Direct Sparse Odometry with Rolling Shutter

David Schubert, Nikolaus Demmel, Vladyslav Usenko,
Jörg Stückler, and Daniel Cremers

Technical University of Munich, Garching b. München, Germany
{schubdav, demmeln, usenko, stueckle, cremers}@in.tum.de

Abstract. Neglecting the effects of rolling-shutter cameras for visual odometry (VO) severely degrades accuracy and robustness. In this paper, we propose a novel direct monocular VO method that incorporates a rolling-shutter model. Our approach extends direct sparse odometry which performs direct bundle adjustment of a set of recent keyframe poses and the depths of a sparse set of image points. We estimate the velocity at each keyframe and impose a constant-velocity prior for the optimization. In this way, we obtain a near real-time, accurate direct VO method. Our approach achieves improved results on challenging rolling-shutter sequences over state-of-the-art global-shutter VO.

Keywords: Direct Monocular Visual Odometry · Rolling Shutter

1 Introduction

Visual odometry for global-shutter cameras has been extensively studied in the last decades (e.g. [6, 18]). Global-shutter cameras capture all pixels in the image within the same time of exposure. Most consumer grade devices such as smartphones or tablets, however, include rolling-shutter cameras which read out image rows sequentially and hence start to expose the rows at sequentially increasing capture times. This leads to image distortions when the camera is moving. Hence,

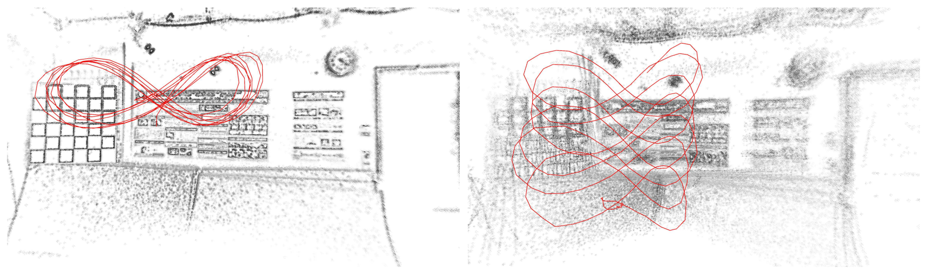


Fig. 1. Qualitative result of our method (DSORS, left) versus DSO (right) on the sequence *infinity-1*. The keyframe trajectory in red shows significant drift without a rolling shutter model. DSORS also produces much cleaner edges in the sparse 3D reconstruction than DSO.

it is necessary to consider the camera pose as a function of the capture time, i.e. row index. Simply neglecting this effect and assuming global shutter can lead to significant drift in the trajectory and 3D reconstruction estimate – see Fig. 1.

For visual odometry two major paradigms exist: direct methods (e.g. [6]) align image intensities based on the photoconsistency assumption, while indirect methods (e.g. [18]) align pixel coordinates of matched keypoints, i.e. they minimize keypoint reprojection error. Direct methods are particularly advantageous in weakly textured and repetitive regions. However, as demonstrated in [6], geometric noise as caused by neglecting rolling shutter significantly downgrades the performance of direct methods. Thus, for direct methods, it is important to model rolling shutter.

While in indirect methods, the row and, hence, the capture time of corresponding keypoints can be assumed known through the extraction process, in direct methods, one has to impose the *rolling shutter constraint* [17] in order to retrieve the capture time. In this paper, we propose a novel direct visual odometry method for rolling-shutter cameras. Our approach performs direct bundle adjustment of a set of recent keyframe poses and the depths of a sparse set of image points. We extend direct sparse odometry (DSO, [6]) by estimating the velocity at each keyframe, imposing a constant-velocity prior for the optimization and incorporating rolling-shutter into the projection model.

We evaluate our method on challenging datasets recorded with rolling-shutter cameras and compare our method to a state-of-the-art approach for global shutter cameras, demonstrating the benefits of modeling rolling shutter adequately in our method.

2 Related Work

Indirect methods: The vast set of literature on indirect methods for visual odometry and SLAM considers global-shutter cameras [19, 18]. Some approaches investigate the proper treatment of rolling-shutter effects. Hedborg et al. [9] propose rolling shutter bundle adjustment which assumes constant-velocity motion between camera poses to determine the camera pose for the row of a keypoint through linear interpolation. Insights into degenerate cases of rolling-shutter bundle adjustment are given in [4]. Essentially, 3D reconstructions collapse to a plane when the camera frame directions are parallel in which the shutter is traversing. Ait-Aider et al. [1] recover 3D reconstruction and motion of a rigidly moving object using a snapshot of a rolling-shutter stereo camera. The method assumes linear motion and solves a non-linear system of equations resulting from the keypoint correspondences. They argue that the use of rolling-shutter cameras can be beneficial over global-shutter cameras for kinetics estimation of fast moving objects. Another line of work addresses the problem of recovering pose and motion in the case of known structure from a single rolling-shutter image [2, 15, 16, 3]. Dai et al. [5] generalize epipolar geometry to the rolling-shutter case and propose linear and non-linear algorithms to solve for the rolling-shutter essential matrix that relates two rolling-shutter images by the relative pose and rotational and translational velocities of their cameras. Some approaches fuse vision with inertial measurements which allows for going beyond constant-velocity assumptions for inter-frame camera motion. Lovegrove et al. [14] approximate the continuous camera motion using B-splines. The approach of [13] considers rolling-shutter for extended Kalman filter based visual-inertial odometry. Saurer et al. [21] develop a pipeline for sparse-to-dense 3D reconstruction that incorporates GPS/INS readings in a rolling-shutter-aware bundle adjustment, prior to performing rolling-shutter stereo to create a dense reconstruction.

