Dense Continuous-Time Tracking and Mapping with Rolling Shutter RGB-D Cameras

Christian Kerl, Jörg Stückler, and Daniel Cremers
Technische Universität München
{kerl,stueckle,cremers}@in.tum.de

Abstract

We propose a dense continuous-time tracking and mapping method for RGB-D cameras. We parametrize the camera trajectory using continuous B-splines and optimize the trajectory through dense, direct image alignment. Our method also directly models rolling shutter in both RGB and depth images within the optimization, which improves tracking and reconstruction quality for low-cost CMOS sensors.

*Using a continuous trajectory representation has a number of advantages over a discrete-time representation (*e.g. *camera poses at the frame interval). With splines, less variables need to be optimized than with a discrete representation, since the trajectory can be represented with fewer control points than frames. Splines also naturally include smoothness constraints on derivatives of the trajectory estimate. Finally, the continuous trajectory representation allows to compensate for rolling shutter effects, since a pose estimate is available at any exposure time of an image. Our approach demonstrates superior quality in tracking and reconstruction compared to approaches with discrete-time or global shutter assumptions.*

1. Introduction

Most current approaches to simultaneous localization and mapping (SLAM) with RGB-D cameras estimate the trajectory at discrete times, *e.g.* one pose for each frame [2, 6, 10]. Implicitly, the discrete pose of a frame needs to apply to all pixels in the image—an assumption that is clearly violated for rolling shutter cameras. In effect, the alignment of two images is biased towards a wrong estimate for such cameras. Since most consumer-grade RGB-D cameras use rolling shutter CMOS sensors, the use of a trajectory representation is desirable that can provide a camera pose estimate for each sensor row individually and allows for compensating for rolling shutter.

In this paper, we propose to optimize a continuous-time trajectory representation in the form of B-splines through direct, dense image alignment. This way, our approach can incorporate parameters of sensor exposure such as exposure durations and time differences of the RGB and depth sensor. The trajectory can be represented with much less parameters, i.e. control points, than pose variables per frame. The spline parametrization also inherently adds constraints on derivatives and produces smooth trajectories. Hence, it regularizes the ill-posed problem of estimating the camera pose for each image row. A further advantage of the spline representation is the easy integration of other sensing modalities such as inertial measurement units (IMU). These sensors can have arbitrarily lower or higher measurement rates than the camera and do not need to measure synchronously with the frames.

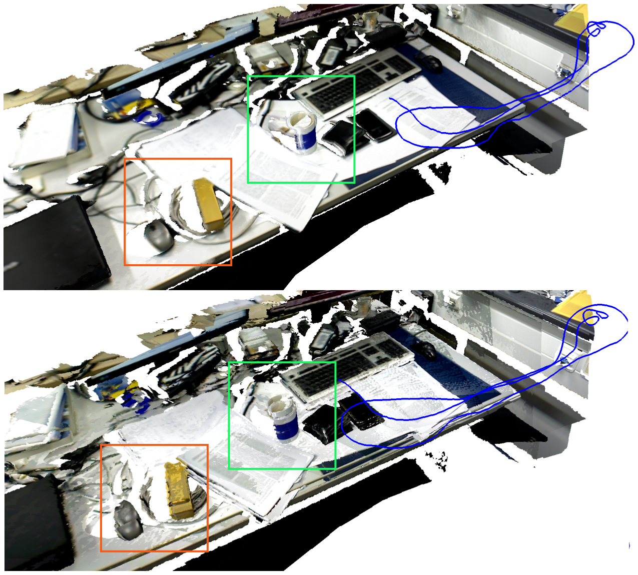


Figure 1: We propose a dense tracking and mapping method to estimate a continuous-time trajectory from a sequence of rolling shutter RGB-D images. Our approach significantly reduces trajectory drift leading to more consistent 3D reconstructions (top) compared to a state-of-the-art algorithm which estimate discrete camera poses w.r.t. to keyframes (bottom). Neither method involves global optimization.

By explicitly modeling rolling shutter in direct image alignment, our method achieves improvements in tracking and reconstruction quality. In our SLAM method, we use the continuous trajectory estimate to fuse RGB-D frames consistently in keyframes over time and remove rolling shutter effects. We also accurately quantify motion blur at each pixel, and use this measure to improve the quality of the color reconstruction. We compare our method with approaches estimating discrete camera poses or relying on global shutter assumptions, and demonstrate superior performance for tracking and reconstruction towards the state-of-the-art. Figure 1 shows a 3D reconstruction of our proposed method (top) in comparison to a baseline algorithm (bottom). The better performance of our method leads to a consistent 3D model even without global optimization.

2. Related Work

Visual SLAM and structure from motion (SfM) has been traditionally formulated with discrete-time trajectories [2, 6, 10]. Recently, Furgale *et al.* [3] proposed a continuous-time spline-based trajectory representation for keypoint-based monocular and stereo SLAM. They explore continuous trajectories for the main purposes of reducing the state-space and for seamlessly integrating high-rate IMU measurements. Lovegrove *et al.* [9] formulate monocular SLAM in a compositional spline representation in the $\mathfrak{se}(3)$ Lie algebra, overcoming problems of the Cayley-Gibbs-Rodrigues representation of poses used in [3]. Their approach compensates for rolling shutter, however, in contrast to our dense method, sparse keypoints are matched between images. They demonstrate SLAM in a small-scale calibration example and on a synthetic sequence.

