Spatiotemporal Optimization for Rolling Shutter Camera Pose Interpolation

Philippe-Antoine Gohard, Bertrand Vandeportaele, Michel Devy

Spatiotemporal Optimization for Rolling Shutter Camera Pose Interpolation

Philippe-Antoine Gohard[1,2], Bertrand Vandeportaele[1], and Michel Devy[1]

[1] LAAS-CNRS, Toulouse University, CNRS, UPS, Toulouse, France
[2] Innersense, Ramonville-Saint-Agne, France
{philippe-antoine.gohard, bertrand.vandeportaele, michel.devy}@laas.fr

Abstract. Rolling Shutter cameras are predominant in the tablet and smart-phone 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 while taking into account the varying dynamic of its motion, using higher density of 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. The non-uniform camera model is integrated into a Bundle Adjustment optimization which is able to converge even from a poor initial estimate. A routine of spatiotemporal optimization is presented in order to optimize both the spatial and temporal positions of the control points. Results on synthetic and real datasets are shown to prove the concepts and future works are introduced that should lead to the integration of our model in a SLAM algorithm.

Keywords: Rolling Shutter · Camera Geometric Model · Bundle Adjustment · Simultaneous Localization And Mapping · B-splines interpolation.

INTRODUCTION

In Augmented Reality applications for mobile devices (smartphones and tablets), the real-time 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: [6]), and Simultaneous Localization and Mapping (SLAM: MonoSLAM[2], PTAM[10] OrbSLAM [15]). 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. if the integration time is neglected, the whole image is the projection of the scene using a single pose for the camera.

However, more than 90% of mobile devices are equipped with Rolling Shutter (RS) cameras because of their lower cost and smaller size compared with 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. Previous RS Camera model presented in[19][16] are achieving camera pose interpolation using B-splines controlled by Control Points (CP) that are sampled in time with a constant interval. This kind of temporal distribution is referred to as an Uniform Time Distribution (UTD) in the following.

This paper extends these models by using an adaptive Non Uniform Time Distribution (NUTD) for the CP of the B-spline as proposed in [21]. This allows to globally reduce the number of CP required to model a given trajectory with the same accuracy compared with the UTD. The NUTD B-Splines are detailed, alongside a Bundle Adjustment (BA) using this model. The NUTD of the CP allows to optimize the timestamps of the CP, and we demonstrate how it can be used to model trajectories on real datasets.

The paper firstly presents existing SLAM algorithms and exhibits problems arising from the use of 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. Then, the iterative minimization used in the Perspective n-Points and bundle adjustment algorithm is explained. Two methods are briefly exposed to efficiently generate the CP, using either an IMU sensor or multiple iterations of reprojection error minimization. A spatiotemporal optimization approach is then proposed to determine adequate NUTD for the CP. Finally, the proposed PnP and BA implementations using the NUTD B-Splines models are evaluated on synthetic datasets to prove the concept, then the spatiotemporal optimization is tested on real datasets. Future works are then discussed.

1 RELATED WORK FOR SLAM

1.1 Global Shutter

A monocular visual SLAM algorithm recovers the position and orientation of a mobile camera and dynamically constructs a model of the environment in real-time. In the Augmented Reality context, the camera parameters are used to synthesize images of virtual elements that are rendered coherently over the acquired image, using the reconstructed model of the environment.

Early real-time methods used to solve the SLAM problem in the literature were based on Extended Kalman Filter (EKF)[2], [18], [5]. The simplicity of this method and it's computing efficiency for small size environment models has made it the most used SLAM method for the past decade.

Other robust and real-time methods based on local BA like PTAM [10] have also been proposed. They minimize the reprojection error over a subset of previously acquired images, called keyframes [3], [15], or over a sliding window of frames [14]. They provide improved robustness thanks to the modeling of outliers, and [20] proved the superiority of these BA-based methods over filtering one.

1.2 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 [11]. 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.

[8] initially used a similar method, and proposed in [7] a BA 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).

[4] also used 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 [19]:

- 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,[19] proposed to use a continuous time trajectory formulation using cumulative B-splines. In precedent works [21], we have shown that the UTD of the B-Splines control poses (CP) leads either to a smoothing of the trajectory or to redundancies. We proposed to use a NUTD of the CP and methods to dynamically generate CP.

This paper is an extension of [21], describing a NUTD B-Spline model suitable for a BA algorithm and proposing a spatiotemporal optimization of the CP. A brief summary of used notations and scientific context is given in the next section.

2 NOTATIONS AND SCIENTIFIC CONTEXT

In this article, the 6 degrees of freedom associated to the extrinsic parameters of the camera are expressed by a matrix 4×4 corresponding to the transformation

from the camera coordinate frame to the world coordinate frame. This matrix $\mathbf{T_w} \in \mathbb{SE}3$ is parameterized by a translation vector $\mathbf{a}$ and a rotation matrix $\mathbf{R}$:

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

A rigid transformation in the Lie algebra $\mathfrak{se}(3)$ can be expressed by a 6D vector $\xi = [\mathbf{w}, \mathbf{a}]^T \in \mathfrak{se}(3)$ where $\mathbf{w} = [\omega_0, \omega_1, \omega_2]^T$ is the rotation component. Its 4×4 associated matrix can be obtained by applying the wedge operator $[\cdot]_\wedge$ on ξ knowing that $[w]_\times$ is the skew-symmetric matrix 3×3 of w:

$$\mathbf{\Omega} = [\xi]_\wedge = \begin{pmatrix} [\mathbf{w}]_\times & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, [\mathbf{w}]_\times = \begin{pmatrix} 0 & -\omega_2 & \omega_1 \\ \omega_2 & 0 & -\omega_0 \\ -\omega_1 & \omega_0 & 0 \end{pmatrix} \tag{2}$$

The logarithmic mapping projects a matrix from $\mathbb{SE}3$ to its tangent space $\mathfrak{se}(3)$ defined locally Euclidean, where compositions of rigid transformations are obtained by additions. The exponential mapping, being the inverse operation of the exponential map, projects back a 6D vector from the tangent space $\mathfrak{se}(3)$ towards $\mathbb{SE}(3)$. These two applications can be resumed (when the magnitude of the angle of rotation is lower than π):

$$\begin{aligned} \mathbf{T_w} &= exp(\mathbf{\Omega}) \\ \mathbf{\Omega} &= log(\mathbf{T_w}) \end{aligned} \tag{3}$$

2.1 Cumulative B-Splines

As a RS camera exposes its lines at different instants, an estimate of the camera pose for each line is needed to project the 3D points of the observed scene onto the image. This estimate can be obtained with continuous-time modeling of the camera trajectory. To interpolate the camera pose associated with one particular image line, multiple timestamped CP, locally controlling the trajectory, are required, the first of these CP being timestamped at time t_i. The corresponding rigid transformations matrices are defined in the world coordinate system and abbreviated $\mathbf{T_{w,i}}$. Various interpolation methods can be considered (linear, B-Spline, Bézier) according to the final application.

