used to implement the cross product. The expressiveness of this representation depends on the number and placement of knots and the order of the B-spline function.

## 6. IMU/Camera calibration

In this section, we apply our proposed continuous-time estimation framework to the well-studied problem of determining the rotation and translation between an IMU and camera that are rigidly mounted to a sensor head. There have been a host of recent papers addressing this problem and because of the high-rate measurements produced by the IMU, the most common solution methods are based on recursive Gaussian filtering (Hol et al., 2010; Jones and Soatto, 2011; Kelly and Sukhatme, 2011; Mirzaei and Roumeliotis, 2008). Our estimator operates on the same type of data required by these algorithms: a dataset collected over a short time interval (up to 3 minutes) consisting of linear acceleration measurements, angular velocity measurements and point landmark measurements derived from camera images.

### 6.1. Problem setup

As depicted in Figure 4, this problem requires definition of several coordinate frames: a world coordinate frame, $\underrightarrow{\mathcal{F}}_w$, which serves as the inertial frame for our estimation problem; the IMU coordinate frame, $\underrightarrow{\mathcal{F}}_b(t)$, in which linear accelerations and angular velocities are measured and the camera coordinate frame, $\underrightarrow{\mathcal{F}}_c(t)$, situated at the camera's optical center with the *z*-axis pointing down the optical axis. Our algorithm estimates time-invariant parameters for (i) gravity, $\mathbf{g}_w$, expressed in $\underrightarrow{\mathcal{F}}_w$, (ii) the locations of $N$ landmarks, $\{\mathbf{p}_w^j \,|\, j=1,\ldots,N\}$ all expressed in $\underrightarrow{\mathcal{F}}_w$ and (iii) the transformation between the camera and the IMU, $\boldsymbol{T}_{bc}$. It also estimates B-spline function coefficients for (iv) the pose of the IMU, $\boldsymbol{T}_{wb}(t)$ and (v) the time-varying accelerometer ($\mathbf{b}_a(t)$) and gyroscope ($\mathbf{b}_\omega(t)$) biases. Following Kelly and Sukhatme (2011), we provide prior information constraining the positions of three landmarks to make the system observable.

Each accelerometer measurement, $\boldsymbol{\alpha}_k$, gyroscope measurement, $\boldsymbol{\varpi}_k$, or landmark measurement, $\mathbf{y}_{j,k}$, is taken to be measured at an instant of time, $t_k$, and corrupted by zero-mean Gaussian noise

$$\boldsymbol{\alpha}_k := \mathbf{C}(\boldsymbol{\varphi}(t_k))^T(\mathbf{a}(t_k) - \mathbf{g}_w) + \mathbf{b}_a(t_k) + \mathbf{n}_{ak} \tag{74a}$$

$$\boldsymbol{\varpi}_k := \mathbf{C}(\boldsymbol{\varphi}(t_k))^T\boldsymbol{\omega}(t_k) + \mathbf{b}_\omega(t_k) + \mathbf{n}_{\omega k} \tag{74b}$$

$$\mathbf{y}_{j,k} := \mathbf{g}\left(\mathbf{T}_{cb}\mathbf{T}_{wb}(t_k)^{-1}\mathbf{p}_w^j\right) + \mathbf{n}_{yk} \tag{74c}$$

where each $\mathbf{n}_{ik} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{ik})$ is assumed to be statistically independent of the others, $\mathbf{g}(\cdot)$ is a nonlinear perspective camera model and there is no requirement that the measurements are captured synchronously, or even periodically.

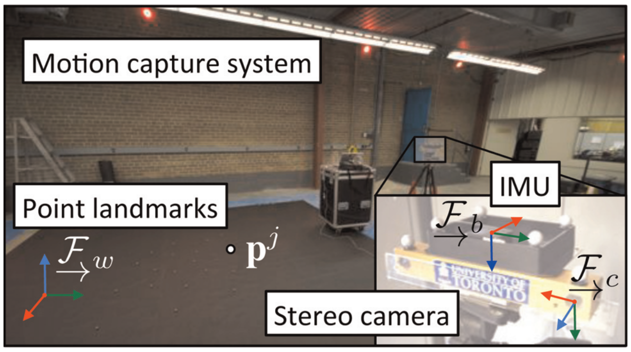
Motion capture system
Point landmarks
$\underrightarrow{\mathcal{F}}_w$
$\mathbf{p}^j$
IMU
$\underrightarrow{\mathcal{F}}_b$
$\underrightarrow{\mathcal{F}}_c$
Stereo camera

**Fig. 4.** The problem setup and experimental apparatus. In the IMU/camera calibration problem, the goal is to estimate the transformation between an IMU, $\underrightarrow{\mathcal{F}}_b$, and a camera $\underrightarrow{\mathcal{F}}_c$ rigidly attached to the same sensor head. To do this, the sensor head is moved through the scene collecting IMU measurements while the camera observes a set of point landmarks, $\{\mathbf{p}^j | j=1,\ldots,N\}$. As part of the estimation process, the time-varying pose of the IMU is estimated with respect to a fixed world frame, $\underrightarrow{\mathcal{F}}_w$. In the second experiment we used the wide-baseline (24 cm) stereo cameras from a Point Grey Research Bumblebee XB3 and a MicroStrain 3DM-GX2 IMU. For evaluation, the sensor head was tracked using a Vicon motion capture system. The motion capture system recorded the trajectory of the sensor head and the positions of the landmarks.

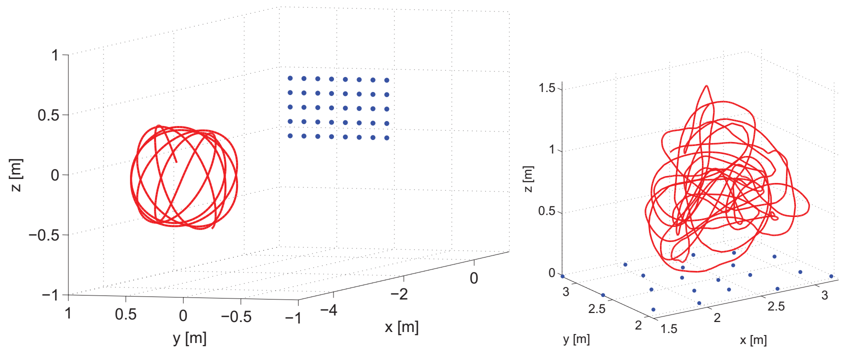

**Fig. 5.** The trajectories taken by the sensor head (red, solid) and the landmarks (blue dots) used in our simulation experiment (left-hand side) and our evaluation on real data (right-hand side).

As the sensor head in this problem is manipulated by hand, we have no control signal to feed into a motion model; hence we model the time-varying components of the system as driven by zero-mean, white GPs

$$\ddot{\mathbf{t}}(t) = \mathbf{w}_t(t), \quad \ddot{\boldsymbol{\varphi}}(t) = \mathbf{w}_\varphi(t), \quad \dot{\mathbf{b}}_a(t) = \mathbf{w}_a(t), \quad \dot{\mathbf{b}}_\omega(t) = \mathbf{w}_\omega(t) \tag{75}$$

where each $\mathbf{w}_i(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_i \delta(t - t'))$. We assume the processes are statistically independent of each other (this means that $E[\mathbf{w}_i(t)\mathbf{w}_j(t')^T] = \mathbf{0}$ for all $t$, $t'$, where $E[\cdot]$ is the expectation operator) and of the measurements in equation (74).

