Direct Semi-dense SLAM for Rolling Shutter Cameras

Jae-Hak Kim, Cesar Cadena and Ian Reid
School of Computer Science, University of Adelaide, Australia

Abstract—In this paper, we present a monocular Direct and Semi-dense SLAM (Simultaneous Localization And Mapping) system for rolling shutter cameras. In a rolling shutter camera, the pose is different for each row of each image, and this yields poor pose estimates and poor structure estimates when using a state-of-the-art semi-dense direct method designed for global shutter cameras. To address this issue in tracking, we model the smooth and continuous camera trajectory using a B-spline curve of degree \( k - 1 \) for poses in the Lie algebra, \( \mathfrak{se}(3) \). We solve for the camera poses at each row-time by a direct optimisation of photometric error as a function of the control points of the spline. Likewise for mapping, we develop generalised epipolar geometry for the rolling shutter case and solve for point depths using photometric error. Although each of these issues has been previously tackled, to the best of our knowledge ours is the first full solution to monocular, direct (feature-less) SLAM. We benchmark our method for pose accuracy and map accuracy against the state-of-the-art semi-dense SLAM system, LSD-SLAM, demonstrating the improved efficacy of our approach when using rolling shutter cameras via synthetic sequences with known ground-truth and real sequences.

I. INTRODUCTION

Many modern cameras making use of CMOS (Complementary metal oxide semiconductor) image sensors — from high-end Digital Single Lens Reflex cameras, portable action camcorders, through to humble web-cams — are potentially affected by the acquisition of their images using a RS (rolling shutter). Although the RS is electronically advantageous to a GS (global shutter), the RS means that each row (or column) of the image integrates light over a slightly different time window, meaning that images of a dynamic scene or acquired by a moving camera, may be distorted. Many algorithms in computer vision and robotics assume that the camera can be modelled via a central projection (i.e. projection through a single point), and this assumption is clearly violated by a moving RS camera, causing errors and other issues with these algorithms.

In recent years, there has been steady progress in the performance of single camera SLAM (Simultaneous Localization And Mapping) systems, with the current state-of-the-art arguably represented by the LSD-SLAM [5], [4] system that tracks and builds a semi-dense, textured map using a direct (feature-less) image warping and alignment technique. Though it has been shown to work successfully even for large scale environments a major drawback of this system is that it assumes a GS camera, and it performs poorly in the presence of significant (and sometimes even mild) RS effects — see e.g. Figure 1-(Left) — as acknowledged explicitly by the authors [4].

Using LSD-SLAM as a base, we propose a number of changes to incorporate a RS model. Although each of these changes is foreshadowed in previous work, it remains a significant challenge to bring these components together to produce a full working system. We demonstrate that our system is capable of building accurate maps and performing accurate camera tracking without recourse to other data such as that provided by inertial sensors in [15], known scene structure in [6] or direct depth as in [9]. To the best of our knowledge, our method proposed in this paper is the first full solution to monocular, direct (feature-less) SLAM that can handle RS cameras.

For tracking, we model the smooth and continuous camera trajectory using a B-spline curve of degree \( k - 1 \) for poses in the Lie algebra, \( \mathfrak{se}(3) \). We solve for the camera poses at each row-time by a direct optimisation of photometric error as a function of the control points of the spline. For mapping, likewise, we develop the generalised epipolar geometry for the RS case (related to push-broom geometry) and stereo matching which we use to solve for point depths using photometric error.

The remainder of the paper is organised as follows: in the next section we consider related work. In Section III we describe the background and our notation for the pose and trajectory representation via B-spline in \( \mathfrak{se}(3) \). In Section IV we introduce our solution to solving for the camera trajectory given a set of (semi-)dense scene points. Section V describes the generalised epipolar geometry required for rolling shutter stereo depth estimation. We show experimental results in Section VI and conclude in Section VII. The appendix gives a summary of equations and notation for quick reference.
II. RELATED WORK

A study of the geometry of RS cameras first appeared in [16] emphasizing the importance of taking into account the effect of RS when a captured scene contains any motion by the camera or objects.

Early work using RS cameras with SLAM appeared in [13] who made use of simple linear interpolation schemes. More recent efforts, inspired by the work of [3] who proposed a planar (3-DOF) ego-motion trajectory representation using cubic B-splines, have considered full 6-DOF representations using splines [15], [11]. In particular, addressing issues of \(C^2\)-continuity and singularities of other representations, [15] proposed to adopt the cumulative basis form, first described for smooth motion representation in computer graphics by [11]. They propose a landmark-based SLAM system that fuses inertial and visual data (from a RS camera). This system is shown to be effective for fusing the two forms of asynchronous measurement.