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 CP $\mathbf{p}_i$. This definition is not suitable for the non euclidean space of rigid body transformation $\mathbb{SE}(3)$ where the elements are composed by matrix multiplication.

In [19], the authors suggest the use of the cumulative form of the B-Splines because of their suitability for the interpolation on the manifold $\mathbb{SE}(3)$ [9]. They chose cubic B-Splines ($k = 4$) for the interpolation to ensure a C^2 continuity of the trajectory, allowing interpolation of velocities and accelerations.

The cumulative basis functions $\tilde{B}(t)_j$ used in cubic cumulative B-Splines are expressed by the j^{th} component of the vector $\tilde{\mathbf{B}}(t)$, starting at index $j = 0$:

$$\tilde{\mathbf{B}}(t) = \frac{1}{6}\begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \tag{4}$$

where $u(t) = \frac{t-t_{i+1}}{\Delta t}$ an intermediate time representation such as $0 \leq u(t) < 1$ between the two CP $\mathbf{T_{w,i+1}}$ and $\mathbf{T_{w,i+2}}$ with $t \in [t_{i+1}, t_{i+2}]$. Δt defines the time interval between the CP, which is considered constant for UTD B-Splines.

Interpolation on the $\mathbb{SE}3$ manifold is defined by a composition of CP variations $\mathbf{\Omega}_{j-1,j} = log(\mathbf{T_{w,j-1}}^{-1}\mathbf{T_{w,j}})$ weighted by basis functions $\tilde{B}(t)_j$ applied to a reference pose $\mathbf{T_{w,i}}$. Thus, in the case of cumulative cubic B-Spline ($k = 4$), the interpolation of the pose $\mathbf{T_w}(t)$ between two CP $\mathbf{T_{w,i+1}}$ and $\mathbf{T_{w,i+2}}$ requires the four neighbor CP $\mathbf{T_{w,i}}$ to $\mathbf{T_{w,i+3}}$. The interpolation function is then expressed as:

$$\mathbf{T_w}(t) = \mathbf{T_{w,i}} \prod_{j=i+1}^{i+k-1} \exp(\tilde{B}(t)_{j-i}\mathbf{\Omega}_{j-1,j}) \tag{5}$$

2.2 Non Uniform Time Distribution for B-Splines

This document is an extension of [21], where we suggested to use a NUTD of the CP in order to adjust the distribution to the camera trajectory dynamic. This enables a more efficient modeling of the trajectory, both in terms of computational cost and memory usage.

Using NUTD, the time interval Δ_t between two CP is not constant anymore. Instead, the time interval $\Delta t_{i-1,i}$ is defined as the difference of timestamps between the two CP $\mathbf{T_{w,i-1}}$ and $\mathbf{T_{w,i}}$.

B-Splines whose CP are non-uniformly sampled in time are named NUTD B-Splines in the remainder of this document. For such B-Splines, the basis functions are defined by a recurrence formula slightly different from the standard (UTD) case [1], see [9] :

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \tag{6}$$

With the stopping condition defined by:

$$B_{i,1}(t) = \begin{cases} 1 & \text{if} \quad t_i < t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The recurrence formula allows to compute the basis functions for different degrees. By developing the formula as in [22] or using Toeplitz matrix [17], the basis function can be expressed as a matrix product as in [19] for cubic B-Spline. Unlike the UTD case, the coefficient matrix used in the expression of basis functions is dependent on the CP timestamps. For cubic B-Splines, four CP $\mathbf{T_{w,i}}$ to $\mathbf{T_{w,i+3}}$ and six CP timestamps $\mathbf{T_{w,i}}$ to $\mathbf{T_{w,i+5}}$ are required to interpolate at a time t within $[t_{i+2}; t_{i+3}[$.

The intermediate time representation between the two CP of the considered interpolated section is henceforth given by:

$$u(t) = \frac{t - t_{i+2}}{t_{i+3} - t_{i+2}} \tag{8}$$

The matrix form for the NUTD basis functions are thus expressed by:

$$\mathbf{B}(t) = \mathbf{M} \begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \tag{9}$$

Details on coefficient matrix $\mathbf{M}$ are given in [22], it is however substantial to note that $\mathbf{M}$ is now relative to the 6 CP timestamps t_i to t_{i+5}.

$\mathbf{C}$ being the cumulative coefficient matrix from $\mathbf{M}$, the cumulative version of the basis functions are now given by:

$$\mathbf{C}_{i,j} = \sum_{l=i}^{k} \mathbf{M}_{l,j} \qquad , \qquad \tilde{\mathbf{B}}(t) = \mathbf{C} \begin{pmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{pmatrix} \tag{10}$$

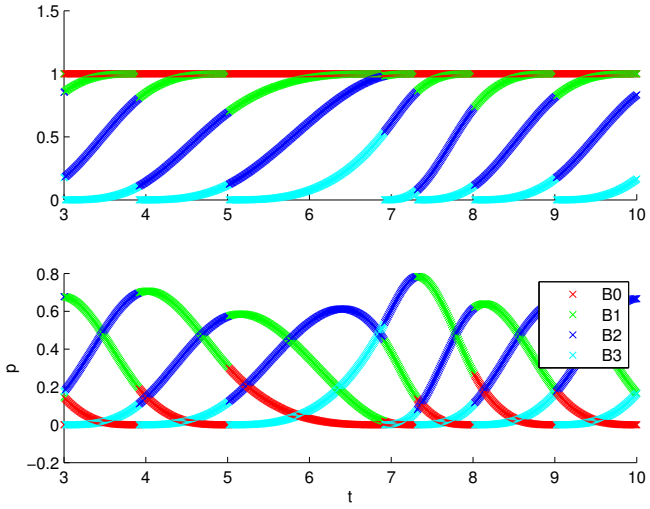

Fig. 1. The evolution of the value of the $k = 4$ cumulative NUTD basis functions $\tilde{B}_{0..k}(t)$(up) and non cumulative $B_{0..k}(t)$(down) w.r.t time. Each basis functions is associated to a color (red, green, blue, cyan), showing the influence of a CP variation(up) or a CP(down) on the interpolated trajectory over time.

The figure 1 shows the value of the 4 NUTD cumulative and non cumulative basis over time. Unlike the UTD case, the basis functions are different for each interpolation section because of the varying time interval between the CP.