## *6.2. Estimator*

We estimate the five quantities defined above using the batch Gauss–Newton algorithm outlined in Section 3.4. Error terms associated with the measurements, equation (74), are constructed as the difference between the measurement and the predicted measurement given the current state estimate (as in equation (13a)). Following Section 3.4, we provide an example of the error term associated with one of the motion models defined in equation (75), $\ddot{\mathbf{t}}(t) = \mathbf{w}_t(t)$, $\mathbf{w}_t(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_t \delta(t - t'))$. In this simple case, the mean function is zero and so the error term from equation (16) collapses to

$$\mathbf{e}_u(t) = \ddot{\mathbf{t}}(t) = \ddot{\boldsymbol{\Phi}}(t)\mathbf{c}_t \approx \underbrace{\ddot{\boldsymbol{\Phi}}(t)\bar{\mathbf{c}}_t}_{=:\bar{\mathbf{e}}_u(t)} + \underbrace{\ddot{\boldsymbol{\Phi}}(t)}_{=:\mathbf{E}_u(t)} \delta\mathbf{c}_t \tag{76}$$

The integrals in equation (21) then become

$$\mathbf{A}_u = \int_{t_0}^{t_K} \ddot{\boldsymbol{\Phi}}(\tau)^T \mathbf{Q}_t^{-1} \ddot{\boldsymbol{\Phi}}(\tau)\, d\tau, \tag{77a}$$

$$\mathbf{b}_u = \int_{t_0}^{t_K} \ddot{\boldsymbol{\Phi}}(\tau)^T \mathbf{Q}_t^{-1} \ddot{\boldsymbol{\Phi}}(\tau)\, d\tau\, \bar{\mathbf{c}}_t \tag{77b}$$

In the case of B-spline basis functions, the integral involved in computing $\mathbf{A}_u$ may be evaluated in closed form and *exactly once* for each estimation problem.

Starting from a dataset, our estimator proceeds as follows. As with any Gauss–Newton estimation algorithm, we require initial guesses for all quantities we are estimating. We initialize the IMU bias splines to zero and assume guesses are available a priori for gravity, the landmark positions and the calibration transformation. For the position of the IMU, $\boldsymbol{T}_{wb}(t)$, we produce an initial guess by first computing a rough estimate of the camera position for each image that has enough landmark measurements using the `solvePnP()` function from the OpenCV library to compute $\boldsymbol{T}_{wc}(t_k)$ for each image time, $t_k$. Applying the initial guess for the calibration transformation we build $\boldsymbol{T}_{wb}(t_k) = \boldsymbol{T}_{wc}(t_k)\boldsymbol{T}_{cb}$. Finally, we produce an initial set of spline coefficients by using the linear algorithm of Schoenberg and Reinsch (De Boor, 2001: ch. XIV). After generating the initial guesses, the Gauss–Newton algorithm is run to convergence, processing all measurements and motion models and iteratively refining the estimate. After convergence, the mean estimate and covariance are recovered using equation (23).

## *6.3. Experimental evaluation*

The algorithm described above was evaluated in two experiments: (1) a simulation designed to test the ability of the estimator to find the calibration transformation and (2) an experiment using real data to highlight the ability of this state representation to represent large problems with fewer state variables.

*6.3.1. Simulation.* Because of the difficulty of obtaining accurate ground-truth data for the calibration transformation, $\boldsymbol{T}_{cb}$, we tested the algorithm in a simulation where the noise models and ground-truth values were known exactly. Anecdotally, we found that large accelerations and angular velocities made the estimate of $\boldsymbol{T}_{cb}$ more accurate. This agrees with the observability analysis and the simulation experiments described in Kelly and Sukhatme (2011).

We generated a 60-second trajectory, shown in Figure 5, based on sinusoidal functions, which provide analytical derivatives used to produce the simulated accelerometer

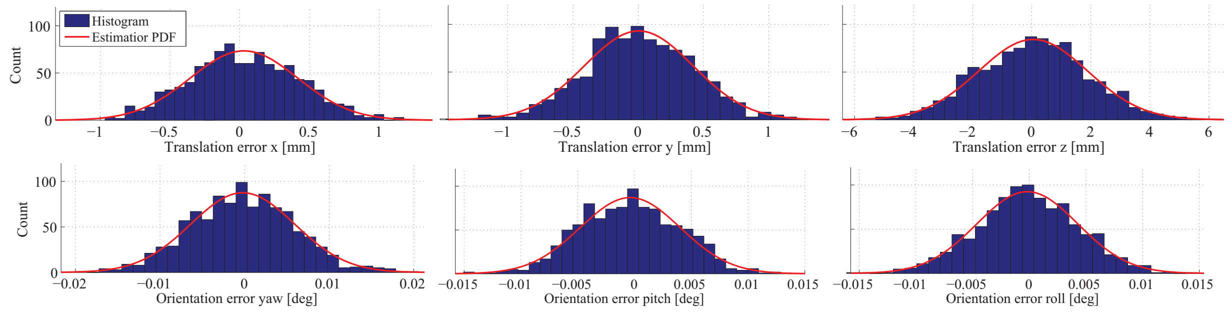

**Fig. 6.** A comparison of the uncertainty returned by the estimator (red, solid line) and a histogram of errors (blue bars) for the estimate of the components of the calibration matrix, $\boldsymbol{T}_{cb}$, over 1000 simulation trials. When the noise properties of the system are known exactly, the uncertainty returned by the estimator is an excellent fit to the experimental results.

and gyroscope measurements. We ran 1000 trials in which we sampled the measurement noise (120 monocular images, 5950 IMU measurements) and IMU biases, initialized the pose spline (300 basis functions) and ran Gauss–Newton to convergence (three or four iterations taking approximately 12 seconds). The main results of this experiment are shown in Figure 6. Over 1000 trials, the estimated uncertainty, plotted as a Gaussian probability density function, is an excellent match for the histogram of errors computed during this simulation.

The uncertainties returned by the estimator were remarkably stable over the 1000 trials; the spread (the difference between the maximum and minimum) of standard deviations was less than 0.2% of the standard deviation value. Based on these results, we believe that, given a sufficiently smooth path of the sensor head and accurate knowledge of the noise models, our algorithm accurately estimates the calibration transformation, $\boldsymbol{T}_{c,i}$, and reports the correct uncertainty for the estimate.

*6.3.2. Real data.* In the second experiment, we used a dataset collected at the University of Toronto Institute for Aerospace Studies in which the sensor head shown in Figure 4 was manipulated by hand over a set of point landmarks for approximately 2 minutes and 20 seconds. A Vicon motion capture system tracked the sensor head and the point landmarks. We believe the output of this system is accurate enough to use as ground-truth for this experiment.

