Int J Comput Vis (2015) 113:208–219
DOI 10.1007/s11263-015-0811-3

A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras

Alonso Patron-Perez · Steven Lovegrove · Gabe Sibley

Received: 12 May 2014 / Accepted: 16 February 2015 / Published online: 27 February 2015

Abstract The use of multiple sensors for ego-motion estimation is an approach often used to provide more accurate and robust results. However, when representing ego-motion as a discrete series of poses, fusing information of unsynchronized sensors is not straightforward. The framework described in this paper aims to provide a unified solution for solving ego-motion estimation problems involving high-rate unsynchronized devices. Instead of a discrete-time pose representation, we present a continuous-time formulation that makes use of cumulative cubic B-Splines parameterized in the Lie Algebra of the group $\mathbb{SE}3$. This trajectory representation has several advantages for sensor fusion: (1) it has local control, which enables sliding window implementations; (2) it is C^2 continuous, allowing predictions of inertial measurements; (3) it closely matches torque-minimal motions; (4) it has no singularities when representing rotations; (5) it easily handles measurements from multiple sensors arriving a different times when timestamps are available; and (6) it deals with rolling shutter cameras naturally. We apply this continuous-time framework to visual–inertial simultaneous localization and mapping and show that it can also be used to calibrate the entire system.

Communicated by Tilo Burghardt , Majid Mirmehdi, Walterio Mayol, Dima Damen.

A. Patron-Perez (✉)
Flyby Media Inc., New York, NY, USA
e-mail: alonso@flybymedia.com

S. Lovegrove
Surreal Vision, London, UK
e-mail: steven@surreal.vision

G. Sibley
Autonomous Robotics and Perception Group,
University of Colorado Boulder, Boulder, CO, USA
e-mail: gsibley@colorado.edu

Keywords Sensor fusion · Visual–inertial · SLAM · Rolling shutter · Calibration

1 Introduction

In many emerging fields, applications have an increased dependence on precise location information. Advances in structure from motion (SfM) and simultaneous localization and mapping (SLAM) research has stemmed largely from the vision and robotics community's interest in building and navigating within a representation of the world built on the fly with no prior information. With the advent of autonomous vehicles, automotive manufacturers are also interested in this capability, as are makers of personal electronic devices who would like to offer more immersive and location aware services. Platforms making use of SLAM may rely on one or more of a range of sensors, commonly vision and/or laser rangers, radar, and inertial measurement units (IMUs), but these have tended to be costly and bespoke. As SLAM begins to be commercialized, manufacturers are interested in reducing its cost and weight whilst increasing its performance envelop, which motivates computer vision solutions such as structure from motion.

In this paper, a method for performing SLAM robustly using inexpensive sensors such as rolling shutter CMOS cameras and MEMS IMUs is described. This method uses a continuous-time model for the trajectory of the camera that naturally allows us to fuse information from many unsynchronized and potentially high-rate sensors whilst limiting state size (here the term unsynchronized is used to describe sensors that capture data at different rates but have a common timestamp system). The rolling shutter of a camera is modelled explicitly (Fig. 1) and can form errors generatively on inertial measurements. This model is not limited to visual–

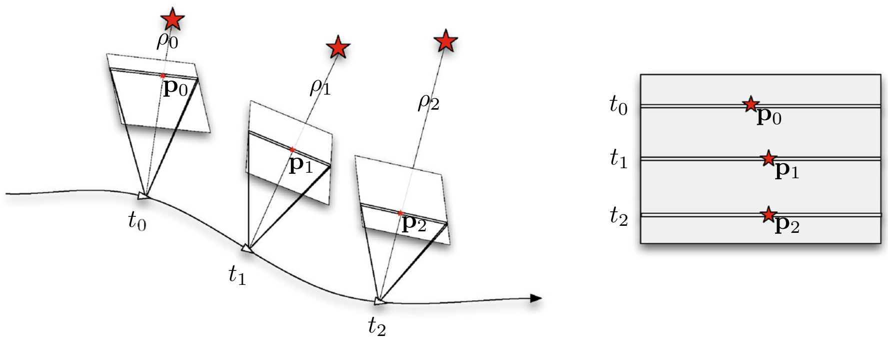


Fig. 1 Rolling shutter cameras are easily modeled with continuous-time SLAM. Landmarks observed at pixel locations $\mathbf{p}_i$ are represented by their inverse depth, ρ_i and time of measurement, t_i. Each scanline is effectively a single push-broom camera (*left*); such scanline-camera measurements are captured over time, which defines the image returned by the actual sensor (*right*) (Lovegrove et al. 2013)

inertial SLAM and may also simplify integration of other sensors such as spinning SICK Laser rangers.

Another important application of the method is unsynchronized multi-sensor intrinsic and extrinsic least-squares calibration. By defining the pose of all sensors within a continuous-time model, all sensor parameters can be estimated jointly, including those that measure time derivatives such as gyroscopes and accelerometers. Given approximate initial sensor configurations, we can find very accurate sensor parameters. This paper is based on a previous publication (Lovegrove et al. 2013), which has been extended to include more detailed explanations and results of the system running with real-world data.

In summary, the contributions of this paper are:

– A spline-based trajectory representation in $\mathbb{SE}3$ that can be used to fuse information from different sensors (Sect. 3).
– A description of how to generate inertial and visual predictions under this framework, including for the case of rolling shutter cameras (Sect. 4).
– A demonstration of self-calibration of a visual–inertial system (Sect. 6.3).

2 Related Work

A number of real-time visual odometry and SLAM systems have been proposed previously, including those that operate on stereo (Comport et al. 2007; Mei et al. 2010; Strasdat et al. 2011) and monocular sequences (Davison 2003; Klein and Murray 2008; Strasdat et al. 2010; Newcombe et al. 2011). These works have focused only on global shutter cameras where all pixels of an image are exposed over the same duration. Treatment of rolling shutter devices is less mature, and has largely focused on removing its influence to create the equivalent global shutter image without distortion (Baker et al. 2010; Jia and Evans 2012). These results can then be fed into a standard visual SLAM pipeline, but at the price of decoupling the two optimization procedures, potentially introducing uncorrectable biases and increasing processing costs. Our approach, in contrast, seeks to leverage rolling shutter directly in the primary optimization. Using rolling shutter measurements directly can be seen as a benefit since measurements are better distributed in time. Additionally, higher quality rolling shutter sensors can be produced much more cheaply than equivalent global shutter sensors since less circuitry is required per pixel and thus more light can be captured.

Among the first to tackle rolling shutter artifacts for visual SLAM were Klein and Murray (2009) when porting PTAM (Parallel Tracking and Mapping) to the iPhone. They use feature tracks over consecutive images to estimate instantaneous rotational velocities for each of the keyframes, and they use this to predict a first order correction to measurements. Hedborg et al. (2012) propose a model that is quite similar but explicitly embedded within the adjustment for rolling shutter cameras. They explicitly model the effect in video sequences by expressing the camera's pose whilst exposing line l as a linear interpolation of neighboring 'key' poses.

Monocular visual SLAM poses a further problem, that of scale drift. When observing landmarks from a single camera, reconstruction is only valid up to some unknown scale, which is often fixed arbitrarily during initialization. When traveling any distance, monocular systems experience drift in scale as well as positional drift since these cannot be observed directly, only through the chain of measurements to wherever it may have been fixed. Strasdat et al. (2010) propose to express this scale as an explicit parameter when closing loops, aiding loop closures across scales and subsequent graph relaxations.