The cumulative coefficient matrix $\mathbf{C}$ is invariant with respect to the interpolation time t. For clarity, we redefine $\Omega_j := \Omega_{j-1,j}$. The derivative of the basis functions with respect to t can be retrieved in the same way as UTD B-Splines:

$$\dot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t_{i+2,i+3}}\mathbf{C}\begin{bmatrix}0\\1\\2u\\3u^2\end{bmatrix}, \ddot{\tilde{\mathbf{B}}}(t) = \frac{1}{\Delta t^2_{i+2,i+3}}\mathbf{C}\begin{bmatrix}0\\0\\2\\6u\end{bmatrix} \tag{11}$$

The first and second derivatives of the interpolated trajectory are then expressed by:

$$\dot{\mathbf{T}}_w(t) = \mathbf{T}_{w,i}(\dot{\mathbf{A}}_1\mathbf{A}_2\mathbf{A}_3 + \mathbf{A}_1\dot{\mathbf{A}}_2\mathbf{A}_3 + \mathbf{A}_1\mathbf{A}_2\dot{\mathbf{A}}_3) \tag{12}$$

$$\ddot{\mathbf{T}}_w(t) = \mathbf{T}_{w,i}\begin{pmatrix}\ddot{\mathbf{A}}_1\mathbf{A}_2\mathbf{A}_3 + \mathbf{A}_1\mathbf{A}_2\ddot{\mathbf{A}}_3 + \mathbf{A}_1\mathbf{A}_2\ddot{\mathbf{A}}_3)+\\ 2*(\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\mathbf{A}_3 + \dot{\mathbf{A}}_1\mathbf{A}_2\dot{\mathbf{A}}_3 + \mathbf{A}_1\dot{\mathbf{A}}_2\dot{\mathbf{A}}_3)\end{pmatrix} \tag{13}$$

$$\mathbf{A}_j = \exp(\tilde{B}(t)_j\boldsymbol{\Omega}_{j+i}) \tag{14}$$

$$\dot{\mathbf{A}}_j = \mathbf{A}_j\dot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} \tag{15}$$

$$\ddot{\mathbf{A}}_j = \dot{\mathbf{A}}_j\dot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} + \mathbf{A}_j\ddot{\tilde{B}}(t)_j\boldsymbol{\Omega}_{j+i} \tag{16}$$

3 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 2.

For a static scene and camera, there is no geometric difference between the GS and RS cameras. But when there is a relative motion between the camera and 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 as $\mathbf{T}(t)$ for each individual line in the RS case.

The pinhole camera perspective projection model is derived below. Let $\mathbf{P}$ be a 3D point defined in homogeneous coordinates in the world coordinate system $\mathbf{w}$. Let $\mathbf{T_{c,w}} = \mathbf{T^{-1}_{w,c}} = \mathbf{T^{-1}_{w}}$ 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 $\mathbf{P}$ onto the image plane to $\mathbf{p} = \begin{bmatrix}p_u\ p_v\end{bmatrix}^T$ by:

$$\mathbf{p} = \begin{bmatrix}p_u\\p_v\end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T_{c,w}}\mathbf{P}) \tag{17}$$

To model the varying pose, $\mathbf{T_{c,w}}$ is parameterized by t as $\mathbf{T_{c,w}}(t)$. The spline being defined by eq.(5) in the spline coordinate frame $\mathbf{s}$ (attached to the IMU for

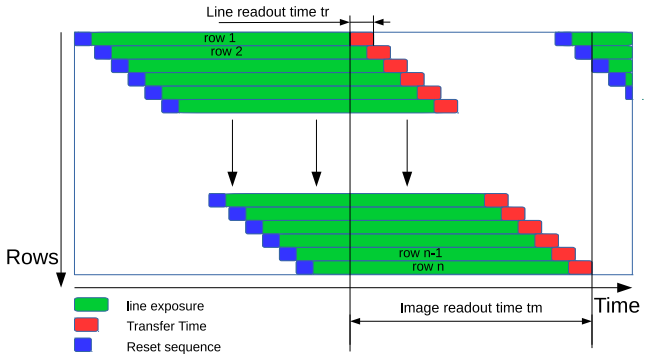

Fig. 2. RS cameras expose the image lines sequentially, thus if the first line is exposed at instant t_0, then the n^{th} line is exposed at $t_0 + n.t_r$ with t_r the readout time of a line. The time taken by the camera to fully expose an image is called the readout time t_m [13]. Image from [21].

instance) different from $\mathbf{c}$, a (constant over time) transformation $\mathbf{T_{c,s}}$ is required to obtain $\mathbf{T_{c,w}}(t)$:

$$\mathbf{T_{c,w}}(t) = \mathbf{T_{c,s}}\mathbf{T_{s,w}}(t) \tag{18}$$

The projection is obtained by:

$$\mathbf{p}(t) = \begin{bmatrix} p_u(t) \\ p_v(t) \end{bmatrix} := \pi([\mathbf{K}|\mathbf{0}]\mathbf{T_{c,w}}(t)\mathbf{P}) = \omega(\mathbf{P}, \mathbf{T_{c,w}}(t)) \tag{19}$$

This model does not yet represent the fact that at a given time t corresponds a single line exposure. So the projection $\left[p_u(t_j)\ p_v(t_j)\right]^T$ of $\mathbf{P}$ at time $t = t_j$ will actually be obtained only if the line $p_v(t_j)$ is exposed at time t_j.

[19] expressed the line exposed at a time t as a function of s being the frame start time, e being the end frame time, and h being the height of the image in pixels by:

$$p_v(t) = h\frac{(t-s)}{(e-s)} \tag{20}$$

The figure 3 shows plotted in blue the image projections $\left[p_u(t)\ p_v(t)\right]^T$ of a single 3D point obtained by eq.(19) using interpolation of the poses defined in eq.(5). 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 $p_v(t)$ 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.

The projection(s) $\left[p_u(t)\ p_v(t)\right]^T$ that is(are) effectively observed by the RS camera is(are) obtained by intersecting the curves corresponding to eq.(19) and eq.(20). Determining t is an optimization problem that [19] solves iteratively using first order Taylor expansion of the 2 equations around a time t:

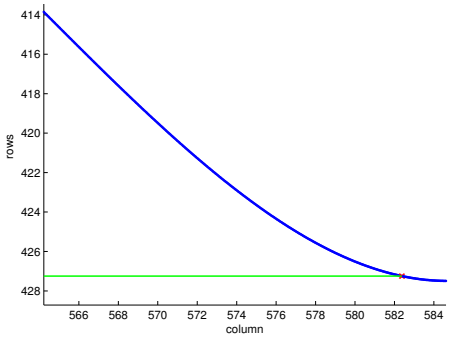

Fig. 3. Projections of a 3D point (in blue) in one image for poses associated to different lines, in the case of a moving camera. The green line shows the exposed rows at the time the 3D point is projected to it. Image from [21].

$$p_v(t+\delta t) = h\frac{(t+\delta t - s)}{(e-s)} \tag{21}$$

$$\begin{bmatrix} p_u(t+\delta t) \\ p_v(t+\delta t) \end{bmatrix} = \omega(P, \mathbf{T_{c,w}}(t)) + \delta t \frac{d\omega(P, \mathbf{T_{c,w}}(t))}{dt} \tag{22}$$

This system of equations is reorganized as:

$$\delta t = -\frac{ht + s(p_v(t) - h) - ep_v(t)}{h + (s-e)\frac{d\omega_{p_v}(P, \mathbf{T_{c,w}}(t))}{dt}} \tag{23}$$

By updating t as $t = t + \delta_t$ and iterating, t converges generally in approximately 3 or 4 iterations. Once t is determined, the corresponding projection is obtained using eq.(19). 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.