Direct methods: Direct methods have been recently shown to achieve state of-the-art performance for visual odometry and SLAM with global-shutter cameras [7, 6]. Since in direct methods, image correspondences are found through projective warping from one image to another, they are more susceptible to errors introduced by neglecting rolling-shutter effects than indirect methods. Estimating the time of projection in the other image requires the solution of the rolling-shutter constraint [17]. The constraint implicitly relates the reprojected image row of a pixel with its capture time, i.e. image row, in the other image. Meingast el al. [17] develop approximations to the constraint for several special cases of camera motion. Saurer et al. [20] present dense multi-view stereo reconstruction for rolling-shutter cameras including image distortion by wide-angle lenses, while they assume the camera motion known. Kerl et al. [11] use B-splines to represent the trajectory estimate continuously for visual odometry with rolling-shutter RGB-D. While we propose a direct method for monocular cameras, similar to our approach they also incorporate the rolling-shutter constraint as a hard constraint by solving for the observation time in the target frame. Most closely related to our method is the approach by Kim et al. [12]. It extends LSD-SLAM [7] to rolling-shutter cameras based on a spline trajectory representation. In contrast to our method they require depth initialization and do not incorporate lens distortion in their model. Their method explicitly incorporates residuals for the rolling-shutter constraint into the non-linear least squares problem by introducing variables for the capture time of each pixel while we directly solve for the capture time. Their implementation runs at approx. 120 s per frame, while our method is faster by orders of magnitude. While their approach separates tracking and mapping, we incorporate a rolling-shutter projection model into a windowed sparse direct bundle adjustment framework (DSO [6]), and represent trajectories using camera poses and velocities at the keyframes. This way, we achieve accurate but run-time efficient visual odometry for rolling-shutter cameras.

3 Direct Sparse Odometry With Rolling Shutter Cameras

We formulate visual odometry as direct bundle adjustment in a recent window of keyframes: we concurrently estimate the camera poses of the keyframes and reconstruct a sparse set of points from direct image alignment residuals (DSO [6]). The method comprises a visual odometry front-end and an optimization backend. The front-end has been left unmodified compared to DSO. It provides initial parameters for the optimization back-end and is responsible for frame and point management. New frames are tracked with respect to the latest keyframe using direct image alignment assuming a constant camera pose across the image.

The need for a new keyframe is determined based on a heuristic that takes optical flow, camera translation and exposure changes into account. The front-end also decides for the marginalization of keyframes and points which drop out of the optimization window: Keyframes are dropped if they do not have at least 5 % of their points visible in the latest keyframe. Also, if the number of keyframes exceeds a threshold ($N = 7$), a keyframe is selected for marginalization using a heuristic that keeps keyframes well-distributed in space, with more keyframes close to the latest one. When a keyframe is marginalized, first all points hosted in the keyframe are marginalized, then the keyframe variables are marginalized. Observations of other points visible in the marginalized keyframe are dropped to maintain the sparsity structure of the Hessian.

The point management aims at keeping a fixed number of active points in the optimization window. The method is sparse, i.e. it does not use all available information. Using more than 2000 image points hardly improves the tracking results of the global shutter method [6], and we also found for our method that the results do not justify the increase in runtime when using more points (see supplementary material). Candidate points are chosen in every new keyframe based on the image gradient, tracked in subsequent frames using epipolar line search and added to the set of active points for the bundle adjustment after old points are marginalized.

Our contribution lies in the optimization backend, where we introduce a model that explicitly accounts for rolling shutter. The energy contains residuals across the window of keyframes and is optimized with respect to all variables jointly using Gauss-Newton optimization.

3.1 Model

In the following, we detail our formulation of direct image alignment in the optimization backend of DSO for rolling shutter cameras. As the rows of an image are not taken at the same time, it is now necessary to find the camera pose as a function of time t. Camera poses $\mathbf{T}_i(t)$ are elements of the special Euclidean group SE(3). We choose a constant velocity model, such that the parametrization of the camera motion while frame i is being taken is given by

$$\mathbf{T}_i(t) = \exp(\hat{\mathbf{v}}_i t)\mathbf{T}_{0,i}, \tag{1}$$

where $\mathbf{v}_i \in \mathbb{R}^6$ is a velocity vector that includes both translational and rotational components and $\hat{\mathbf{v}}_i \in \mathfrak{se}(3) \subset \mathbb{R}^{4\times 4}$ the corresponding Lie Algebra element. We assume that the time t when a pixel is read out is linearly related to the vertical y-coordinate of the pixel, i.e.