Recently, several works on RS cameras for stereo, bundle adjustment and calibration have been studied. Saurer et al. [20] showed a RS stereo method which improves accuracy compared to a GS based stereo method. Hedborg et al. [8] showed a bundle adjustment method on RS videos. Also calibration methods for RS camera have been proposed in [16], [18]. Direct SLAM approaches, which use dense textured maps and photometric based image warping (in contrast to landmark based approaches) have been introduced [17], [4]. The latter, in particular incorporates a number of “best practice” aspects to their system such as handling scale-drift error via \(\mathbb{SE}(3)\) pose-graph optimization over keyframes.

It has been shown to be effective at mapping quite large-scale areas. Nevertheless both its camera tracking and depth recovery are fundamentally based on a GS model which breaks down even for modest amounts of RS distortion.

Our main contribution is showing how the key ideas from a direct SLAM approach such as LSD-SLAM can be effective within a RS system, by use of splines to represent the camera trajectory, generalised image warping (and alignment) for camera tracking, and generalised epipolar geometry for photometric depth recovery.

III. BACKGROUND AND NOTATION

Our notation for Lie algebra is summarised as follows: A hat \(\hat{\cdot}\) operator on a 6-vector \(\xi\) in Lie algebra \(\mathfrak{se}(3)\) gives twist coordinates as a \(4 \times 4\) matrix as follows:

\[
[\xi]_\wedge = \begin{bmatrix}
[w]_\times & u \\
0_{1 \times 3} & 0
\end{bmatrix}
\]

for \(\xi = \begin{bmatrix}u^T & w^T \end{bmatrix}^T \in \mathfrak{se}(3)\), where \(u = [u_1, w_2, w_3]^T\) and \(w = [w_1, w_2, w_3]^T\) are translational and rotational parameters, respectively, and \([w]_\times\) is a \(3 \times 3\) skew-symmetric matrix from \(w\). A vee \(\langle \cdot \rangle_\vee\) operator on the \(4 \times 4\) matrix returns back to the original 6-vector \(\xi\) in \(\mathfrak{se}(3)\) as follows:

\[
[\xi]_\vee = \xi \in \mathfrak{se}(3).
\]

An exponential map of twist coordinates of a 6-vector \(\xi\) in Lie algebra \(\mathfrak{se}(3)\) gives a \(4 \times 4\) transformation matrix in the Lie group \(\mathbb{SE}(3)\) as

\[
\exp([\xi]_\wedge) := e^{[\xi]_\wedge} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in \mathbb{SE}(3),
\]

where \(R\) is a rotation matrix in \(\mathbb{SO}(3)\) and \(t\) is a translation vector. A logarithmic map of a \(4 \times 4\) transformation matrix in \(\mathbb{SE}(3)\) followed by a vee \(\langle \cdot \rangle_\vee\) operator gives a 6-vector in \(\mathfrak{se}(3)\) as follows

\[
\log \left( \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \right)_\vee = \log e^{[\xi]_\wedge} = \xi.
\]

For specifying a pose with respect to a coordinate system, we use the \(\mathfrak{se}(3)\) representation \(B^\cdot \xi_A\), for a pose \(A\) (bottom-right subscript) with respect to the coordinate system \(B\) (top-left superscript). It can be used to transform a 3D point \(A^\cdot X\) in the coordinate system \(A\) to a 3D point \(B^\cdot X\) in the coordinate system \(B\) as follows:

\[
B^\cdot X = e^{[B^\cdot \xi_A]}A^\cdot X.
\]

A concatenation operator \(\circ\) for pose multiplication is

\[
C^\cdot X = e^{[C^\cdot \xi_A]}A^\cdot X = e^{[C^\cdot \xi_B \circ B^\cdot \xi_A]}A^\cdot X,
\]

where the operator \(\circ : \mathfrak{se}(3) \times \mathfrak{se}(3) \rightarrow \mathfrak{se}(3)\) is for transforming a point in \(A\) to \(C\) via \(B\)'s coordinate system.

Our model for RS cameras assumes that: (1) the direction of electronic/mechanical RS scanning a scene is known (e.g. vertically for rows); (2) all pixels in a single scanning row of the RS image are exposed over the same period; (3) rows are exposed sequentially from top to bottom with a regular (i.e. linear) timing.

