Pose Interpolation for Rolling Shutter Cameras using Non Uniformly Time-Sampled B-splines

Bertrand Vandeportaele[1], Philippe-Antoine Gohard[1,2], Michel Devy[1] and Benjamin Coudrin[2]
[1]*LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France*
[2]*Innersense, Ramonville-Saint-Agne, France*
{*bertrand.vandeportaele, philippe-antoine.gohard, michel.devy*}*@laas.fr, benjamin.coudrin@innersense.fr*

Keywords: Rolling Shutter, Camera Geometric Model, Perspective n-Points Algorithm, Simultaneous Localization and Mapping, B-splines interpolation.

Abstract: Rolling Shutter (RS) cameras are predominant in the tablet and smartphone market due to their low cost and small size. However, these cameras require specific geometric models when either the camera or the scene is in motion to account for the sequential exposure of the different lines of the image. This paper proposes to improve a state-of-the-art model for RS cameras through the use of Non Uniformly Time-Sampled B-splines. This allows to interpolate the pose of the camera taking into account the varying dynamic of the motion by adding more control points where needed while keeping a low number of control points where the motion is smooth. Two methods are proposed to determine adequate distributions for the control points, using either an IMU sensor or an iterative reprojection error minimization. Results on simple synthetic data sets are shown to prove the concept and future works are introduced that should lead to the integration of our model in a SLAM algorithm.

1 INTRODUCTION

In Augmented Reality applications for mobile devices (smartphones and tablets), the realtime localization of the device camera and the 3D modeling of the environment are used to integrate virtual elements onto the images of the real environment. This task is usually performed by algorithms of Structure From Motion (SFM: (Hartley and Zisserman, 2004)), and Simultaneaous Localization and Mapping (SLAM: MonoSLAM(Davison, 2003), PTAM(Klein and Murray, 2007) OrbSLAM (Mur-Artal et al., 2015)). Most existing implementations assume that the cameras are using a Global Shutter (GS), for which all the lines of the image are exposed at the same time, ie. the image is a projection of the scene at one instant t.

However, more than 90% of mobile devices are equipped with Rolling Shutter (RS) cameras because of their lower cost and smaller size than the classic GS cameras. The advantages of RS cameras come with some drawbacks; by the way they are designed, they cause image distortions when observing a dynamic scene or when the camera is moving. In these sensors, all the lines of the image are exposed and transferred sequentially at different times.

More complex geometric models are thus required for RS cameras, to account for the the varying pose of the camera. This paper extends the RS Camera model presented in(Steven Lovegrove, 2013) and (Patron-Perez et al., 2015) for the camera pose interpolation using B-splines controlled by Control Points (CP) that are Uniformly Distributed (UD) in time. Our improvement of the model consists in using an adaptive Non Uniform Distribution (NUD) of the CP of the B-spline over time. This allows to globally reduce the number of CP required to model a given trajectory with the same accuracy than using UD.

The paper firstly presents existing SLAM algorithms and exhibits problems arising from using images captured with RS cameras. It then presents the theoretical framework used for the modeling of the RS cameras. First, the cumulative B-splines are introduced to allow the interpolation of 6-dof camera pose in continuous time. Second, the pinhole camera model using interpolated poses is derived. Finally, the iterative minimization used in the Perspective n-Points algorithm is explained. Our contribution focused on the distribution of the CP is then studied and two methods are proposed to determine adequate NUD of the CP, using either information from an IMU sensor or multiple iterations of reprojection error minimization. The two methods are evaluated on sim-

Vandeportaele B., Gohard P., Devy M. and Coudrin B.
Pose Interpolation for Rolling Shutter Cameras using Non Uniformly Time-Sampled B-splines.
DOI: 10.5220/0006171802860293
In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, pages 286-293
ISBN: 978-989-758-227-1

ple synthetic data sets to prove the concept and future works are introduced that should lead to the integration of our model in a SLAM algorithm.

2 RELATED WORK FOR SLAM

2.1 Global Shutter

A SLAM algorithm can recover the position and orientation of a mobile camera. In the Augmented Reality context, these camera parameters can be used to synthesize images of virtual elements as if they were seen by the real camera. To do so without prior knowledge of the environment, it is necessary to map it in realtime.