Another approach to fixing scale drift is to augment the platform with a device capable of measuring absolute scale. Nuetzi et al. (2010) compare scale correction within both an Extended Kalman Filter (EKF) and spline fitting approach, though optimization does not exist on the spline but is used to adjust scale to match an IMU. The observability of scale in visual–inertial calibration was demonstrated by Kelly and

Sukhatme (2010). A different observability analysis of this problem was given by Jones et al. (2007), who propose a robust version of an EKF to simultaneously calibrate the system and merge measurements. The authors show that, when provided with visual measurements and non-constant acceleration, the gravity vector becomes observable. However, they also note that fixing the initial pose is not sufficient to achieve full observability of the problem since this constraint is not actively enforced by the filter. The proposed solution is to fix the global reference frame by selecting three points in the first image and tracking them during the whole sequence. Mirzaei and Roumeliotis (2008) presented a similar filter-based visual–inertial calibration method, and provided an observability analysis based on Lie derivatives. This method requires a calibration target but achieves a high level of accuracy.

All of the previously mentioned methods use discrete-time state representations. A 2D spline-based continuous trajectory representation for a SLAM system is introduced by Bibby and Reid (2010), where splines are used to represent the trajectories of tracked objects, providing a substantial reduction of the state space. Closer to our approach is Furgale et al. (2012), where a continuous representation is used. Like ours, their method is based on a spline representation of the vehicle trajectory. The authors demonstrate that by using a continuous-time representation, significant parameter reduction could be achieved, and visual/inertial objectives could be unified within their least squares minimization. In their framework, rotation and translation components are parameterized by independent splines. Although a great proof of concept, the representation did not lend itself to accurate representation with splines because of the choice to parameterize pose using the Cayley–Gibbs–Rodrigues formulation and then to interpolate in this space. This representation suffers from several drawbacks: (1) it has a singularity at 180°, (2) interpolations in this space will not reflect minimum distances in the group of rotations, and (3) it does not well approximate torque-minimum trajectories. Using the same parameterization, Anderson and Barfoot (2013) showed a relative spline representation for visual SLAM that could be extended to handle loop-closure events. The approach presented here employs a spline parameterization of rotation-translation that does not suffer from the problems outlined above.

3 Spline-Based Trajectories

Central to our approach is a continuous-time trajectory representation. For this representation to be practical in a visual–inertial SLAM system as well as serve as a unifying framework for fusing information from other sensors, it should have certain characteristics:

1. Local control, allowing the system to function online as well as in batch.
2. C^2 continuity, to enable inertial predictions.
3. Good approximation of minimal torque trajectories.
4. A parameterization of rigid-body motion devoid of singularities.

A well-known representation for continuous trajectories in $\mathbb{R}^3$ are B-Splines. B-Splines provide local control, and cubic B-Splines are C^2-continuous in $\mathbb{R}^3$. However, B-Splines are not so easily applied when dealing with 3D rotations, such as interpolation in $\mathbb{SO}3$. Methods for interpolating rotations such as piecewise Spherical Linear Interpolation, SLERP (Shoemake 1985), suffer from discontinuities, while Spherical Quadratic Interpolation, SQUAD (Shoemake 1987), does not necessarily preserve C^2 continuity (Kim et al. 1995a). More details on interpolation methods for rotations are provided in Dam et al. (1998).

We choose to parameterize a continuous trajectory using cumulative basis functions (Sect. 3.2) formed using the Lie Algebra $\mathfrak{se}3$ of the matrix group $\mathbb{SE}3$ (Sect. 3.1), equivalent to that proposed by Crouch et al. (1999). This decision is based on two primary factors. First, using cumulative B-Spline basis functions is not only C^2-continuous, but it also provides a very simple second derivative formulation useful for generating inertial predictions. Second, the Lie Algebra parameterization, when applied locally, is free from any singularities and offers a very good analytical approximation to minimum torque trajectories.

3.1 Pose Transformations in $\mathbb{SE}3$

Transformations between sensors' positions are represented by 4×4 matrices. For example, $\mathbf{T}_{b,a}$ represents the matrix that transforms homogeneous points defined in frame a to the equivalent points in frame b, such that $\mathbf{x}_b \propto \mathbf{T}_{b,a}\mathbf{x}_a$. We can decompose $\mathbf{T}_{b,a}$ as follows:

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{R}_{b,a} & \mathbf{a}_b \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_{b,a} \in \mathbb{SE}3, \quad \mathbf{R}_{b,a} \in \mathbb{SO}3,$$

where $\mathbf{R}_{b,a}$ is a 3×3 orthonormal rotation matrix representing rotation-only point transfer between frames a and b. Here, $\mathbf{a}_b$ represents the position of the origin of frame a in the frame of reference b. If $\mathbf{T}_{b,a}$ represents the motion undergone in time Δt seconds between frames $\mathbf{T}_{w,a}$ and $\mathbf{T}_{w,b}$ (where w is a world coordinate frame) traveling at constant angular and linear velocity, this velocity can be expressed in matrix form as

$$\boldsymbol{\Omega} = \frac{1}{\Delta t} \log\left(\mathbf{T}_{b,a}\right), \quad \boldsymbol{\Omega} \in \mathbb{R}^{4\times4},$$

where log is the matrix logarithm. For the matrix group $\mathbb{SE}3$, the logarithmic map and its inverse (the exponential map) can be computed in closed form (Strasdat 2012). By explicitly writing $\mathbf{T}_{b,a} = \mathbf{T}_{w,b}^{-1}\mathbf{T}_{w,a}$, it is easier to see the similarity between $\boldsymbol{\Omega}$ and the computation of velocity in Euclidean space: $\mathbf{v} = \frac{1}{\Delta t}(\mathbf{p}_b - \mathbf{p}_a)$, $\mathbf{v}, \mathbf{p}_a, \mathbf{p}_b \in \mathbb{R}^n$. A point in a trajectory between a and b in Euclidean space is defined by $\mathbf{p}(t) = \mathbf{p}_a + t\mathbf{v}_{b,a}$, $t \in [0, \Delta t]$. The equivalent formulation for $\mathbb{SE}3$ is:

$$\begin{aligned}\mathbf{T}_w(t) &= \mathbf{T}_{w,a}\exp(t\log(\mathbf{T}_{w,b}^{-1}\mathbf{T}_{w,a}))\\ &= \mathbf{T}_{w,a}\exp(t\boldsymbol{\Omega}_{b,a})\end{aligned}$$