Several approaches correct for rolling shutter in monocular, stereo, or RGB-D images. Klein and Murray [8] estimate camera motion in a monocular image from the movement of keypoints between frames and use this motion to determine a rolling-shutter compensated image. Baker *et al.* [1] estimate translational pixel motion in a frame from optical flow. The approach in [15] estimates a linear interpolation spline of rotational camera motion from KLT feature tracks in a short window of frames. A similar approach for modeling rolling shutter is used in the bundle adjustment approach in [5]. Here, the pose of the top row of each image is estimated and interpolated linearly between frames. For RGB-D cameras, only a few approaches exist. Ringaby *et al.* [14] remove rolling shutter in the depth image of a structured light sensor by applying the keypoint-based correction method [15] to the infrared image. In [13], a 3-axis gyro is used to determine the rotation of the camera instead. Meilland *et al.* [11] assume the camera velocity to be linear during the exposure of a frame and determine the motion through direct image alignment.

Our method introduces a generalized continuous-time spline-based trajectory representation to consider rolling shutter for direct RGB-D image alignment and SLAM. We model rolling shutter in both the RGB and depth image. We also propose to fuse the rolling shutter corrected RGB-D images in keyframes and reduce the effects of motion blur based on the motion estimate of individual pixels.

3. Approach

The RGB-D camera provides us with a sequence of RGB and depth images in the time interval $[t_{\text{min}}, t_{\text{max}})$. Our goal is to estimate a trajectory function $\boldsymbol{T}(t) : \mathbb{R} \rightarrow \text{SE}(3)\, \forall\, t \in [t_{\text{min}}, t_{\text{max}})$ from the image sequence using dense image alignment.

An RGB image will be denoted by $\mathcal{C} : \Omega_{\mathcal{C}} \rightarrow \mathbb{R}^3$, the derived intensity image by $\mathcal{I} : \Omega_{\mathcal{I}} \rightarrow \mathbb{R}$ and a depth image by $\mathcal{Z} : \Omega_{\mathcal{Z}} \rightarrow \mathbb{R}$. The corresponding timestamps are $t_{\mathcal{C}}$, $t_{\mathcal{I}}$ and $t_{\mathcal{Z}}$. We do not require the timestamps of an RGB-D image pair to be synchronized. For simplicity we assume that all image domains have the same dimensions with width w and height h.

3.1. Camera Model

We model the RGB and depth cameras as rolling shutter cameras with pinhole projection including radial and tangential lens distortion. We assume the intrinsic and extrinsic parameters of both cameras to be known. The focal length will be denoted by f and the offset of the projection center by o_x along the image x-axis and o_y for the y-axis respectively. The relative transformation between RGB and depth camera is $\boldsymbol{T}_{\text{cam}}$. For clarity of presentation we omit the lens distortion in the projection equation.

The projection function π mapping a 3D point $\boldsymbol{p} = (X, Y, Z)^{\mathsf{T}}$ to a 2D pixel $\boldsymbol{x} = (x, y)^{\mathsf{T}}$ is defined as:

$$\boldsymbol{x} = \pi(\boldsymbol{p}) = \left(\frac{X}{Z} f + o_x, \frac{Y}{Z} f + o_y \right). \tag{1}$$

The inverse projection function π^{-1} reconstructs a 3D point given a pixel location $\boldsymbol{x}$ and a depth image $\mathcal{Z}$:

$$\boldsymbol{p} = \pi^{-1}(\boldsymbol{x}, \mathcal{Z}(\boldsymbol{x})) = \left(\frac{x - o_x}{f}, \frac{y - o_y}{f}, 1 \right)^{\mathsf{T}} \mathcal{Z}(\boldsymbol{x}). \tag{2}$$

For a rolling shutter camera the pose which transforms a world point into the camera frame depends on the pixel location the point projects to. This can be formalized as the following constraint:

$${}^{t_r}\!/_{h} \left[\pi(\boldsymbol{T}^{-1}(t_0 + t_x)\, \boldsymbol{p})\right]_y \overset{!}{=} t_x \tag{3}$$

where t_0 is the time of the first row, t_r is the read out time of the camera, and h is the number of rows. The read out time specifies the time difference between the capture of the first and the last row. Here we assume that the camera reads the image row-wise. To project a point with arbitrary $\boldsymbol{T}(t)$ we solve (3) for t_x using the Newton-Raphson method.

3.2. Trajectory Representation

We use the cumulative cubic B-spline representation introduced by Lovegrove *et al.* [9] for the trajectory function $\boldsymbol{T}(t)$. Here we briefly repeat the definition of $\boldsymbol{T}(t)$. For a detailed derivation and formulas for the time derivatives we refer the interested reader to [9].

The trajectory is defined over a time interval $[t_{\text{min}}, t_{\text{max}})$. The shape of the trajectory is determined by the set $\mathcal{T}_C$ of m control points $\boldsymbol{T}_{C,l} \in \mathrm{SE}(3)$ placed at discrete times (knots) $t_0 \dots t_l \dots t_{m-1}$ in the time interval. The knots are uniformly distributed in the interval. Each of these knot intervals is Δt wide. In our case we use $\Delta t = 0.05\,\mathrm{s}$. Every point on the trajectory is influenced by four control points. At time t these are the control points at times $\{t_{l-1}, t_l, t_{l+1}, t_{l+2}\}$ if $t \in [t_l, t_{l+1})$. Then the pose at t is:

$$\boldsymbol{T}(t) = \boldsymbol{T}_{C,l-1} \prod_{k=1}^{3} \exp(\boldsymbol{B}_j(u) \boldsymbol{\Omega}_{l-1+k}) \tag{4}$$

with $\boldsymbol{\Omega}_l = \log(\boldsymbol{T}_{C,l-1}^{-1} \boldsymbol{T}_{C,l}) \in \mathfrak{se}(3)$. $u = (t - t_l)/\Delta t$ normalizes t to the interval $[0, 1)$. The cumulative basis functions $\boldsymbol{B}(u)$ are:

$$\boldsymbol{B}(u) = \frac{1}{6} \begin{pmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ u \\ u^2 \\ u^3 \end{pmatrix}. \tag{5}$$

$\boldsymbol{B}_k(u)$ selects the k-th element from $\boldsymbol{B}(u)$ with indices starting at 0. A pose $\boldsymbol{T}(t)$ transforms a point from local coordinates to world coordinates.