$$t(x, y) = y - y_0 \tag{2}$$

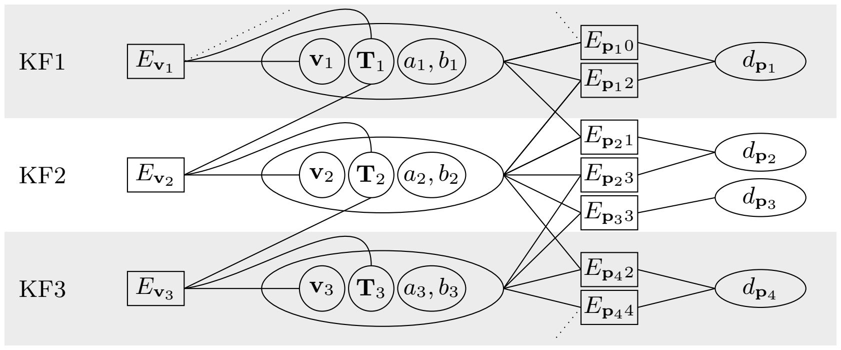


Fig. 2. Factor graph of the objective energy. The ith point observed in the jth keyframe contributes a photometric energy $E_{\mathbf{p}_i j}$, which depends on the variables of the host and the target keyframe plus the point's inverse depth in the host frame. In addition, energy terms $E_{\mathbf{v}_i}$ representing the velocity prior create correlations between velocities and poses. Not shown are the camera intrinsics, which influence every photometric residual.

which usually is well satisfied for rolling shutter cameras (though technically the shutter might also be along the horizontal x-coordinate, in which case the image can simply be rotated). The optimization backend estimates the reference poses $\mathbf{T}_{0,i} \in \mathrm{SE}(3)$ of the keyframes and the velocities $\mathbf{v}_i \in \mathbb{R}^6$.

In our model, the row y_0 has been taken at the camera pose $\mathbf{T}_{0,i}$. We set y_0 in the middle of the vertical range of the image. Of course, y_0 can be chosen arbitrarily in the image and we could just choose $y_0 = 0$, but with our choice we assume that the optimal $\mathbf{T}_{0,i}$ is in best agreement with its initialization from tracking towards the last keyframe which assumes global shutter.

If a point $\mathbf{p}$ that is hosted in image I_i is observed in image I_j, it contributes to the energy as

$$E_{\mathbf{p}j} = \sum_{k \in \mathcal{N}_\mathbf{p}} w_{\mathbf{p}_k} \|r_k\|_\gamma \,, \tag{3}$$

with photometric residuals

$$r_k = (I_j[\mathbf{p}'_k] - b_j) - e^{a_j - a_i}(I_i[\mathbf{p}_k] - b_i). \tag{4}$$

The indices in $\mathcal{N}_\mathbf{p}$ denote pixels in the neighborhood of point $\mathbf{p}$. As in [6], we use an 8-pixel neighborhood and a gradient-based weighting $w_{\mathbf{p}_k}$ that down-weights pixels with strong gradient. For robustness, the Huber norm $\|\cdot\|_\gamma$ is used. The parameters a_i, b_i describe an affine brightness transfer function $\exp(-a_i)(I_i - b_i)$ which is used to account for possibly changing exposure times or illumination.

For photometrically perfect images (synthetic data), they are not required. For the real data in our experiments with constant exposure we can include a prior which keeps them at zero.

The pixel position $\mathbf{p}'_k$ in the host frame is calculated from $\mathbf{p}_k$ with a composition of inverse projection, rigid body motion and projection,

$$\mathbf{p}'_k = \Pi_{\mathbf{c}}(\mathbf{R}\Pi_{\mathbf{c}}^{-1}(\mathbf{p}_k, d_{\mathbf{p}}) + \mathbf{t})\,, \tag{5}$$

where $d_{\mathbf{p}}$ is the depth of point $\mathbf{p}$ in its host frame and the projection $\Pi_{\mathbf{c}}$ depends on the four internal camera parameters f_x, f_y, c_x, c_y which are the components of the vector $\mathbf{c}$.

The rotation $\mathbf{R}$ and the translation $\mathbf{t}$ are calculated as

$$\begin{aligned}\begin{bmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0} & 1\end{bmatrix} &= \mathbf{T}_j(t^*)\mathbf{T}_i^{-1}(t(\mathbf{p})) &(6)\\ &= \exp(\hat{\mathbf{v}}_j t^*)\mathbf{T}_{0,j}\mathbf{T}_{0,i}^{-1}\exp(-\hat{\mathbf{v}}_i t(\mathbf{p})) &(7)\end{aligned}$$

Note that we know the time when the point has been observed in the host frame, as we know its pixel coordinates. It is not straightforward, however, to obtain the time t^* of observation in the target frame. It depends on the y-coordinate $\mathbf{p}'_y$ of the projected point (eq. (2)), but the y-coordinate of the projected point also depends on time through the time-dependent pose $\mathbf{T}_j(t)$. This interdependency is formulated as the *rolling shutter constraint* [17], where we choose t^* such that it satisfies

$$t^* = t(\mathbf{p}'(t^*)). \tag{8}$$

In short: pose time equals row time. Here, t is the function defined in eq. (2). Apart from some specific cases, there is no closed-form solution for t^* [17]. In practice, it turns out to be sufficient to iterate the update $t^* \leftarrow t(\mathbf{p}'(t^*))$ for a few times to obtain a solution.

We remove lens distortion from the images through undistortion in a preprocessing step. Consequently, the mapping between pixel coordinates and time in eq. (2) changes: Instead of the row in the undistorted image, we need to consider the row of the corresponding point in the original distorted image,

$$t(x, y) = f_{\mathrm{d}}(x, y)_y - \tilde{y}_0. \tag{9}$$

A point in the undistorted image with coordinates x and y is mapped by the distortion function f_{d} into the original image, where the y-coordinate determines the time. The offset $\tilde{y}_0 = f_{\mathrm{d}}(x_0, y_0)_y$ is chosen to be the original y-coordinate of the midpoint (x_0, y_0) of the undistorted image. We calculate $f_{\mathrm{d}}(x, y)_y$ for all pixels as a preprocessing step and then interpolate. This is computationally less expensive than using the distortion function each time. Also, it facilitates the incorporation of new distortion models, as we later need the derivatives of the function.

