# Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras

Steven Lovegrove
slovegrove@gwu.edu
Alonso Patron-Perez
apatron@gwu.edu
Gabe Sibley
gsibley@gwu.edu

Department of Computer Science
The George Washington University
Washington, DC, USA

**Abstract**

This paper describes a general continuous-time framework for visual-inertial simultaneous localization and mapping and calibration. We show how to use a spline parameterization that closely matches the torque-minimal motion of the sensor. Compared to traditional discrete-time solutions, the continuous-time formulation is particularly useful for solving problems with high-frame rate sensors and multiple unsynchronized devices. We demonstrate the applicability of the method for multi-sensor visual-inertial SLAM and calibration by accurately establishing the relative pose and internal parameters of multiple unsynchronized devices. We also show the advantages of the approach through evaluation and uniform treatment of both global and rolling shutter cameras within visual and visual-inertial SLAM systems.

## 1 Introduction

In this paper, we describe a method for performing SLAM robustly using inexpensive sensors such as rolling shutter CMOS cameras and MEMS IMUs. We make use of 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. We model the rolling shutter of a camera explicitly (Figure 1) and can form errors generatively on inexpensive measurements. This model is not limited to visual-inertial SLAM and may also simplify integration of other sensors such as spinning SICK Laser rangers.

Another important application of our 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.

A number of real-time visual odometry and SLAM systems have been proposed previously, including those that operate on stereo sequences, such as [6, 20, 29], and monocular

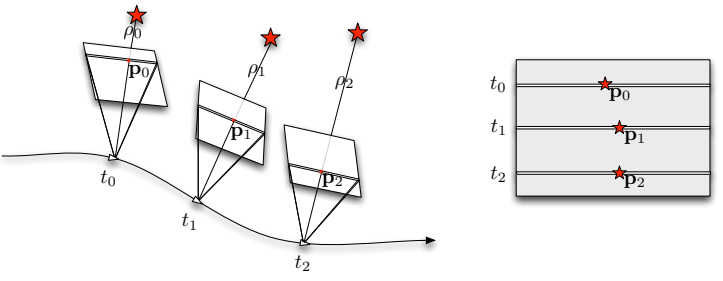


Figure 1: Rolling shutter cameras are easily modeled with continuous-time SLAM. Landmarks observed at pixel locations $\mathbf{p}_i$ are represented by their inverse depth, $\rho_i$ and time of measurement, $t_i$. Each scanline is effectively a single push-broom camera (left); such scanline-camera measurements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

sequences, such as [9, 18, 24, 28]. 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 [3, 12]. 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.

Among the first to tackle rolling shutter artifacts for visual SLAM were Klein *et al.* when porting PTAM to the iPhone [19]. They use feature tracks over consecutive images to estimate instantaneous rotational velocities for each of the keyframes, and use this to predict a first order correction to measurements. Hedborg *et al.* propose quite a similar model, 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 [11].

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.* propose to express this scale as an explicit parameter when closing loops, aiding loop closures across scales and subsequent graph relaxations [28].

Another approach to fixing scale drift is to augment the platform with a device capable of measuring absolute scale. Nuetzi *et al.* compare scale correction within both an Extended Kalman Filter and spline fitting approach, though optimization does not exist on the spline but is used to adjust scale to match an IMU [25]. Kelly *et al.* demonstrate the observability of scale in visual-inertial calibration [14]. A different observability analysis of this problem was given in [13]. They propose a robust version of an EKF to simultaneously calibrate the system and merge measurements. A similar filter-based visual-inertial calibration method was presented in [22], which provides an observability analysis based on Lie derivatives. This method requires a calibration target but achieves a high level of accuracy.