Early realtime methods used to solve the SLAM problem in the literature were based on Extended Kalman Filter (EKF)(Davison, 2003), (Roussillon et al., 2012), (Gonzalez, 2013). The simplicity of this method and it's computing efficiency for small size environment model has made it the most used SLAM method for the past decade.

Other robust and realtime methods based on local Bundle Adjustment like PTAM (Klein and Murray, 2007) have also been proposed. They minimize the reprojection error over a subset of previously acquired images, called keyframes (Engel et al., 2014), (Mur-Artal et al., 2015), or over a sliding window of frames (Mouragnon et al., 2006). They provide improved robustness thanks to the modeling of outliers, and (Strasdat et al., 2010) proved the superiority of these Bundle Adjustment-based methods over filtering one.

2.2 Rolling Shutter

Using a SLAM method designed for GS camera model with RS camera produces deviations of the estimated trajectory and reconstructed 3D points. These deviations increase with the velocity of the camera. One of the first use of a RS camera model in the SLAM context was the adaptation of PTAM for smartphone (Klein and Murray, 2009). They estimated the angular velocity of the camera at the keyframe using keypoints tracked between the previous and the next frame. This angular velocity was then used to correct the measurements of the points in the image using a first order approximation so they can be used as if they where obtained by a GS camera.

(Hedborg et al., 2011) initially used a similar method, and proposed in (Hedborg et al., 2012) a bundle adjustment using a RS camera model. To estimate the varying camera pose inside a frame, they interpolated independently the rotations (using SLERP) and the translations (using linear interpolation).

(Furgale et al., 2012) also used a system based on B-splines to have a continuous time representation. In their work, they interpolate rotations and translations by two independent B-splines. Their Cayley-Gibbs-Rodriguez formulation used for the poses had two major issues according to (Steven Lovegrove, 2013):

- The used Rodrigues parameterization has a singularity for the rotation at π radians.
- The interpolation in this space does not represent the minimum distance for the rotation group hence the generated trajectories can correspond to unrealistic motions.

To address these issues,(Steven Lovegrove, 2013) proposed to use a continuous time trajectory formulation using cumulative B-splines. Their theoretical framework is the basis of our work and is derived next.

3 THEORETICAL FRAMEWORK FOR RS CAMERAS

3.1 Cumulative B-splines for Pose Interpolation

A standard B-spline is defined by constant polynomial basis functions $B_{i,k}$ ($k-1$ being the degree of the used polynomial) and variable Control Points (CP) p_i:

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

Let $\tilde{B}_{i,k}(t) = \sum_{j=i}^{k-1} B_{j,k}(t)$ be a cumulative basis functions defined further in eq. (8). The eq. (1) can be reformulated using CP differences $(\mathbf{p_i} - \mathbf{p_{i-1}})$ to obtain the cumulative B-splines expression:

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

This definition of the B-spline stands for multidimensional spaces and thus allows the interpolation for camera poses $p(i) \in \mathbb{R}^6$ independently over its different components (for instance 3 translation parameters and 3 rotation parameters using Rodrigues parameterization). However this does not account for the correlations between the components of the pose thus some adaptations are required to overcome this, using 6DoF camera poses defined in the $\mathbb{SE}3$ space. Such poses can be expressed by transformation matrices T

parameterized by a translation a and a rotation matrix R:

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \mathbf{T} \in \mathbb{SE}3, \mathbf{R} \in \mathbb{SO}3, \mathbf{a} \in \mathbb{R}^3 \quad (3)$$

While the cumulative B-splines interpolation in $\mathbb{R}^6$ involves the differentiation of CP $(\mathbf{p_i} - \mathbf{p_{i-1}})$, in the special Euclidean group $\mathbb{SE}3$ it requires more complex operations. Let $\mathbf{T}_{w,i}$ be the i^{th} Control Poses (indistinctly abbreviated as CP) relatively to the world coordinate frame $\mathbf{w}$, the transformation matrix $\Delta\mathbf{T}_{i-1,i}$ relating two poses $\mathbf{T}_{w,i-1}$ and $\mathbf{T}_{w,i}$ is obtained by composition:

$$\Delta\mathbf{T}_{i-1,i} = \mathbf{T}_{w,i-1}^{-1}\mathbf{T}_{w,i} \quad (4)$$

The log function maps a pose (or variation) defined in $\mathbb{SE}3$ to its tangent space $\mathfrak{se}3$, where operations for composing transformation are simpler, and allows to express the summation of weighted CP variation by simple addition. Let $\mathbf{T_{w,s}}(t)$ be the interpolated pose along the B-spline at instant t and Δ_t be the time between the 2 poses. Let $\Omega_{i-1,i} = \frac{1}{\Delta_t}\log(\Delta\mathbf{T}_{i-1,i})$, the eq. (2) can be adapted to $\mathfrak{se}3$:

$$log(\mathbf{T}_{w,s}(t)) = \tilde{B}_0(t)log(T_{w,0}) + \sum_{i=1}^{k-1}\tilde{B}_i(t)\Omega_{i-1,i} \quad (5)$$

The inverse of the logarithmic map, the exponential map, projects a pose from $\mathfrak{se}3$ to $\mathbb{SE}3$ and can be used to recover $\mathbf{T}_{w,s}(t)) = exp(log(\mathbf{T}_{w,s}(t)))$. Observing that additions in $\mathfrak{se}3$ maps to multiplications in $\mathbb{SE}3$, the eq.(5) can be reformulated for $\mathbb{SE}3$. The pose $\mathbf{T}_{w,s}(t)$ (for the B-spline $\mathbf{s}$ and expressed in world coordinates $\mathbf{w}$) is interpolated between the two CP $\mathbf{T}_{w,i}$ and $\mathbf{T}_{w,i+1}$ for different time t using :

$$\mathbf{T}_{w,s}(t) = exp(\tilde{B}_0(t)log(\mathbf{T}_{w,0}))\prod_{i=1}^{k-1}\exp(\tilde{B}_i(t)\Omega_{i-1,i}) \quad (6)$$

A change of variable is introduced to map the real time t variable between two CP associated to times t_i and $t_i + \Delta_t$ to a normalized time $u \in [0,1]$ interval, using a constant time interval Δ_t between the CP:

$$u = (t - t_i)/\Delta_t \quad (7)$$

Using this normalized time, the time between 2 CP is equal to one, so the factor $\frac{1}{\Delta_u}$ vanishes and the expression of $\Omega_{i-1,i}$ can be reformulated as $\Omega_{i-1,i} = \log(\Delta\mathbf{T}_{i-1,i})$

(Steven Lovegrove, 2013) have chosen to use cubic B-splines because of their C^2 continuity, which allow to predict angular velocities and linear accelerations. Hence the degree of the polynomial for the basis is 3 and the B-splines are locally controlled by 4 CP (as $k = 4$), meaning that the knowledge of the 4 surrounding CP is sufficient to interpolate on the B-spline. The used cumulative basis functions $\tilde{B}_{i,k}(u)$ for $k = 4$ are expressed in the i^{th} component of the vector $\tilde{B}(u)$ by:

$$\tilde{B}(u) = \frac{1}{6}\begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \end{pmatrix} \quad (8)$$

Using normalized time variable u and noting that the first line of $\tilde{B}(u)$ is $\tilde{B}_0(u) = 1$, the eq.(6) can be simplified in:

$$\mathbf{T}_{w,s}(u) = \mathbf{T}_{w,0}\prod_{i=1}^{k-1}\exp(\tilde{B}_i(u)\Omega_{i-1,i}) \quad (9)$$

3.2 Rolling Shutter Camera Model

An image from a RS camera can be seen as the concatenation of one dimensional images (rows) exposed at different times as seen in the figure 1.

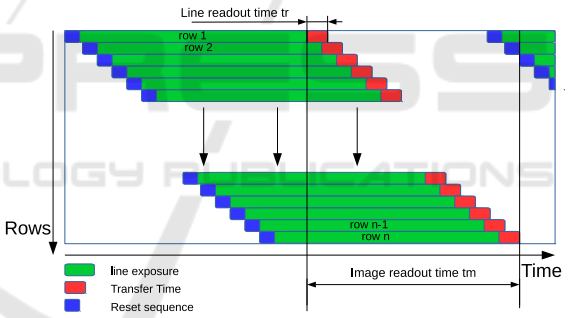

Figure 1: RS cameras expose the image lines sequentially, thus if the first line is exposed at instant t_0, then the n^{th} line is exposed at $t_0 + n.t_r$ with t_r the readout time of a line. The time taken by the camera to fully expose an image is called the readout time t_m. (Li et al., 2013).