3.2 Cumulative Basis B-Spline Representation

Cumulative B-Spline basis functions were first proposed for quaternion interpolation (Kim et al. 1995b) in the context of computer animation. The standard basis function representation of a B-Spline curve of degree $k-1$ in Euclidean space is given by:

$$\mathbf{p}(t) = \sum_{i=0}^{n}\mathbf{p}_i B_{i,k}(t), \tag{1}$$

where $\mathbf{p}_i \in \mathbb{R}^N$ are control points at times t_i, $i \in [0, \dots, n]$ and $B_{i,k}(t)$ are basis functions (Fig. 2a), which can be computed with the De Boor - Cox recursive formula (Boor 1972; Cox 1972). Eq. (1) can be reorganized into its cumulative form as done by Kim et al. (1995b):

$$\mathbf{p}(t) = \mathbf{p}_0\tilde{B}_{0,k}(t) + \sum_{i=1}^{n}(\mathbf{p}_i - \mathbf{p}_{i-1})\tilde{B}_{i,k}(t), \tag{2}$$

where $\tilde{B}_{i,k}(t) = \sum_{j=i}^{n} B_{j,k}(t)$ are cumulative basis functions (Fig. 2b). Although apparently different, these two formulations are equivalent as can be seen in Fig. 2c and d. Making use of the logarithmic and exponential maps described in the previous section, we can re-write Eq. (2) to describe trajectories in $\mathbb{SE}3$ by substituting the control point differences with the logarithmic map $\boldsymbol{\Omega}_i = \log(\mathbf{T}_{w,i-1}^{-1}\mathbf{T}_{w,i}) \in \mathfrak{se}3$ between control poses, and the summation by multiplication of exponents:

$$\mathbf{T}_{w,s}(t) = \exp\left(\tilde{B}_{0,k}(t)\log(\mathbf{T}_{w,0})\right)\prod_{i=1}^{n}\exp\left(\tilde{B}_{i,k}(t)\boldsymbol{\Omega}_i\right), \tag{3}$$

where $\mathbf{T}_{w,s}(t) \in \mathbb{SE}3$ is the pose along the spline at time t, and $\mathbf{T}_{w,i} \in \mathbb{SE}3$ are the control poses. We use the subscript w to emphasize that the pose at time t and control poses are given in the world coordinate frame.

3.3 Cumulative Cubic B-Splines

Because of the C^2 continuity requirement, we use cumulative *cubic* B-Splines ($k = 4$) to represent the trajectory. Control points are assumed to be distributed at uniform time intervals Δt. In a cubic B-Spline, four control points influence the value of the spline curve at time t. These control points are defined to be the set $[t_{i-1}, t_i, t_{i+1}, t_{i+2}]$ for $t \in [t_i, t_{i+1})$. The notation can be further simplified by using a uniform time representation $s(t) = (t - t_0)/\Delta t$. This transforms the control point times t_i into uniform times $s_i \in [0, 1, \dots, n]$. Given a time $s_i \leq s(t) < s_{i+1}$ we define $u(t) = s(t) - s_i$. Using this time formulation and based on the matrix representation of the De Boor–Cox formula (Qin 2000), we can write the matrix representation of a cubic cumulative basis $\tilde{\mathbf{B}}(u)$ and its time derivatives $\dot{\tilde{\mathbf{B}}}(u)$, $\ddot{\tilde{\mathbf{B}}}(u)$ as:

$$\tilde{\mathbf{B}}(u) = \mathbf{C}\begin{bmatrix}1\\u\\u^2\\u^3\end{bmatrix}, \quad \dot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t}\mathbf{C}\begin{bmatrix}0\\1\\2u\\3u^2\end{bmatrix},$$

$$\ddot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t^2}\mathbf{C}\begin{bmatrix}0\\0\\2\\6u\end{bmatrix}, \quad \mathbf{C} = \frac{1}{6}\begin{bmatrix}6 & 0 & 0 & 0\\5 & 3 & -3 & 1\\1 & 3 & 3 & -2\\0 & 0 & 0 & 1\end{bmatrix}$$

From Eq. (3), and using uniform times and matrix notation, the pose in the spline trajectory can now be defined as:

$$\mathbf{T}_{w,s}(u) = \mathbf{T}_{w,i-1}\prod_{j=1}^{3}\exp\left(\tilde{\mathbf{B}}(u)_j\,\boldsymbol{\Omega}_{i+j}\right), \tag{4}$$

where the ith index is taken from the index of the interval where $u(t)$ is defined, the subscript in $\tilde{\mathbf{B}}(u)_j$ indexes the jth element of $\tilde{\mathbf{B}}(u)$ (0 based), and $\boldsymbol{\Omega}$ is defined as before. Note that $\tilde{\mathbf{B}}(u)_0 = 1, \forall u$. We can express first and second derivatives of the pose w.r.t. time as follows:

$$\dot{\mathbf{T}}_{w,s}(u) = \mathbf{T}_{w,i-1}\begin{pmatrix}\dot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2 +\\ \mathbf{A}_0\dot{\mathbf{A}}_1\mathbf{A}_2 +\\ \mathbf{A}_0\mathbf{A}_1\dot{\mathbf{A}}_2\end{pmatrix}, \tag{5}$$

$$\ddot{\mathbf{T}}_{w,s}(u) = \mathbf{T}_{w,i-1}\begin{pmatrix}\ddot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2 + \mathbf{A}_0\ddot{\mathbf{A}}_1\mathbf{A}_2 +\\ \mathbf{A}_0\mathbf{A}_1\ddot{\mathbf{A}}_2 + 2\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2 +\\ 2\dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2 + 2\mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\end{pmatrix}, \tag{6}$$

$$\begin{aligned}\mathbf{A}_j &= \exp\left(\boldsymbol{\Omega}_{i+j}\tilde{\mathbf{B}}(u)_j\right),\\ \dot{\mathbf{A}}_j &= \mathbf{A}_j\boldsymbol{\Omega}_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j,\\ \ddot{\mathbf{A}}_j &= \dot{\mathbf{A}}_j\boldsymbol{\Omega}_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j + \mathbf{A}_j\boldsymbol{\Omega}_{i+j}\ddot{\tilde{\mathbf{B}}}(u)_j\end{aligned}$$

4 Spline as a Generative Model

4.1 Inertial Predictions

The cumulative B-Spline parameterization allows the computation of analytical time derivatives as was shown in the previous section. From these derivatives, accelerometer and gyroscope measurements can be trivially synthesized, which