We use the symbol \(p = [p_x, p_y, 1]^T\) to represent an image pixel in a RS image, with its depth denoted \(p_z\). In a rolling shutter image, the pose at which a pixel is acquired is a function of the row; we refer to this as the row-pose of the pixel. For any two pixels \(p\) and \(q\) (either in the same or different images), the relative row-pose — i.e. the \(\mathbb{SE}(3)\) transformation between the poses at which they were acquired — is denoted \(q_{\xi_p}\). Given a pixel location \(p\) and its depth \(p_z\), along with the relative pose to a new view, we easily can transform the pixel into the new view using standard geometry. We denote this mapping by \(W(\cdot)\), a warp function:

\[
q = W(p, 1/p_z, q_{\xi_p}),
\]

where we assume that the cameras are calibrated and so omit these parameters from the equation.

We represent a camera trajectory by B-spline curve as in Lovegrove et al. [15]. We then may write a row-pose of \(p\) with respect to the row-pose of \(q\) as a function of \(k\) control points of B-spline and a time parameter \(u\) such that

\[
q_{\xi_p} = -\log(e^{[q_{\xi_0}]} \prod_{j=1}^{k-1} e^{[B(u)_{j+1}]}_{\xi_{i+1}}),
\]
where $B(u)$ is a vector-valued function for cumulative basis as $B(u) = M_k \begin{bmatrix} 1 & u & u^2 & \ldots & u^{k-1} \end{bmatrix}^T$ with a $k \times k$ matrix $M_k$. Here $k$ is the order of the B-spline, where we assume $k \leq 4$ (see [19] for details). The parameter $u \in [0, 1]$ selects a single point along a B-spline segment. We develop the theory below independent of the frequency of control points and the order of B-spline, with the actual values used in our experiments discussed in Section VI.

Figures 2 and 3 illustrate the relationship between the camera trajectory modelled via a B-spline and the images. The former conceptualises the timing and geometry, while the latter shows a graphical model of the dependencies between images, B-spline segments and control points. As shown in these examples, a B-spline segment may span multiple images. The row-pose of a pixel $q$ is related to the time of acquisition specified by $u_q$, which is a function of $q_y$, the row value of $q$:

$$u_q = (q_y + S_1 - S_s)/N_s.$$  

Here, $S_1$ is the image starting row, $S_s$ is the B-spline segment starting row and $N_s$ is the number of image rows in a B-spline segment. Note that if, say, there are two images per B-spline segment (as in Figure 2), then $N_s$ has twice the number of rows as an image. This assumes regular timing for rows and for simplicity of exposition does not admit any delay between the capture time of the last row of the current frame and the first row of the next frame, but this is easily incorporated.

IV. POSE OPTIMIZATION FOR TRACKING

As noted above, we model the trajectory of the camera as a B-spline parameterized by $k$ control points. The objective in any camera tracking system is to determine the pose – in our case the B-spline control points – using the image data of successive frames. We now consider how we achieve this in a direct-SLAM framework.

A. Direct Image Warping on B-spline

In LSD-SLAM, the current pose of the global shutter camera is obtained by optimizing a whole-image warp to minimize the photometric error (Equation (13) in [5]) between the warped current frame and its keyframe, as a function of the 6 pose parameters.

In the rolling shutter case the pose of each row, rather than each frame, is distinct. Thus we must formulate a multi-pose warping problem for RS images that maps a set of multiple row-poses to another set of multiple row-poses via the B-spline representation of pose. This is similar to a motion estimation problem of generalised cameras [14], [10].

More specifically we seek $k$ control points and the time parameters $u$ of the B-spline which describe all camera row-poses $^T_p \xi_S$ warping each semi-dense point $p$ in the keyframe image $I_S$ onto each point $q$ in the current image $I_T$. In a direct-SLAM framework we seek to do so by optimizing photometric error with respect to the B-spline control points. Of course this introduces various complications compared with the global shutter case.

The most pressing issue that arises is that warping under rolling shutter induces a chicken-and-egg problem: the warp of a pixel $p$ in the keyframe to $q$ on the target frame depends on the depth of $p$ and the (relative) pose parameters of the target camera. The relative pose parameters are determined by $u$, the position along the B-spline. A particular choice of pose-time $u$, however, implies a row-time for $q$, and thus the value of $q_y$ (the row to which the point projects). But the row $q_y$ is not known until we have performed the warping $q = W(p, 1/p_z, u)$. We refer to any difference between the pose-time $u$ and the row-time $u_q$ (obtained from the row coordinate of $q$) as the row-time difference.