During this experiment, the sensor head collected 1639 stereo images and 14,211 IMU measurements. While it is clear from the related work that it is possible to process a dataset of this size using a Gaussian filter, the outlook for a discrete-time batch algorithm is much more grim; one must estimate the state at each measurement time. In terms of our 12 time-varying parameters (six state and six bias) this would involve estimating $12 \times 15{,}850 = 190{,}200$ state variables in addition to our time-invariant parameters in a single batch. While this may be possible using sparse matrix methods it is certainly not ideal. In this experiment, we show that using our method we may estimate the trajectory of the sensor head in continuous time using far fewer parameters.

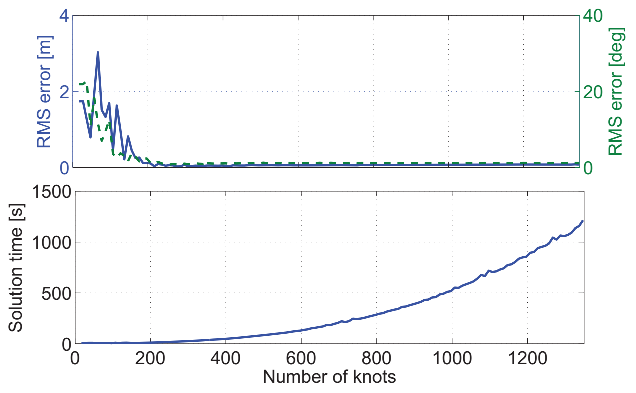

**Fig. 7.** Using a dataset with 1639 images and 14,211 IMU measurements, the number of knots in the spline encoding $\boldsymbol{T}_{w,i}(t)$ was varied from 10 to 1350 in steps of 10. For each setting we ran our estimator and evaluated the root-mean-square (RMS) error of both the translation and orientation parts of $\boldsymbol{T}_{w,i}(t)$ by comparing them to the track from the Vicon motion capture system shown in Figure 4. The estimator was limited to four iterations and timed. The plots in this figure show that, when too few knots are used, the estimation errors can be very large, but as the number of knots is increased, the spline is able to represent the trajectory. For the dataset used in Section 6.3.2, between 200 and 300 knots is sufficient to accurately estimate the path of the sensor head but further knots only increase the execution time of the estimator. Note that we do not overfit as we increase the number of knots because we included the motion model into our formulation, which serves as a regularizer.

As the IMU biases are moving slowly we fixed the number of knots in the bias spline at 15. For the pose spline encoding $\boldsymbol{T}_{wb}(t)$, we ran the estimator many times, varying the number of knots from 10 to 1350, increasing by 10 each time. For each setting we ran our estimator and evaluated the root-mean-square (RMS) error of both the translation and orientation parts of $\boldsymbol{T}_{wb}(t)$ by comparing them to the track from the Vicon motion capture system. The estimator was limited to four iterations and timed on a MacBook Pro with a 2.66 GHz Core 2 Duo and 4 GB of 1067 MHz DDR3 RAM.

Figure 7 shows the main result of this experiment. When too few knots are used, the estimation errors can be very

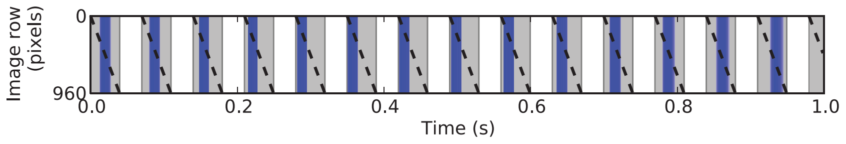

**Fig. 8.** An illustration of the timing of rolling-shutter image capture on this experiment. The gray box indicates the capture time for the image, the vertical blue lines are the observation times for individual dot grid points (see Figure 9) and the diagonal line represents the start of exposure time for an individual row of the image. It is this offset of the start of exposure for subsequent rows that causes the rolling-shutter wobble effect.

large but, as the number of knots is increased, the spline is able to represent the trajectory. For the dataset used in this section, between 200 and 300 knots is sufficient to accurately estimate the path of the sensor head but additional knots only serve to increase the execution time of the estimator. Iterating to convergence with 300 knots took approximately 26 seconds. This is far less than the length of the dataset (2 minutes and 20 seconds) and thus the approach holds promise for realtime implementation. In the timing plot of Figure 7, the execution time is dominated by the solution of the $\mathbf{A}\delta\boldsymbol{\theta}^{\star} = -\mathbf{b}$ system in (22).

Work on this topic has since been extended to include estimation of the unknown time delay between the camera and the IMU (Furgale et al., 2013) and this code is available publicly as a calibration toolbox.

## 7. Perspective localization with a rolling-shutter camera

In this section we present a simple example of perspective localization using a rolling-shutter camera. Rolling-shutter cameras violate one of the major assumptions of image-based localization that the pixels were all captured at the same time instant. Instead, the rows of the image are exposed sequentially, causing the well-known ‘rolling-shutter wobble’ effect. Figure 8 illustrates the timing of the rolling-shutter camera used in this experiment.

By using the basis function approach we build errors for every observed point by looking up the measurement time, based on the image row at which the line was observed. A similar formulation has been used by several other authors for both rolling-shutter cameras (Forssén and Ringaby, 2010; Hedborg et al., 2011, 2012; Li et al., 2013; Lovegrove et al., 2013; Oth et al., 2013; Ringaby and Forssén, 2011, 2012) and continuously scanning lasers (Anderson and Barfoot, 2013b; Dong and Barfoot, 2012; Tong and Barfoot, 2013; Tong et al., 2014). In this experiment, we provide a direct comparison of perspective localization between a rolling-shutter camera and a global-shutter camera, which were mounted side-by-side and synchronized to capture data simultaneously. The camera rig was manipulated by hand in front of a planar pattern board with a regular grid of circles. For each camera, the motion estimate was computed with respect to the pattern board and compared to the motion estimate returned by a Vicon motion capture system. To the best of our knowledge, this is the first direct comparison of global-shutter and rolling-shutter localization.

### 7.1. Problem setup

Figure 9 shows our the sensor head we used in this experiment. The world coordinate frame (measured by a Vicon motion capture system), $\underrightarrow{\mathcal{F}}_w$, which serves as the inertial frame for our estimation problem and the camera coordinate frame, $\underrightarrow{\mathcal{F}}_c(t)$, situated at the camera’s optical center with the $z$-axis pointing down the optical axis. Our rig has two cameras one with a rolling shutter and the other with a global shutter, but we consider them independently so that we can compare the results. Both cameras captured $1280 \times 960$ grayscale images at 15 frames per second.

Our algorithm estimates the B-spline function coefficients for the pose of the camera, $\boldsymbol{T}_{wc}(t)$. Each landmark measurement, $\mathbf{y}_{j,k}$, is taken to be measured at an instant of time, $t_{j,k}$ and corrupted by zero-mean Gaussian noise

$$\mathbf{y}_{j,k} := \mathbf{g}\left(\mathbf{T}_{wc}(t_{j,k})^{-1}\mathbf{p}_w^j\right) + \mathbf{n}_{j,k} \tag{78}$$

where $\mathbf{n}_{j,k} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_{j,k}\right)$ and $\mathbf{g}(\cdot)$ is a standard perspective camera model. The measurement time is inferred from the row of the corner observation as

$$t_{j,k} = t_k + dy_{j,k} \tag{79}$$

where $t_k$ is the timestamp for image $k$ and $d$ is the *line delay*, the time between the start of exposure of successive lines in the rolling-shutter model. In this experiment, we include no motion model for the platform. The estimation is performed only by minimizing reprojection error.