For a static scene and camera, there is no geometric difference between the GS and RS camera. But if the camera is in motion relatively to the scene, each line is a projection from a different camera viewpoint. While the camera pose $\mathbf{T} \in \mathbb{SE}3$ is common to all the pixels in a GS image, it varies over time as $\mathbf{T}(t)$ for each individual line in the RS case.

The pinhole camera perspective projection model is derived below. Let $\mathbf{X}$ be a 3D point defined in homogeneous coordinates in the world coordinate system $\mathbf{w}$. Let $\mathbf{T_{w,c}}$ be the transformation matrix from world $\mathbf{w}$ to camera $\mathbf{c}$ coordinate frame. Let $\mathbf{K}$ be the camera matrix containing the intrinsic parameters and $\pi(.)$ be the perspective projection (mapping from $\mathbb{P}^2$

to $\mathbb{R}^2$). The pinhole model projects $\mathbf{X}$ onto the image plane to $\begin{bmatrix} x & y \end{bmatrix}^T$ by:

$$\begin{bmatrix} x \\ y \end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T}_{\mathbf{w},\mathbf{c}}\mathbf{X}) \tag{10}$$

To model the varying pose over time $\mathbf{T}_{\mathbf{w},\mathbf{c}}$ is parameterized by t as $\mathbf{T}_{\mathbf{w},\mathbf{c}}(t)$. The spline being defined by eq.(9) in the spline coordinate frame $\mathbf{s}$ (attached to the IMU for instance) different from $\mathbf{c}$, a (constant over time) transformation $\mathbf{T}_{\mathbf{s},\mathbf{c}}$ is required to obtain $\mathbf{T}_{\mathbf{w},\mathbf{c}}(t)$:

$$\mathbf{T}_{\mathbf{w},\mathbf{c}}(t) = \mathbf{T}_{\mathbf{s},\mathbf{c}}\mathbf{T}_{\mathbf{w},\mathbf{s}}(t) \tag{11}$$

The projection at varying time t is obtained by:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T}_{\mathbf{w},\mathbf{c}}(t)\mathbf{X}) = \omega(\mathbf{X}, \mathbf{T}_{\mathbf{w},\mathbf{c}}(t)) \tag{12}$$

This model does not yet represent the fact that at a given time t corresponds a single line exposure. So the projection $\begin{bmatrix} x(t_1) & y(t_1) \end{bmatrix}^T$ of $\mathbf{X}$ at time $t = t_1$ will actually be obtained only if the line $y(t_1)$ is exposed at time t_1.

(Steven Lovegrove, 2013) expressed the line exposed at a time t as a function of s the frame start time, e is the end frame time, and h is the height of the image in pixels by:

$$y(t) = h\frac{(t-s)}{(e-s)} \tag{13}$$

The figure 2 shows plotted in blue the image projections $\begin{bmatrix} x(t) & y(t) \end{bmatrix}^T$ of a single 3D point obtained by eq.(12) using interpolation of the poses defined in eq.(9). The value of t is sampled to the different exposure time of each individual row. The green line indicates the row that is actually exposed when the $y(t)$ corresponds to the row number and the red cross at the intersection is the resulting projection. For highly curved trajectories, multiple projections of a single 3D point can be observed in a single image.

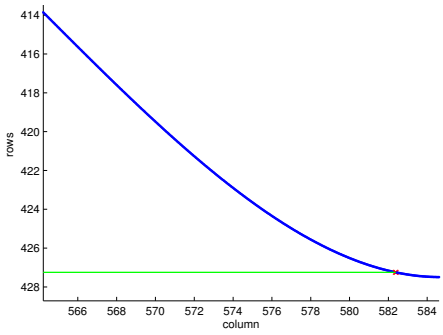

Figure 2: Here is an example of the different projections of a 3D point (in blue) in one image for each of the line poses, in the case of a moving camera. The green line shows the exposed rows at the time the 3D point is projected to it.