The first component of the total energy is the summation over all photometric residuals,

$$E_{\text{ph}} = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \sum_{j \in \text{obs}(\mathbf{p})} E_{\mathbf{p}j}, \tag{10}$$

where $\mathcal{F}$ is the set of frames, $\mathcal{P}_i$ the set of points in frame i and $\text{obs}(\mathbf{p})$ the set of frames in which $\mathbf{p}$ is visible.

Optimizing E_{ph} alone, however, is not reliable. It has been shown that rolling shutter images are prone to ambiguities [4] and we also found out that softly constraining the velocities leads to a more stable system. Thus, we add an additional energy term E_{vel} to the total energy,

$$E = E_{\text{ph}} + \lambda E_{\text{vel}}. \tag{11}$$

The term

$$E_{\text{vel}} = \sum_{i \in \mathcal{F}} \|\mathbf{v}_i - \mathbf{v}_{i,\text{prior}}\|^2 \tag{12}$$

is a prior on the velocities, with

$$\mathbf{v}_{i,\text{prior}} = \log(\mathbf{T}_i^{-1}\mathbf{T}_{i-1})^{\vee} \frac{\Delta t_{\text{r}}}{t_i - t_{i-1}}, \tag{13}$$

where $\log(\cdot)^{\vee}$ is the composition of the matrix logarithm and the inverse hat transform which extracts twist coordinates from Lie algebra elements, so that $\mathbf{v}_{i,\text{prior}} \in \mathbb{R}^6$. Here, we need actual times: $t_i - t_{i-1}$ is the time difference between the capture of the ith and the $(i-1)$th keyframe and Δt_{r} is the time difference between two consecutive pixel rows due to the rolling shutter. The prior intuitively favors that the velocity between the latest existing keyframe and the new keyframe is similar to the velocity while taking the new keyframe. As initially many keyframes are taken, such a smoothness assumption is reasonable. The resulting constraints are visualized in Fig. 2, where a factor graph of the different energy terms is shown. After marginalizing a keyframe's variables, the prior still acts through the marginalization term.

3.2 Optimization

We minimize the cost in equation (11) using Gauss-Newton optimization. The linearized system is

$$\mathbf{H}\boldsymbol{\delta} = \mathbf{b}, \tag{14}$$

with

$$\mathbf{H} = \mathbf{J}^T\mathbf{W}\mathbf{J}, \tag{15}$$

$$\mathbf{b} = -\mathbf{J}^T\mathbf{W}\mathbf{r}. \tag{16}$$

The matrix $\mathbf{J}$ is the Jacobian of the residual vector $\mathbf{r}$ for the variables to optimize. The diagonal weight matrix $\mathbf{W}$ contains the weights of the residuals. The Hessian $\mathbf{H}$ has a large diagonal block of correlations between the depth variables, which makes efficient inversion using the Schur complement possible. Compared to [6], which has 8 variables per frame (6 for the camera pose plus 2 for the affine brightness transfer function), we now additionally have 6 velocity components, which gives 14 variables per frame. In addition, there are 4 internal camera parameters $\mathbf{c}$ shared among all keyframes. Each point adds one inverse depth variable.

One single row in the Jacobian, belonging to one single pixel $\mathbf{p}_k$ in the neighborhood of $\mathbf{p}$, is given by

$$\mathbf{J}_k = \frac{\partial r_k(\boldsymbol{\delta} \boxplus \boldsymbol{\zeta})}{\partial \boldsymbol{\delta}} \,. \tag{17}$$

The state vector $\boldsymbol{\zeta}$ contains all variables in the system, i.e. keyframe poses, velocities, affine brightness parameters, inverse depths and intrinsic camera parameters. The symbol $\boxplus$ denotes standard addition for all variables except for poses where it means $\boldsymbol{\delta}_\mathrm{p} \boxplus \mathbf{T} = \exp(\hat{\boldsymbol{\delta}}_\mathrm{p})\mathbf{T}$, with $\boldsymbol{\delta}_\mathrm{p} \in \mathbb{R}^6$ and $\mathbf{T} \in \mathrm{SE}(3)$.

The Jacobian can be decomposed as

$$\mathbf{J}_k = \left[\mathbf{J}_I \mathbf{J}_\mathrm{geo}, \mathbf{J}_\mathrm{photo}\right] \,. \tag{18}$$

The image gradient

$$\mathbf{J}_I = \frac{\partial I_j}{\partial \mathbf{p}'_k} \tag{19}$$

as well as the photometric Jacobian

$$\mathbf{J}_\mathrm{photo} = \frac{\partial r_k(\boldsymbol{\delta} \boxplus \boldsymbol{\zeta})}{\partial \boldsymbol{\delta}_\mathrm{photo}} \tag{20}$$

are evaluated at pixel $\mathbf{p}'_k$, where $\boldsymbol{\delta}_\mathrm{photo}$ corresponds to the photometric variables a_i, b_i, a_j, b_j. The geometric Jacobian $\mathbf{J}_\mathrm{geo}$ contains derivatives of the pixel location (not the intensity, as it is multiplied with $\mathbf{J}_I$) with respect to the geometric variables $\mathbf{T}_i, \mathbf{T}_j, \mathbf{v}_i, \mathbf{v}_j, d, \mathbf{c}$. It is approximated as the derivative of the central pixel $\mathbf{p}'$ for all pixels $\mathbf{p}'_k$ in the neighborhood. For $\mathbf{J}_\mathrm{geo}$, we also have to take into account that the observation time t^* depends on the geometric variables. Thus,

$$\mathbf{J}_\mathrm{geo} = \frac{\partial \mathbf{p}'}{\partial \delta_\mathrm{geo}} + \frac{\partial \mathbf{p}'}{\partial t^*} \frac{\mathrm{d}t^*}{\mathrm{d}\boldsymbol{\delta}_\mathrm{geo}} \,. \tag{21}$$