Equipped with the rolling shutter camera model and the continuous-time trajectory representation we define in the next sections the dense image alignment terms, which we use to estimate the control points of the spline.

3.3. Dense Image Alignment

To estimate the trajectory function $\boldsymbol{T}(t)$ we use a dense, direct image alignment method, because they are more robust in scenes with little features and on blurred images. In general, direct image alignment methods estimate the motion parameters to align a given *current* image to a *reference* image or model. Iteratively we take a set of points from the reference image, apply a warping function to transfer these points to the current image, where we evaluate an error function measuring consistency between the two, finally the motion parameters are adjusted to increase consistency. In this process the data association between reference and current image is solved implicitly, which is in contrast to feature based methods. In our case the warping function is the composition of back projection, rigid body motion and projection.

For rolling shutter cameras the parameters of the warping function depend on its result as previously described. Therefore, the warping involves for general rigid body motions and projection functions itself an iterative optimization, which increases the computational load. Especially for direct, dense methods, because they have to apply the warping function to a large number of points. Therefore, previous approaches resort to linear motion approximations. In contrast, we present a dense image alignment term, which does not require the projection into a rolling shutter image, and as alternative an efficient strategy to project even with our more general trajectory representation.

We use a geometric and a photometric error term in the image alignment. In the following we describe both error terms.

Geometric Error For the geometric error we take advantage of the fact that the roles of reference and current image can be swapped, *i.e.*, points from the current depth image can be warped to the reference image. For the current depth image we know the capture time for every row and can evaluate the trajectory function at the appropriate point in time. The geometric error for one point is given by

$$\begin{aligned} r_{\mathcal{Z},i,j} = & \mathcal{Z}_{\text{ref}}(\pi(\boldsymbol{T}^{-1}(t_{\mathcal{Z},\text{ref}} + t_x)\, \boldsymbol{p}'_j)) - \\ & [\boldsymbol{T}^{-1}(t_{\mathcal{Z},\text{ref}} + t_x)\, \boldsymbol{p}'_j]_Z \end{aligned} \tag{6}$$

where $\boldsymbol{p}'_j = \boldsymbol{T}(t_{\mathcal{Z},i} + {}^{t_r}/_{h}\, [\boldsymbol{x}_j]_y)\, \pi^{-1}(\boldsymbol{x}_j, \mathcal{Z}_i(\boldsymbol{x}_j))$ is the point from the i-th image transformed into the world frame. $[\cdot]_Z$ selects the Z component of the point. The time offset t_x is either 0, if the reference image is rectified, or it is determined by solving (3). Here, *rectified* means that we removed the rolling shutter effects from the image and a global shutter projection model applies. Note, if we have rectified the reference image, we can solve the alignment problem based on the geometric error without the need to project into a rolling shutter image. This insight is equally applicable to ICP-like error functions.

Photometric Error We define the photometric error for a point visible in the reference intensity image $\mathcal{I}_{\text{ref}}$ and the i-th intensity image $\mathcal{I}_i$ as:

$$\begin{aligned} r_{\mathcal{I},i,j} = & \mathcal{I}_i(\pi(\boldsymbol{T}_{\text{cam}}\, \boldsymbol{T}^{-1}(t_{\mathcal{I},i} + t_{i,x})\, \boldsymbol{p}'_j)) - \\ & \mathcal{I}_{\text{ref}}(\pi(\boldsymbol{T}_{\text{cam}}\, \boldsymbol{T}^{-1}(t_{\mathcal{I},\text{ref}} + t_{\text{ref},x})\, \boldsymbol{p}'_j)) \end{aligned} \tag{7}$$

where $\boldsymbol{p}'_j = \boldsymbol{T}(t_{\mathcal{Z},\text{ref}} + {}^{t_r}/_{h}\, [\boldsymbol{x}_j]_y)\, \pi^{-1}(\boldsymbol{x}_j, \mathcal{Z}_{\text{ref}}(\boldsymbol{x}_j))$ is a 3D point from the reference depth image transformed into the world frame. The time offsets $t_{i,x}$ and $t_{\text{ref},x}$ are determined by solving (3). In case the depth reference image is rectified and has a registered intensity image (7) can be simplified. *Registered* means that we determined the intensity of every 3D point in the reference depth image beforehand. Basically, this can be done by evaluating the second part of (7).

However, it requires an estimate of $\boldsymbol{T}(t)$ during the frame interval of $\mathcal{I}_{\text{ref}}$. The simplified error term is:

$$r_{\mathcal{I},i,j} = \mathcal{I}_i(\pi(\boldsymbol{T}_{\text{cam}}\,\boldsymbol{T}^{-1}(t_{\mathcal{I},i} + t_{\text{cur},x})\,\boldsymbol{p}'_j)) - \mathcal{I}_{\text{ref}}(\boldsymbol{x}_j) \quad (8)$$

with $\boldsymbol{p}'_j = \boldsymbol{T}(t_{\mathcal{Z},\text{ref}})\,\pi^{-1}(\boldsymbol{x}_j, \mathcal{Z}_{\text{ref}}(\boldsymbol{x}_j))$. In this case it is equivalent to the photometric error described by Meilland *et al*. [11] except for our more general trajectory function. In the following we will only use the simplified version (8) of the photometric error assuming we have a rectified and registered reference image. In Section 3.6 we give additional details about frame rectification and registration.