B. SE(3) tracking with B-spline

We address the chicken-and-egg problem above by introducing a new term to the optimization. In addition to
photometric error, we penalize any non-zero row-difference:

$$\arg \min_{\Xi, u} \frac{1}{2} \sum_{i=1}^{\kappa} \sum_{p \in D_i} \rho \left( E_p^2 + \alpha E_r^2 \right)$$

\[ E_p = I_{S_i}(p) - I_{T_i}(q) \]

\[ E_r = u_q - u , \]

where we now optimize not only over the B-spline control points \( \Xi \) but over a per-point pose-time parameter \( u \). When the relative pose between \( p \) and \( q \) is correctly estimated, the pose parameter \( u \) and \( q \)’s row-time \( u_q \) (calculated from \( q_\theta \)) will coincide and there is no penalty. Incorrect control points, or an incorrect pose-time \( u \) will both lead to non-zero \( E_r \).

Equation (2) is the usual photometric residual (intensity or pose parameter \( u \)) of the residual photometric and row-difference errors for all semi-dense points, as a function of the \( k \) control points of the current B-spline segment and a per-point time parameter \( u \). The constant \( \alpha \), which we set empirically to \( \sqrt{500} \), balances the contributions of the photometric and row-difference error terms. Note that points \( p \) are defined semi-densely in a keyframe, only in areas with a large intensity gradient.

C. Multi-frame pose estimation

Although the section above shows how to obtain the current pose by (semi-)dense alignment of a keyframe and the current image, it does so by changing the keypoints of the B-spline, which in turn affects the row-poses of any images influenced by those keypoints (see Figure 3). In practice we find that the trajectory accuracy is greatly increased by considering the warp not only for the current frame in isolation, but for the full set of recent frames affected. This is similar in spirit to the sliding window Bundle Adjustment approach of [21], [2].

We refer to this set of frames as the “neighbour window”. This neighbour window can be deduced from the graphical model in Figure 3. The current frame belongs to a B-spline segment defined by \( k \) control points. A change in any one of these control points affects \( 2k - 1 \) segments in total, \( k \) of which are in the past. Each of these \( k \) most recent segments, and the images they generate (one or more image per segment) will participate in the optimization. For example, in Figure 3, if the current image is \( I_4 \) with a keyframe \( I_1 \), then control points \( \xi_2, \xi_3, \xi_4 \) and \( \xi_5 \) influence its pose via segment \( S_2 \). These control points also affect recent segment \( S_1 \) and through this the poses of images \( I_2 \) and \( I_3 \) are affected. Figure 4 illustrates this differently, showing the keypoints that participate when considering the pose of the current tracking frame \( I_2, \ldots, I_8 \) relative to keyframe \( I_1 \).

The penalty we pay for the improved accuracy and stability of the estimation over multiple frames is in the time taken for optimization, since we have greatly increased the number of variables that must be determined. We seek \( k \) control points, so in total \( 6k \) pose parameters. However in addition we optimize over one \( u \) for each semi-dense point. This yields an extra \( n \times k \times f \) variables where \( n \) is the number of semi-dense points in the current key-frame and \( f \) is the number of frames per spline segment.

D. Implementation

1) Jacobian computation: Jacobians with respect to the photometric error and row-time-difference error are computed easily thanks to automatic differentiation of Ceres-solver [1], and they are used in a nonlinear least-square optimization with a local parameterization of exponential map in Lie algebra for updating rules.

2) Control points in local coordinates: In our descriptions above the B-spline control points are represented with respect to the world coordinate system. However when we estimate the control points for image alignment by optimization, in practice we transform them into a local coordinate system defined at the pose of the first row of the oldest current keyframe image. Optimizing in such a local coordinate system yields more stable pose estimation.

3) Choice of keyframes: The condition for creation of a new keyframe is that a weighted combination of relative distance and angle to the current keyframe exceeds a threshold; this is identical to LSD-SLAM. However, when selecting a new keyframe from previously tracked images, we make sure to choose the oldest tracked image within the neighbour window, since we assume this to be the most accurately located, and least influenced by changing the current set of control points.

V. ROLLING SHUTTER STEREO

The fact that each row of a RS image is acquired from a different pose also complicates depth mapping. LSD-SLAM uses an epipolar search, seeking a best photometric match for a pixel in the current frame along the corresponding epipolar line in the (GS) keyframe. In the RS scenario, the epipolar line becomes a more general curve which depends on the camera motion and the scene depth.