The derivative of a function $y(x)$ that is defined as the root of a function $R(x, y)$ is given as

$$\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{\partial R / \partial x}{\partial R / \partial y} \,. \tag{22}$$

As t^* is defined by equation (8), we can use this rule to calculate $\frac{\mathrm{d}t^*}{\mathrm{d}\delta_{\mathrm{geo}}}$. In our case,

$$R(\boldsymbol{\zeta}, t^*) = t^* - t(\mathbf{p}'(\boldsymbol{\zeta}, t^*)) \tag{23}$$

$$= t^* - \left(f_{\mathrm{d}}(\mathbf{p}'(\boldsymbol{\zeta}, t^*))_y - \tilde{y}_0\right), \tag{24}$$

so that

$$\frac{\mathrm{d}t^*}{\mathrm{d}\boldsymbol{\delta}_{\mathrm{geo}}} = \frac{\frac{\partial f_{\mathrm{d},y}}{\partial \mathbf{p}'} \frac{\partial \mathbf{p}'}{\partial \boldsymbol{\delta}_{\mathrm{geo}}}}{1 - \frac{\partial f_{\mathrm{d},y}}{\partial \mathbf{p}'} \frac{\partial \mathbf{p}'}{\partial t^*}} . \tag{25}$$

The Jacobians $\mathbf{J}_{\mathrm{geo}}$ and $\mathbf{J}_{\mathrm{photo}}$ are approximated using first-estimates Jacobians [10]. This means that the evaluation point of these Jacobians does not change once a variable is part of the marginalization term, while the image gradient $\mathbf{J}_I$ and the residual r_k are always evaluated at the current state. The authors of [6] argue that $\mathbf{J}_{\mathrm{geo}}$ and $\mathbf{J}_{\mathrm{photo}}$ are smooth so that we can afford evaluating them at a slightly different location for the benefit of keeping a consistent system in the presence of non-linear null-spaces such as absolute pose and scale. As the translational component of the velocity is also affected by scale ambiguity, we decide to include velocities in the first-estimate Jacobian approximation.

When variables are marginalized, we follow the procedure of [6]: We start from a quadratic approximation of the energy that contains residuals which depend on variables to be marginalized. Variables are marginalized using the Schur complement, which results in an energy that only depends on variables which are still active in the optimization window. This energy acts like a prior and can be added to the energy of active residuals. We also include our velocity priors in the marginalization term if they depend on a variable that is marginalized, so that velocities of active keyframes are still constrained by marginalized keyframes which are temporally and spatially close. We refer the reader to [6] for further details on the marginalization process.

4 Experimental Evaluation On Real And Synthetic Data

4.1 Datasets

Along with the rolling shutter RGB-D SLAM method in [11], **synthetic sequences** were published. They are re-renderings of the ICL-NUIM dataset [8], containing 4 different trajectories in a living room, named *kt1*, *kt2*, *kt3*, and *kt4*. As the data is photometrically perfect, we do not estimate affine brightness parameters on these sequences.

We also show results for the sequence *freiburg1_desk*, an **office sequence** from the TUM RGB-D benchmark [22]. It has already been used by [12], but a quantitative comparison is not possible, as they only show a trajectory plot. The row time difference Δt_{r} is not available. We were successful with our first guess of $\Delta t_{\mathrm{r}} = 0.06\,\mathrm{ms}$, a value that leaves only a small time gap between the last row and the first row of the next image.

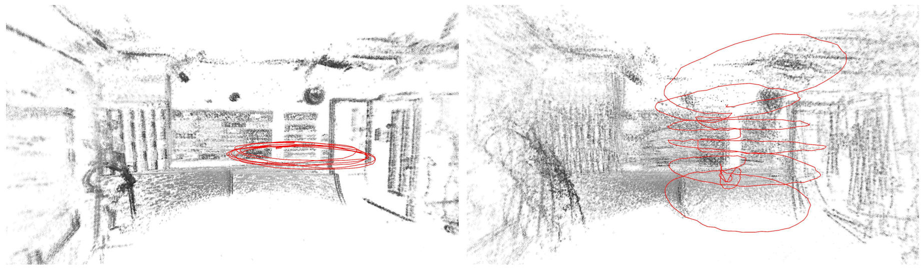


Fig. 3. Qualitative result of DSORS (left) versus DSO (right) on the sequence *alt-circle-1*. Even after many circles, the sparse 3D reconstruction of DSORS shows little drift, whereas DSO shows the same edges at clearly distinct locations.

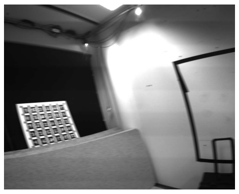


Fig. 4. Our faster sequences show significant distortion and blur due to motion (from *alt-circle-2*).