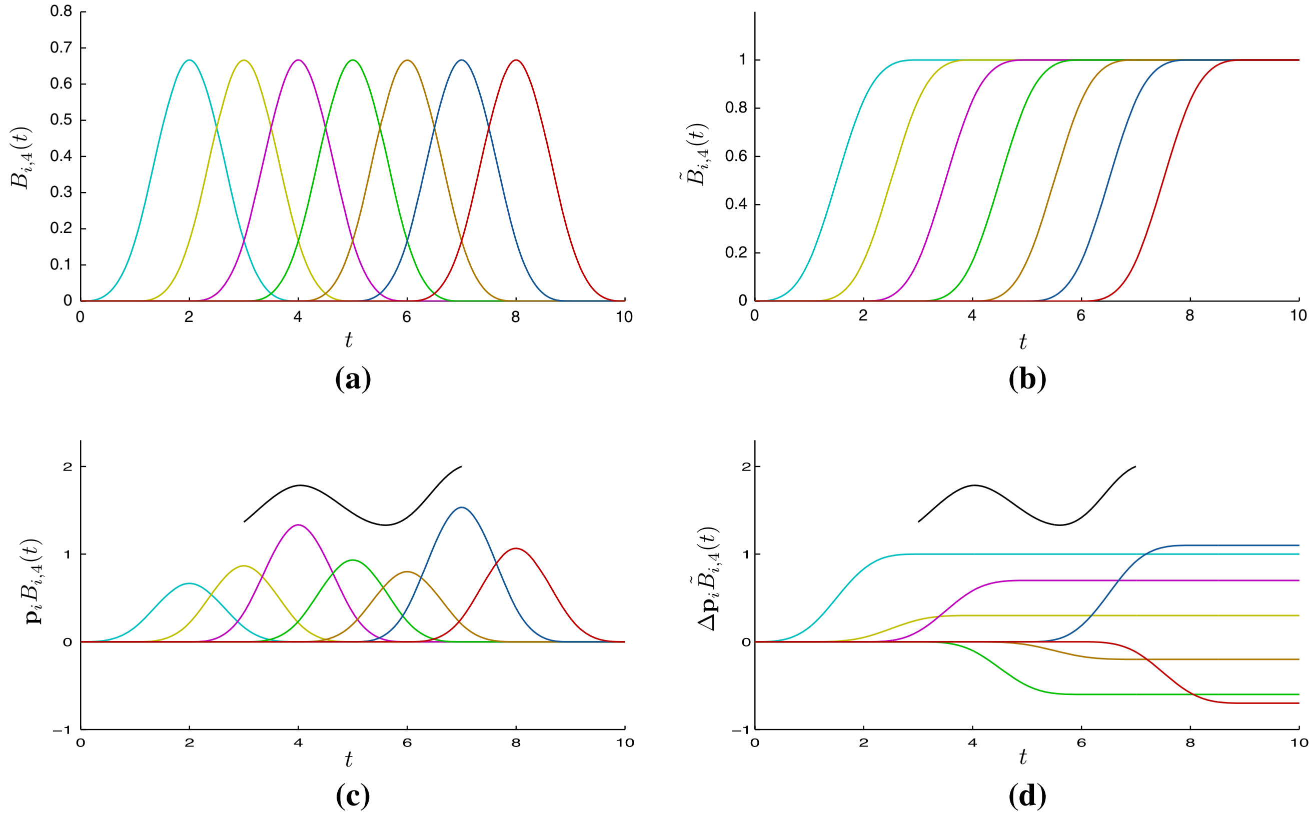


Fig. 2 Cubic basis functions in their standard (**a**), and cumulative forms (**b**). **c** and **d** A segment of a 1D cubic B-Spline (*black curve*) computed using the standard and cumulative basis representations respectively (with the same control points). Both representations produce the same curve

can in turn be used to form direct errors on observed measurements (see Sect. 5). The generative models of angular velocity and acceleration are given by:

$$f_\omega(u) = \mathbf{R}_{w,s}^\intercal(u) \cdot \dot{\mathbf{R}}_{w,s}(u) + \mathbf{b}_\omega, \tag{7}$$

$$f_a(u) = \mathbf{R}_{w,s}^\intercal(u) \cdot (\ddot{\mathbf{s}}_w(u) + g_w) + \mathbf{b}_a, \tag{8}$$

where $\dot{\mathbf{R}}_{w,s}$ and $\ddot{\mathbf{s}}_w$ are the appropriate sub-matrices of $\dot{\mathbf{T}}_{w,s}$ and $\ddot{\mathbf{T}}_{w,s}$ respectively (as defined in Sect. 3.1), $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are gyro and accelerometer biases, and g_w is the acceleration due to gravity in the world coordinate frame. The multiplication by $\mathbf{R}_{w,s}^\intercal(u)$ is applied so that the resulting velocity and acceleration are given in the local coordinate frame.

4.2 Visual Predictions

Landmarks are parameterized in our system using inverse depth $\rho \in \mathbb{R}^+$ from the first observing pose along our spline (Pietzsch 2008). It has been shown that inverse depth representation allows for simple treatment of points at infinity, greatly easing monocular feature initialization (Montiel et al. 2006). We can transfer an inverse depth point observed in a projective camera frame a at image coordinates $\mathbf{p}_a \in \mathbb{R}^2$ to its corresponding image position $\mathbf{p}_b \in \mathbb{R}^2$ in frame b via $\mathcal{W}$:

$$\begin{aligned}\mathbf{p}_b &= \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}, \rho) \\ &= \pi\left([\mathbf{K}_b \mid \mathbf{0}]\, \mathbf{T}_{b,a} \left[\mathbf{K}_a^{-1} \begin{bmatrix}\mathbf{p}_a \\ 1\end{bmatrix}; \rho\right]\right)\end{aligned} \tag{9}$$

where $\pi(\mathbf{P}) = \frac{1}{P_2}[P_0, P_1]^\intercal$, $\mathbf{P} \in \mathbb{R}^3$ is the homogeneous projection function and $\mathbf{K}_a, \mathbf{K}_b \in \mathbb{R}^{3\times3}$ are the camera intrinsics for frames a and b respectively. global shutter assumption (treatment of rolling shutter is given in the following section), $\mathbf{T}_{b,a}$ is defined in our continuous-time spline representation as $\mathbf{T}_{w,s}(t_b)^{-1}\mathbf{T}_{w,s}(t_a)$. t_a and t_b are the times when the corresponding landmark was imaged in frames a and b. Converting these times into uniform times u_a and u_b (Sect. 3.3) and using (4) the transformation from a to b can be recovered.

4.3 Projection into a Rolling Shutter camera

Although the projective geometry of a rolling shutter camera remains the same as that of a global shutter camera, every line of the image is exposed for a different period, each one more delayed than the last. When the camera is in motion, this can cause the image to appear distorted and skewed. Using a continuous-time model for the motion of the camera, we are free to treat every line of the image as its own exposure. Doing so, however, will require that we project each landmark separately into each line. Although each whole pixel has only a single exposure time, our point landmarks are measured by observing the location of their image patches, even though these patches have spatial extent spanning multiple lines.

We instead favor treating the y-axis as a continuous parameterization of time, where sub-pixel values represent differing time intervals. Transfering a known feature $\mathbf{p}_a$ first observed in frame a at fixed time t_a into a frame b at varying time t_b can be written:

$$\mathbf{p}_b(t_b) = \begin{bmatrix} x_b(t_b) \\ y_b(t_b) \end{bmatrix} = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t_b), \rho). \tag{10}$$

In the remainder of this section we drop the subscript b of the time parameter for clarity. This approximation balances patch measurements that occur over multiple lines. At first it seems to complicate projection of a landmark into the camera as we cannot know in advance into which vertical sub-pixel value, and hence instance in time, a landmark will fall. Meingast et al. (2005) analyze this problem of projection into a rolling shutter camera when using a constant velocity motion model under different motions and show that it can be solved analytically in constrained circumstances. Though we do not assume constant velocity trajectories, we follow an approach very similar to their projection under general motion.