3.1 Rolling Shutter PnP

The PnP (perspective-N-Points) algorithm allows to estimate the pose of one or many cameras from n 2D-3D matching between 3D points of the scene $\mathbf{P_k}$ and their observations in the images $\mathbf{p}_{i,k}$.

Having an initial estimate of the camera pose $\mathbf{T_w}$ and a set of 2D-3D matching, a reprojection error can be computed as a function of the image i and point k:

$$Err(i,k) = \mathbf{p}_{i,k} - \pi([K|0]{\mathbf{T_w}}^{-1}\mathbf{P_k}) \tag{24}$$

This PnP reprojection error function can be extended to RS camera, taking into account the exposure time of the line where the 3D point is projected:

$$Err(i,k) = \mathbf{p}_{i,k} - \pi([K|0]\mathbf{T_w}(t)^{-1}\mathbf{P_k}) \tag{25}$$

In that case, the parameters to optimize θ are the complete set of CP for the B-spline instead of independent camera pose for each image i. Refined parameters $\hat{\theta}$ are obtained by minimizing the sum of squared reprojection error:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_i \sum_k Err(i,k)^2 \tag{26}$$

3.2 Rolling Shutter Bundle Adjustment

Similarly to the PnP, the BA is an optimization problem that aims to refine a set of parameters by minimizing a cost function but unlike the PnP, it also integrates the 3D points of the scene to the optimization. The parameters to optimize are then defined by $\theta_b = [\theta, P_0..P_M]$, and are refined by minimizing the following reprojection error:

$$\hat{\theta_b} = \underset{\theta_b}{\operatorname{argmin}} \sum_i \sum_k Err(i,k)^2 \tag{27}$$

The minimization can be achieved through Levenberg-Marquardt algorithm, the initial value for θ_b being initialized for instance from a SLAM using a GS camera model. In that case, the CP θ may be initialized directly on the interpolated trajectory.

3.3 Graph representation

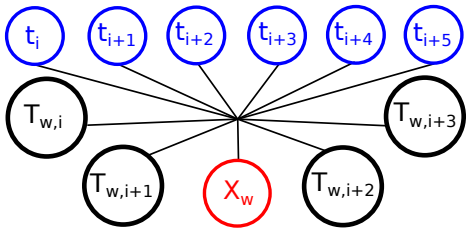

Fig. 4. Graph representation of an observation of a 3D Point $\mathbf{X_w}$ on an image line exposed at time $t_{i+2} < t < t_{i+3}$. The camera pose associated to this line is constrained by the 4 CP $\mathbf{T_{w,i}}$ to $\mathbf{T_{w,i+3}}$ and by the 6 timestamps t_i to t_{i+5}.

We chose to use a graph representation for the optimization problem and used the g2o library[12] as an open source graph optimization tool and we developed a

dedicated C++ solver for both the PnP and BA. The graph representation had to be adapted for the continuous time trajectory model because it is different from the standard formulation where each keyframe corresponds to a single camera pose. In both cases, the observations (2d projections) are represented as edges that connect nodes representing the unknown variables (camera poses and 3D features of the map). In the standard formulation for global shutter cameras, the edges connect two nodes whereas the continuous-time modeling of the camera trajectory requires an observation to be dependent of the 4 neighboring CP and the 6 neighboring timestamps as represented in the figure 4. Thus, a multi-edge is used in the graph to model these dependencies between multiple nodes.

Using this multi-edge representation, a single graph for the PnP and BA is created, the values for the X_W parameters being fixed for the PnP while all the parameters have to be estimated for the BA. g2o offers the numeric evaluation of the jacobians by finite differences.

4 NON UNIFORM TIME DISTRIBUTION OF THE CP

4.1 CP Generation Methods

As our approach uses varying time intervals adapted to the local properties of the motion, it requires the generation of 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 real-time applications as the CP can be generated online using IMU measurements while the second involves an iterative process that predisposes it for offline computation.

Before providing the details of the CP generation methods, it is worth noting that the NUTD 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 5). 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 a chronologically ordered list and using hash tables for fast access.

CP Generation based on IMU

An IMU sensor is generally composed of gyroscopes and accelerometers that respectively measure the angular velocities and linear accelerations. On current tablets and smartphones, it generally delivers measurements at about 100Hz, which is a 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 6 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

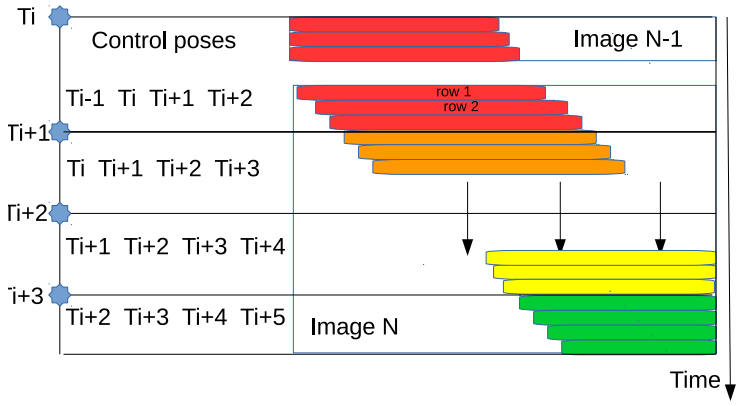

Fig. 5. An example of multiple CP inside a single image, and their influence for interpolating the pose for different rows. 3 CP are time-stamped between the beginning and the end of the image N: $\mathbf{T_{i+1}}$, $\mathbf{T_{i+2}}$ and $\mathbf{T_{i+3}}$. The quartet of CP used for the interpolation of the pose at each row changes three time during the image exposure. Image from [21].

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 as a trade-off 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.(26).

CP Generation based on Reprojection Error We propose a second method to determine the temporal location of the CP when an IMU in not available or as a post processing step after the application of the IMU based method. This method involves an iterative estimation of the trajectory using initial CP (set for instance with an UTD or with an NUTD obtained from the IMU based method) as shown in eq.(26). Let $\overline{res_{t_1,\Delta_t}}$ be the mean of the residuals $Err(i,k)$ defined in eq.(25) between times t_1 and $t_1 + \Delta_t$. The proposed method uses 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. Maxima 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.(26) is achieved with the added CP to refine the whole set of CP.

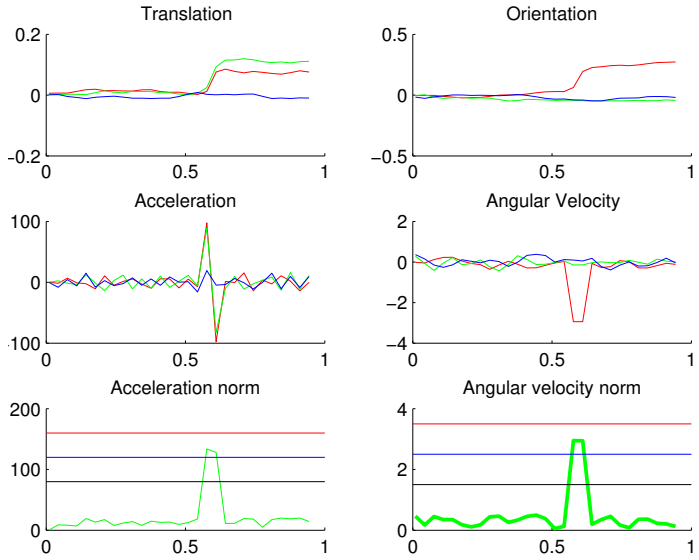

Fig. 6. Determination of the number of required CP per unit of time using IMU measurements analysis (See the text for details). International system of units are used for the axes. Image from [21].