<table>
<thead>
<tr>
<th>Key-frame</th>
<th>Track-frame</th>
<th>Control points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>○ ○ ○ ○ ○ ○ ○</td>
</tr>
</tbody>
</table>

Fig. 4. Optimization over multiple frames. For a cubic B-spline with 30Hz and 15Hz for camera and segment frequency, respectively, the current tracking frame 8 is associated with 4 control points marked as red circles, which are being estimated in the neighbour window approach. Filled (red/gray) circles with edges are control points affecting a B-spline segment given a pair of keyframe and tracking frame.
Sauer et al. [20] proposed a RS stereo method which involves specifying a prior motion model and solving a polynomial equation. Although we could use this, the LSD-SLAM framework separates pose calculation and depth computations (this separation was first very successfully pioneered in PTAM [12]), and this means a simpler solution is available to us, namely finding the generalised epipolar curve as a function of the pose and searching in a manner analogous to any other stereo disparity search.

Our implementation separates this searching process over a generalised epipolar curve into two steps: (1) Extract image points along the generalised epipolar curve followed by sorting in the order of the inverse depth, and copy them into a buffer; and (2) Search for the matching disparity from the buffer.

A. Generalised Epipolar Curve.

Let us consider two RS images $I_0$ and $I_1$ as shown in Figure 5. The image ray L is a back-projection of an image point $x$ in the left RS image $I_0$. The $i$-th scan-line in the right RS image $I_1$ is back-projected as a plane in space, and this plane intersects with ray L at a point $X_i$. This point projects onto the right RS image $I_1$ at an image point $x'_i$. Note that $x'_i$ is of course a point on the conventional epipolar curve of a GS image acquired at the pose associated with row $i$ of $I_1$, and is at the intersection of the conventional epipolar line and the scan-line. The generalised epipolar curve for RS images is now built by finding the locus of all intersection points $x'_i$ by sweeping the set of $I_1$'s rows.

Note that we must exclude points with negative depth, and ensure that our buffer is sorted by increasing (inverse) depth, which may differ from the row sweeping order. We must also in practice consider the fact that each scan line is of finite width; thus the intersection between a conventional epipolar line and the scan line may span multiple pixels. We deal with this case (which arises regularly during horizontal translational motion) by back-projecting both the top and bottom of the scanline to planes which intersect L in two distinct 3D points. These points span a range of possible depths, and project to a consecutive set of pixels along the scanline in $I_1$.

Sauer et al. [20] proposed a RS stereo method which involves specifying a prior motion model and solving a polynomial equation. Although we could use this, the LSD-SLAM framework separates pose calculation and depth computations (this separation was first very successfully pioneered in PTAM [12]), and this means a simpler solution is available to us, namely finding the generalised epipolar curve as a function of the pose and searching in a manner analogous to any other stereo disparity search.

Our implementation separates this searching process over a generalised epipolar curve into two steps: (1) Extract image points along the generalised epipolar curve followed by sorting in the order of the inverse depth, and copy them into a buffer; and (2) Search for the matching disparity from the buffer.

A. Generalised Epipolar Curve.

Let us consider two RS images $I_0$ and $I_1$ as shown in Figure 5. The image ray L is a back-projection of an image point $x$ in the left RS image $I_0$. The $i$-th scan-line in the right RS image $I_1$ is back-projected as a plane in space, and this plane intersects with ray L at a point $X_i$. This point projects onto the right RS image $I_1$ at an image point $x'_i$. Note that $x'_i$ is of course a point on the conventional epipolar curve of a GS image acquired at the pose associated with row $i$ of $I_1$, and is at the intersection of the conventional epipolar line and the scan-line. The generalised epipolar curve for RS images is now built by finding the locus of all intersection points $x'_i$ by sweeping the set of $I_1$'s rows.

Note that we must exclude points with negative depth, and ensure that our buffer is sorted by increasing (inverse) depth, which may differ from the row sweeping order. We must also in practice consider the fact that each scan line is of finite width; thus the intersection between a conventional epipolar line and the scan line may span multiple pixels. We deal with this case (which arises regularly during horizontal translational motion) by back-projecting both the top and bottom of the scanline to planes which intersect L in two distinct 3D points. These points span a range of possible depths, and project to a consecutive set of pixels along the scanline in $I_1$.