3.4. Non-linear Trajectory Optimization

With the point-wise photometric and geometric errors defined in the previous section we formulate an error function over a whole image, which we minimize with respect to the spline control points parameterizing our trajectory (*c.f.* Section 3.2). Every error term depends on multiple control points. Therefore, multiple images constrain the same control points and require us to minimize the error function jointly over multiple images. This joint optimization is in contrast to all previous dense, direct image alignment methods, which only optimize the relative transformation of the most recent image to some reference image and ignore the temporal relationship of the images.

Given a rectified and registered reference RGB-D image and a set of N rolling shutter RGB and depth images we define the joint photometric error as:

$$r_{\mathcal{I}} = \sum_i^N \sum_j^{M_i} w_{\mathcal{I},i,j}\, r^2_{\mathcal{I},i,j} \quad (9)$$

where M_i is the number of points visible in image $\mathcal{I}_i$ and $r_{\mathcal{I},i,j}$ is the photometric error for one point as defined in (8).

Similarly, we define the joint geometric error for these N images as:

$$r_{\mathcal{Z}} = \sum_i^N \sum_j^{M_i} w_{\mathcal{Z},i,j}\, r^2_{\mathcal{Z},i,j} \quad (10)$$

with M_i being the number of points in the i-th depth image, which project into the reference image.

We also include a per residual weight in the joint errors $w_{\mathcal{I},i,j} = w(r_{\mathcal{I},i,j})$ for (9) and $w_{\mathcal{Z},i,j} = w(r_{\mathcal{Z},i,j})$ for (10). The weight function $w(r)$ is derived from the Student t-distribution [7]:

$$w(r) = \frac{\nu + 1}{\nu + (r/\sigma)^2} \quad (11)$$

where ν are the degrees of freedom (fixed to $\nu = 5$) and scale parameter σ. We estimate $2\,N$ scale parameters σ. One for the photometric and one for the geometric errors of each of the N images.

When we align two images only the relative pose is observable. However, with our trajectory representation we optimize the poses w.r.t. the world frame. Therefore, we fix the control points defining the pose of the rectified reference image $\boldsymbol{T}(t_{\mathcal{Z},\text{ref}})$. With one pose fixed every point-wise error influences four control points. As the capture interval of one image $[t_0,\, t_0 + t_r)$ might span multiple knot intervals of the spline, one image influences four or more spline control points. Let $\widetilde{\mathcal{T}}_C \subset \mathcal{T}_C$ denote the set of control points influenced by the N images excluding the fixed ones of the reference pose. Then we define our joint trajectory optimization problem for the set of optimal control point poses $\widetilde{\mathcal{T}}_C^*$ as:

$$\widetilde{\mathcal{T}}_C^* = \arg\min_{\widetilde{\mathcal{T}}_C} r_{\mathcal{I}} + r_{\mathcal{Z}} \quad (12)$$

This is a non-linear weighted least squares problem, which we solve iteratively using the Gauss-Newton method. The error function is linearized with respect to small increments $\Delta\boldsymbol{T}_C$ to the control points. The increments are represented using the Lie algebra $\mathfrak{se}(3)$. In every iteration we solve the normal equations $\boldsymbol{A}\,\Delta\boldsymbol{T}_C = -\boldsymbol{b}$. The Hessian matrix $\boldsymbol{A}$ has a band diagonal structure. Every point-wise error contributes a dense 24×24 block to $\boldsymbol{A}$. In Section 3.7 we provide details how to efficiently compute the Jacobians to construct the normal equations.

As common in dense image alignment we use a coarse-to-fine scheme to increase the convergence radius. The optimization for $\widetilde{\mathcal{T}}_C^*$ starts with a low resolution version of each of the N images and sequentially increases the resolution as the affected control points converge. To choose the appropriate resolution for an image we use the following strategy. We associate with every control point $\boldsymbol{T}_{C,l}$ an image resolution. Furthermore, we keep track of the magnitude of the increments to the control point. Once this magnitude falls below a threshold or after a maximum number of iterations we increase the resolution associated with the control point. We choose the lowest resolution among all affected control points per image. When control points converge on the highest resolution they are excluded from further optimization.

3.5. Online Trajectory Estimation

We apply our optimization strategy in an online system where new RGB and depth images become available incrementally. In our system we choose certain frames as keyframes. The latest keyframe serves as reference image in the direct image alignment. Previous keyframes are merely kept for visualization purposes. For a new image the spline is expanded by adding new knots such that the capture interval $[t_0,\, t_0 + t_r)$ of the new frame lies entirely inside the trajectory interval $[t_{\text{min}}, t_{\text{max}})$. We initialize the new control points simply with the value of the last control point. Optimization for the new image starts on the lowest resolution.

Furthermore, we can remove old frames from the optimization for which all control points converged. We fuse these converged frames into the latest keyframe as described in the next section. For the online estimation we choose the image resolution only once before the non-linear optimization and keep it fixed during the iterations.

At some point, the latest image will have too little overlap with the keyframe to reliably estimate the pose. Therefore, we have to choose a new one. The simplest strategy is to choose the last converged frame as new keyframe. An alternative is to choose the last frame and initially only use the geometric error term with rolling shutter reference. After the control points of this new keyframe converge it can be rectified and the photometric and geometric term can be used. This more involved approach helps to keep track during camera motions with fast viewpoint changes. Furthermore, this strategy allows to bootstrap the system, because the geometric term does not require a rectified reference. We always propagate the depth and RGB values from the last keyframe to the new, rectified keyframe. Therefore, we loose all information about the scene, which is not represented in the new keyframe, rendering our trajectory estimation an odometry method. Alternatively, if a consistent model of the scene is available, *e.g.*, as signed distance function volume [12] or a set of keyframes [10], the reference frames can be synthesized from this model.