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

These two methods allow to generate new CP specifically on sections of the trajectory that exhibit high dynamic. Despite the fact that the IMU based method is more intuitive because it is based on a physical measurement of the trajectory, the reprojection based method is generally preferable for two main reasons: First, the thresholds in the IMU based method are required to be set cautiously, which is not the case of the reprojection error based method for which the threshold is just expressed in pixel unit. Second,the reprojection error based method uses the same metric than the PnP or BA algorithms, hence it can be efficiently integrated in those contexts.

The figure 7 shows one translation component w.r.t time of a simple simulated ground truth trajectory (a) on which the reprojection error method is tested. The UTD interpolation gives poor results (b) and the associated reprojection error (c) is significant when the high dynamic motion occurs ($t \approx 0.5$). By adding a single CP at the right time, the proposed method accurately models the trajectory (d), dividing the reprojection error by approximately fifteen (e)).

4.2 Spatiotemporal Optimisation of the Control Poses

The NUTD B-Splines permit the optimization of the CP timestamps, who only operate on the cumulative coefficient matrix $\mathbf{C}$ (eq. 10). However, these added parameters have to respect the following constraints to forbid the optimization to

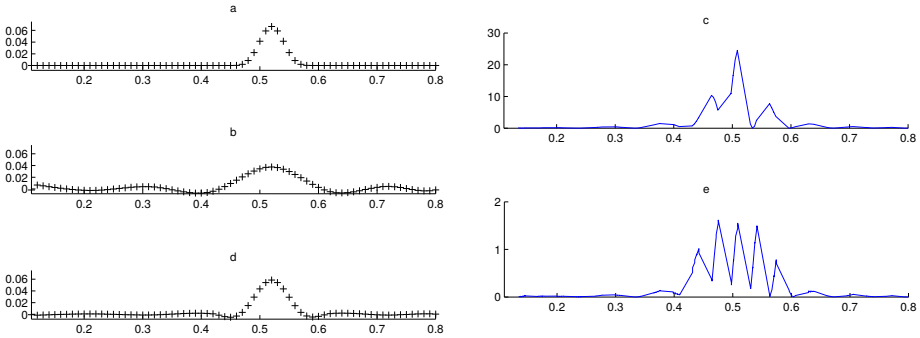

Fig. 7. Results for the reprojection error based method (see text for details).

diverge, Δt_{min} being the minimum temporal distance between two timestamps and t_{total} being the duration of the whole trajectory:

$$\forall i : \Delta t_{i-1,i} > \Delta t_{min} > 0 \tag{28}$$

$$\sum_i \Delta t_{i-1,i} = t_{total} \tag{29}$$

Starting from a poor estimate, simultaneous optimization of the CP and their timestamps can converge to an erroneous local solution, due to local minima. Trying to overcome this problem, we propose to firstly alternate spatial and temporal only optimizations. A complete spatiotemporal optimization can then be applied to refine the estimated trajectory and adapt the CP time distribution to its actual dynamic.

5 RESULTS

5.1 Synthetic Datasets

Results on simulated dataset are firstly presented to avoid noise induced by image processing (such as corner extraction) and problems arising from erroneous data association (between the extracted corners and 3D map points).

Initially, a set of CP is generated to model different types of reference trajectories and a simple 3D point cloud is used as geometric model of the environment. The figure 8 shows an example of generated data. Gaussian noise is added both to these CP and 3D points. The time intervals between the CP are randomly generated to obtain a NUTD and validate the model.

The camera frequency is arbitrary chosen ($fps = 3$), and the start and end exposure time of each image is set accordingly. The camera poses are interpolated using these parameters and the 3D points are projected to the images using the RS projection model. Gaussian noise is then added to these projections to generate the measurements used for the optimization.

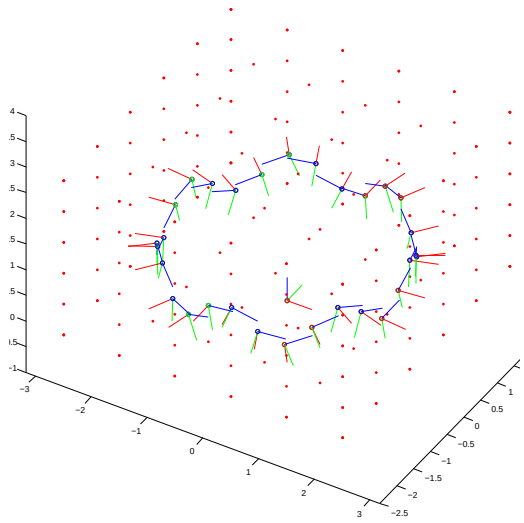

Fig. 8. Example of CP along a circular trajectory, the viewing direction being aligned with the tangent to the circle. An additional sinusoidal translation component is added perpendicularly to the circle plane. The point cloud is displayed in red.

To demonstrate the PnP, the interpolated camera poses using the noisy CP are used as initial estimates for the CP to optimize. A hundred camera poses are interpolated from 60 CP and are used to project the 3D point cloud. The PnP is achieved by our C++ solver and example of results are shown in the figure 9. Initial camera poses (green) converge toward the ground truth (red) after optimization (blue). Only 5 iterations were sufficient to refine those camera poses despite significantly noisy initial estimates.

The BA is demonstrated with the same synthetic trajectories, but using the noisy point cloud as initial estimate for the map, which is to be refined by optimization. The results are shown in the figures 10 and 11.

5.2 Real dataset

To motivate the proposed model, we first validate that it can be used to interpolated the trajectory of a real hand-held camera. This camera is equipped with motion capture markers and tracked by a VICON MOCAP providing the camera poses at 200 Hz which are used as a ground truth. Distinct images sequences and associated ground truth trajectories are captured, with different dynamics in translation and rotation.