### Table I

<table>
<thead>
<tr>
<th>Sequence</th>
<th>LSD-SLAM</th>
<th>Ours</th>
<th>LSD-SLAM</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence77</td>
<td>98.24</td>
<td>2.36</td>
<td>9.41</td>
<td>0.34</td>
</tr>
<tr>
<td>Sequence154</td>
<td>15.71</td>
<td>3.62</td>
<td>2.51</td>
<td>0.51</td>
</tr>
</tbody>
</table>

#### VI. Experiments

A. Synthetic sequences

For synthetic data generation, we first rendered photo-realistic GS images of a 3D model scene by using POV-Ray from ICL-NUIM RGBD-benchmark dataset generation code [7], then built a RS sequence by gathering all scanlines from the images corresponding to the ground truth pose of RS camera. Two image sequences were created for the same trajectory but different speeds of camera motion. Sequence154 contains 154 RS images which are extracted and accumulated from 73,920 GS images of 640 × 480 resolution. Sequence77 comprises 77 RS images generated from 36,960 rendered images as shown in Figure 6. The motion in this synthetic data is translational only, with no rotational component. First the camera translates in the xy-plane (perpendicular to the viewing direction) in a small loop (the so-called “SLAM-waggle”), then proceeds forward (+z), to the right (+x), backward (−z) and left (−x), followed by another forward motion (+z) to close a full loop. This motion is within a 2m × 0.4m × 2.25m volume.

For processing the synthetic sequences we used B-spline segment frequency of 30Hz (i.e. one segment per image) with 2 control points per segment for a linear B-spline. Translational and rotational RMSE errors are reported in Table I. In Figure 8 and Figure 9, the estimated positions in the $x$, $y$ and $z$ axes are compared with the ground truth. The comparative results of mapping and tracking by our method and LSD-SLAM are shown in Figure 7. These experimental results show our method significantly improves accuracy against the ordinary LSD-SLAM, both in terms of pose estimation and 3D mapping.

Experiments on a real sequence was conducted from TUM/RGBD-SLAM-Benchmark dataset [22]. The real data we selected was “fr1/desk” sequence captured by a handheld RGBD sensor with a fast motion (average translational velocity 0.413 m/s) causing RS distortion. Our method shows improved accuracy in trajectory estimation and 3D reconstruction against the ordinary LSD-SLAM as shown in Figure 10 and Figure 11.
Fig. 6. **Sample of Sequence77.** Images of the living room model are rendered using [7] then scanlines are gathered to generate a synthetic RS sequence. Significant RS distortion can be seen in particular the middle and last sample image.

Fig. 7. **Comparison of our Rolling-Shutter Direct-SLAM method to the GS based ordinary LSD-SLAM method [4].** (Top-row) LSD-SLAM results from Sequence77 – first three columns – and Sequence154 – last three columns. (Bottom-row) Our method results from Sequence77 – first three columns – and Sequence154 – last three columns. Gray dots indicate a recovered structure by mapping. Green lines indicate estimated trajectory by tracking. Keyframes are marked by blue camera frustum. Initial ground truth depth at the first frame was given for all experiments. Loop-closing optimization and relocalization were not used. Sequence77 (first to third column) has a faster camera motion than Sequence154 (fourth to sixth column).

Fig. 10. **Results of tracking from real sequence “fr1/desk”.** (a) First keyframe image. (b) Ground-truth (red) and estimated trajectories by our method (green) and LSD-SLAM (blue) starting from (0, 0, 0). Initial ground truth depth at the first frame was given for all experiments. Loop-closing optimization and relocalization were not used.

Fig. 11. **Results of mapping from real sequence “fr1/desk”.** (a) 3D point cloud of depths reconstructed by LSD-SLAM. (b) Point cloud obtained by our method. (c) A top-side view (point cloud only) of (b). Initial ground truth depth at the first frame was given for all experiments. Loop-closing optimization and relocalization were not used.
end operating asynchronously. “front end”, corrected by a slower but more accurate back-
ioverall, we believe we have developed a formulation for rolling shutter direct-
SLAM that is capable of real-time processing. The current state-of-the-art in semi-dense tracking and mapping is essential to achieve accurate
Taking into account the rolling shutter effect within semi-
dense tracking and mapping is crucial. We have developed a formulation for rolling shutter direct-
SLAM which models the camera trajectory using a B-spline. Camera tracking is performed by optimizing photometric error which is a function of the multiple-pose warp that maps
one rolling shutter image to another, with respect to parameters of a B-spline which represents the camera trajectory. Depth estimation is performed by minimising a photometric error cost along generalized epipolar curves. We have shown that taking into account the rolling shutter effect within semi-
dense tracking and mapping is essential to achieve accurate reconstruction, using which our method outperforms the current state-of-the-art in semi-dense tracking and mapping (a method which uses a global shutter camera model).