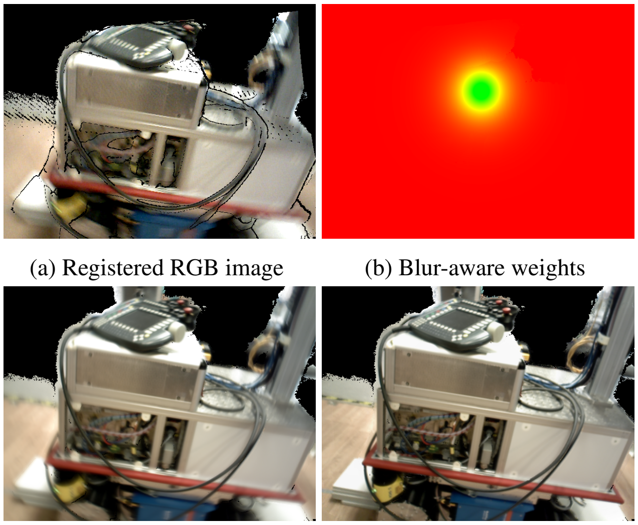
3.6. Frame Rectification and Fusion

Once all control points for an RGB-D frame converged, we rectify the frame and fuse it into the keyframe. To fuse the depth image, we first warp it to the keyframe using the GPU-based technique described by Meilland *et al.* [10]. To correct for the rolling shutter distortion we use a different transformation per row, which we obtain by evaluating $\boldsymbol{T}(t)$ at the corresponding timestamps. Afterwards, we update for each pixel in the keyframe the weighted average of the depth value. The weights take the distance based uncertainty of the depth values into account. The forward warping has the benefit, that we can fill pixels with missing data.

For the fusion of the RGB images we warp every point of the keyframe into the RGB image, taking the rolling shutter constraint into account, and lookup the new RGB measurement. In practice, we also handle self occlusions of those points. All RGB measurements for a point in the keyframe are combined using a moving average. We propose to use the following weighting function for the RGB measurements, which models the amount of motion blur:

$$w_{\mathcal{C},i}(\boldsymbol{p}) = \exp\left(-\frac{\|\boldsymbol{x}_t - \boldsymbol{x}_{t-t_e}\|}{\sigma^2}\right) \qquad (13)$$

where $\boldsymbol{x}_t = \pi(\boldsymbol{T}_{\text{cam}}\,\boldsymbol{T}^{-1}(t_{\mathcal{C},i} + t_{i,x})\,\boldsymbol{p})$ is the pixel location the point $\boldsymbol{p}$ (in world coordinates) projects to in the i-th RGB image, and $\boldsymbol{x}_{t-t_e} = \pi(\boldsymbol{T}_{\text{cam}}\,\boldsymbol{T}^{-1}(t_{\mathcal{C},i} + t_{i,x} - t_e)\,\boldsymbol{p})$ is the pixel at the beginning of the frame exposure. Therefore, $\|\boldsymbol{x}_t - \boldsymbol{x}_{t-t_e}\|$ is the length of a line approximating the path of the point in the image during exposure time t_e. The more pixels a point covers the stronger motion blur is and correspondingly $w_{\mathcal{C},i}(\boldsymbol{p})$ will be lower. In all experiments we set $\sigma = 1$. Figure 2 illustrates the effects of our blur-aware weight function in comparison to a simple constant weight. Figure 2a shows an RGB image, which was registered to the keyframe. In Figure 2b we show the corresponding weights where green indicates high and red low weights. Note how the green spot matches the sharp area in the center of the registered RGB image. The images in the bottom row show the color image of the keyframe after fusing multiple images with constant weights (Figure 2c) and our proposed weights (Figure 2d). It is clearly visible that our weights suppress the influence of blurred images on the fusion result. Our weighting function, which quantifies the amount of motion blur, is complementary to other, *e.g.* normal-based, weight functions previously proposed. Whenever we propagate the fused RGB and depth values from the last keyframe to a new one we also propagate their corresponding weights.


(a) Registered RGB image

(b) Blur-aware weights

(c) Fused keyframe using constant weights

(d) Fused keyframe using blur-aware weights

Figure 2: Illustration of our weighting to suppress motion blur in the fused keyframes: a) rectified input image with motion blur, b) computed blur-aware weights where green indicates a high and red a low value, c) fused keyframe averaged with constant weights, and d) keyframe fused using blur-aware weights for the average. Note how our weighting scheme retains sharp details.

3.7. Implementation Details

The main computational burden in dense, direct image alignment is the evaluation of the error function and the construction of the normal equations. In both parts we found operations which evaluate the trajectory $\boldsymbol{T}(t)$ most costly.

For the geometric alignment a separate transformation for every point depending on its image row is required. Therefore, we precompute the pose of every row for all the images currently involved in the optimization once per iteration. Similarly, for the projection into a rolling shutter image we need to repeatedly evaluate the pose and its derivative w.r.t. time. We found it sufficient to use the pose and time derivative of the closest integer row index, which we also precompute for every image included in the optimization. Note, that the projection still results in subpixel locations. An even coarser sampling of the poses and velocities, *e.g*. every 5th or 10th row, might be sufficient. Using only one pose and velocity per image results in a constant velocity model.

Another costly operation is the computation of the Jacobians and building the normal equations. Lovegrove *et al*. use numerical derivatives [9] that is feasible for a small number of feature points, but prohibitively expensive for the amount of points involved in dense image alignment. Therefore, we derive analytic expressions for the photometric and geometric error terms and a decomposition, which reduces the additional effort for the spline representation compared to a single pose for each image to a factor proportional to the height h of the images instead of the number of pixels $h \times w$. The derivative of the simplified photometric error (8) w.r.t. the increments to the four control points defining $\boldsymbol{T}(t_{\mathcal{I},i} + t_{i,x})$ is:

$$\frac{\partial r_{\mathcal{I},i,j}}{\partial \Delta \boldsymbol{T}_C} = \left.\frac{\partial \mathcal{I}_i(\pi(\boldsymbol{T}_{\text{cam}}\, \boldsymbol{T}^{-1}\, \boldsymbol{p}'_j))}{\partial \boldsymbol{T}}\right|_{\boldsymbol{T}=\boldsymbol{T}(t_{\mathcal{I},i}+t_{i,x})} \left.\frac{\partial \boldsymbol{T}(t_{\mathcal{I},i}+t_{i,x})}{\partial \Delta \boldsymbol{T}_C}\right|_{\Delta \boldsymbol{T}_C=\mathbf{0}} \tag{14}$$

where the first part is a 1×12 matrix well known from dense photometric alignment estimating a single rigid body motion. The second part is a 12×24 matrix and specific to the spline trajectory representation. For the geometric error with a rectified reference image we obtain a similar expression:

$$\frac{\partial r_{\mathcal{Z},i,j}}{\partial \Delta \boldsymbol{T}_C} = \left.\frac{\partial \mathcal{Z}_{\text{ref}}(\pi(\boldsymbol{T}_{\text{ref}}^{-1}\, \boldsymbol{T}\, \boldsymbol{p}_j)) - [\boldsymbol{T}_{\text{ref}}^{-1}\, \boldsymbol{T}\, \boldsymbol{p}_j]_Z}{\partial \boldsymbol{T}}\right|_{\boldsymbol{T}=\boldsymbol{T}(t_{\mathcal{Z},i}+t_{j,x})} \left.\frac{\partial \boldsymbol{T}(t_{\mathcal{Z},i}+t_{j,x})}{\partial \Delta \boldsymbol{T}_C}\right|_{\Delta \boldsymbol{T}_C=\mathbf{0}} \tag{15}$$

with $\boldsymbol{T}_{\text{ref}} = \boldsymbol{T}(t_{\mathcal{Z},\text{ref}})$, $\boldsymbol{p}_j = \pi^{-1}(\boldsymbol{x}_j, \mathcal{Z}_i(\boldsymbol{x}_j))$, and $t_{j,x} = {}^{t_r}\!/_{h}\, [\boldsymbol{x}_j]_y$. The key insight is that the second part of the Jacobian in (15) is constant for all pixels in the same row. Therefore, we just have to form the outer product of 1×12 Jacobians and sum 12×12 matrices for all pixels in the same row instead of 1×24 Jacobians and 24×24 matrices to build the normal equations. The same holds for (14) if we evaluate the Jacobian only at times corresponding to integer row indices. Nevertheless, for the photometric error points from the same row in the reference image will be scattered to different rows in the i-th image. Therefore, it is useful to sort the residual terms by row before computing the Jacobians. This ensures that the Jacobian of the pose w.r.t. the control points has to be computed only once for each row. We also derived an analytic expression for the Jacobian of a pose w.r.t. its control points, which we provide in the supplementary material due to space limitations.

4. Evaluation

We evaluate our trajectory estimation approach on simulated and real world datasets. For the experiments on synthetic data we adapt the ICL-NUIM dataset proposed by Handa *et al*. [4]. We perform the experiments with real data on a set of sequences, which we recorded along with the groundtruth trajectory from a motion capture system.

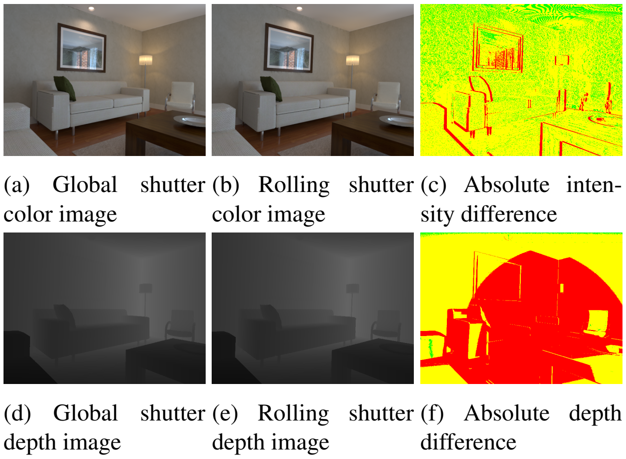
We run all experiments on a PC with Intel Core i7-2600 CPU and 8GB RAM. The alignment with geometric error requires on average 210ms/frame and 775ms/frame with photometric and geometric error. Our C++ implementation computes the error terms and normal equations for the not yet converged images in parallel. We expect to achieve further speedup using a GPU implementation.

4.1. Synthetic Rolling Shutter Dataset

The ICL-NUIM dataset is a synthetic, ray-casted RGB-D dataset. It comprises eight datasets for two scenes, a living room and an office, with four different trajectories each. Besides the groundtruth trajectories the authors also provide a model of the living room scene and tools to evaluate the reconstruction accuracy. However, Handa *et al*. created the datasets with the global shutter assumption. Therefore, we extend this dataset by creating four sequences with the rolling shutter model for the living room scene. To do so we raycast every row of each image from a separate camera pose. We obtain the pose of every row by fitting a cumulative B-spline trajectory to the groundtruth trajectories and evaluating it at the appropriate time. We also generate new groundtruth trajectories from the fitted spline, which differ slightly from the original ones. Figure 3 shows an example RGB-D frame from the lr kt2 sequence rendered with global shutter and rolling shutter model, and the absolute difference of the intensity and depth values. Note how rolling shutter affects the error of most pixels in the depth images, but only pixels close to edges in the color images. We did not simulate the motion blur effect, because it would have increased the raycasting time tremendously.

4.2. Real World Dataset

We ran a second set of experiments on six datasets captured with a PrimeSense Carmine sensor inside a mo-


(a) Global shutter color image (b) Rolling shutter color image (c) Absolute intensity difference

(d) Global shutter depth image (e) Rolling shutter depth image (f) Absolute depth difference

Figure 3: Example of simulated RGB-D images and comparison between global shutter and rolling shutter models. To visualize the influence of the rolling shutter model we show the intensity difference c) between a) and b), and the depth difference f) between images d) and e). Green corresponds to zero and red to a difference of 10% for the intensity and 0.01 m for depth. The color images differ mainly on the small number of edge pixels. In contrast, there is a large difference for many pixels in the depth images.

tion capture volume. We could not use any of the existing RGB-D benchmark datasets with groundtruth trajectory (*e.g.* TUM RGB-D benchmark [16]), because they only provide pre-registered depth images, which destroys the correspondence between rows in the depth image and their capture time. The six sequences comprise three different motion patterns: translation along horizontal camera axis (robot t1/t2), rotation around camera z-axis (robot r1/r2), and a more general scanning motion (table1/2). Table 1 lists length, average translational and rotational velocity for each of these datasets. The translational velocities are among the fastest in comparison to the TUM RGB-D benchmark datasets and the rotational velocities are larger than any present in the TUM RGB-D benchmark. We plan to release all of these new datasets to the public with this publication.

4.3. Results

In total we have ten datasets to evaluate our approach. We compare a baseline method (a variant of [10]) and four variants of our approach. The baseline method performs frame to keyframe tracking estimating a single pose per frame with a geometric error term and global shutter assumption. Furthermore, we fuse all frames into the current keyframe and choose new keyframes based on the same overlap criterion as in our approach. We generate the four variants of our approach by using either only the geometric error term (G) or in addition the photometric error term (P+G), and by switching between global shutter (GS) and rolling shutter (RS) models. To evaluate the different algorithms we use the absolute trajectory error (ATE) and the relative pose error per second (RPE/s) as proposed by Sturm *et al.* [16]. Table 2 shows the root mean squared error (RMSE) of the translational and rotational RPE/s for all five algorithms and the ten sequences. For the synthetic and real world datasets we separately show the average drift and the improvement relative to the baseline algorithm. The results of the best performing algorithm for each sequence are marked with a bold font. Similarly, Table 3 shows the RMSE of the ATE.

Table 1: Statistics of real world datasets we recorded with groundtruth trajectories from a motion capture system. We performed three different motions: translation along camera x-axis (robot t1/t2), rotation around camera z-axis (robot r1/r2) and general motion (table1/2).

Dataset	Length [m]	avg. transl. velocity [m/s]	avg. rot. velocity [deg/s]
robot t1	7.678	0.356	15.807
robot t2	7.671	0.359	13.782
robot r1	2.592	0.123	62.172
robot r2	3.203	0.153	66.847
table1	11.494	0.423	23.223
table2	5.273	0.153	38.582

In general, our algorithms, which estimate a continuous-time trajectory, outperform the baseline algorithm in terms of the RPE and ATE metrics. The large difference on table1/table2, kt0, kt1 and kt3 is mainly due to temporary failures of the baseline algorithm, because of too fast motion or few frames only observing a wall. The spline representation helps in these cases to stabilize the optimization. There is only a small difference between the geometric and combined photometric and geometric error terms on the real world datasets. We attribute this to the presence of enough geometric structure and that we neglect the motion blur effects in the photometric error term. In contrast, on the synthetic datasets the inclusion of the photometric error term improves the performance, because of the missing motion blur and little structure in parts of the sequences. Furthermore, modeling rolling shutter improves the translational RPE and the ATE about 10%. More interestingly, the improvement w.r.t. rotational drift is around 30%. Considering that rotational drift causes most of the error in large scale mapping we find this a remarkable improvement.

To summarize, we demonstrate the superior performance of our continuous-time tracking and mapping algorithm on a number of synthetic and real world datasets in comparison to a standard baseline algorithm. The continuous-time trajectory representation improves tracking performance during fast motions. The combination of photometric and geometric error terms helps to stabilize tracking in structureless regions. Modeling of the rolling shutter effects increases the precision even further.

Table 2: RMSE values of translational and rotational drift per second (RPE/s) for five different trajectory estimation methods on six real world and four synthetic datasets. The four spline-based methods use different combinations of geometric (G) and photometric (P) error terms, and global shutter (GS) and rolling shutter (RS) models. The results for real world and synthetic datasets are separately summarized by average RPE/s values and relative improvement w.r.t. baseline method. All algorithms estimating a continuous-time trajectory clearly outperform the baseline method.

Dataset	Baseline		Spline+GS+G		Spline+RS+G		Spline+GS+P+G		Spline+RS+P+G	
	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]	[m/s]	[deg/s]
robot t1	0.0149	0.7761	0.0168	0.8098	0.0116	0.5389	0.0133	0.9368	**0.0085**	**0.4857**
robot t2	0.0102	0.6679	0.0104	0.6649	0.0100	0.4962	0.0108	0.7730	**0.0089**	**0.4961**
robot r1	0.0180	2.1351	0.0177	2.1407	0.0173	**1.1222**	0.0160	2.1507	**0.0156**	1.1624
robot r2	0.0112	2.3313	0.0108	2.2961	0.0110	**1.1619**	0.0114	2.0803	**0.0105**	1.1911
table1	0.0827	2.2656	0.0362	1.4801	**0.0158**	0.9305	0.0371	1.4749	0.0160	**0.9297**
table2	0.1160	4.2465	0.0117	1.6441	0.0132	0.5973	**0.0103**	1.5749	0.0126	**0.5464**
average	0.0422	2.0704	0.0173	1.5059	0.0132	0.8078	0.0165	1.4984	**0.0120**	**0.8019**
	0%	0%	59.0%	27.3%	68.8%	61.0%	60.9%	27.6%	**71.5%**	**61.3%**
lr kt0	0.0162	2.9869	0.0804	0.2673	0.0268	**0.1059**	0.0075	0.2394	**0.0056**	0.1170
lr kt1	0.1569	30.7227	0.0047	0.1700	0.0029	0.0625	0.0064	0.1709	**0.0024**	**0.0590**
lr kt2	0.0054	0.2744	0.0068	0.2973	0.0044	0.0723	0.0068	0.2128	**0.0031**	**0.0655**
lr kt3	0.7420	35.6279	0.0075	0.9161	**0.0068**	0.9427	0.0164	**0.1900**	0.0100	0.2482
average	0.2301	17.4030	0.0249	0.4127	0.0102	0.2958	0.0093	0.2033	**0.0053**	**0.1224**
	0%	0%	89.2%	97.6%	95.6%	98.3%	96.0%	98.8%	**97.7%**	**99.3%**

Table 3: RMSE values of absolute trajectory error (ATE) *c.f.* caption of Table 2 for structure and abbreviations. Modeling rolling shutter improves the ATE on the real world sequences by 10%. On the synthetic sequences the spline representation and inclusion of the photometric term stabilize the trajectory estimate.

Dataset	Baseline [m]	Spline+GS+G [m]	Spline+RS+G [m]	Spline+GS+P+G [m]	Spline+RS+P+G [m]
robot t1	0.0129	0.0169	0.0129	0.0133	**0.0092**
robot t2	0.0116	0.0161	**0.0082**	0.0287	0.0145
robot r1	0.0170	0.0148	0.0145	**0.0135**	0.0137
robot r2	0.0099	0.0105	0.0108	0.0092	**0.0083**
table1	0.2521	0.0842	0.0379	0.0909	**0.0110**
table2	0.4675	0.0127	**0.0111**	0.0144	0.0112
average	0.1285 (0%)	0.0259 (79.9%)	0.0159 (87.6%)	0.0283 (78.0%)	**0.0113 (91.2%)**
lr kt0	0.1129	0.3980	0.1088	0.0259	**0.0186**
lr kt1	0.2651	0.0111	0.0108	0.0092	**0.0054**
lr kt2	0.0209	0.0173	0.0109	0.0082	**0.0079**
lr kt3	1.2669	0.0607	0.0676	0.0498	**0.0210**
average	0.4165 (0%)	0.1218 (70.8%)	0.0495 (88.1%)	0.0233 (94.4%)	**0.0132 (96.8%)**

5. Conclusion

In this paper we introduced a dense tracking method to estimate a continuous-time trajectory from a sequence of unsynchronized and unregistered RGB-D images. The continuous-time representation helps to better constrain individual image poses and to model the rolling shutter of color and depth cameras. Furthermore, this representation has several benefits, which are now available to dense, direct tracking methods. Additionally, we show how to use the continuous-time trajectory to easily quantify the amount of motion blur and suppress its influence on the fused 3D model. Quantitative evaluation on both synthetic and real world datasets shows that the proposed dense alignment with continuous-time trajectory and rolling shutter model lead to drastic improvements in the trajectory error.

Acknowledgments

We thank the Chair of Information-oriented Control (ITR) at TUM for access to their motion capture system. This work has been partially funded through ERC grant Convex Vision (#240168) and ERC Proof of Concept grant CopyMe3D (#632200).

References

[1] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2392–2399, June 2010. 2

[2] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3d mapping with an RGB-D camera. *IEEE Transactions on Robotics (T-RO)*, 30(1):177–187, 2013. 1, 2

[3] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2088–2095. IEEE, 2012. 2

[4] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1524–1531, May 2014. 6

[5] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434–1441, June 2012. 2

[6] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106, Nov 2013. 1, 2

[7] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3748–3754, May 2013. 4

[8] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86, Oct 2009. 2

[9] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. *Proceedings of the British machine vision conference*, pages 93–1, 2013. 2, 3, 6

[10] M. Meilland and A. Comport. On unifying key-frame and voxel-based dense visual slam at large scales. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3677–3683, Nov 2013. 1, 2, 5, 7

[11] M. Meilland, T. Drummond, and A. Comport. A unified rolling shutter and motion blur model for 3d visual registration. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2016–2023, Dec 2013. 2, 4

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136, Oct 2011. 5

[13] H. Ovrén, P.-E. Forssén, and D. Törnqvist. Improving rgb-d scene reconstruction using rolling shutter rectification. In Y. Sun, A. Behal, and C.-K. R. Chung, editors, *New Development in Robot Vision*, volume 23 of *Cognitive Systems Monographs*, pages 55–71. Springer Berlin Heidelberg, 2015. 2

[14] E. Ringaby and P.-E. Forssén. Scan rectification for structured light range sensors with rolling shutters. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1575–1582, Nov 2011. 2

[15] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, 96(3):335–352, 2012. 2

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580, Oct 2012. 7