### 7.2. Estimator

We again use the batch Gauss–Newton algorithm outlined in Section 3.4. Reprojection error terms associated with the measurements, equation (78), are once again constructed as the difference between the measurement and the predicted measurement given the current state estimate (as in equation (13a)). We initialize the B-spline encoding the camera’s pose, $\boldsymbol{T}_{wc}(t)$, by fitting the coefficients to an initial guess generated at each image time using the eight-point method from Zhang (1998). Note that Zhang’s method does not account for the rolling-shutter effect, so the

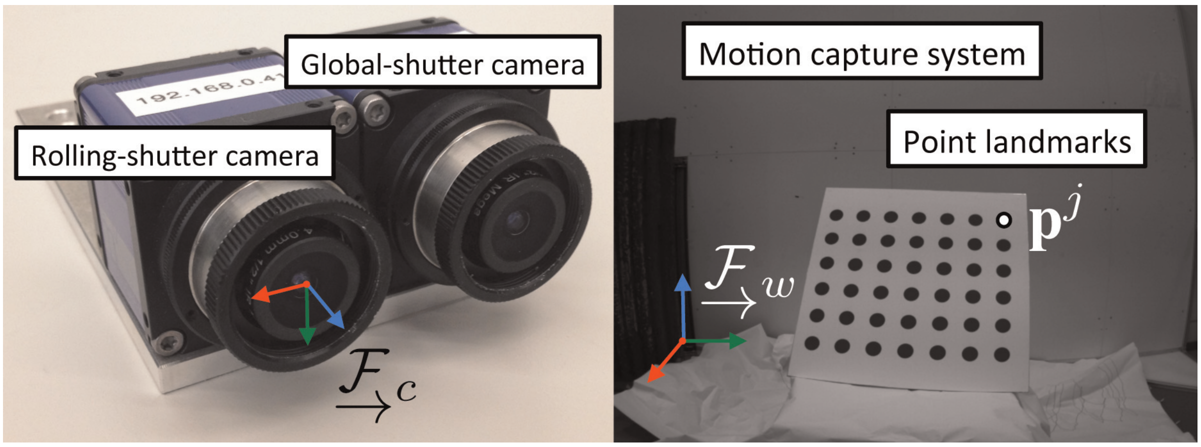

**Fig. 9.** The problem setup and experimental apparatus for the rolling-shutter perspective localization problem. Our rig has a pair of synchronized cameras, a CMOS rolling-shutter camera (Matrix Vision BlueCougar-X102d) and a CMOS global-shutter camera (Matrix Vision BlueCougar-X012b). Each camera is considered individually so that we can compare the estimates on similar data. The goal is to estimate the time-varying pose of the camera frame, $\underrightarrow{\mathcal{F}}_c$, with respect to the world frame, $\underrightarrow{\mathcal{F}}_w$, given a sequence of timestamped images as the camera is waved in front of the pattern board. The positions of the circles on the pattern board, $\{\mathbf{p}^j | j = 1,\ldots,N\}$, are assumed to be known. For evaluation, the sensor head was tracked using a Vicon motion capture system.

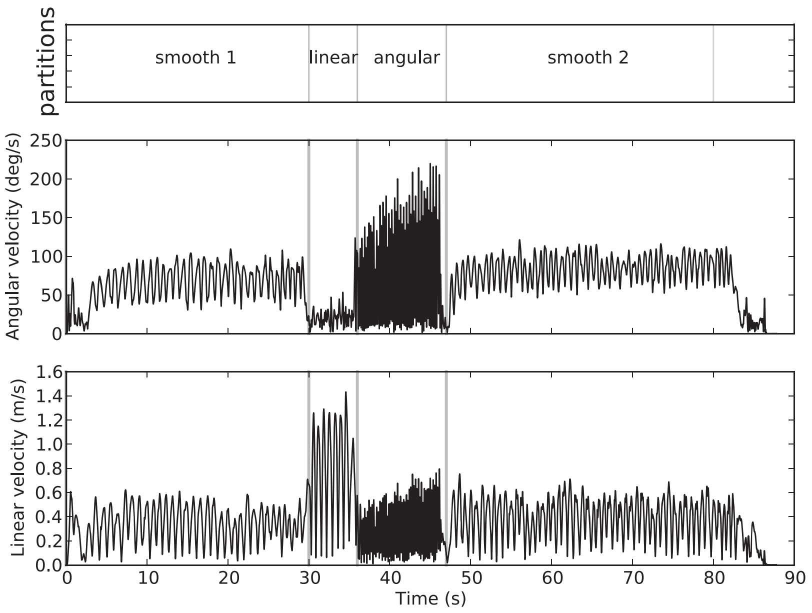

**Fig. 10.** This figure shows the linear and angular velocity of the camera rig during the dataset used for evaluation in this section. For close analysis we have partitioned the dataset into four sections. Two sections exhibit smooth motion (smooth 1 and smooth 2), one section consists of fast linear motion (linear) and one section consists of fast angular motion (angular). We will examine these sections in later plots.

initialization of the rolling-shutter camera is worse than that of the global-shutter camera when the sensor head is in motion.

## *7.3. Experimental evaluation*

The dataset used in this section is approximately 80 seconds long in which the camera rig is waved around in front of the dot pattern. We tried to push the localization to its limits by including different types of motion as shown in Figure 10. This resulted in four partitions of this dataset: two smooth motion sections, a fast linear motion section with velocities up to 1.4 m/s and high accelerations and a fast angular motion section with rotation speeds as high as 200 degrees per second and high angular acceleration.

*7.3.1. Influence of increasing the number of knots.* With a fixed order, the representational power of a B-spline can be

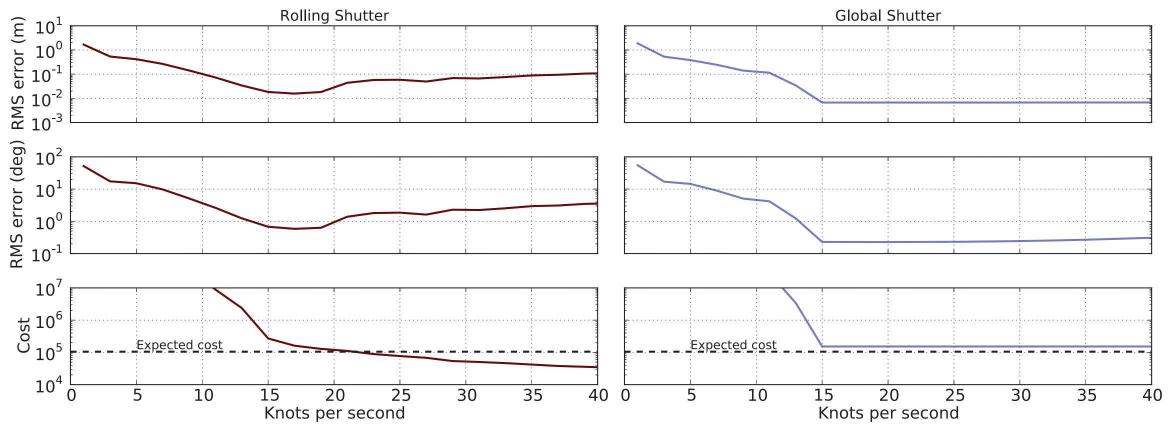

**Fig. 11.** The effect of increasing the number of knots on the accuracy of the estimation for both rolling-shutter and global-shutter localization. The top and middle plots show that the root-mean-square (RMS) error decreases as the number of knots is increased. In the rolling-shutter case, the RMS error starts to increase as the spline starts to overfit to the data. In the global-shutter case, the error plateaus when the spline has enough knots to represent the trajectory, then increases slowly as the spline becomes unconstrained between the images. The bottom plot shows that the cost function deterministically decreases as more knots are added and that the expected value of the cost function nearly coincides with the minimum RMS error.