To date we have not considered issues of scale, conducting all of our trajectory modelling in SE(3). A future extension will involve reformulating the problem to admit SIM(3) to incorporate scale changes and drift. We also hope to incorporate lens distortion; however this is not a trivial task, since a simple non-linear “un-warp” that is usually applied in the global shutter case results in points changing image rows, in which a naive treatment would also alter their timestamp and pose.

Processing time is currently a significant weakness of our system. On an Intel Core i5 CPU 3.2GHz with 4 threads, we dedicated one thread to tracking and three threads to mapping. For Sequence77, our system took about 3 hours (i.e. 140 sec per frame) which is substantially slower than LSD-SLAM that is capable of real-time processing. The majority of the processing cost is associated with computing Jacobians (using Ceres-solver’s automatic differentiation mode of operation) during the pose and depth estimation iterations. We are currently implementing analytic Jacobians to improve this, but it seems likely we will only be able to achieve real-time operation by having a fast approximate “front end”, corrected by a slower but more accurate back-end operating asynchronously.

VIII. APPENDIX

In this section, we summarize and explain all the equations and notations used throughout this paper for SE(3) tracking in detail.

\[
E_p = \mathcal{I}_{S_i}(p) - \mathcal{I}_{T_i}(q) \\
E_r = u_q - u \\
p = \begin{bmatrix} p_x & p_y \end{bmatrix}^\top = \begin{bmatrix} \mathbf{I} \end{bmatrix} \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} \mathbf{P} \end{bmatrix} \\
q = \begin{bmatrix} q_x & q_y \end{bmatrix}^\top = \begin{bmatrix} \mathbf{I} \end{bmatrix} \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} \mathbf{Q} \end{bmatrix} \\
\begin{bmatrix} \mathbf{P} \end{bmatrix} = \begin{bmatrix} \frac{1}{P_c} (K^-1\mathbf{p})^\top \end{bmatrix}^\top \\
\begin{bmatrix} \mathbf{Q} \end{bmatrix} = e^{[\mathbf{Q}] p} \begin{bmatrix} \mathbf{P} \end{bmatrix} \\
q = W(p, \frac{1}{p_2}, q_\xi p) \\
\Xi = \{W^j_\xi(j) | j = 1, \ldots, k \} \\
\begin{bmatrix} p_\xi(j) \end{bmatrix} = (-W^j_\xi p) \circ W_\xi(j) \\
q_\xi p = -\begin{bmatrix} p_\xi q \end{bmatrix} = [\log(PT_{q-1})]_\forall \\
PT_q = e^{[\mathbf{P}] \xi_{(1)}_\land} \prod_{j=1}^{k-1} e^{[B(u)] \xi_{(j+1)}_\land} \\
\begin{bmatrix} (j) \xi_{(j+1)} \end{bmatrix} = \left[\log \left( e^{[\mathbf{P}] \xi_{(j)}_\land} \prod_{j=1}^{k-1} e^{[\mathbf{P}] \xi_{(j+1)}_\land} \right) \right]_\forall \\
B(u) = \mathbf{M}_k \begin{bmatrix} 1 & u & u^2 & \ldots & u^{k-1} \end{bmatrix}^\top,
\]

where

- \rho(\cdot) is Huber loss function.
- \(D_i\) is a set of semi-dense points in the keyframe \(S_i\).
- \(E_p\) is a photometric error.
- \(\mathcal{I}_{S_i}(\cdot)\) is image intensity of a given pixel in the \(i\)-th keyframe image in the neighbour window. Note that the size of neighbour window is \(k\), which is equal to the order of B-spline curve.
- \(\mathcal{I}_{T_i}(\cdot)\) is image intensity of a given pixel in the image of the \(i\)-th current frame in the neighbour window.
- \(p\) is a vector for 2D coordinates of the image point in the keyframe image \(I_{S_i}\) as a projection of the 3D point \(\begin{bmatrix} \mathbf{P} \end{bmatrix}\) by a calibrated perspective camera.
- \(q\) is the warped image point in the frame image \(I_{T_i}\). It is a projection of 3D point \(\begin{bmatrix} \mathbf{Q} \end{bmatrix}\) by a calibrated perspective camera after a transformation into the tracking frame coordinate system for the row-pose of \(q\).
- \(E_r\) is a row-time-difference error.
- \(\begin{bmatrix} \mathbf{P} \end{bmatrix}\) is a vector for 3D point with respect to the keyframe coordinate system for the row of \(p\), and is obtained by back-projection and the inverse depth \(1/p_2\).
- \(\begin{bmatrix} \mathbf{Q} \end{bmatrix}\) is a 3D point with respect to the current frame coordinate system for a row of \(q\). It is obtained by a transformation of the 3D point \(\begin{bmatrix} \mathbf{P} \end{bmatrix}\) via \(\begin{bmatrix} \mathbf{Q} \end{bmatrix}\).
- \(\Xi\) is a set of poses of \(k\) control points with respect to the world coordinate system \(W\).
- \(\begin{bmatrix} p_\xi(j) \end{bmatrix}\) is the \(j\)-th control points with respect to the row-pose \(p\) in the keyframe image \(S_i\). The pose of \(p\) with respect to the world \((W_\xi p)\) is known in the first keyframe or estimated from a previous tracking step. Then, \(W_\xi(p)\) is required to be estimated.
- \(\begin{bmatrix} p_\xi q \end{bmatrix}\) is a row-pose of \(q\) in the current frame coordinate system with respect to a row-pose of \(p\) in the keyframe coordinate system in \(se(3)\), i.e. a transformation moving a point \(q\) in the frame system to a point \(p\) in the
keyframe system. It is expressed as a B-spline equation which is multiplications over exponential mappings of poses of control points with B-spline basis coefficients.

- $W(\cdot)$ is an image warping function taking the image point $p$ in the keyframe image $S_i$ in the neighbour window and warps to the image point $q$ in the current frame $T_i$ with a known depth $\frac{1}{z}$ by a transformation $\xi_p$. It is expressed by control points $\Xi$ to be estimated.

- $\xi^j_{(j+1)}$ is a pose of the $(j + 1)$-th control point with respect to the $j$-th control point in $s\Xi(3)$. It may be obtained by a concatenation operator of poses of two control points in $s\Xi(3)$.

- $B(u)$ is the cumulative basis as a $k \times 1$ vector by a $k \times k$ matrix $M_k$ as follows:

$$
M_2 = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}, \\
M_3 = \frac{1}{2} \begin{bmatrix}
2 & 1 & 0 \\
0 & 2 & 0 \\
0 & -1 & 1
\end{bmatrix}, \\
M_4 = \frac{1}{6} \begin{bmatrix}
6 & 0 & 0 & 0 \\
5 & 3 & -3 & 1 \\
1 & 3 & 3 & -2 \\
0 & 0 & 0 & 1
\end{bmatrix},
$$

and $B(u)_j$ indicates $j$-th element in the vector.

- $u_q$ is a time value in terms of the $y$-coordinate of $q$ related to the number of rows in a segment.

- $M_k$ is a $k \times k$ matrix for cumulative B-spline basis as [19], [15].

- $\xi^i_{p}$ is a transformation of a pose of the image point $p$ in $S_i$ into a pose of the image point $q$ in $T_i$.

- $W^i_p$ is a row-pose (i.e. a pose of the image row) for an $p$ in an image point $p$ with respect to the world. It is determined from a time parameter $u_p$, the B-spline curve segment where the row-pose lies on.

- $S_i$ is a keyframe image from a set of keyframes as multiple tracking references. It is a keyframe image corresponding to the current frame $T_i$.

- $T_i$ is a current frame image from a set of tracking frames in the neighbour window (section IV). It is a current tracking frame corresponding to the keyframe $S_i$. $W$ is the world coordinate system.

- $k$ is the number of control points, i.e. the order of polynomial for the B-spline, such that $\{k \in Z | 2 \leq k \leq 4\}$.

- $K$ is a $3 \times 3$ calibration matrix.

- $u$ is the time parameter value between 0 and 1 which determines a pose on a B-spline segment. $u_p$ and $u_q$ denote time parameter for image point $p$ in the image $S_i$ and $q$ in the image $T_i$, respectively.

- $S_T$ is the starting row for the tracking image.

- $S_k$ is the starting row on the current B-spline segment.

- $N_k$ is the number of rows per B-spline segment.

- $\alpha$ is a weight for the row-time-difference error.

IX. ACKNOWLEDGEMENTS

We are extremely grateful to the Australian Research Council for funding this research through project DP130104413, the ARC Centre for Robotic Vision CE140100016, and through a Laureate Fellowship FL130100102 to IDR. We thank Ankur Handa for his POV-Ray code [7], and Computer Vision Group in Technische Universität München for their LSD-SLAM code and RGBD-benchmark dataset. We also highly appreciate all discussion with and comments from Tom Drummond, Alireza Khosrovian, Yasis Latif, Zygmunt Szpak and others in the Australian Centre for Robotic Vision and Australian Centre for Visual Technologies.

REFERENCES