Taking a rolling shutter camera whose lines are sampled uniformly in time from the first to the last row, the moment a 2D observation is recorded can be expressed as $y_b(t) = h(t-s)/(e-s)$, where s is the frame start time, e is the end frame time, and h is the height of the image in pixels. We can see that there is no reason that this implied time will match that of Eq. (10). We first project the landmark at a time within the frame interval that corresponds to our best guess of its y-axis location, perhaps initialized from the last frame, or more simply taking the images central y-axis value. We then perform a 1st order Taylor series expansion of the landmark projection around this time for Δt at time t, and similarly for the y-ordinate:

$$\mathbf{p}_b(t+\Delta t) = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho) + \Delta t \frac{d\,\mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{d\,t} \tag{11}$$

$$y_b(t+\Delta t) = \frac{h(t+\Delta t - s)}{e-s}. \tag{12}$$

Equations 11 and 12 can be equated for the y-ordinate and rearranged to solve for Δt:

$$\Delta t = -\frac{ht_0 + s(y_b(t) - h) - ey_b(t)}{(s-e)\dfrac{d\mathcal{W}_y(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{dt} + h} \tag{13}$$

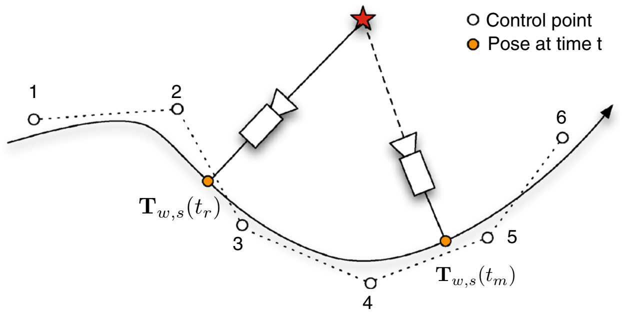


Fig. 3 Reprojection cost. The reference pose (at time t_r) is parameterized by control points 1–4 and the measurement pose (at time t_m) by control points 3–6

This can be repeated, setting $t \leftarrow t + \Delta t$ after every iteration. After only two or three iterations, we find t to converge with high precision to a consistent projection / observation time, even for severe rolling shutter.

5 Model Optimization

An optimization for a visual–inertial SLAM system is presented in this section. First, we define w, s and c as the world, spline and camera reference frames (Fig. 3). For convenience, the reference frame of the inertial sensor has been chosen to be the same as that of the spline (i.e., the spline defines the motion of the IMU). Following our previous notation, a pose on the spline at time t in the world coordinate frame is denoted by $\mathbf{T}_{w,s}(t)$. At any point in time, the location of the camera with respect to the spline (and in our case w.r.t. the IMU) can be obtained by applying the transformation $\mathbf{T}_{s,c}$. This transformation is considered to be fixed at any point of the spline and as such is not dependent on time.

Given our generative models for visual and inertial data, we can solve for our spline and camera parameters in batch or over a window by minimizing an objective function formed from the difference of measured to predicted observations. By using a continuous-time formulation, reprojection errors and inertial errors can be treated uniformly, weighted by their respective information matrices Σ computed from device specifications or calibration. For inverse depth landmark measurements, reprojection errors are constructed between the first measurement of a landmark (reference pose) and subsequent observations (measurement poses). Using the definition of the transfer function $\mathcal{W}$ given in Eq. (9), we need to compute the transformation between the reference and measurement poses. Taking into account the transformations between different reference frames, this is:

$$\mathbf{T}_{m,r} = \mathbf{T}_{c,s}\mathbf{T}_{w,s}(u_m)^{-1}\mathbf{T}_{w,s}(u_r)\mathbf{T}_{s,c}, \tag{14}$$

where u_r and u_m are the uniform times for the reference and measurement poses. Because of the spline representation, the reference and measurement poses depend on four control points. These control points can overlap depending on the time when the measurement was taken (see Fig. 3). Therefore, the number of parameters that are involved in the transfer function can vary. As we receive new visual–inertial measurements, we iteratively attempt to find $\operatorname{argmin}_\theta E(\theta)$:

$$E(\boldsymbol{\theta}) = \sum_{\hat{\mathbf{p}}_m} \left(\hat{\mathbf{p}}_m - \mathcal{W}(\mathbf{p}_r; \mathbf{T}_{m,r}, \rho) \right)^2_{\Sigma_p} + \sum_{\hat{\boldsymbol{\omega}}_m} \left(\hat{\boldsymbol{\omega}}_m - f_\omega(u_m) \right)^2_{\Sigma_{\boldsymbol{\omega}}} + \sum_{\hat{\mathbf{a}}_m} \left(\hat{\mathbf{a}}_m - f_a(u_m) \right)^2_{\Sigma_{\mathbf{a}}} \quad (15)$$

where summations are taken over all visual and inertial measurements $\hat{\mathbf{p}}_m$, $\hat{\boldsymbol{\omega}}_m$ and $\hat{\mathbf{a}}_m$ (or those within a sliding window). u_m is the time when the corresponding measurement (landmark, gyro, or acceleration) occurs. f_ω and f_a are the spline-generated inertial measurements, as defined in Sect. 4.1. θ represents the vector of parameters to optimize (which may vary by application), including spline control poses, camera intrinsics, landmark inverse depth values ρ, camera-to-IMU transformation ($\mathbf{T}_{c,s}$), and IMU biases. We achieve this via iterative non-linear least squares, which requires that we linearize $E(\theta)$, find the minima of this approximate function, update θ, and repeat. We further parameterize pose transform updates by their corresponding Lie Algebra in $\mathfrak{se}3$, see e.g., Strasdat (2012). We use the flexible least squares solver Ceres in order to minimize this objective (Agarwal and Mierle 2012).

6 Experiments

Experiments have been conducted using both simulated and real data to evaluate our flexible continuous-time approach. First, batch optimization of a spline trajectory is shown to match ground truth precisely when given perturbed inertial measurements and visual correspondences with known data association. Sliding window visual odometry results are then shown on a simulated monocular rolling shutter dataset. Next, we demonstrate the systems ability to establish visual–inertial calibration parameters on a real dataset when data association is made more straight-forward by observing a pattern. Finally, the system is validated with real data for joint visual–inertial SLAM and self-calibration.

6.1 Simulated Visual–Inertial Odometry with Known Data Association

We constructed an analytical trajectory and a set of random landmarks on the surface of a sphere from which we could synthesize measurements. The analytical trajectory allowed us to compute time derivatives for use when generating inertial measurements. Figure 4 shows a plot of the batch solution using our method when estimating a continuous spline trajectory and also landmark 3D positions simultaneously. Correspondence of 2D observations to 3D landmarks were given and projected into a virtual rolling shutter 30 fps camera without considering the image formation and association process. Accerometer and gyroscope measurements were generated with 0.1 g and 0.1 rad variance Gaussian noise, respectively. Knots are placed regluarly every 0.1 s. The estimated trajectory and 3D positions match very closely the ground-truth solution and cannot be distinguished visually at this scale. Visual–inertial calibration parameters were known and fixed.

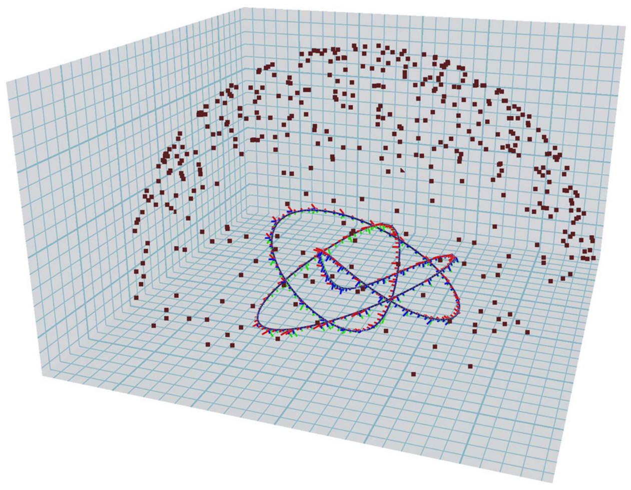


Fig. 4 Simulated rolling shutter, accelerometer and gyroscope measurements and the estimated trajectory and landmark 3D locations. Data association between landmarks is assumed known

In our limited trials, our batch solution was able to converge reliably to the true solution in spite of poor initialization; in our case, each control pose was set to the identity and the inverse depth of 3D points set to 0. A visual only gradient based optimization would have particular problems with a singularity in this configuration, but these are broken by inertial data.