All the previously mentioned methods use discrete time state representations. A 2D spline-based continuous trajectory representation for a SLAM system is introduced in [4],

样条函数插值
where splines are used to represent the trajectories of tracked objects, providing a substantial reduction of the state space. Closer to our approach is [10], where a continuous representation is used. Like ours, their method is based on a spline representation of the vehicle trajectory. Rotation and translation components are parameterized by independent splines. The authors test their approach for camera-IMU calibration, but it requires known landmark locations. Our approach employs a better spline parameterization of rotation-translation, and we also present a framework for joint visual-inertial self-calibration using only feature tracks extracted from a calibration pattern.

# 2 Continuous-time representation

Furgale *et al*. demonstrated 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 [10]. 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: (*i*) it has a singularity at 180°, (*ii*) interpolations in this space will not reflect minimum distances in the group of rotations, and (*iii*) it does not well approximate torque-minimum trajectories. We instead choose the Lie Algebra $\mathfrak{se}3$ of the matrix group $\mathbb{SE}3$ in order to undertake smooth trajectories in the manifold space of rigid body translations and rotations, but we do so only locally, not in a global frame. This parameterization is free from any singularities and offers a very good analytical approximation to minimum torque trajectories. More details follow in section 2.2.

## 2.1 Camera pose transformations

We represent transformations between cameras by the $4 \times 4$ matrix representing homogeneous point transfer between these frames. 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 $3 \times 3$ 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 $\Delta t$ 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 $\Omega = \frac{1}{\Delta t} \log\left(\mathbf{T}_{b,a}\right), \Omega \in \mathbb{R}^{4\times4}$, 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 [7].

## 2.2 $C^2$-continuous curves in $\mathbb{SE}3$

At the heart of our approach lies a continuous trajectory representation. We chose a formulation which offers: (*i*) local control, allowing the system to function online as well as in batch, (*ii*) $C^2$ continuity to enable us to predict IMU measurements, and (*iii*) good approximation of minimal torque trajectories. Cubic B-Splines are a well-known representation for

trajectories in $\mathbb{R}^3$, but are not so easily applied when dealing with 3D rotations, such as interpolation in $\mathbb{SO}3$. For example, $C^2$ continuity is not necessarily preserved [15]. We choose to parameterize a continuous trajectory using cumulative basis functions formed using the Lie Algebra, equivalent to that proposed in [8]. Using cumulative B-Spline basis functions were first proposed for quaternion interpolation in [16] in the context of computer animation. This representation is not only $C^2$ continuous, but it also provides a very simple second derivative formulation.

### 2.2.1 Representing B-Splines with cumulative basis functions

The standard basis function representation of a B-Spline curve of degree $k-1$ 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, which can be computed with the De Boor - Cox recursive formula [5, 7]. Equation 1 can be reorganized into its cumulative form as [16]:

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

where $\tilde{B}_{i,k}(t) = \sum_{j=i}^{n} B_{j,k}(t)$ are cumulative basis functions. Making use of the logarithmic and exponential maps described in the previous section, we can re-write Equation 2 to describe trajectories in $\mathbb{SE}3$ by substituting the control point differences with the logarithmic map $\Omega_i = \log(\mathbf{T}_{w,i-1}^{-1}\mathbf{T}_{w,i}) \in \mathfrak{se}3$ between control poses, and the summation by multiplication of exponents:

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

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.

### 2.2.2 Cumulative cubic B-Splines

In this paper we focus on the particular case of cumulative cubic B-Splines ($k = 4$). We assume that control points are distributed at uniform time intervals $\Delta t$. In a cubic B-Spline, four control points influence the value of the spline curve at time $t$. We define these control points to be the set $[t_{i-1}, t_i, t_{i+1}, t_{i+2}]$ for $t \in [t_i, t_{i+1})$. We can further simplify the notation 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 for the De Boor - Cox formula [26], we can write the matrix representation of a cumulative basis $\tilde{\mathbf{B}}(u)$ and it's time derivatives $\dot{\tilde{\mathbf{B}}}(u), \ddot{\tilde{\mathbf{B}}}(u)$ as:

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

The pose in the spline trajectory can now be defined as:

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

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

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

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

$$\mathbf{A}_j = \exp\left(\Omega_{i+j}\tilde{\mathbf{B}}(u)_j\right), \quad \dot{\mathbf{A}}_j = \mathbf{A}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j,$$

$$\ddot{\mathbf{A}}_j = \dot{\mathbf{A}}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j + \mathbf{A}_j\Omega_{i+j}\ddot{\tilde{\mathbf{B}}}(u)_j$$

# 3 Generative model of visual-inertial data

## 3.1 Parameterization

We parameterize landmarks in our system using inverse depth $\rho \in \mathbb{R}^+$ from the first observing pose along our spline. It has been shown that inverse depth representation allows for simple treatment of points at infinity, greatly easing monocular feature initialization [23]. 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 $\mathcal{W}$:

$$\mathbf{p}_b = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}, \rho) = \pi\left(\left[\mathbf{K}_b \,|\, \mathbf{0}\right]\mathbf{T}_{b,a}\left[\mathbf{K}_a^{-1}\begin{bmatrix}\mathbf{p}_a\\1\end{bmatrix};\rho\right]\right), \tag{7}$$

where $\pi(\mathbf{P}) = \frac{1}{\mathbf{P}_2}[\mathbf{P}_0, \mathbf{P}_1]^\intercal$ 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.

The cumulative B-Spline parameterization allows us to compute analytical time derivatives, as seen in Section 2.2.2. This allows us to trivially synthesize accelerometer and gyroscope measurements, which we can in turn use to form errors on observed measurements.

$$\text{Gyro}(u) = \mathbf{R}_{w,s}^\intercal(u)\cdot\dot{\mathbf{R}}_{w,s}(u) + \text{bias}, \tag{8}$$

$$\text{Accel}(u) = \mathbf{R}_{w,s}^\intercal(u)\cdot(\ddot{\mathbf{s}}_w(u) + g_w) + \text{bias}, \tag{9}$$

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 Section 2.1), and $g_w$ is the acceleration due to gravity in the world coordinate frame.

### 3.2 Minimization

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 $\Sigma$ computed from device specifications or calibration. As we receive new visual-inertial measurements, we iteratively attempt to find $\text{argmin}_\theta E(\theta)$:

$$\begin{aligned} E(\theta) &= \sum_{\hat{\mathbf{p}}_m} \left( \hat{\mathbf{p}}_m - \mathcal{W}(\mathbf{p}_r; \mathbf{T}_{c,s}\mathbf{T}_{w,s}(u_m)^{-1}\mathbf{T}_{w,s}(u_r)\mathbf{T}_{s,c}, \rho) \right)^2_{\Sigma_p} + \\ &\quad \sum_{\hat{\omega}_m} \left( \hat{\omega}_m - \text{Gyro}(u_m) \right)^2_{\Sigma_\omega} + \sum_{\hat{\mathbf{a}}_m} \left( \hat{\mathbf{a}}_m - \text{Accel}(u_m) \right)^2_{\Sigma_\mathbf{a}}, \end{aligned} \tag{10}$$

where summations are taken over all visual and inertial measurements $\hat{\mathbf{p}}_m, \hat{\omega}_m$ and $\hat{\mathbf{a}}_m$ (or those within a sliding window). Measurement $m$ occurs at time $u_m$. For inverse depth landmark measurements, $u_r$ denotes the reference time at which the landmark was first observed. $\theta$ represents the vector of parameters to optimize (which may vary by application), including spline control poses, camera intrinsics, landmark inverse depth values $\rho$, 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 $\theta$ and repeat. We further parameterize pose transform updates by their corresponding Lie Algebra in $\mathfrak{se}3$, see e.g. [27]. We use the flexible least squares solver Ceres in order to minimize this objective [1].