increased by using more knots. Figure 11 shows the effect of increasing the number of evenly spaced knots when processing both the rolling-shutter and global-shutter images. The camera rig was shaken quite violently during this dataset so, when the spline does not have enough knots, the error is high, both in the cost function and the RMS error when compared to the motion capture system. Around 15 knots per second, the spline is able to represent the trajectory and the RMS errors plateau.

Since we do not have a motion model, the plateau dips and the RMS error starts to increase for the rolling-shutter camera as more knots are added, while the cost (the sum of inverse-covariance-weighted squared reprojection errors) continues to decrease monotonically. In contrast, the cost does not decrease significantly as knots are added for the global-shutter camera. The difference here is that the rolling-shutter camera's observations are spread out in time (see Figure 8). Consequently, adding more knots allows the estimator to overfit to the noise on the observations by increasing the velocity of the spline. In the global-shutter case, all of the observations for an image come in simultaneously so the ability of the estimator to fit the data is constrained by geometry, the estimator cannot reduce the cost with additional motion; however, the error in the global-shutter estimate starts to grow as more knots are added *even though* the cost does not significantly decrease. This is because the absence of regularization terms allows the unconstrained spline to oscillate between the images.

As introduced in Oth et al. (2013) and discussed in Anderson et al. (2014), the normalized innovation squared (NIS) (Bar-Shalom et al., 2002) allows us to test if the sum of inverse-covariance-weighted squared errors (the cost function) matches the expected value given the noise models. Briefly, weighting the squared errors by the inverse covariance whitens the noise and the expected value of cost, $E[J]$, is simply the dimension of the stacked error vector. For our results, we derived the covariance matrices for the circle grid observations by computing the covariance of the residuals after intrinsic camera calibration. The expected cost is drawn as a dotted horizontal line in Figure 11. For both cameras, the expected cost is a good predictor of the minimum error when compared to ground truth; however, it is not a perfect fit. There are two possible reasons for this. First, the fit of the model to the cost may not be constant over the whole dataset (as we will examine later) so the use of a uniform knot sequence and a single average is too coarse a measure for proper analysis. Second, the intrinsic calibration dataset was collected under ideal conditions with a static circle grid, so the covariance estimated from those residuals may underestimate the covariance of the circle detector under motion blur or motion distortion. This seems likely as the expected cost is slightly higher than the actual cost in both cases.

Next, we examine the estimates produced in both the rolling-shutter and global-shutter cases using 18 knots per second, the uniform knot sequence that produced the best estimate when compared to ground truth in Figure 11.

*7.3.2. Continuous- versus discrete-time estimation.* Figure 12 shows the benefits of using a continuous-time model for rolling-shutter motion estimation. Both the continuous and discrete-time estimators use Gauss–Newton.

The estimate is significantly better than the discrete-time estimate that does not account for the temporal offset of the individual image measurements. These results essentially confirm what has been demonstrated previously for sweeping lasers and rolling-shutter cameras. We know that the continuous-time model improves the accuracy of motion estimation for sweeping imaging sensors. But does it make the rolling-shutter estimate as accurate as a

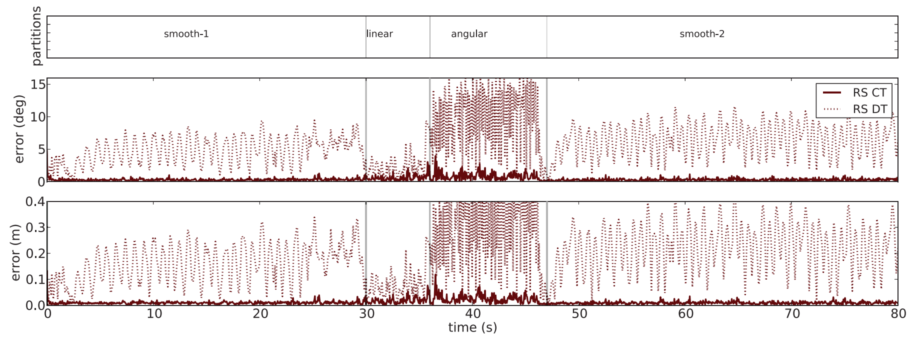

**Fig. 12.** A comparison of the discrete-time (DT) Gauss–Newton and continuous-time (CT) models when performing motion estimation for a rolling-shutter (RS) camera. The continuous-time model produces a significantly better estimate the discrete-time equivalent that does not account for motion distortion.

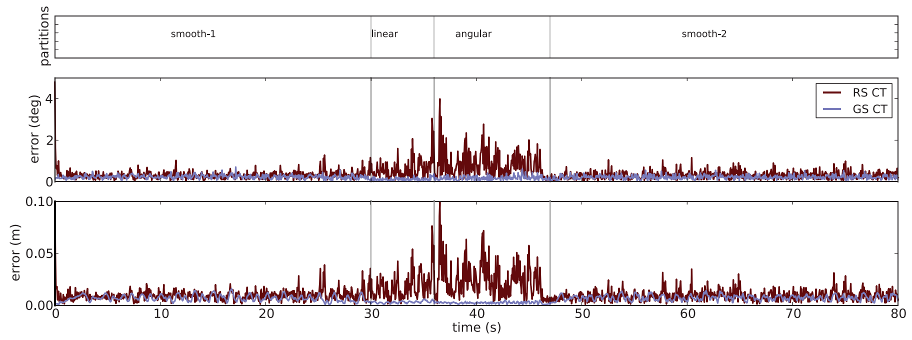

**Fig. 13.** A comparison of the global and rolling-shutter continuous-time motion estimates. The accuracy of the estimates is comparable when the motion is smooth. When the motion becomes extreme, the rolling-shutter (RS) estimate becomes worse than the global-shutter (GS) estimate. As these errors are computed at the timestamps of the Vicon motion capture system (200 Hz) the accurate performance of the global-shutter estimate implies that the spline using 18 knots per second is sufficient to capture the dynamics of the dataset.

global-shutter camera? Figure 13 compares the final motion estimate of the global- and rolling-shutter cameras using the continuous-time model. During periods of smooth motion, the estimate from either camera is similarly accurate; however, during periods of extreme motion (especially angular velocity) the estimate produced by the rolling-shutter camera becomes worse, while the global-shutter estimate remains highly accurate.

### 7.4. Discussion

There are several possibilities for why the rolling-shutter estimate is less accurate than the global-shutter one.