Due to the lack of rolling shutter datasets for monocular visual odometry, we recorded six **own sequences** with ground truth. Sequences were captured at 1280x1024 resolution with a handheld uEye UI-3241LE-M-GL camera by IDS and a Lensagon BM4018S118 lens by Lensation. The camera was operated at 20 Hz and provides an approximate value of $\varDelta t_{\mathrm{r}} = 0.033\,\mathrm{ms}$. Ground truth was recorded with a Flex 13 motion capture system by OptiTrack. It uses IR-reflective markers and 16 cameras distributed around the room. The ground truth poses are hand-eye and time shift calibrated, meaning that they provide poses of the camera frame to an external world system and timestamps of the ground truth poses are given in the same time system as timestamps for the camera frames. We fixed the exposure, which means our algorithm uses a prior that prefers small affine brightness parameters. We also use lens vignetting correction on our own sequences. The sequences can be divided into three categories:

- **infinity:** The camera motion repeatedly draws an infinity symbol in front of a wall.
- **circle:** The camera moves on a circle that lies in the horizontal plane while the camera is pointing outwards
- **alt-circle:** Same as *circle*, but with alternating directions.

Each of the categories has been captured twice (e.g. *infinity-1* and *infinity-2*), with the second one always being faster than the first one, but none of them is really slow in order to have sufficient rolling shutter effect.

Our camera includes two identical cameras in a stereo setup. For comparison with methods processing global shutter images, we simultaneously recorded sequences using the second camera set to global shutter mode.

Our dataset contains significant blur and rolling-shutter distortion due to fast motion as can be seen in Fig. 4. Due to flickering of the illumination in our recording room, the rolling-shutter images contain alternating stripes of brighter and darker illumination which are also visible in Fig. 4. While this is not consistent with the illumination model in DSO and DSORS, none of the two has an obvious advantage. Note that the global shutter images do not exhibit this effect.

4.2 Evaluation Method

For each sequence, we compare the performance of DSORS versus DSO. DSO originally allows a maximum of 6 iterations for each Gauss-Newton optimization. We found slight improvements for our method if we increase them to 10. To make the comparison fair, we also allow a maximum of 10 iterations for DSO, though still both methods can break early when convergence is reached. The number of active points is set to 2000, and there are maximally 6 old plus one new keyframe in the optimization window, which are the standard settings for DSO. Compared to DSO, we only introduced one model parameter, the weight of the velocity prior λ. The same value is used for all sequences.

We use the *absolute trajectory error* (ATE) to evaluate our results quantitatively. Given ground truth keyframe positions $\hat{\mathbf{p}}_i \in \mathbb{R}^3$ and corresponding tracking results $\mathbf{p}_i \in \mathbb{R}^3$, it is defined as

$$e_{\text{ate}} = \min_{\mathbf{T} \in \text{Sim}(3)} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{T}(\mathbf{p}_i) - \hat{\mathbf{p}}_i\|^2} \,. \tag{26}$$

It is necessary to align with a 7D similarity transform[1] $\mathbf{T} \in \text{Sim}(3)$, since scale is not observable for monocular methods. We run the methods 20 times on each sequence. To randomize the results, different random number seeds are used for the point selection. We show two types of visualization for the quantitative results: the color plots show e_{ate} for each run and for each sequence individually. The cumulative error histograms contain all sequences of each dataset. Here, the function value at position e gives the number of sequences with $e_{\text{ate}} \leq e$.

4.3 Results

On the **synthetic sequences**, DSORS clearly outperforms DSO, as can be seen in Fig. 5. Not only is the overall performance in the cumulative error histogram visibly more accurate, but also on each sequence as can be seen in the color plot. Only on the sequence *kt0* it is not entirely stable, but here DSO also has outliers. The RGB-D method in [11] reports ATEs of (0.0186, 0.0054, 0.0079, 0.0210) (after SE(3) alignment) for the four trajectories, while our median ATEs over 20 runs are (0.0037, 0.0197, 0.0045, 0.0062) (after Sim(3) alignment).

For the **office sequence**, the difference between DSORS and DSO in the cumulative error histogram in Fig. 6 is even more obvious. The performance of DSORS is much more stable and accurate. On the right side of the figure, typical

[1] In equation (26) $\mathbf{T}$ is used as an operator on 3D points $\mathbf{T} : \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{p} \mapsto \mathbf{T}(\mathbf{p})$.

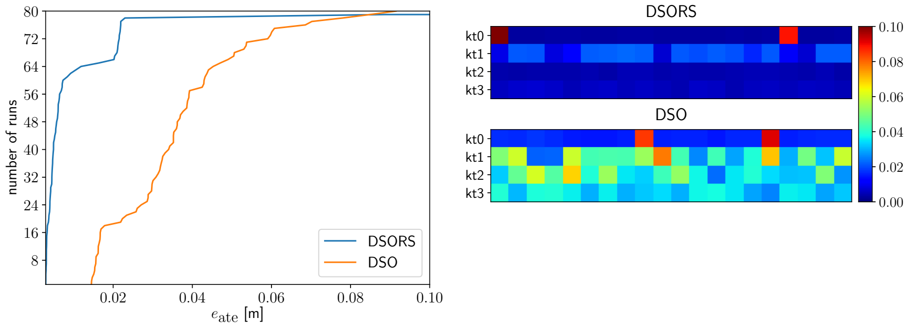


Fig. 5. On the left, the cumulative error histogram for 20 runs on each synthetic sequence clearly shows that DSORS produces more accurate results than DSO. The plot on the left indicates the error for each individual run and shows that DSORS is also superior on each sequence.

tracked trajectories are plotted with ground truth after Sim(3) alignment. The red lines between corresponding points make visible how large the error is for DSO, compared to much smalles errors for DSORS.