6.2 Simulated Rolling Shutter Visual Odometry with Unknown Data Association

To tackle monocular visual odometry from a rolling shutter camera, a simulated sequence has been generated that exhibits severe rolling shutter effects (Fig. 5a). This dataset simulates a person walking quickly along a city block over 200 frames. The motion was constructed analytically once again in order to derive accurate synthetic inertial measurements.

In order to synthesize a rolling shutter image we used OpenGL, rendering each horizontal line in a new pass using

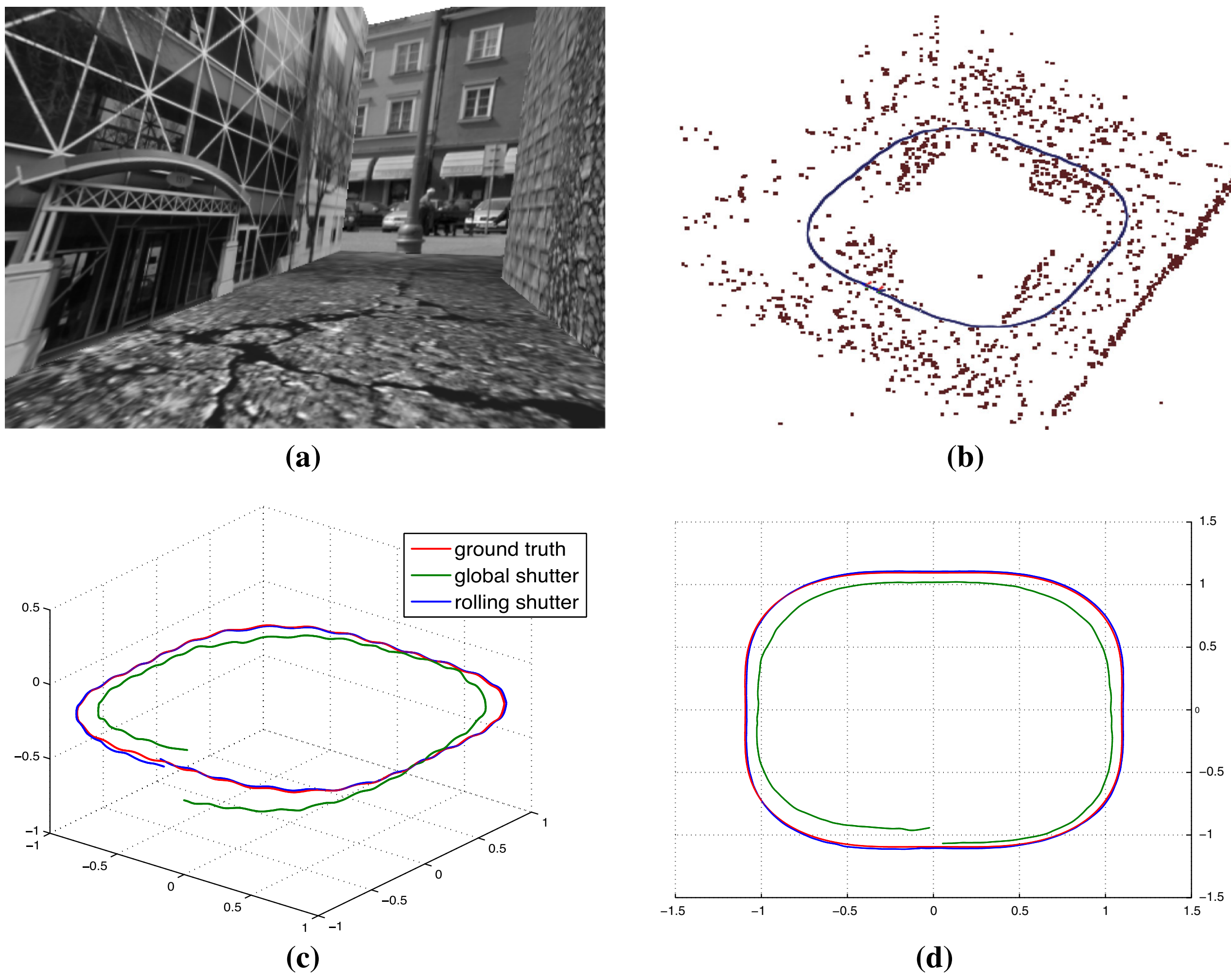


Fig. 5 Results from simulated monocular rolling shutter experiment. **a** Still from simulated sequence exhibiting large rolling shutter warp. **b** Final estimated trajectory and sparse structure when modelling rolling shutter. **c** and **d** Comparison of ground truth trajectory (*red*) against estimated trajectories when modelling (*blue*) and not modelling (*green*) rolling shutter (Lovegrove et al. 2013)

an updated ModelView matrix. Each line represents a different moment in time sampled from the analytical description of the trajectory. We generated 'clean' images without further noise added, though in the future more realistic scenes and aquisition would be interesting for evaluation. With this image data, we were able to test the integration of a SLAM front-end with inperfect data-association into our system.

For constant time operation, we consider a fixed-size sliding window of cubic spline control points, adding new knots as required and optimizing as new visual landmark measurements are made.

To bootstrap the system, we start by using the initialization approach described in Klein and Murray (2007). This method assumes global shutter landmark observations, but it does well enough as an initial guess, particularly when the camera is moving slowly. This provides us with an initial transformation between two selected initialization frames, which can then be used to triangulate features that are added to the spline. Feature points are represented by their 2D coordinates, time (derived from image line) and inverse depth at the point when they were first seen, as described in Sect. 4.2.

When a new frame is captured, the image projection of previously observed landmarks is predicted from the spline using the method described in Sect. 4.3. From these predictions, data association is made by appearance matching using FREAK descriptors (Alahi et al. 2012) against nearby features. The last N control points of the spline are optimized, holding the first three fixed. A very weak motion model is also included at each control point to favor low acceleration—this keeps the spline well constrained even if there are too few measurements within a knot interval. This is implemented as a sum of squares cost over the components of the linear and angular acceleration at the knot, as defined in Eq. 6. If the number of matched landmarks is below a threshold (64