1. *Reduced accuracy of the corner detector*. The motion distortion of the rolling-shutter causes the circles in the circle grid to deform under motion. It is possible that this deformation biases the pixel location returned by the circle detector and that this unmodelled effect reduces the estimator accuracy.
2. *Inferring the measurement time from the corner position*. In equation (79) we set the measurement time used for prediction to be based on the measurement in image space. This is not the only option. Oth et al. (2013) also use this measurement time but then adapt the covariance of the image measurement to account for uncertainty of the time parameter. Lovegrove et al. (2013) estimate each measurement time every iteration of Gauss–Newton using a fixed-point method, based on the current values of their pose spline. A comparison of these methods on simulated and real data would be interesting future work.
3. *Not enough knots to represent the motion*. Rolling-shutter cameras represent a higher-frequency signal than their global-shutter counterparts. It is possible that this requires more representational power in the spline during parts of the trajectory with extreme motion. While this is most likely not the case here as the global-shutter camera estimate is able to capture the true motion, this effect is worth considering in other applications when vibration or jerky motion could disrupt rolling-shutter estimators.

The question of the amount of representational power needed is an interesting and open topic for continuous-time motion estimation. Both Oth et al. (2013) and Anderson

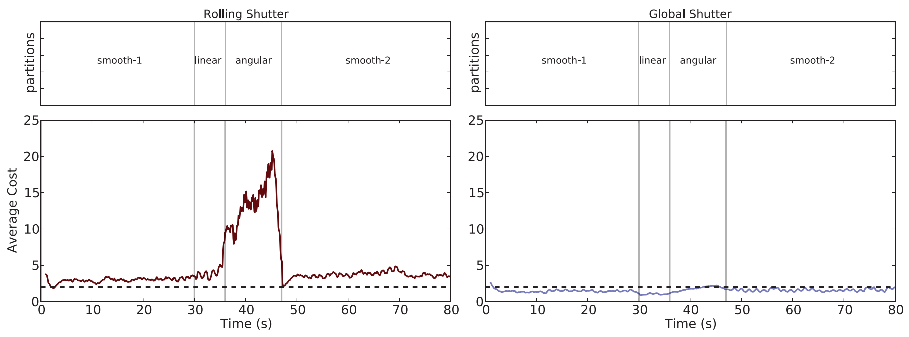

**Fig. 14.** Using a window of one second to compute the average NIS ($E[J]$). The expected average cost is plotted as a dashed horizontal line. The left-hand side plot shows that, for the rolling-shutter camera, the model and the knot sequence are both a good fit for the smooth sections, but not for the section with high angular motion. The right-hand side plot shows that the noise model for the global-shutter camera slightly underestimates the variance of the dot grid detection (resulting in a lower than expected average cost) but this is nearly constant over the whole dataset, even during extreme motion.

et al. (2014) showed that there is some “optimal” number of knots to use for motion estimation on a particular dataset. In Oth et al. (2013), the knot-picking strategy and examination of the NIS was absolutely necessary to realize a reliable and repeatable calibration method; uniform knot sequences were not good enough. Figure 14 tries to give some intuition about this. It plots the average NIS computed over a one second window after estimation using 18 knots per second. The dashed horizontal line is the expected value (this is two, as reprojection error has two components). Here we can see that during the smooth motion sections, both cameras produce NIS values that are near the expected value. In the sections with more extreme motion, the rolling-shutter errors are much higher than expected. In our experience, this is common for under-constrained problems in continuous-time estimation. Remember that, in this case, only reprojection errors are used; there is no motion model or regularization. NIS tells us that our model does not fit the measurements. In the case that all other effects are properly accounted for, this can be an excellent measure of underfitting or overfitting. In cases such as this one, where we have several hypotheses as to the reason for the errors, NIS simply tells us that our model is not a good match for the data.

In situations where a high-rate measurement is available to regularize all the degrees of freedom of the curve (such as an IMU or a motion capture system) we have found that it is possible to use a dense uniform knot sequence and let the measurements regularize the curve (cf. Furgale et al. (2013)). In situations where high-rate measurements are not available, or it is important to keep the number of knots as small as possible, NIS is the tool we use to automatically analyze the fit to the data; however, there is still significant work to be done on this topic. First, the proposed methods for adaptively choosing the number of basis functions (Anderson et al., 2014; Oth et al., 2013) were both considered in the offline batch context. Bringing adaptive knot picking to online motion estimation will be a major step. Second, it is not clear that all degrees of freedom of the curve need the same number of basis functions. Consider a six-degrees-of-freedom motion estimate of a robot traveling in a plane where $z$ (the height above the plane) pitch and roll are all constant. It is clear that these degrees-of-freedom (DOFs) need very few parameters, whereas the other DOFs need to be examined based on the motion of the platform. A similar situation is encountered in the dataset used above. During the `linear` section, linear velocity is large while the angular velocity is close to zero. In this section, why should the attitude spline require the same amount of representational power as the translation spline? And in cases like this one, if our only signal is reprojection error, how should we separate the effects of attitude and translation on the measurements? These are interesting and important questions that must be addressed.

Finally, we would like to comment on the computational complexity of the basis function approach compared to a discrete-time implementation. The time spent solving a nonlinear optimization problem using Gauss–Newton can be sorted into four categories: initialization, error evaluation, Jacobian evaluation and linear system solution. We will discuss each of these individually.

1. *Initialization*. In discrete time it is common to use a linear method to produce the initial guess before nonlinear optimization. When using the basis function approach one has to perform an additional spline fit by solving a large sparse linear system of equations (e.g. Section 6.1). As such, spline-based methods will accrue additional computational load. For B-splines the computational cost of this initial fitting step does not significantly increase with spline order and the number of unknowns is less than or equal to the number of parameters in the full nonlinear least-square problem. Consequently, this extra step is no more expensive than a single iteration of Gauss–Newton.
2. *Error evaluation*. In a discrete-time approach, each quantity required for error evaluation is directly available as a parameter in the estimation problem. Using

the basis function approach, these quantities must be built by evaluating the curve equations (e.g. equation (60)). If the evaluation time is not changing during optimization much of this work can be cached, such as the process of building the matrices in equation (49). If the evaluation time is changing, as it does when estimating time offsets (Furgale et al., 2013), extra work is needed to build these matrices every time the error is evaluated. Although each individual error equation may not require much extra computation this additional work can become significant when a problem has many error equations.

3. *Jacobian evaluation*. The situation for Jacobian evaluation is similar; additional work is required to compute the Jacobian of the quantity being estimated (e.g. equation (72)). For many nonlinear estimation problems in robotics the solution time is dominated by the evaluation of Jacobians (see, for example, Sibley et al. (2009)). Depending on the specific problem, this may represent a significant drawback to the basis function formulation.
4. *Linear system solution*. There are two factors that affect the computational complexity of the solution of the linear system in equation (22): the number of parameters to be estimated and matrix sparsity. As discussed above, basis function approaches have the potential to drastically reduce the size of this system of equations by reducing the number of parameters that need to be estimated. This reduction always results in a speedup; however, basis function approaches will generally produce matrices that are less sparse than their discrete-time counterparts because equations such as (72) produce multidimensional block diagonals in (22). Higher-order B-splines produce wider block diagonals. The complexity of solving these block-banded systems remains linear, but at a higher cost than purely block-diagonal systems. That said, if your problem has a motion model, off-diagonal entries will exist even for discrete-time approaches. The effect of changing sparsity patterns on solution time is difficult to predict for all but the simplest cases, so the tradeoff between discrete- and continuous-time approaches is something that must be empirically tested for each specific problem.