We seek to fit interpolated trajectories to the ground truth ones with UTD and NUTD B-Splines model following different optimization schemes. The initial CP are set onto the MOCAP trajectory using UTD. Error $Err_T \in \mathbb{R}6$ between interpolated poses $\mathbf{T_w}(t)$ and the ground truth $\mathbf{T_{GT}}(t)$ is minimized for all timestamps t_j of the samples acquired by the MOCAP:

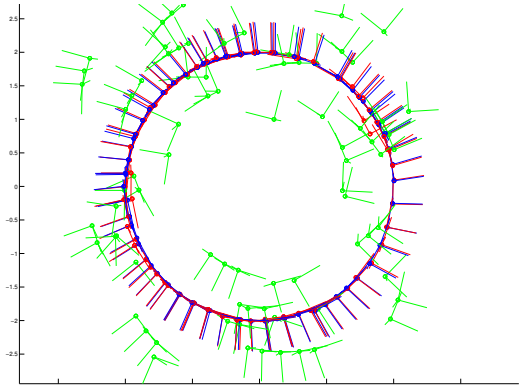

Fig. 9. Illustration of the PnP optimization on a synthetic dataset. Camera poses interpolated from initial CP (green), optimized CP (blue) and the ground truth (red).

$$Err_T = \sum_j log(\mathbf{T_{GT}}(t_j)^{-1}\mathbf{T_w}(t_j)) \tag{30}$$

The minimization is achieved using the Levenberg-Marquardt algorithm and jacobians computed by finite differences. This optimization allows to refine spatially the CP. As described earlier, additional CP are iteratively generated between the two CP where surrounding the highest error after each optimization step.

The figure 12 shows the mean translation error after spatial optimization with respect to the number of CP, for a UTD (red) and a NUTD (green). As expected, at equal number of CP, the NUTD allows a more accurate modeling of the trajectory than with a UTD, because the algorithm adds CP where required. For a greater number of CP, the UTD and NUTD tend to produce similar results as seen in the left plot, due to the fact that with a sufficient amount of CP, even a UTD accurately represents the trajectory.

It is also important to note that for a greater number of CP, the error (mainly for UTD but also for NUTD) can increase as it is shown in the right plot, because the initial estimates of the CP can be set such that the trajectory cannot be correctly approximated.

The CP generation method adds CP at arbitrary time between the timestamps of the two neighboring CP, basically at $\frac{t_{i+2}+t_{i+3}}{2}$. The time distribution is then refined by optimizing the timestamps of the CP.

The spatiotemporal optimization offers in the majority of the cases a significant reduction of the error. However, better results have been observed by following an alternate optimization scheme. A first spatial optimization is done until it converges. Then these 3 steps are applied n times:

- Temporal optimization
- Adding a CP where the error is maximal
- Spatial optimization

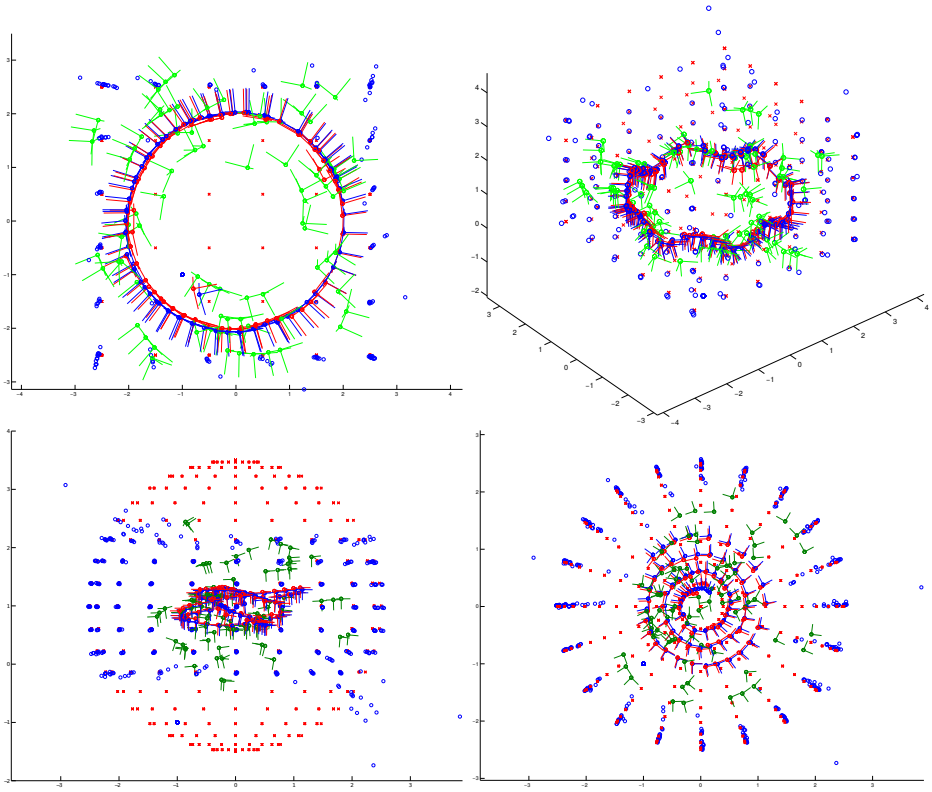

Fig. 10. Different views for the BA using the NUTD model: The initial CP (green), the CP after optimization (blue) and the ground truth (red) are displayed alongside the optimized point cloud (using the same color coding).

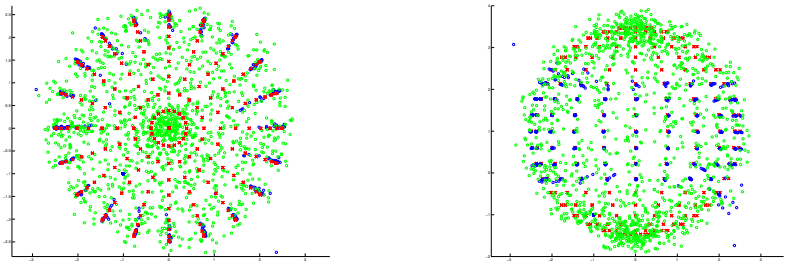

Fig. 11. Different views of the point cloud before (green) and after BA (blue). The ground truth is displayed in red. Only the 3D points that have been observed in the images have been optimized, hence the blue dots do not cover the whole scene.

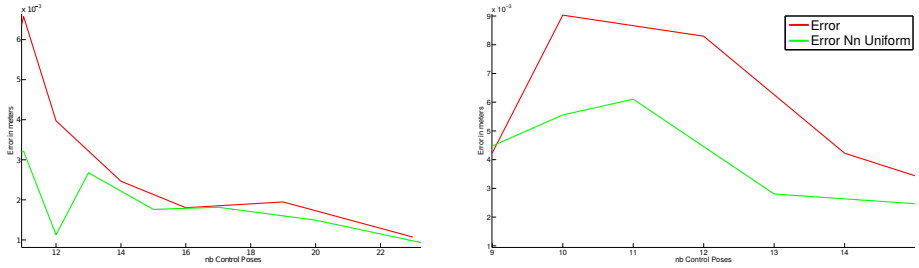

Fig. 12. The mean translation error with respect to the number of CP for two different trajectories (left and right) with a UTD (red) and NUTD (green) for the CP.