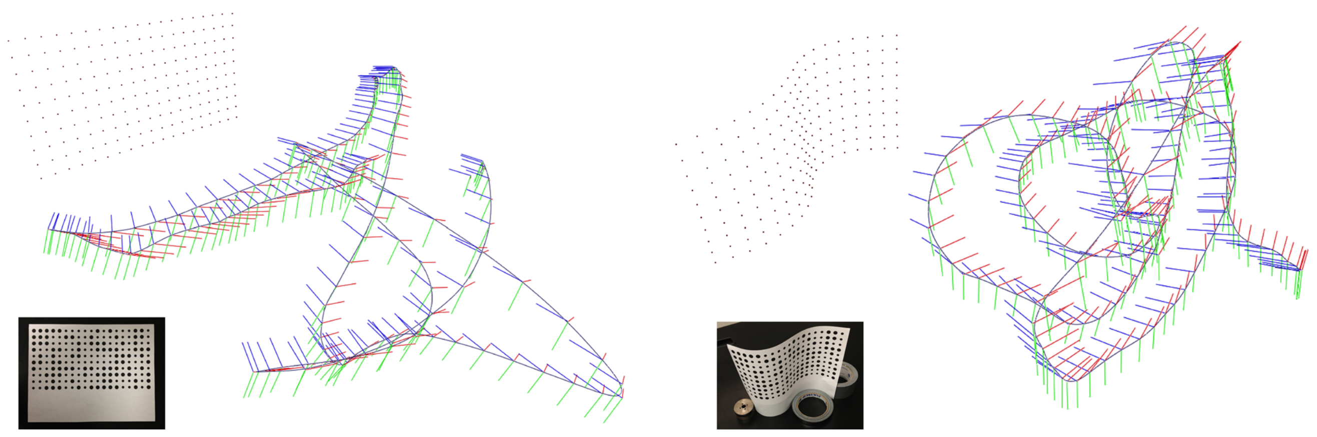


Fig. 6 Example estimates computed for full-SLAM and self-calibration including camera-to-IMU, IMU biases, gravity vector, camera intrinsics (with distortion), 3D landmark locations and sensor trajectory. The grid is only used to simplify data association (Lovegrove et al. 2013)

in our case), new landmarks are initialized at infinity from feature points in the current image and added to the spline.

Figure 5 demonstrates the described method operating on our simulated sequence when modelling rolling shutter and also when incorrectly modelling time as per a global shutter camera. We managed to recover the structure of the simulated city block without serious distortions when rolling shutter was taken into account (Fig. 5b). Ignoring rolling shutter, a more pronounced drift is observed and, as expected, artifacts in the estimated trajectory are particularly noticeable when rolling shutter distortions are larger (i.e., when turning around a corner). The loop closure position error (normalized by the distance traveled) when modelling rolling shutter is 0.0057, compared to 0.0472 for global shutter.

6.3 Self-Calibration and Scale

Our system can also perform full SLAM and self-calibration. Figure 6 demonstrates two results of self-calibration estimating camera intrinsics, camera-to-IMU extrinsics, bias, gravity vector, platform trajectory, and landmark 3D locations jointly. For this experiment, we assume known point feature data association but unknown 3D location. Data association is established using a grid target, first laid out as a plane and then curved into a more interesting shape (to show the calibration target is not required). An Xtion camera was used to obtain rolling shutter images fixed rigidly to a Microstrain 3DM-GX3-35 for inertial data. Images were captured at 30fps while IMU measurements were sampled at 100Hz.

To initialize the system, all visual and inertial measurements are added to the spline in batch, and all control knots are set to the identity transform (control points are added every 0.1 s). The inverse depth of all observed feature points is set to 0, representing a point at infinity along its ray. Camera intrinsics are initialized to a central principle point and arbitrary focal length. The camera-to-IMU transform is initialized to a fair guess by hand (within 45 degrees). While not required, the gravity vector can be initialized to a sensible value by assuming that the camera is stationary at the very start and reading the initial accelerometer value. This is a good approximation for the down direction even if the camera is not stationary, providing it is not undergoing severe accelerations.

A purely monocular system would be unable to initialize using a single frame from points initialized at infinity with no estimate of translation, but inertial measurements in this experiment allow the system to converge. Importantly, it should be noted that the IMU provides us with absolute scale, which is not true of a purely monocular configuration. Table 1 shows landmark mean square reprojection error (MSRE) for the two sequences recorded from a rolling shutter camera. Results are shown for calibration when accurately modelling the rolling shutter and when instead assuming a global shutter. We can see that MSRE is reduced when considering the true shutter model, and scale is established more accurately for the planar sequence where we were able to measure the ground truth scale from the pattern using a ruler. The planar sequence lasts for 12.86 s, which is long enough to establish scale accurately—just 0.3 % scale error when considering the rolling shutter model, and 0.6 % when ignoring it.

6.4 Visual Odometry on Realistic Data

To demonstrate the stability of the spline representation, when dealing with more challenging data, preliminary experiments of monocular visual odometry were performed. Neither rolling shutter nor inertial data is present in either sequence. In both experiments, control points are dropped at 0.1 s intervals.

In the first experiment, the Tsukuba dataset (Martull et al. 2012) was employed. This dateset is a realistic 3D rendering of a trajectory around an office environment. Despite being

Table 1 Results of calibration assuming global or rolling shutter for the two sequences depicted in Fig. 6

Sequence	Mod. shutter	MSRE (pix)	Length (cm)
Planar	Global	0.227	25.3094
Planar	Rolling	0.160	25.3746
Curved	Global	0.319	N/A
Curved	Rolling	0.163	N/A

The actual target length for the planar pattern is 25.45 cm, fairly close to both global and rolling shutter estimates

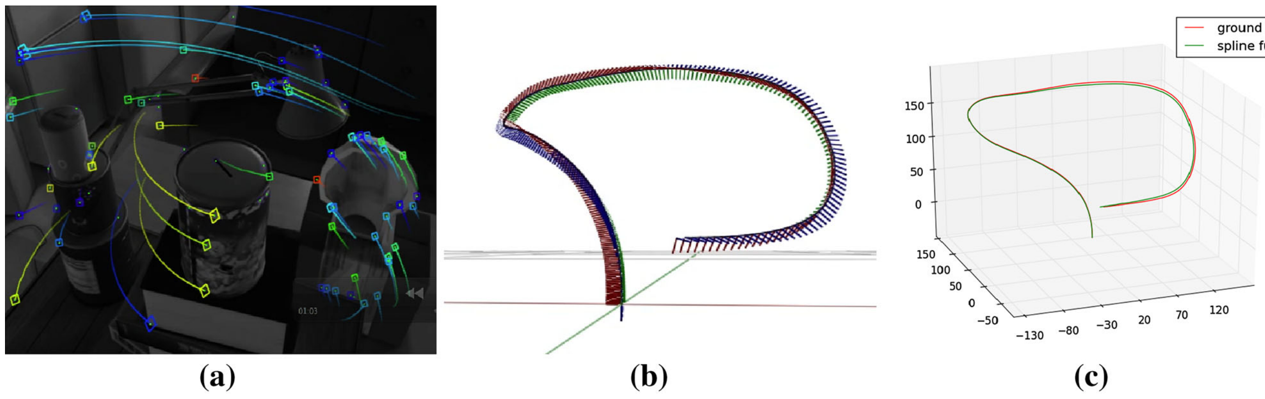
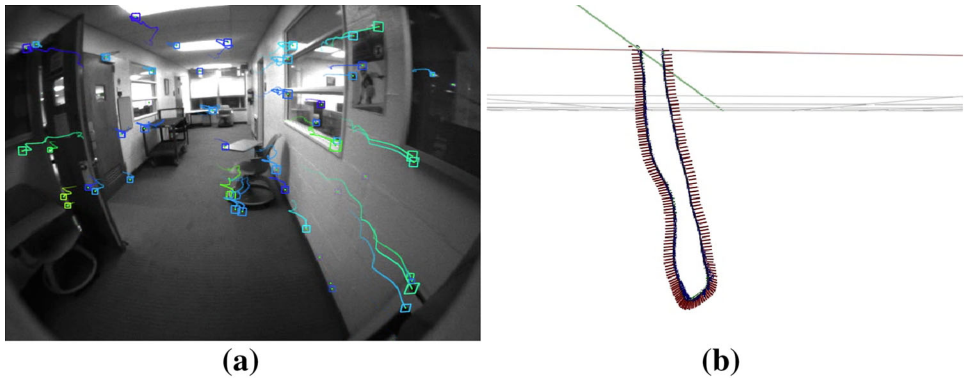


Fig. 7 Results on realistic data from the Tsukuba dataset (Martull et al. 2012). **a** Feature tracks. **b** Estimated continuous trajectory. **c** Comparison with ground truth data, RMSE: 1.96 units

Fig. 8 Results on real data from a hallway. Even without inertial sensing the monocular spline based system starts and maintains scale without specialized initialization maneuvers as is done in Klein and Murray (2008) (this is an arbitrary scale not a metric one)



simulated data, it presents several fast turns. To timestamp the frames, it is assumed a capture rate of 20 fps. In order to compare the resulting trajectory with the provided ground truth, a scale factor has to be computed. This was estimated by calculating the average ratio between the differences in position of the first 100 frames of the ground truth and the the differences in position of the corresponding positions of the estimated trajectory. We then compute the error in position along the trajectory. Figure 7 shows the trajectory estimated after 550 frames and a comparison with the ground truth. The distance travelled over these 550 frames is 791.61 units, and the root-mean-squared error (RMSE) on position is 1.96 units.

A more qualitative experiment was performed with real data collected with a wide angle camera, capturing images at 30fps. This dataset consists of a short loop around a college hallway (Fig. 8). Note that in both experiments no severe scale-drift is observed.

7 Conclusion and Future Work

We have demonstrated in simulation sliding window visual odometry for a rolling shutter sequence based on a continuous-time model for the trajectory of the camera. In this mode, we are able to estimate accurate camera trajectories and scene structure from a single camera alone, and we have shown that ignoring the rolling shutter of the camera leads to poor results compared to our method. We have also shown visual–inertial calibration of camera intrinsics and camera-to-IMU extrinsics from real data given accurate correspondences. Although this calibration is shown with a rolling shutter camera, our method is flexible and naturally supports a wide range of sensors. Recent experiments with real data (Fig. 8) have shown encouraging results, even without the use of an inertial sensor.

In the future we hope to speed up our work to make it applicable for real-time uses. It currently only operates at just a few frames per second within its sliding window mode, but the formulation of the spline does not change the overall complexity when compared to traditional approaches. We hope real-time performance can be obtained through thorough software optimization. One particular optimization would be to take greater care when computing derivatives with respect to spline parameters—we currently make use of Ceres Jet types for computing derivatives automatically, but sparsity exists within these derivatives that cannot automatically be accounted for with this package.

A further line of future work is to estimate any unknown time discrepencies and slew between device clocks. This should theoretically by quite simple within our formulation by taking partial derivatives of the cost function (Eq. 15) with respect to these parameters and estimating how they should change to reduce this objective. The remaining issue is one of book-keeping in order to update the spline interval to which a measurement belongs.

Acknowledgments This work was made possible by generous support from NSF MRI grant 1337722, Toyota Motor Engineering & Manufacturing North America, Inc, and Google, Inc.

References

Agarwal, S., & Mierle, K. (2012). *Ceres solver: Tutorial & reference*. California: Google Inc.

Alahi, A., Ortiz, R. & Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *Conference on Computer Vision and Patter Recognition*.

Anderson, S. & Barfoot, T. D. (2013). Towards releative continuous-time slam. In *IEEE Conference on Robotics and Automation*.

Baker, S., Bennett, E. P., Kang, S. B. & Szeliski, R. (2010). Removing rolling shutter wobble. In *Conference on Computer Vision and Pattern Recognition*.

Bibby, C. & Reid, I. (2010). A hybrid slam representation for dynamic marine environments. In *International Conference on Robotics and Automation*.

Boor, C. D. (1972). On calculating with b-splines. *Journal of Approximation Theory*, *6*, 50–62.

Comport, A. I., Malis, E. & Rives, P. (2007). Accurate quadri-focal tracking for robust 3d visual odometry. In *International Conference on Robotics and Automation*.

Cox, M. G. (1972). The numerical evaluation of b-splines. *Journal of Applied Mathematics*, *10*(2), 134–149.

Crouch, P., Kun, G., & Leite, F. S. (1999). The de casteljau algorithm on lie groups and spheres. *Journal of Dynamical and Control Systems*, *5*(3), 397–429.

Dam, E. B., Koch, M. & Lillholm, M. (1998). Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, University of Copenhagen, Department of Computer Science.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*.

Furgale, P., Barfoot, T.D. & Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *International Conference on Robotics and Automation*.

Hedborg, J., Forssen, P., Felsberg, M. & Ringaby, E. (2012). Rolling shutter bundle adjustment. In *Conference on Computer Vision and Pattern Recognition*.

Jia, C. & Evans, B. L. (2012). Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *International Workshop on Multimedia Signal Processing*.

Jones, E., Vedaldi, A. & Soatto, S. (2007). Inertial structure from motion with autocalibration. In *ICCV Workshop on Dynamical Vision*.

Kelly, J., & Sukhatme, G. S. (2010). Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research*, *30*(1), 56–79.

Kim, M. J., Kim, M. S. & Shin, S. (1995a). A c^2-continuous b-spline quaternion curve interpolating a given sequence of solid orientations. In *Computer Animation*, pp. 72–81.

Kim, M. J., Kim, M. S. & Shin, S. (1995b). A general construction scheme for unit quaternion curves with simple high order derivatives. In *SIGGRAPH*, pp. 369–376.

Klein, G. & Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed and Augmented Reality*.

Klein, G. & Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*.

Klein, G. & Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality*.

Lovegrove, S., Patron-Perez, A. & Sibley, G. (2013). Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *British Machine Vision Conference*.

Martull, S., Martorell, M. P. & Fukui, K. (2012). Realistic cg stereo image dataset with ground truth disparity maps. In *ICPR workshop*.

C, Mei, Sibley, G., Cummins, M., Newman, P., & Reid, I. (2010). RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, *94*, 1–17.

Meingast, M., Geyer, C. & Sastry, S. (2005). Geometric models for rolling shutter cameras. In *OmniVis Workshop*.

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

Montiel, J., Civera, J. & Davison, A. J. (2006). Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems*.

Newcombe, R. A., Lovegrove, S. J. & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *International Conference on Computer Vision*.

Nuetzi, G., Weiss, S., Scaramuzza, D. & Siegwart, R. (2010). Fusion of imu and vision for absolute scale estimation in monocular slam. In *International Conference on Unmanned Aerial Vehicles*.

Pietzsch, T. (2008). Efcient feature parameterisation for visual slam using inverse depth bundles. In *British Machine Vision Conference*.

Qin, K. (2000). General matrix representations for b-splines. *The Visual Computer*, *16*(3–4), 177–186.

Shoemake, K. (1985). Animating rotation with quaternion curves. In *SIGGRAPH*, pp. 245–254.

Shoemake, K. (1987). Quaternion calculus and fast animation. In *SIGGRAPH Course Notes*.

Strasdat, H. (2012). Local accuracy and global consistency for efficient visual slam. Ph.D. thesis, Imperial College London.

Strasdat, H., Montiel, J. M. M. & Davison, A. (2010). Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems*.

Strasdat, H., Davison, A. J., Montiel, J. M. M. & Konolige, K. (2011). Double window optimisation for constant time visual slam. In *International Conference on Computer Vision*.