The projection(s) $\begin{bmatrix} x(t) & y(t) \end{bmatrix}^T$ that is(are) effectively observed by the RS camera is(are) obtained by intersecting the curves corresponding to eq.(12) and eq.(13). Determining t is an optimization problem that (Steven Lovegrove, 2013) solve iteratively using first order Taylor expansion of the 2 equations around a time t:

$$y_b(t+\delta t) = h\frac{(t+\delta t - s)}{(e-s)} \tag{14}$$

$$\begin{bmatrix} x(t+\delta t) \\ y(t+\delta t) \end{bmatrix} = \omega(X, T_{w,c}(t)) + \delta t \frac{d\omega(X, T_{w,c}(t))}{dt} \tag{15}$$

This system of equations is reorganized as:

$$\delta t = -\frac{ht + s(y_b(t) - h) - ey_b(t)}{h + (s-e)\frac{d\omega_y(X, T_{w,c}(t))}{dt}} \tag{16}$$

By updating t as $t = t + \delta_t$ and iterating, t converges generally in approximately 3 or 4 iterations (see Figure 3). Once t is determined, the corresponding projection is obtained using eq.(12). For slow motions, the initial value for t can be set at the time corresponding to the middle row of the image and the iterative algorithm is very likely to converge. However, for high dynamic motions, the initial value has to be set wisely, as different initial values could lead to different projections that correspond indeed to different actual projections.

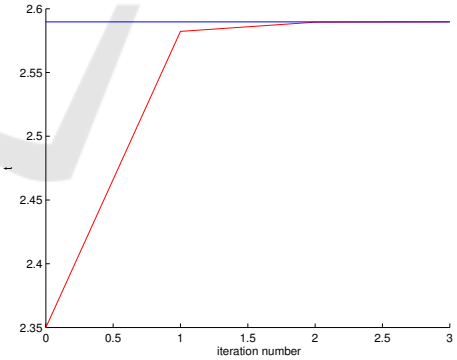

Figure 3: Evolution of t during three iterations.

3.3 Perspective n-Points Algorithm for RS Camera

The Perspective n-Points, or PnP algorithm consists in determining the pose of a camera given n correspondences between 3D known points and their observations in the image. Here we describe the iterative portion of the PnP algorithm that is used to refine a first estimate.

Let $\mathbf{u_{i,k}}$ be a set of image measurements (corresponding to time t_i) of M 3D points $\mathbf{X_k}$ inside a set of N images and $\mathbf{T_{w,c}}$ be related to a set of CP $\mathbf{T_{w,s}}$ through eq.(11).The reprojection error $res(i,k)$ for individual point is defined by:

$$res(i,k) = ||\mathbf{u_{i,k}} - \pi([\mathbf{K}|\mathbf{0}]\mathbf{T_{w,c}}(\mathbf{t_i})\mathbf{X_k})|| \quad (17)$$

The following cost function is minimized to determine $\hat{\theta}$, the set of parameters $\mathbf{T_{w,s}}$ that best fits the model to the set of observations:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N}\sum_{k=1}^{M} res(i,k)^2 \quad (18)$$

While a PnP algorithm for a GS camera optimizes the pose of the camera itself, the RS version has to optimize the CP used for the interpolation. Thus, the parameters vector θ contains the required CP parameters.

The minimization is achieved using the Levenberg-Marquardt algorithm, an iterative nonlinear least squares solver behaving between Gauss-Newton and Gradient Descent. The initial value for the parameters in θ is set from the IMU measurements or by applying firstly a GS PnP. The levenberg Marquardt algorithm uses the partial derivatives (obtained by finite difference) of the cost function with respect to the parameters to optimize. It generally converges after several iterations but is prone to converge to local minimum if the cost function is not convex or if the initial value is too far from the global minimum.

4 CP TEMPORAL SAMPLING

4.1 Uniform Distribution

(Steven Lovegrove, 2013) use an UD in time for the CP, and use normalized times as shown in eq.(7). This UD of CP (chosen every 0.1 seconds in (Steven Lovegrove, 2013)) leads to some limitations in these two cases:

- For a camera grabbing frames at 30FPS, a CP is defined each 3 frames leading to relatively smoothed trajectory for fast camera motions, failing to accurately model it.
- For steady movement, the algorithm creates unnecessary redundant CP.

The motion of hand-held device used for augmented reality can be decomposed in three distinct type of motions: high speed motion, low speed motion, and low range but high frequency motions when the user's hand is shaking. While the use of a small UD interval Δ_t between CP is a a simple solution to model theses motions, it leads to an increase of the CP number and consequently of the computational cost. Doing so, unnecessary CP are used for low speed and steady motions, adding no information/constraints to the interpolated trajectory.