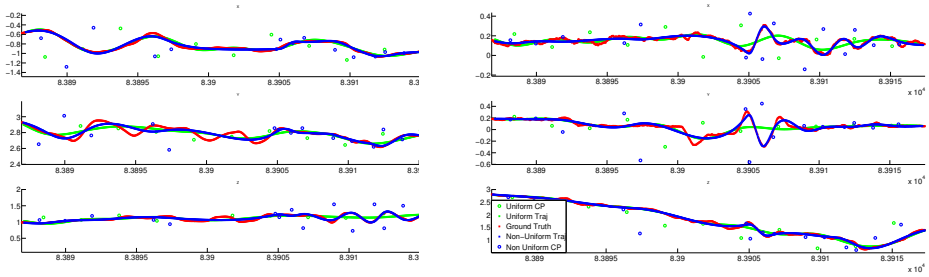

Fig. 13. X, Y, et Z translation (left) and rotation (right) components of the interpolated trajectories after spatial optimization only (green) compared with the interpolated trajectories after 3 iterations of the spatiotemporal optimization scheme (blue) with respect to time. The ground truth is shown in red and is better approximated by the NUTD interpolation (blue) than by the UTD one (green).

The figure 13 shows the interpolated trajectory components after spatial optimization (green), and after $n = 3$ spatiotemporal optimization cycles (blue), alongside the ground truth (red). The temporal and spatial distributions of the CP are adapted automatically to the trajectory dynamic. This is noticeable on the rotation components (right) and the translation z component (lower left) of the trajectories, where the UTD model fail to model fast oscillations.

Close-up view on different trajectories are given in figures 14 and 15 illustrating the final CP distributions. For the NUTD, more CP are used and optimized both temporally and spatially to accurately model fast motions, while less significant motions are smoothed. Note that with additional optimization and CP addition cycles, the CP distribution would adapt to the whole trajectory, however the error reduction would be less significant. There is obviously a trade-off between the increase in accuracy and the induced increase in terms of computational cost.

The figure 16 highlights the evolution of the error (red) during the optimization iterations. It is important to note that the plots begin after the convergence of a first spatial optimization. Hence, without the proposed approach, the minimum achievable error through standard spatial optimization corresponds to the

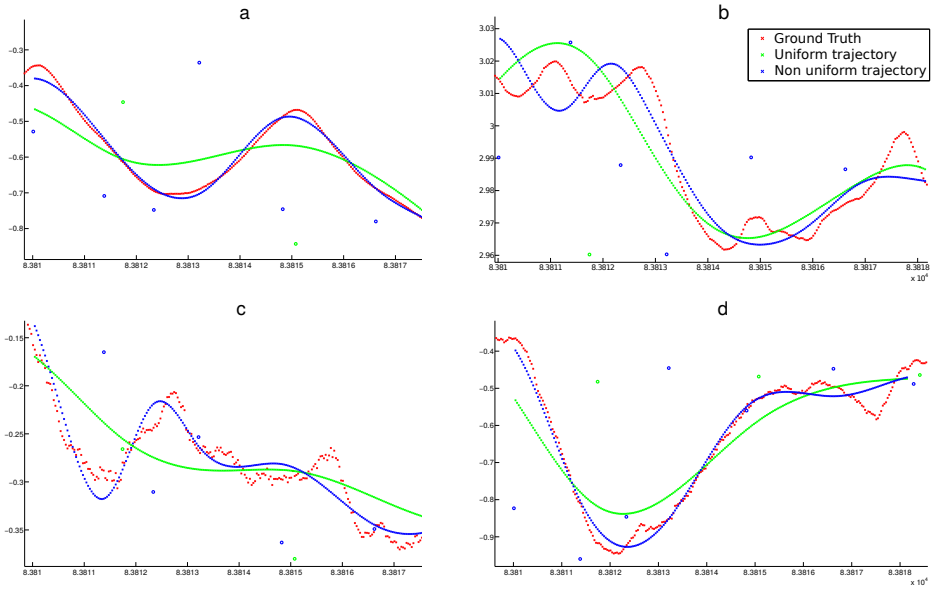

Fig. 14. X translation (a), Y translation (b), X rotation (c) and Y rotation (d) components of the interpolated trajectories in sequence 1.

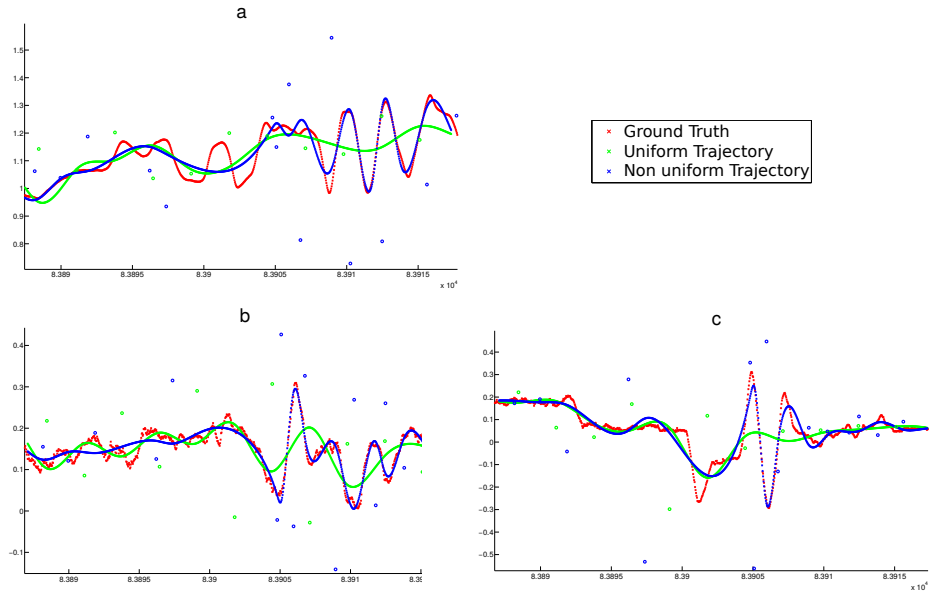

Fig. 15. Z translation (a), X rotation (b) and Y rotation (c) components of the interpolated trajectories in sequence 4.

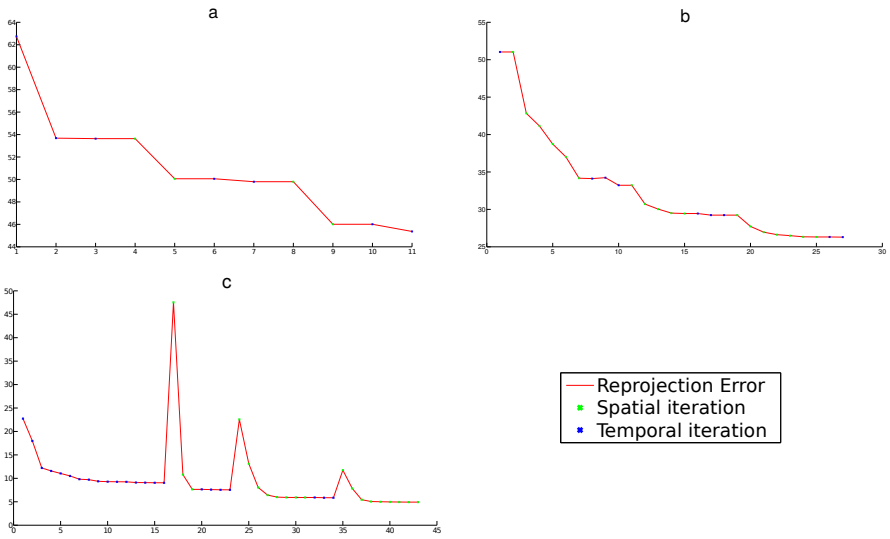

Fig. 16. Evolution of the translation error (red) as a function of space (green) and time (blue) optimization iterations. The optimization follows temporal then spatial optimization steps with without addition of CP in two first cases (a,b) whereas it required additional CP in the third case (c). In this test, the non optimal initial values for the added CP at iterations 16,24 and 35 are the reason of the temporarily increased error. The error is then minimized, taking into account these added CP.

first error displayed in the plots. The temporal optimization allows to reduce this error by a significant factor using either no (a,b) or a low number (c) of additionnal CP.

6 FUTURE WORKS

The conducted experiments demonstrated that it is possible to adapt the spatiotemporal distribution of the CP to the dynamic of real trajectories through the proposed optimization process. The used cubic interpolation model allows an accurate modeling of the trajectory. Quadratic or even linear models can be sufficient depending on the type of trajectory, however the availability of interpolated velocities and accelerations can be useful for applications using inertial measurements. The cost function used in our experiments on real datasets is relative to a ground truth that is not available in the case of a SLAM process whereas the available inertial measurements would be compared with the derivatives of the interpolated trajectory. Hence it could be possible to integrate the inertial measurments to the optimization as done in [19], which would allow to recover the metric scale of the scene and to perform auto-calibration.

The presented work is still an early work, and new tests within a complete SLAM process using the NUTD model must be driven on real datasets, involving the complete image processing pipeline. We also plan to integrate other geometric

features such as segments, lines and planar patches to describe the environment and adapt their observation models to the RS cameras.

CONCLUSION

A NUTD cumulative B-Spline model maintaining the C^2 continuity of the interpolated trajectory have been presented. This model have been tested to fit interpolated trajectories to real and synthetic ones using different optimization schemes.The improvement offered by the optimization of the CP timestamps have been demonstrated for both simulated datasets and real trajectories. The integration of the model within a BA have been shown for simulated datasets only as it did not involved to operate image processing and because the ground truth for the environment was directly available.

The continuous-time trajectory model using a NUTD for the CP allows a reduction of both the memory usage and the computational cost. For trajectories with large partially linear parts, as encountered in automotive applications, a large amount of camera poses can be efficiently parameterized by a small number of CP. For trajectories with varying dynamics, it is possible to accurately model the high speed motion by locally increasing the CP density.

References

1. de Boor, C.: On calculating with b-splines. Journal of Approximation Theory **6**(1), 50 – 62 (1972). https://doi.org/https://doi.org/10.1016/0021-9045(72)90080-9, http://www.sciencedirect.com/science/article/pii/0021904572900809
2. Davison, A.J.: 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. pp. 1403–1410 (2003). https://doi.org/10.1109/ICCV.2003.1238654, http://dx.doi.org/10.1109/ICCV.2003.1238654
3. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: ECCV (September 2014)
4. Furgale, P., Barfoot, T.D., Sibley, G.: Continuous-time batch estimation using temporal basis functions. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on. pp. 2088–2095 (May 2012). https://doi.org/10.1109/ICRA.2012.6225005
5. Gonzalez, A.: Localisation par vision multi-spectrale. Application aux systèmes embarqués. Theses, INSA de Toulouse (Jul 2013), https://tel.archives-ouvertes.fr/tel-00874037
6. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edn. (2004)
7. Hedborg, J., Forssn, P.E., Felsberg, M., Ringaby, E.: Rolling shutter bundle adjustment. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 1434–1441 (June 2012). https://doi.org/10.1109/CVPR.2012.6247831
8. Hedborg, J., Ringaby, E., Forssn, P.E., Felsberg, M.: Structure and motion estimation from rolling shutter video. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on. pp. 17–23 (Nov 2011). https://doi.org/10.1109/ICCVW.2011.6130217

9. Kim, M.J., Kim, M.S., Shin, S.Y.: A general construction scheme for unit quaternion curves with simple high order derivatives. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques - SIGGRAPH '95 pp. 369–376 (1995). https://doi.org/10.1145/218380.218486, http://portal.acm.org/citation.cfm?doid=218380.218486
10. Klein, G., Murray, D.: 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 (November 2007)
11. Klein, G., Murray, D.: Parallel tracking and mapping on a camera phone. In: Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09). Orlando (October 2009)
12. Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp. 3607–3613. Shanghai, China (May 2011). https://doi.org/10.1109/ICRA.2011.5979949
13. Li, M., Kim, B., Mourikis, A.I.: 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. pp. 4697–4704. Karlsruhe, Germany (May 2013)
14. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Real time localization and 3d reconstruction. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 1, pp. 363–370 (2006). https://doi.org/10.1109/CVPR.2006.236
15. Mur-Artal, R., Montiel, J.M.M., Tards, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (Oct 2015). https://doi.org/10.1109/TRO.2015.2463671
16. Patron-Perez, A., Lovegrove, S., Sibley, G.: A spline-based trajectory representation for sensor fusion and rolling shutter cameras. Int. J. Comput. Vision **113**(3), 208–219 (Jul 2015). https://doi.org/10.1007/s11263-015-0811-3, http://dx.doi.org/10.1007/s11263-015-0811-3
17. Qin, K.: General matrix representations for B-splines. The Visual Computer **16**(3), 177–186 (2000). https://doi.org/10.1007/s003710050206, https://doi.org/10.1007/s003710050206
18. Roussillon, C., Gonzalez, A., Solà, J., Codol, J., Mansard, N., Lacroix, S., Devy, M.: RT-SLAM: A generic and real-time visual SLAM implementation. CoRR **abs/1201.5450** (2012), http://arxiv.org/abs/1201.5450
19. Steven Lovegrove, Alonso Patron-Perez, G.S.: 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 (2013)
20. Strasdat, H., Montiel, J., Davison, A.J.: Real-time monocular slam: Why filter? In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. pp. 2657–2664. IEEE (2010)
21. Vandeportaele, B., Gohard, P.A., Devy, M., Coudrin, B.: Pose interpolation for rolling shutter cameras using non uniformly time-sampled b-splines. In: Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 6: VISAPP, (VISIGRAPP 2017). pp. 286–293. INSTICC, SciTePress (2017). https://doi.org/10.5220/0006171802860293
22. Yang, H., Yue, W., He, Y., Huang, H., Xia, H.: The Deduction of Coefficient Matrix for Cubic Non-Uniform B-Spline Curves. 2009

First International Workshop on Education Technology and Computer Science (3), 607–609 (2009). https://doi.org/10.1109/ETCS.2009.396, http://ieeexplore.ieee.org/document/4959111/