## 8. Conclusion and future work

In this paper, we have presented a derivation of the SLAM problem in continuous time. We showed how to derive an MAP estimator for this problem by first assuming the probability densities and processes involved are Gaussian, representing the state as a weighted sum of continuous temporal basis functions and estimating the state and other parameters using the Gauss–Newton method. We then presented one possible implementation of this solution method using B-splines to represent the robot state and evaluated it by implementing a batch estimator addressing (i) the calibration of a camera and IMU rigidly mounted to the same sensor head and (ii) the localization from a perspective rolling-shutter camera. To the best of our knowledge, this is the first derivation of this kind presented in robotics and we have made every attempt to present our work in steps of increasingly specific assumptions to encourage the development of different continuous-time estimators that make different assumptions.

There are many possible avenues of future work, both theoretical and practical. It will be important to explore the tradeoffs between state-of-the-art discrete-time estimators and continuous-time estimators, both parametric as presented in this paper and non-parametric (Anderson et al., 2015a; Tong et al., 2013), when applied to the canonical problems in robotics. There is also further work to be done on the use of temporal basis functions to represent robot states, especially on the choice of basis and knot sequence. Finally, we would like to see basis-function-formulated SLAM make the leap to an online algorithm (much the way bundle adjustment did (Konolige et al., 2010; Sibley et al., 2010)) as we believe it presents many benefits in terms of reduced state variable size and easy handling of asynchronous measurements (Bibby and Reid, 2010).

**Acknowledgements**

Thanks are due to the great many people who influenced this work with challenging and thought-provoking discussion including Mike Bosse, Jörn Rehder, Luc Oth, Jon Kelly, Hannes Sommer and many others.

**Funding**

This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant number 269916 (V-Charge) and grant number 610603 (EUROPA2).

## References

Ait-Aider O and Berry F (2009) Structure and kinematics triangulation with a rolling shutter stereo rig. In: *Proceedings of the IEEE international conference on computer vision (ICCV)*, pp. 1835–1840.

Alismail H, Baker LD and Browning B (2014) Continuous trajectory estimation for 3D SLAM from actuated lidar. In: *Proceedings of the IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, pp. 6096–6101.

Almqvist H, Magnusson M, Stoyanov T, et al. (2013) Improving point-cloud accuracy from a moving platform in field operations. In: *Proceedings of the international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, pp. 725–730.

Anderson S and Barfoot TD (2013a) Ransac for motion-distorted 3D visual sensors. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, 3–7 November, pp. 2093–2099.

Anderson S and Barfoot TD (2013b) Towards relative continuous-time SLAM. In: *Proceedings of the IEEE international* *conference on robotics and automation (ICRA)*, Karlsruhe, Germany, pp. 1025–1032.

Anderson S, Dellaert F and Barfoot TD (2014) A hierarchical wavelet decomposition for continuous-time slam. In: *Proceedings of the IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–7 June, pp. 373–380.

Anderson S, Barfoot TD, Tong CH, et al. (2015a) Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression. *Autonomous Robots*. (in press).

Anderson S, MacTavish K and Barfoot TD (2015b) Relative continuous-time SLAM. *International Journal of Robotics Research*. (in press).

Bailey T and Durrant-Whyte H (2006) Simultaneous localization and mapping (SLAM): Part II. *Robotics & Automation Magazine, IEEE* 13(3): 108–117.

Bar-Shalom Y, Kirubarajan T and Li XR (2002) *Estimation with Applications to Tracking and Navigation*. New York, USA: John Wiley & Sons.

Barfoot TD, Forbes JR and Furgale PT (2011) Pose estimation using linearized rotations and quaternion algebra. *Acta Astronautica* 68(1–2): 101–112.

Barfoot TD and Furgale PT (2014) Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics* 30(3): 679–693.

Barfoot TD, Tong C and Särkkä S (2014) Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In: *Proceedings of robotics: science and systems (RSS)*, Berkeley, USA.

Bartels RH, Beatty JC and Barsky BA (1987) *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Los Altos, USA: Morgan Kaufmann Publishers.

Bauchau O and Trainelli L (2003) The vectorial parameterization of rotation. *Nonlinear Dynamics* 32(1): 71–92.

Bibby C and Reid I (2010) A hybrid slam representation for dynamic marine environments. In: *2010 IEEE international conference on robotics and automation (ICRA)*, Alaska, USA, 3–7 May, pp. 257–264. IEEE.

Bosse M and Zlot R (2009) Continuous 3D scan-matching with a spinning 2D laser. In: *Proceedings of the 2009 IEEE international conference on robotics and automation*, Piscataway, USA, pp. 4244–4251. IEEE.

Bosse M, Zlot R and Flick P (2012) Zebedee: Design of a spring-mounted 3D range sensor with application to mobile mapping. *IEEE Transactions on Robotics* 28(5): 1104–1119.

De Boor C (2001) *A Practical Guide to Splines*. New York, USA: Springer Verlag.

De Boor C (2002) Spline basics. In: *Handbook of Computer Aided Geometric Design*, Amsterdam, The Netherlands: Elsevier Science B.V., pp.141–163.

Dellaert F and Kaess M (2006) Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research (IJRR)* 25(12): 1181–1204.

Dong H and Barfoot TD (2012) Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation. In: *Proceedings of the international conference on field and service robotics (FSR)*, Matsushima, Japan, pp. 327–342.

Durrant-Whyte H and Bailey T (2006) Simultaneous localization and mapping: part I. *Robotics and Automation Magazine, IEEE* 13(2): 99–110.

Fleps M, Mair E, Ruepp O, et al. (2011) Optimization based IMU camera calibration. In: *2011 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3297–3304. IEEE.

Forssén PE and Ringaby E (2010) Rectifying rolling shutter video from hand-held devices. In: *IEEE conference on computer vision and pattern recognition*. San Francisco, USA, pp. 507–514. IEEE.

Furgale P, Barfoot T and Sibley G (2012) Continuous-time batch estimation using temporal basis functions. In: *2012 IEEE international conference on robotics and automation (ICRA)*, pp. 2088–2095.

Furgale P, Rehder J and Siegwart R (2013) Unified temporal and spatial calibration for multi-sensor systems. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, pp. 1280–1286.

Gustafsson F (2010) Particle filter theory and practice with positioning applications. *Aerospace and Electronic Systems Magazine, IEEE* 25(7): 53–82.

Hedborg J, Ringaby E, Forssén PE, et al. (2011) Structure and motion estimation from rolling shutter video. In: *ICCV workshops*, pp. 17–23.

Hedborg J, Forssen PE, Felsberg M, et al. (2012) Rolling shutter bundle adjustment. In: *2012 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1434–1441.

Hol J, Schön T and Gustafsson F (2010) Modeling and calibration of inertial and vision sensors. *The International Journal of Robotics Research* 29(2–3): 231–244.

Hughes PC (1986) *Spacecraft Attitude Dynamics*. New York: John Wiley & Sons.