4.2 Non Uniform Distribution

To overcome this problem, we propose to relax the UD constraint for the intervals Δ_t between the CP. This allows to locally increase the number of CP where needed by lowering the time interval Δ_t during fast motion while keeping a low number of CP for portions of the trajectory that correspond to slow motions.

The mapping between u and t is then defined differently from eq.(7) by:

$$u = \frac{t - t_i}{t_{i+1} - t_i} \quad (19)$$

While the retrieving of the t_i and $t_i + 1$ CP involved only multiplication and rounding in the UD case, it requires more complex computations with NUD as the time value t_i can be defined freely.

NUD provides the ability to create locally multiple CP inside a single image. This leads to many CP quartets required to interpolate the pose inside different portions of the image (see Figure 4). As retrieving the CP associated to a time t is an operation that is done thousands of times per frame, some care is required to avoid wasting time searching over all the CP. This is achieved by storing CP indices in an chronologically ordered list and using hash tables for fast access.

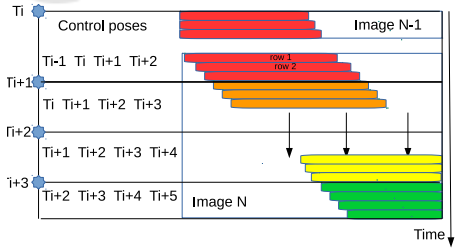

Figure 4: An example of multiple CP inside a single image, and their influence for interpolating the pose for different rows. 3 CP are time-stamped between the beginning and the end of the image N: T_{i+1}, T_{i+2} and T_{i+3}. The quartet of CP used for interpolating pose at each row change three time during the image exposure.

4.3 CP Generation Methods

As our approach uses varying time intervals function of the local properties of the motion, it requires to generate the CP at the right time. We investigated two different approaches involving either an IMU sensor or the analysis of the reprojection error in the images. The first one is better fitted for realtime applications as the CP can be generated online using IMU measurements while the second involves an iterative process that predisposes it for offline computation.

4.3.1 Based on IMU

An IMU sensor is generally composed of a gyroscope that measures the angular velocities and accelerometers that provide accelerations. On current tablets and smartphones, it generally delivers measurements at about 100Hz, which is an higher frequency than the frame rate of the camera.

We propose a simple analysis of the measurements done by the IMU to determine when the CP are required. The figure 5 shows a generated trajectory (in the two top plots). The left (resp. right) plots corresponds to the translation (resp rotation) components. The data provided by the IMU are plotted in the middle plots. The bottom plots show respectively the norm of the linear accelerations and angular velocities. Different thresholding values (shown in red, blue and black) are used to determine when more CP are required. These threshold values th_i are stored in two (for acceleration and angular velocity) look up table providing $n(th_i)$, the number of CP required per unit of time. These lists of threshold values are generated empirically for the moment and correspond to a tradeoff between accuracy of the motion modeling and computational cost. For instance, if the angular velocity norm is below the threshold th_1, $n(th_1)$ CP per unit of time are used while if its norm rises above th_1 but below th_2, $n(th_2)$ CP per unit of time are used and so on. Once the CP are created, their parameters are optimized using eq.(18).

4.3.2 Based on Reprojection Error

We propose a second method to determine the temporal location of the CP when a IMU in not available. This method involves an iterative estimation of the trajectory using initial CP (set for instance with an UD) as shown in eq.(18). Let $\overline{res_{t_1,\Delta_t}}$ be the mean of the residuals $res(i,k)$ defined in eq.(17) between times t_1 and $t_1+\Delta_t$. Our method use an iterative scheme consisting of the following two steps:

- The residuals are computed along the trajectory and analyzed using a sliding window to measure locally $\overline{res_{t_1,\Delta_t}}$ for varying t_1. Maximums values are detected and additional CP are generated at the middle of the two corresponding neighboring CP for time $t_1+\frac{\Delta_t}{2}$.
- An iterative estimation of the trajectory eq.(18) is achieved with the added CP to refine the whole set of CP.

The process is iterated until $\forall t_1 : \overline{res_{t_1,\Delta_t}} < threshold$.