## 4 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 to treat the y-axis as a continuous parameterization of time, where sub-pixel values represent differing time intervals. This approximation balances patch measurements that occur over multiple lines.

$$\mathbf{p}_b(t) = \begin{bmatrix} x_b(t) \\ y_b(t) \end{bmatrix} = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho). \tag{11}$$

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*. 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 [21]. Though we do not assume constant

velocity trajectories, we follow an approach very similar to their projection under general motion. The moment a 2D observation is exposed 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 Equation 11. 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. We then perform a 1st order Taylor series expansion of the landmark projection around this time, and solve for the unknown time discrepancy:

$$\mathbf{p}_b(t+\Delta t) = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho) + \Delta t \frac{d\,\mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{dt} \tag{12}$$

$$y_{b(t+\Delta t)} = \frac{h(t+\Delta t - s)}{e-s}, \quad \Delta t = -\frac{h.t_0 + s.(y_b(t) - h) - e.y_b(t)}{(s-e)\frac{d\,\mathcal{W}_y(\mathbf{p}_a; \mathbf{T}_{b,a}(t), \rho)}{dt} + h} \tag{13}$$

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 Experiments

We have conducted experiments in both simulation and with real data to evaluate our flexible continuous-time approach. We first present sliding window visual odometry results on a simulated monocular rolling shutter dataset. We continue by demonstrating our system on real data for joint visual-inertial SLAM and self-calibration.

## 5.1 Simulated rolling shutter visual odometry

To tackle monocular visual odometry from a rolling shutter camera, we have generated a simulated sequence exhibiting severe rolling shutter effects (Figure 2a). This dataset simulates a person walking quickly through a city block over 200 frames. 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 [17]. 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 Section 3.1.

When a new frame is captured, the image projection of previously observed landmarks is predicted from the spline using the method described in Section 4. From these predictions, data association is made by appearance matching using FREAK descriptors [2] 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. If the number of matched landmarks is below a threshold (64 in our case),

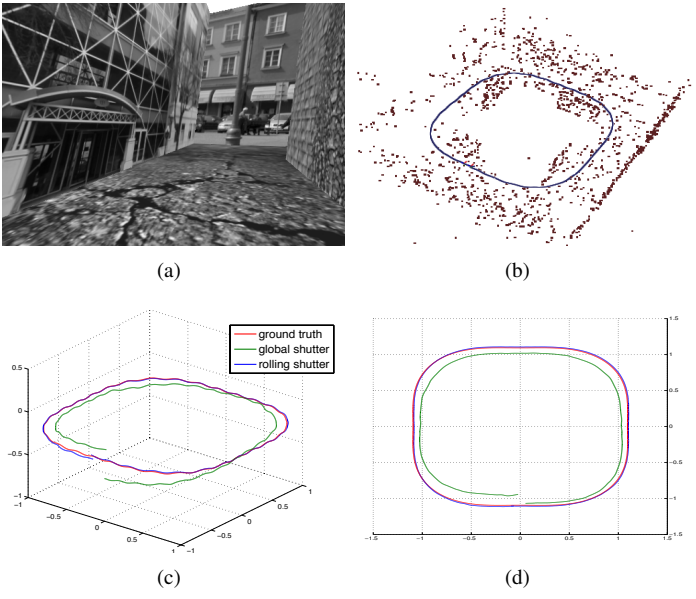


Figure 2: Results from simulated monocular rolling shutter experiment. (a) Still from simulated sequence exhibiting large rolling shutter warp. (b) Final estimated trajectory and sparse structure when modelling rolling shutter. (c) and (d) Comparison of ground truth trajectory (red) against estimated trajectories when modelling (blue) and not modelling (green) rolling shutter.

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

Figure 2 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 (Figure 2b). 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.

## 5.2 Self-calibration and scale

Our system can also perform full SLAM and self-calibration. Figure 3 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). We used an Xtion 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

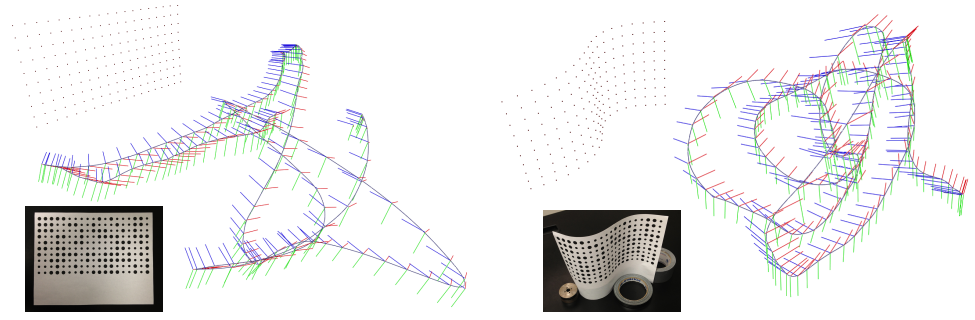

Figure 3: Example estimates computed for full-SLAM and self-calibration including camera-to-IMU, IMU biases, gravity vector, camera intrinsics (with distortion), 3D landmark locations and sensor trajectory. Again, the structure and scale in these figures are accurately estimated without any prior knowledge of the scene. This being our first experimental analysis, full and robust front-end feature tracking has not been completed, and the grid is only used to simplify data-association. This still allows evaluation and demonstration of full SLAM and visual-inertial self-calibration.

| Rolling Shutter Sequence | Modelled Shutter | Landmark MSE (pixels) | 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 |

Table 1: Results of calibration assuming global or rolling shutter for the two sequences depicted in Figure 3. The actual target length for the planar pattern is 25.45cm, fairly close to both global and rolling shutter estimates.

in batch, and all control knots are set to the identity transform (control points are added every 0.1 sec). 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 Error 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 MSE 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 seconds, 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 Conclusions 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 IMU-to-camera 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 wider range of sensors than previous methods. By fusing sensor data jointly within a big window, we reduce the bias that can be induced by linearization within a Kalman filter.

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 a second within its sliding window mode, but the formulation of the spline does not change the overall complexity when compared to traditional approaches.

**Acknowledgements**. This work was made possible by generous support from NSF EAGER grant 1249409 and Toyota Motor Engineering & Manufacturing North America, Inc.

# References

[1] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc., 2012.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Conference on Computer Vision and Pattern Recognition*, 2012.

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

[4] C. Bibby and I. Reid. A hybrid slam representation for dynamic marine environments. In *International Conference on Robotics and Automation*, 2010.

[5] C. De Boor. On calculating with b-splines. *Journal of Approximation Theory*, (6): 50–62, 1972.

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

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

[8] P. Crouch, G. Kun, and F. Silva Leite. The de casteljau algorithm on lie groups and spheres. *Journal of Dynamical and Control Systems*, 5(3):397–429, July 1999.

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

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

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

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

[13] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In *ICCV Workshop on Dynamical Vision*, 2007.

[14] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration". *International Journal of Robotics Research*, 2010.

[15] M-J. Kim, M.-S. Kim, and S. Shin. A $c^2$-continuous b-spline quaternion curve interpolating a given sequence of solid orientations. In *Computer Animation*, 1995.

[16] M-J. Kim, M-S. Kim, and S. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. In *SIGGRAPH*, 1995.

[17] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *International Symposium on Mixed and Augmented Reality*, 2007.

[18] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*, 2008.

[19] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality*, 2009.

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

[21] M. Meingast, C. Geyer, and S. Sastry. Geometric models for rolling shutter cameras. In *OmniVis Workshop*, 2005.

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

[23] J. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Robotics: Science and Systems*, 2006.

[24] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*, 2011.

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

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

[27] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual SLAM*. PhD thesis, Imperial College London, 2012.

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

[29] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. In *International Conference on Computer Vision*, 2011.