Jazwinski AH (1970) *Stochastic Processes and Filtering Theory.* New York, USA: Academic Press.

Jia C and Evans BL (2012) Probabilistic 3D motion estimation for rolling shutter video rectification from visual and inertial measurements. In: *Proceedings of the IEEE international workshop on multimedia signal processing*, pp. 203–208.

Jones ES and Soatto S (2011) Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research* 30(4): 407–430.

Jung SH and Taylor C (2001) Camera trajectory estimation using inertial sensor measurements and structure from motion results. In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, (CVPR)*, vol. 2. pp. 732–737.

Kaess M, Johannsson H, Roberts R, et al. (2012) iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research* 31(2): 216–235.

Kalman R and Bucy R (1961) New results in linear filtering and prediction theory. *Transactions of the ASME. Series D, Journal of Basic Engineering* 83: 95–107.

Karpenko A, Jacobs D, Baek J, et al. (2011) Digital video stabilization and rolling shutter correction using gyroscopes. Technical report, Stanford University.

Kelly J and Sukhatme GS (2010) A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors. In: *12th International symposium on experimental robotics*, Delhi, India, pp. 195–209.

Kelly J and Sukhatme GS (2011) Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research* 30(1): 56–79.

Klein G and Murray D (2009) Parallel tracking and mapping on a camera phone. In: *Proceedings of the eighth IEEE and ACM* *international symposium on mixed and augmented reality (ISMAR)*, Orlando, USA, pp. 83–86.

Ko J and Fox D (2011) Learning GP-BayesFilters via gaussian process latent variable models. *Autonomous Robots* 30(1): 3–23.

Konolige K, Bowman J, Chen J, et al. (2010) View-based maps. *The International Journal of Robotics Research* 29(8): 941–957.

Leutenegger S, Furgale P, Rabaud V, et al. (2013) Keyframe-based visual-inertial slam using nonlinear optimization. In: *Proceedings of robotics: science and systems*, Berlin, Germany, pp. 1–24.

Li M, Kim B and Mourikis AI (2013) Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera. In: *Proceedings of the IEEE international conference on robotics and automation*, Karlsruhe, Germany, pp. 4697–4704.

Lovegrove S, Patron-Perez A and Sibley G (2013) Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In: *British machine vision conference (BMVC)*, pp. 1–12.

Mair E, Fleps M, Suppa M, et al. Spatio-temporal initialization for IMU to camera registration. In: *2011 IEEE international conference on robotics and biomimetics (ROBIO)*, pp. 557–564. IEEE.

Mirzaei F and Roumeliotis S (2008) A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics* 24(5): 1143–1156.

Nocedal J and Wright SJ (2006) *Numerical Optimization*. vol. 2. New York: Springer.

Nützi G, Weiss S, Scaramuzza D, et al. Fusion of IMU and vision for absolute scale estimation in monocular SLAM. *Journal of Intelligent and Robotic Systems* 61: 287–299.

Oth L, Furgale P, Kneip L, et al. (2013) Rolling shutter camera calibration. In: *2013 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1360–1367.

Patron-Perez A, Lovegrove S and Sibley G (2015) A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision* 113(3): 1–12.

Plagemann C, Kersting K, Pfaff P, et al. (2007) Gaussian beam processes: A nonparametric Bayesian measurement model for range finders. In: *Proceedings of robotics: science and systems (RSS)*, Atlanta, USA, pp. 137–144.

Qin K (2000) General matrix representations for B-splines. *The Visual Computer* 16: 177–186.

Rasmussen CE and Williams C (2006) *Gaussian Processes for Machine Learning*. Cambridge, USA: MIT Press.

Rehder J, Beardsley P, Siegwart R, et al. (2014) Spatio-temporal laser to visual/inertial calibration with applications to handheld, large scale scanning. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Chicago, USA, pp. 459–465.

Ringaby E and Forssén PE (2011) Scan rectification for structured light range sensors with rolling shutters. In: *IEEE international conference on computer vision*. Barcelona, Spain, pp. 1575–1582. IEEE Computer Society.

Ringaby E and Forssén PE (2012) Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision* 96(3): 335–352.

Rosen DM, Huang G and Leonard JJ (2014) Inference over heterogeneous finite-/infinite-dimensional systems using factor graphs and gaussian processes. In: *Proceedings of the IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, pp. 1261–1268.

Sheehan M, Harrison A and Newman P (2013) Continuous vehicle localisation using sparse 3d sensing, kernelised rényi distance and fast gauss transforms. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, pp. 398–405.

Sibley G, Mei C, Reid I and Newman P (2009) Adaptive relative bundle adjustment. In: *Robotics science and systems (RSS)*, Seattle, USA.

Sibley G, Mei C, Reid I, et al. (2010) Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research* 29(8): 958–980.

Smith RC and Cheeseman P (1986) On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research* 5(4): 56–68.

Smith RC, Self M and Cheeseman P (1990) Estimating uncertain spatial relationships in robotics. In: Cox IJ and Wilfong GT (eds) *Autonomous Robot Vehicles*. New York: Springer Verlag, pp.167–193.

Strasdat H, Montiel J and Davison A (2010) Real-time monocular SLAM: Why filter? In: *2010 IEEE international conference on robotics and automation (ICRA)*, Anchorage, USA, pp. 2657–2664. IEEE.

Thrun S, Burgard W and Fox D (2001) *Probabilistic Robotics*. Cambridge, USA: The MIT Press.

Thrun S and Montemerlo M (2006) The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research* 25(5–6): 403–429.

Thrun S and Leonard JJ (2008) Simultaneous localization and mapping. In: Siciliano B and Khatib O (eds) *Springer Handbook of Robotics*. Berlin: Springer, pp. 871–889.

Tong C, Furgale PT and Barfoot TD (2012) Gaussian process Gauss–Newton: Non-parametric state estimation. In: *9th Canadian conference on computer and robot vision (CRV)*, Toronto, Canada, pp. 206–213.

Tong C and Barfoot TD (2013) Gaussian Process Gauss–Newton for 3D laser-based visual odometry. In: *Proceedings of the IEEE conference on robotics and automation (ICRA)*, Karlsruhe, Germany, pp. 5184–5191.

Tong C, Furgale P and Barfoot TD (2013) Gaussian Process Gauss–Newton for non-parametric simultaneous localization and mapping. *International Journal of Robotics Research (IJRR)* 32(5): 507–525.

Tong C, Anderson S, Dong H, et al. (2014) Pose interpolation for laser-based visual odometry. *Journal of Field Robotics* 31(5): 731–757.

Yan X, Indelman V and Boots B (2014) Incremental sparse gp regression for continuous-time trajectory estimation & mapping. In: *Neural information processing systems (NIPS) workshop on autonomously learning robots*, Montreal, Canada.

Zhang J and Singh S (2014) Loam: Lidar odometry and mapping in real-time. In: *Proceedings of robotics: science and systems*, Berkeley, USA, pp. 161–195.

Zhang Z (1998) Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision* 27(2): 161–195.

Zlot R and Bosse M (2012) Efficient large-scale 3D mobile mapping and surface reconstruction of an underground mine. In: *Proceedings of the international conference on field and service robotics (FSR)*, Matsushima, Japan, pp. 479–493.