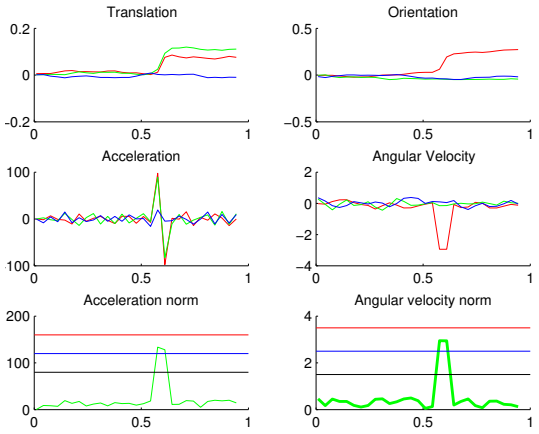

Figure 5: Determination of the number of required CP per unit of time using IMU measurements analysis (See the text for details).

5 RESULTS

We evaluated our methods on simulated data sets consisting of simple scenes (defined by sparse points clouds) and trajectories (defined by B-splines using manually defined ground truth CP that are NUD). While the proposed methods have been tested to model complex 6-dof trajectories, they cannot be easily drawn in this paper. Hence the plotted results are for simple trajectories to allow the reader to interpret them in 2D and serve as proof of concept. The synthetic scene is projected to synthetic images using the RS camera model, and the CP are optimized using the iterative minimization. Even if the plotted results only show the trajectories, they are computed using the whole framework presented above ie. using the PnP algorithm to compare the poses through the reprojection error in the images using the RS camera model.

5.1 Based on IMU

To test IMU-based CP generation, IMU measurements are synthesized at 100Hz from the ground truth trajectory using the first and second derivatives of

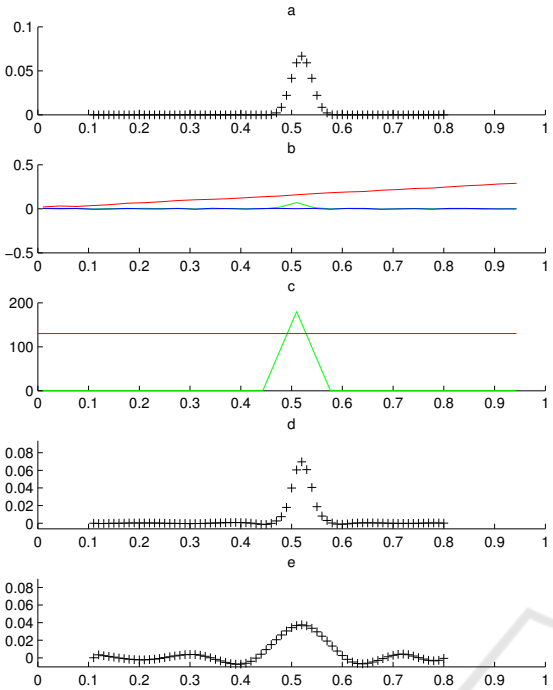

Figure 6: Results for the IMU method (see text for details).

eq.(6) (see (Steven Lovegrove, 2013) for details). We set the first thresholds for the angular velocity to $th_dRot = 2rad/s$ and for the linear acceleration to $th_Acc = 130cm/s^2$.

The figure 6 (a) shows a simple trajectory consisting of a linear motion with a perturbation at t=0.5. The plot 6 (b) shows the position of the camera during the motion (using different colors for different axes, the green being used for the vertical direction). The plot 6 (c) shows the norm of the acceleration in green while the first threshold is plotted in red. The angular velocities are not plotted in this simple example as they have null values. The plot 6 (d) shows the B-spline trajectory generated by our method while 6 (e) shows the B-spline generated with the same number of CP but using an UD. As expected, the NUD B-spline is much closer to the ground truth while the UD B-spline artificially smooths the trajectory, and creates larger oscillations.

5.2 Based on Reprojection Error

For this second test, the same trajectory is used as shown in figure 7 (a), but the synthetic IMU measurements are not exploited. A first UD for the CP is used to initiate the process and the resulting B-spline after one reprojection error minimization step is shown in (b). The residuals mean value for the sliding windows are shown in (c), and the maximum detected at $t = 0.5$ permits to add a few (2) CP in the neighborhood and iterate with error minimization step. The process is stopped once the number of CP reaches the same value than in the plot 6 (d) and (e) for comparison purpose. The obtained B-spline trajectory is shown in 7 (d) and the corresponding residuals mean value for the sliding windows in (e) shows a decrease by an approximate 20X factor compared to (c) for a increase of only 2 CP.