The results on our **own sequences** also demonstrate that DSORS is superior to DSO when dealing with rolling shutter data. Qualitatively, this is clearly visible in Figs. 1 and 3. The sparse 3D reconstructions look much cleaner for DSORS, while DSO produces inconsistent edges when revisiting a section of the room. Even more striking is the large systematic drift of the camera trajectories.

In Fig. 7, the quantitative difference also becomes apparent. DSORS outperforms DSO both in terms of accuracy and stability. Only the sequence *infinity-2*

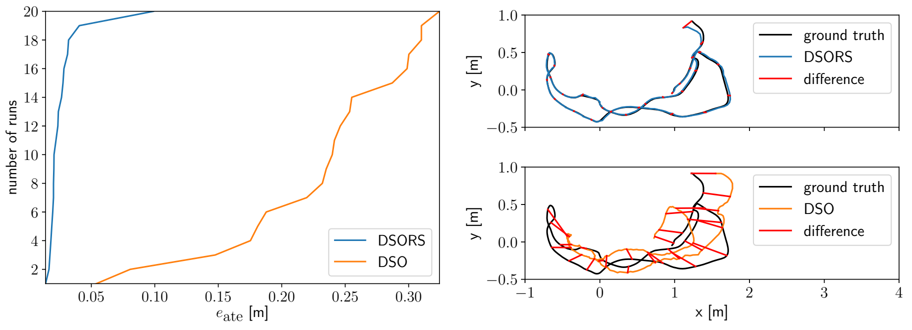
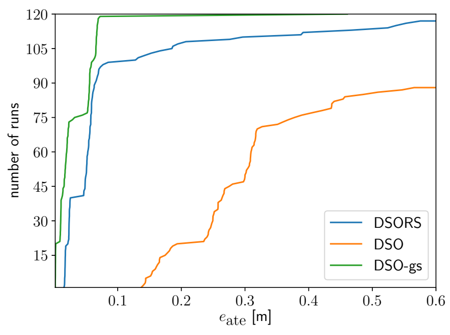


Fig. 6. On the left, the cumulative error histogram over 20 iterations on the *freiburg1_desk* sequence from the TUM-RGBD benchmark shows that DSORS can repeatably track the sequence accurately while DSO has large drift. On the right, the top down view shows the ground truth and estimated trajectories after Sim(3) alignment. For each method we select the iteration with median e_{ate} (DSORS: 0.022 m, DSO: 0.241 m). The red lines indicate corresponding points for every 10th keyframe.



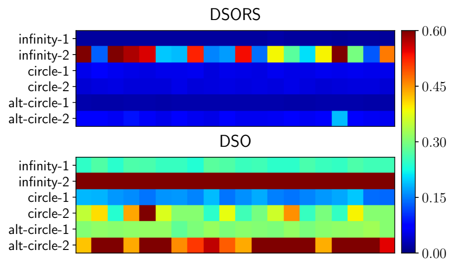


Fig. 7. Cumulative error histogram and color plot for the absolute trajectory error e_{ate} on our new sequences. In the cumulative histogram, the green line also gives a comparison to DSO running on global shutter data, which has been captured in parallel with a stereo setup. For trajectory plots as in Fig. 6, see supplementary material.

remains a challenge, but DSORS in approximately half of the cases produces reasonable results, whereas DSO always fails in our runs.

We also show results of DSO operating on global shutter data. The sequences are very comparable to the rolling shutter sequences, as they use the same hardware, and cameras are triggered at the same time in a stereo setup. Running DSO on global shutter images is still better than running DSORS on rolling shutter images. The difference in stability visible in the cumulative histogram mainly comes from the challenging sequence *infinity-2*. On the remaining sequences, the advantage of using global shutter images is not as dominant.

4.4 Runtime

The computationally most expensive part of our method is the creation of a new keyframe, i.e. calculating derivatives, accumulating the Hessian and solving the linear system. Our derivatives are more involved compared to DSO, and the number of variables is larger. As keyframes are not selected with equal time spacing, but based on motion, it is not possible to specify how many frames

Table 1. Figures about trajectories and runtime. The number of keyframes that were created is denoted n_{KF}, the realtime factor is given by r.

Sequence	Length / Duration	$n_{\mathrm{KF}}^{\mathrm{DSORS}}$	r^{DSORS}	$n_{\mathrm{KF}}^{\mathrm{DSO}}$	r^{DSO}
infinity-1	20.9 m / 33.3 s	249	0.174	240	0.433
infinity-2	36.3 m / 29.9 s	324	0.128	277	0.458
circle-1	44.8 m / 58.9 s	547	0.149	543	0.378
circle-2	30.6 m / 29.5 s	424	0.099	431	0.248
alt-circle-1	24.6 m / 41.9 s	392	0.153	399	0.362
alt-circle-2	31.6 m / 27.1 s	353	0.109	356	0.288

per second we can process in general. With a fast moving camera, more new keyframes are required per time, which affects the runtime. In Table 1, some figures about the trajectories and the performance of DSORS and DSO run on an Intel Core i5-2500 CPU are given. The realtime factor r is calculated as the real duration of the sequence divided by the processing time of the algorithm. By comparing each slower sequence (*...-1*) to its faster variant (*...-2*), one can confirm that the realtime factor depends on the speed of the camera motion. Only r^{DSO} for the sequence *infinity-2* is an exception, but this sequence is very unstable for DSO, thus many outlier points are dropped during the optimization, which speeds up the execution. Also, the number of keyframes is rather related to the total length of the trajectory than to the duration.

The results also prove that DSORS is slower than DSO, by a factor roughly around 2.5 (except for *infinity-2*). It might seem surprising that not even DSO is real-time here, but this is due to the fact that all results were created in a linearized mode, where the coarse tracking waits for the keyframe creation to finish. Given that DSO is generally a real-time capable method, further optimization of our method or using a faster processor might produce real-time results in the future. In fact, by reducing the number of active points to 800 and enforcing real-time execution (with the coarse tracking continuing while a new keyframe is created), it was possible to obtain results for the slower sequences that were close to our non-real-time results, but not yet for the faster sequences.

5 Conclusions

In this paper, we have integrated a rolling shutter model into direct sparse visual odometry. By extending keyframe poses with a velocity estimate and imposing a constant-velocity prior in the optimization, we obtain a near real-time but accurate direct visual odometry method.

Our experiments on sequences from rolling shutter cameras have demonstrated that the model is well-suited and can drastically improve accuracy and stability over methods that neglect rolling shutter. Our method makes accurate direct visual odometry available to rolling shutter cameras which are often present in consumer grade devices such as smartphones or tablets and in the automotive domain.

For direct rolling shutter approaches, real-time capability is a challenge. With our formulation, we are already much closer to real-time processing than the alternative approach in [12]. In future work, we will investigate further speedups of our implementation. The integration with inertial sensing could further increase the accuracy and stability of the visual odometry.

Acknowledgment

This work was partially supported through the grant “For3D” by the Bavarian Research Foundation and through the grant CR 250/9-2 “Mapping on Demand” by the German Research Foundation.

References

1. Ait-Aider, O., Berry, F.: Structure and kinematics triangulation with a rolling shutter stereo rig. In: IEEE International Conference on Computer Vision. pp. 1835–1840 (2009)
2. Ait-Aider, O., Andreff, N., Lavest, J.M., Martinet, P.: Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) Computer Vision – ECCV 2006. pp. 56–68. Springer (2006)
3. Albl, C., Kukelova, Z., Pajdla, T.: R6P - rolling shutter absolute pose problem. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2292–2300 (2015)
4. Albl, C., Sugimoto, A., Pajdla, T.: Degeneracies in rolling shutter SfM. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 36–51. Springer (2016)
5. Dai, Y., Li, H., Kneip, L.: Rolling shutter camera relative pose: Generalized epipolar geometry. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 4132–4140 (2016)
6. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(3), 611–625 (2018)
7. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 834–849. Springer (2014)
8. Handa, A., Whelan, T., McDonald, J., Davison, A.J.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: IEEE International Conference on Robotics and Automation. pp. 1524–1531 (2014)
9. Hedborg, J., Forssén, P.E., Felsberg, M., Ringaby, E.: Rolling shutter bundle adjustment. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1434–1441 (2012)
10. Huang, G.P., Mourikis, A.I., Roumeliotis, S.I.: A first-estimates jacobian EKF for improving SLAM consistency. In: Khatib, O., Kumar, V., Pappas, G.J. (eds.) Experimental Robotics. pp. 373–382. Springer (2009)
11. Kerl, C., Stückler, J., Cremers, D.: Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In: IEEE International Conference on Computer Vision. pp. 2264–2272 (2015)
12. Kim, J.H., Cadena, C., Reid, I.: Direct semi-dense SLAM for rolling shutter cameras. In: IEEE International Conference on Robotics and Automation. pp. 1308–1315 (2016)
13. Li, M., Kim, B.H., Mourikis, A.I.: Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In: IEEE International Conference on Robotics and Automation. pp. 4712–4719 (2013)
14. Lovegrove, S., Patron-Perez, A., Sibley, G.: Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In: Burghardt, T., Damen, D., Mayol-Cuevas, W., Mirmehdi, M. (eds.) Proceedings of the British Machine Vision Conference. pp. 93.1–93.12. BMVA Press (2013)
15. Magerand, L., Bartoli, A.: A generic rolling shutter camera model and its application to dynamic pose estimation. In: International Symposium on 3D Data Processing, Visualization and Transmission (2010)
16. Magerand, L., Bartoli, A., Ait-Aider, O., Pizarro, D.: Global optimization of object pose and motion from a single rolling shutter image with automatic 2D-3D

matching. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) Computer Vision – ECCV 2012. Springer (2012)
17. Meingast, M., Geyer, C., Sastry, S.: Geometric models of rolling-shutter cameras. arXiv preprint cs/0503076 (2005)
18. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017)
19. Nister, D., Naroditsky, O., Bergen, J.: Visual odometry. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. I-652–I-659 Vol.1 (2004)
20. Saurer, O., Köser, K., Bouguet, J.Y., Pollefeys, M.: Rolling shutter stereo. In: 2013 IEEE International Conference on Computer Vision. pp. 465–472 (2013)
21. Saurer, O., Pollefeys, M., Lee, G.H.: Sparse to dense 3D reconstruction from rolling shutter images. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3337–3345 (2016)
22. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 573–580 (2012)