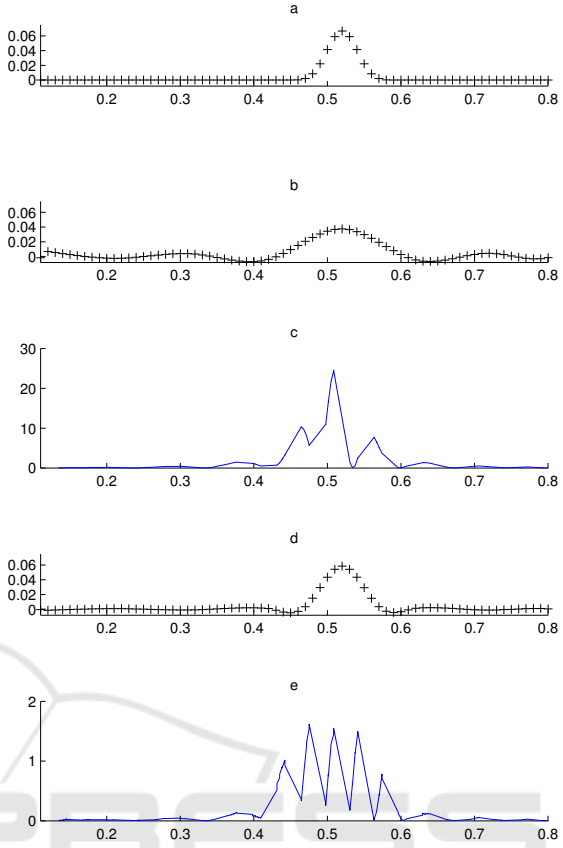

Figure 7: Results for the reprojection error method (see text for details).

6 FUTURE WORKS

Our future researches will focus on multiple points:

- Some tests on real motions using hand held devices will be conducted, the ground truth being provided by a motion capture system.
- We will attempt to intricate in the same optimization process the minimization of the residuals and the CP generation in order to lower the time requirements, especially for the method based on reprojection error.
- Whereas the time associated to the CP are fixed in our current study, it would be interesting to

optimize their values alongside the pose parameters. This added degree of freedom would allow to adapt dynamically the distribution in time of the CP avoiding the generation of new ones.

- We will try to integrate our model in a realtime SLAM.

7 CONCLUSION

In this paper, we have enriched the B-splines trajectory model from (Steven Lovegrove, 2013) to integrate NUD for the CP in the RS Camera SLAM context. We have shown that this NUD allowed to better model trajectories than using an UD for the same number of CP. We have proposed two methods to generate the CP, using either IMU measurements or iterative reprojection error minimization. The two methods have been tested on synthetic trajectories and proven to be efficient for the PnP problem resolution. This enriched model will be integrated in a SLAM in future works.

REFERENCES

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *9th IEEE International Conference on Computer Vision (ICCV 2003), 14-17 October 2003, Nice, France*, pages 1403–1410.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*.

Furgale, P., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2088–2095.

Gonzalez, A. (2013). *Localisation par vision multispectrale. Application aux systèmes embarqués*. Theses, INSA de Toulouse.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Hedborg, J., Forssén, P. E., Felsberg, M., and Ringaby, E. (2012). Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434–1441.

Hedborg, J., Ringaby, E., Forssén, P. E., and Felsberg, M. (2011). Structure and motion estimation from rolling shutter video. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 17–23.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan.

Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, Orlando.

Li, M., Kim, B., and Mourikis, A. I. (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*, pages 4697–4704, Karlsruhe, Germany.

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370.

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Patron-Perez, A., Lovegrove, S., and Sibley, G. (2015). A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *Int. J. Comput. Vision*, 113(3):208–219.

Roussillon, C., Gonzalez, A., Solà, J., Codol, J., Mansard, N., Lacroix, S., and Devy, M. (2012). RT-SLAM: A generic and real-time visual SLAM implementation. *CoRR*, abs/1201.5450.

Steven Lovegrove, Alonso Patron-Perez, G. S. (2013). Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

Strasdat, H., Montiel, J., and Davison, A. J. (2010). Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE.