Fast Rolling Shutter Correction in the Wild

Delin Qu, Bangyan Liao, Huiqing Zhang, Omar Ait-Aider, Yizhen Lao

Fast Rolling Shutter Correction in the Wild

Delin Qu*, Bangyan Liao*, Huiqing Zhang, Omar Ait-Aider, and Yizhen Lao†

Abstract—This paper addresses the problem of rolling shutter correction (RSC) in uncalibrated videos. Existing works remove rolling shutter (RS) distortion by explicitly computing the camera motion and depth as intermediate products, followed by motion compensation. In contrast, we first show that each distorted pixel can be implicitly rectified back to the corresponding global shutter (GS) projection by rescaling its optical flow. Such a point-wise RSC is feasible with both perspective, and non-perspective cases without the pre-knowledge of the camera used. Besides, it allows a pixel-wise varying RSC framework called DRSC that handles locally varying distortion caused by various sources, such as camera motion, moving objects, and depth variation in a scene. More importantly, our approach is an efficient CPU-based solution that enables undistorting RS video in real-time (40fps for 480p). We evaluate our approach across a broad range of cameras and video sequences, including fast motion, dynamic scenes, and non-perspective lenses, demonstrating the superiority of our proposed approach over state-of-the-art methods in both effectiveness and efficiency. We also evaluated the ability of the RSC results to serve for downstream 3D analysis, such as visual odometry and structure-from-motion, which verifies preference for the output of our algorithm over other existing RSC methods.

Index Terms—Rolling Shutter, Distortion correction, Optical flow, Epipolar geometry.

1 INTRODUCTION

THE Rolling shutter (RS) is a widely used shutter mechanism in most of today's consumer cameras due to its low cost and compact design [1]. Compared to a global shutter (GS) camera, which exposes each row simultaneously, the RS camera sequentially captures the image pixels row by row from top to bottom [2]. The time delay between consecutive rows causes different distortions, *e.g.*, skew, smear and wobble when a camera is moving or filming dynamic objects [3]. Such distortion, namely the RS effect, not only brings a visual unpleasantness against user perception but also defeats further geometry analysis solutions developed based on GS assumptions, such as structure-from-motion (SfM) [4] and SLAM [5]. Therefore, by removing distortions from images, rolling shutter correction (RSC) benefits various vital applications, such as 3D computer vision, robotics, and consumer mobile photography, and attracted much attention in the last decade. Existing works on RS correction are generally categorized into inertial measurement unit-based (IMU-based) methods [6], [7], [8], [9], [10], single-frame methods [11], [12], [13], [14], multi-frame methods [15], [16], [17], [18], [19] and learning-based methods [20], [21], [22], [23], [24].

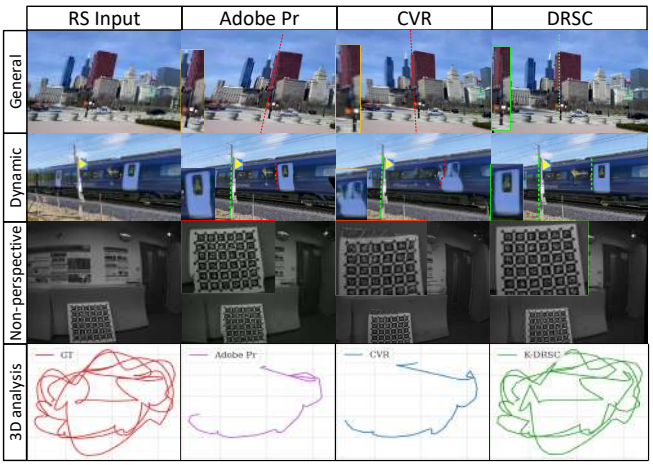

Fig. 1: Quantitative evaluations of the real-world RSC. From top to bottom are static, dynamic, and non-perspective lens scenes. RSC results to support the downstream application, such as visual-odometry, are shown in the last row. Our proposed method generates significantly higher quality images than commercial software Adobe Pr [25] and state-of-the-art learning-based RSC method CVR [26].

1.1 Related Work

IMU-based methods. Previous works [6], [7], [8], [9] utilize gyroscopes to measure camera instantaneous motion of the camera during video caption and assist compensation of RS effects. Recently, the work of [10] incorporated IMU data into a deep network for RS rectification. However, these methods strongly depend on external sensors, limiting their practical use without mentioning the requirement for precise multi-sensor calibration and synchronization.

Single-frame methods. Most methods in this category use line features since they are generally abundant in artificial environments. After detecting line features, performs RSC following the "straight lines must be straight" principle in [11], vanishing direction constraints in [12], [13], or analytical 3D straight line RS projection model in [14]. However, such solutions commonly rely on strong assumptions not

- *D. Qu, B. Liao, H. Zhang and Y. Lao are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, 410082.*
- *O. Ait-Aider is with Institut Pascal, Universite Clermont Auvergne / CNRS, France.*

* *Authors contributed equally*
† *Corresponding author: yizhenlao@hnu.edu.cn*
Manuscript received XXXXXX.

only of the filmed scene, namely abundant 3D line [11], [12], [13], [14] but also simplified camera motion, namely rotation-only model [12], [14], [20], Ackermann model [12]. Obviously, these approaches will fail in the wild scenes lacking line features or when the camera is under translation.

Multi-frame methods. Vasu *et al.* [27], and Zhuang *et al.* [28] present two RS removal methods using multiple 1-dimension and 2-dimension local homography fields between two successive frames, respectively. While the works of [16] and [29] address the general unordered RS images (no need to be successive) case and respectively propose a row-by-row local mixture homography and global homography for general unordered RS images. Beyond homography, Zhuang *et al.* [18] develop an epipolar-geometry-based RS correction solution with 8pt and 9pt linear solvers for two consecutive frames under constant velocity and acceleration. In contrast to a 2-frame solution, the method of [19] tries to recover the trajectory of the camera and compensate RS effects by scanline realignment via interpolation. Nevertheless, the above methods strongly depend on prior lens calibration or suffer from high computational load and simplified motion assumption.

Learning-based methods. Rengarajan *et al.* [20] first develop a learning-based RS correction method using CNN. After that, Zhuang *et al.* [21] expand [20] to learn depth and camera motion from a single RS image. Recently, Liu *et al.* [22] present an end-to-end RS correction network with a differentiable forward warping module by giving two consecutive RS images. The work of [23] proposes a simultaneous RS correction and motion deblur network with a deformable attention module to fuse features. The work of [24] presents a deep network that consists of a PWC-based [30] undistortion flow estimator and a time-centred GS image decoder. Lately, Fan *et al.* [26] introduce a context-aware deep RSC model that enables occlusion reasoning and motion compensation simultaneously.

Nevertheless, learning-based approaches require significant storage and computational resources, as well as a substantial financial investment, due to the large size of the training model and the need for GPU support, without mentioning the cost of preparing qualified training datasets.

1.2 Motivation

Although RSC has made significant progress in certain scenes, such as perspective projection, rigid scenes, and specific camera motion, no methods have yet addressed the 'wild' scenarios, namely the following three challenges:

General dynamic scene. Most conventional RSC methods assume the underlying scene is static and the RS effect is caused by camera ego motion. Though the homography [16] based or epipolar [18] based criterion can be utilized in RANSAC to filter out the dynamic objects features. However, such approaches will fail when encountering massive moving objects. Learning-based methods have been seen as a potential tool to handle such cases once the training dataset has covered sufficient dynamic scenes [23]. However, as shown in Fig. 1 (second row), even the state-of-the-art works are in a nascent stage for dynamic scenes. Since general dynamic scenes are common in the wild, correcting the distortion caused by the camera motion and moving objects simultaneously is challenging but vital.

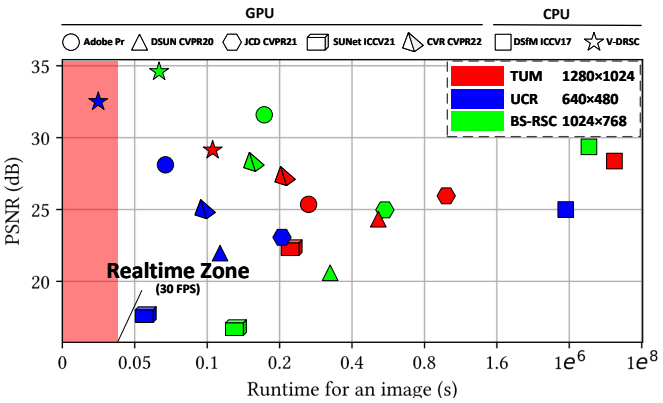

Fig. 2: PSNR(dB) vs. runtime(ms) of state-of-the-art RSC methods and our method on different datasets. The region indicates real-time inference at 30 fps. The red icons, blue icons and green icons are methods on the 1280×1024 dataset, 640×480 dataset and 1024×768 dataset, respectively. Our method is better not only in the efficiency aspect but also in the accuracy aspect.

Calibration-free RSC with lens distortion. Since both the RS effect and non-perspective lens will produce image distortion, correcting the RS effect independently from lens distortion is challenging. To the best of our knowledge, except for the works of [31] considering both the RS effect and radial distortion in camera tracking, no RSC methods before addressed the RS effect with a non-perspective model. One possible solution is to rectify the lens distortion by using the lens calibration as a prior, followed by RSC later. However, This prevents using these algorithms where only the video is available, without further knowledge or access to the lens calibration. Note that calibration-free RSC methods [16], [27], [28], [32], which assume homography for the perspective camera, are not feasible in non-perspective cases. As shown in Fig. 1 (third row), the existing learning-based methods lost their generalization ability in such scenarios without preparing specific training datasets. Therefore, since non-perspective images/videos are common in consumer cameras (*e.g.*, sports cameras) and robotic vision, calibration-free RSC to a non-perspective camera is essential. Unfortunately, this case has not been well addressed.

Light and efficient RSC. Notice that RSC is common in (i) consumer image/video editing apps on PC or mobile phones and (ii) robot vision systems (*e.g.*, cameras on autonomous vehicles), where real-time correction is often required (*e.g.*, SLAM/visual odometry) under the limited power, storage, and computation resources. However, the existing geometry-based RSC [18] is time-consuming, while the learning-based approaches are common with massive model sizes and power-consuming GPU acceleration, without mentioning the cost of preparing the training data. Thus, an RSC solution that is light (small size, low-cost, and low-power consuming) and efficient (real-time performance) is vital for practical applications.

1.3 Contribution and Paper Organization

To fill the gap between the three challenges of in-the-wild RSC and existing works, we introduce a novel RSC

TABLE 1: Comparison of the proposed method *DRSC* vs. the state-of-the-art RSC solutions.

Method	Learning-based				Geometry-based				
	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	Mixed homography [16]	Global homography [32]	Multiple homography [28]	DSfM [18]	DRSC (ours)
Dynamic Scene	√	√	√	√					√
Depth-dependent RS effect	√	√	√	√				√	√
Calibration-free	√	√	√	√	√	√	√		√
Lens distortion									√
GPU-free					√	√	√	√	√
Runtime (FPS)	8	5	17	11	5	0.2	-	0.005	40

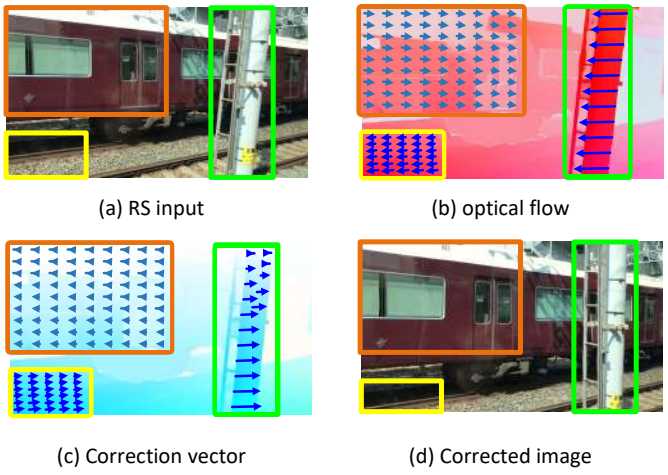

Fig. 3: (a) RS frame in the dynamic scene. (b) Raw optical flow. (c) Our pixel-wise varying correction vector field. (d) Output corrected image.

framework called direct RS correction (***DRSC***) that can effectively and efficiently remove the RS effect from two successive frames. Our novel insight is that an optical flow between two successive RS frames implicitly models the instantaneous motion and scene depth. Thus, we draw the mathematical link (Fig. 6) between the optical flow (1st RS to 2nd RS frame) and its corresponding correction flow (1st RS to 1st GS frame) based on epipolar geometry. In other words, we express the RSC of each pixel parametrically as a rescaling of its corresponding optical flow vector. Therefore, by tracking the optical flow of a single point feature between two successive RS frames, the proposed method directly brings this RS feature back to the GS projection coordinates. As summarized in Tab. 1, this simple but effective strategy leads to **five advantages** over the existing works:

1. General 6 DoF motion and depth-dependent RS effect. The proposed method can handle the general 6 DoF motion and depth-dependent RS effect while the single-frame-based method [11], [12], [13], [14] and homography-based method [16], [27], [28], [29] fails in correcting RS effect produced by translation and depth.

2. Dynamic scenes. As shown in Fig. 1(second row), the proposed method with a pixel-wise correction scheme can handle locally varying distortion caused by camera motion and moving objects better than the existing RSC works without expensive image segmentation.

3. Calibration-free for non-perspective cameras. The proposed method performs RSC in pixel-level image space

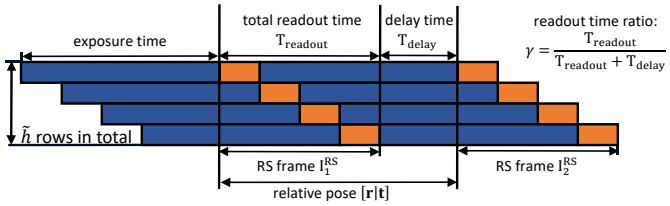

Fig. 4: Illustration of the exposure mechanism between two RS consecutive frames.

without requiring any prior calibration. Besides, we theoretically and experimentally verify that the proposed method is feasible for both perspective and non-perspective models (Fig. 1(third row)). To the best of our knowledge, this is the first calibration-free RSC solution for the non-perspective case.

4. Serve for downstream 3D analysis. Thanks to the pixel-wise correction scheme, the proposed method corrects the matched or tracked features which can serve for downstream 3D vision applications such as SfM/SLAM. Thus, as shown in Fig. 1(fourth row), we can easily use DRSC as a flexible tool to plug in most of the famous SfM (*e.g.*, VisualSfM [4]) and SLAM frameworks (*e.g.*, ORB-SLAM2 [33]).

5. Fast RSC in real-time. The proposed solution is based on a novel single-point analytical solver free from iterative optimization or RANSAC selection. As shown in Fig. 2, *DRSC* runs efficiently over the existing works without GPU acceleration.

In summary, our contributions include:

- A novel geometry-based pixel-wise pose-free correction solver that corrects a single image pixel parametrically by rescaling its corresponding optical flow vector, which faithfully models the general 6 DoF motion and depth-depend RS effect.
- A calibration-free RSC method that handles both perspective and non-perspective cameras in dynamic scenes.
- Extensive evaluations of visual correction quality across a broad range of cameras and scenes demonstrate the superiority of our proposed approach over state-of-the-art methods in terms of effectiveness and efficiency. Besides, we experimentally verify that the proposed method can effectively augment GS SfM/SLAM to RS-aware ones.
- A highly efficient RSC solution that corrects RS video at 40 fps can support the RS-VO and RS-SLAM in real-time.

2 OPTICAL FLOW FOR GS AND RS INSTANTANEOUS MOTION

In this section, we provide a brief description of the derivation of optical flow that relates to the instantaneous motion between two successive GS frames and two RS frames, respectively. Since this section does not include our contributions, we give only the basic information to follow the rest of this paper. More details of the algorithm can be found in our supplementary materials and [18], [34], [35].

2.1 Instantaneous GS Optical Flow Model [34] [35]

Assuming a 3D point $\mathbf{P} = [X, Y, Z]^\top$ is projected as normalized image points $\mathbf{q}_1^{\mathrm{GS}} = [x_1^{\mathrm{GS}}, y_1^{\mathrm{GS}}, 1]^\top$ and $\mathbf{q}_2^{\mathrm{GS}} = [x_2^{\mathrm{GS}}, y_2^{\mathrm{GS}}, 1]^\top$ in two consecutive GS frames $\mathrm{I}_1^{\mathrm{GS}}$ and $\mathrm{I}_2^{\mathrm{GS}}$ with the constant instantaneous motion integrated by translation velocity $\mathbf{t} = [t_1, t_2, t_3]^\top$ and rotational velocity $\mathbf{r} = [r_1, r_2, r_3]^\top$. The GS optical flow $\mathbf{u}_{(1:2)}^{\mathrm{GS}}$ induced by $(\mathbf{r}, \mathbf{t})$ of a certain $\mathbf{q}_1^{\mathrm{GS}}$ is first introduced in [34] as:

$$\mathbf{u}_{(1:2)}^{\mathrm{GS}} = \frac{\mathbf{A}(x_1^{\mathrm{GS}}, y_1^{\mathrm{GS}})\mathbf{t}}{Z} + \mathbf{B}(x_1^{\mathrm{GS}}, y_1^{\mathrm{GS}})\mathbf{r}, \tag{1}$$

$$\mathbf{A}(x_1^{\mathrm{GS}}, y_1^{\mathrm{GS}}) = \begin{bmatrix} -1 & 0 & x_1^{\mathrm{GS}} \\ 0 & -1 & y_1^{\mathrm{GS}} \end{bmatrix}, \tag{1a}$$

$$\mathbf{B}(x_1^{\mathrm{GS}}, y_1^{\mathrm{GS}}) = \begin{bmatrix} x_1^{\mathrm{GS}} y_1^{\mathrm{GS}} & -(1 + x_1^{\mathrm{GS}2}) & y_1^{\mathrm{GS}} \\ (1 + y_1^{\mathrm{GS}2}) & -x_1^{\mathrm{GS}} y_1^{\mathrm{GS}} & -x_1^{\mathrm{GS}} \end{bmatrix}, \tag{1b}$$

where the global shutter homogeneous optical flow vector $\mathbf{u}^{\mathrm{GS}}$, $\mathbf{u}^{\mathrm{GS}} \in \mathbb{R}^3$ and $\mathbf{u}_{(3)}^{\mathrm{GS}} = 0$, indicates the displacement of a pixel over two consecutive GS frames, which is equal to $\mathbf{q}_2^{\mathrm{GS}} - \mathbf{q}_1^{\mathrm{GS}}$. We denote $\mathbf{u}_{(i:j)}^{\mathrm{GS}}$ as a vector consisting of the i^{th} to j^{th} elements of $\mathbf{u}^{\mathrm{GS}}$. Z is the corresponding depth of each pixel and can be eliminated to yield the differential epipolar constraint, which is first given in [35] as:

$$\mathbf{u}^{\mathrm{GS}\top}[\mathbf{t}]_\times \mathbf{q}_1^{\mathrm{GS}} - \mathbf{q}_1^{\mathrm{GS}\top}\mathbf{s}\mathbf{q}_1^{\mathrm{GS}} = 0, \tag{2}$$

$$\text{with,} \quad \mathbf{s} = \frac{1}{2}([\mathbf{t}]_\times[\mathbf{r}]_\times + [\mathbf{r}]_\times[\mathbf{t}]_\times), \tag{2a}$$

where the velocities $\mathbf{r}$ and $\mathbf{t}$ indicate the poses of two GS frames. Specifically, $[\mathbf{I}|\mathbf{0}]$ for $\mathrm{I}_1^{\mathrm{GS}}$ and $[\mathbf{R}|\mathbf{t}]$ for $\mathrm{I}_2^{\mathrm{GS}}$, where the relative rotation is modelled using small rotation approximation $\mathbf{R} = \mathrm{Exp}(\mathbf{r}) \approx \mathbf{I} + [\mathbf{r}]_\times$, where $[\mathbf{r}]_\times$ is the skew-symmetric matrix of $\mathbf{r}$.

2.2 Instantaneous RS Optical Flow Model [18]

The work of [18] extends the instantaneous GS optical flow model to RS case. As the rolling shutter mechanism shown in Fig. 4, the sensor scans the scene rapidly and sequentially. Thus, every single scanline holds an individual camera pose relevant to the row index and readout time ratio γ. Assuming a 3D point $\mathbf{P}$ is filmed by two consecutive RS frames $\mathrm{I}_1^{\mathrm{RS}}$ and $\mathrm{I}_2^{\mathrm{RS}}$ as $\mathbf{q}_1^{\mathrm{RS}} = [x_1^{\mathrm{RS}}, y_1^{\mathrm{RS}}, 1]^\top$ and $\mathbf{q}_2^{\mathrm{RS}} = [x_2^{\mathrm{RS}}, y_2^{\mathrm{RS}}, 1]^\top$. As shown in Fig. 5, with the temporal varying pose during the exposure, the relative pose between $\mathbf{q}_1^{\mathrm{RS}}$ and $\mathbf{q}_2^{\mathrm{RS}}$ becomes:

$$\begin{aligned} \mathbf{r}_{1\to2}^{\mathrm{RS}} &= (1 + \frac{\gamma}{h}(y_2^{\mathrm{RS}} - y_1^{\mathrm{RS}}))\mathbf{r}, \\ \mathbf{t}_{1\to2}^{\mathrm{RS}} &= (1 + \frac{\gamma}{h}(y_2^{\mathrm{RS}} - y_1^{\mathrm{RS}}))\mathbf{t}, \end{aligned} \tag{3}$$

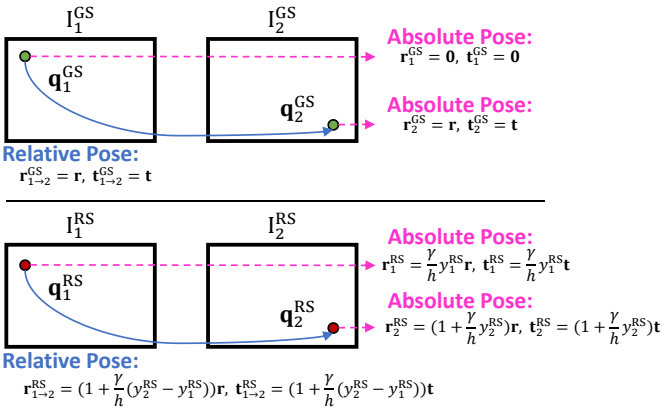

Fig. 5: Configurations of instantaneous GS optical flow [35] (top) and instantaneous RS optical flow [18] (bottom).

where h is the normalized image height, and γ is the readout time ratio. By substituting Eq. (3) into Eq. (2), we obtain the constraint of homogeneous RS optical flow $\mathbf{u}^{\mathrm{RS}}$

$$\frac{\mathbf{u}^{\mathrm{RS}\top}}{1 + \frac{\gamma}{h}(y_2^{\mathrm{RS}} - y_1^{\mathrm{RS}})}[\mathbf{t}]_\times \mathbf{q}_1^{\mathrm{RS}} - \mathbf{q}_1^{\mathrm{RS}\top}\mathbf{s}\mathbf{q}_1^{\mathrm{RS}} = 0, \tag{4}$$

where $\mathbf{s}$ is defined in Eq. (2a). The rolling shutter homogeneous optical flow vector $\mathbf{u}^{\mathrm{RS}}$, $\mathbf{u}^{\mathrm{RS}} \in \mathbb{R}^3$ and $\mathbf{u}_{(3)}^{\mathrm{RS}} = 0$, indicates the displacement of a pixel over two consecutive RS frames, which is equal to $\mathbf{q}_2^{\mathrm{RS}} - \mathbf{q}_1^{\mathrm{RS}}$. Removing the common factor of $\mathbf{t}$ and $\mathbf{r}$ in Eq. (2) and (4), we have the following Lemma illustrating the relation between $\mathbf{u}^{\mathrm{GS}}$ and $\mathbf{u}^{\mathrm{RS}}$.

Lemma 1. *The homogeneous RS optical flow vector* $\mathbf{u}^{\mathrm{RS}}$ *when scaled by dimensionless scaling factor* $1 + \frac{\gamma}{h}(y_2^{\mathrm{RS}} - y_1^{\mathrm{RS}})$ *is equivalent to the homogeneous GS optical flow vector* $\mathbf{u}^{\mathrm{GS}}$:

$$\mathbf{u}^{\mathrm{GS}} = \frac{1}{1 + \frac{\gamma}{h}(y_2^{\mathrm{RS}} - y_1^{\mathrm{RS}})}\mathbf{u}^{\mathrm{RS}}. \tag{5}$$

□

3 METHODOLOGY

By Lemma 1, the work of [18] rescales all $\mathbf{u}^{\mathrm{RS}}$ vectors followed by the conventional GS-based linear 8pt algorithm [35] to recover the instantaneous motion $\mathbf{r}$ and $\mathbf{t}$ and depth. Finally, RSC can be done jointly using camera motion and depth variation in a static scene. However, we found out that such a solution does not hold well in computing the depth map since with a short baseline and is time-consuming with RANSAC filtering and nonlinear refinement.

In contrast, our insight is to correct an RS projection $\mathbf{q}_1^{\mathrm{RS}}$ directly by using its optical flow $\mathbf{u}^{\mathrm{RS}}$ without the need to compute the camera instantaneous motion and depth as an intermediate product or any joint process (*e.g.*, RANSAC).

In this section, we first present the pixel-wise pose-free RSC solver in calibrated case (Sec. 3.1.2) and then show that it still holds in the uncalibrated case with both perspective (Sec. 3.1.3) and non-perspective projection (Sec. 3.1.4) models. In Sec. 3.2, we introduce our direct RSC framework (DRSC) based on pixel-wise pose-free RSC solver for image

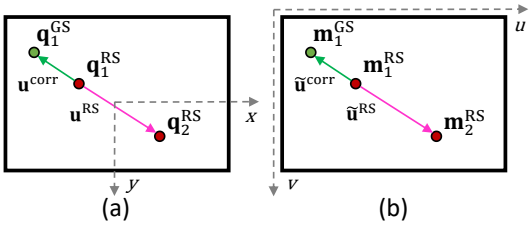

Fig. 6: Two examples of proposed pixel-wise pose-free RSC solver (a) on the normalized image plane (Proposition 1) and (b) pixel measurement plane (Corollary 1 and Corollary 2).

correction (Sec. 3.2.1) and keypoint correction (Sec. 3.2.2) respectively, and followed by its error analysis details in Sec. 4.1.

3.1 Pixel-wise pose-free RSC Solver

3.1.1 Correction Vector

In order to correct RS projection $\mathbf{q}_1^{RS}$ back to GS projection $\mathbf{q}_1^{GS}$ as Fig. 6(a), we model the homogeneous correction vector $\mathbf{u}^{corr} = \mathbf{q}_1^{GS} - \mathbf{q}_1^{RS}$. As shown in Fig. 5, all the points in I_1^{GS} share the same pose as $\mathbf{r}_1^{GS} = \mathbf{0}$ and $\mathbf{t}_1^{GS} = \mathbf{0}$. While the pose of $\mathbf{q}_1^{RS}$ in I_1^{RS} is

$$\mathbf{r}_1^{RS} = \frac{\gamma y_1^{RS}}{h}\mathbf{r}, \quad \mathbf{t}_1^{RS} = \frac{\gamma y_1^{RS}}{h}\mathbf{t}, \tag{6}$$

where the pose of scanline y_1^{RS} is a linear scaling of $\mathbf{r}$ and $\mathbf{t}$. Thus, we have the following Lemma.

Lemma 2. *The GS optical flow vector $\mathbf{u}^{GS}$ when scaled by dimensionless scaling factor $-\frac{\gamma y_1^{RS}}{h}$ is equivalent to the correction vector $\mathbf{u}^{corr}$:*

$$\mathbf{u}^{corr} = -\frac{\gamma y_1^{RS}}{h}\mathbf{u}^{GS}. \tag{7}$$

Proof. Since $\mathbf{r}_1^{GS} = \mathbf{0}$ and $\mathbf{t}_1^{GS} = \mathbf{0}$, the relative pose from $\mathbf{q}_1^{RS}$ to $\mathbf{q}_1^{GS}$ is the same as Eq. (6). By substituting $\mathbf{r}_1^{RS}$, $\mathbf{t}_1^{RS}$ into Eq. (1)

$$\begin{aligned}\mathbf{u}_{(1:2)}^{corr} &= \frac{\mathbf{A}(x_1^{GS}, y_1^{GS})(\mathbf{t}_1^{GS} - \mathbf{t}_1^{RS})}{Z} + \mathbf{B}(x_1^{GS}, y_1^{GS})(\mathbf{r}_1^{GS} - \mathbf{r}_1^{RS}) \\ &= -\frac{\gamma y_1^{RS}}{h}\left(\frac{\mathbf{A}(x_1^{GS}, y_1^{GS})\mathbf{t}}{Z} + \mathbf{B}(x_1^{GS}, y_1^{GS})\mathbf{r}\right) \\ &= -\frac{\gamma y_1^{RS}}{h}\mathbf{u}_{(1:2)}^{GS}\end{aligned} \tag{8}$$

where $\mathbf{A}$ and $\mathbf{B}$ are defined in Eq. (1a) and (1b), respectively. In addition, $\mathbf{u}^{corr}$ is the correction homogeneous optical flow vector, where $\mathbf{u}^{corr} \in \mathbb{R}^3$ and $\mathbf{u}_{(3)}^{corr} = 0$. □

Lemma 2 describes the relationship between $\mathbf{u}^{corr}$ and $\mathbf{u}^{GS}$, namely, they are equivalent by a dimensionless scaling factor $-\frac{\gamma y_1^{RS}}{h}$.

3.1.2 Pixel-wise pose-free RSC Solver to Calibrated Image

Note that we aim to obtain the correction vector $\mathbf{u}^{corr}$ as Fig. 6(b) by using $\mathbf{u}^{RS}$ directly without computing the depth or instantaneous motion in [18]. To this end, we introduce the following proposition.

Proposition 1. *Given two consecutive RS images I_1^{RS} and I_2^{RS}, the RS image point $\mathbf{q}_1^{RS}$ can be corrected to the corresponding GS image point $\mathbf{q}_1^{GS}$ by following transform:*

$$\mathbf{q}_1^{GS} = \mathbf{q}_1^{RS} - \frac{y_1^{RS}}{\frac{h}{\gamma} + (y_2^{RS} - y_1^{RS})}\mathbf{u}^{RS}. \tag{9}$$

Proof. By Lemma 1 and Lemma 2, we substitute $\mathbf{u}^{corr}$ from Eq. (7) and $\mathbf{u}^{GS}$ from Eq. (5) cascadingly into $\mathbf{q}_1^{GS} = \mathbf{q}_1^{RS} + \mathbf{u}^{corr}$, we obtain:

$$\begin{aligned}\mathbf{q}_1^{GS} &= \mathbf{q}_1^{RS} + \mathbf{u}^{corr} = \mathbf{q}_1^{RS} - \frac{\gamma y_1^{RS}}{h}\mathbf{u}^{GS} \\ &= \mathbf{q}_1^{RS} - \frac{\frac{\gamma y_1^{RS}}{h}}{1 + \frac{\gamma}{h}(y_2^{RS} - y_1^{RS})}\mathbf{u}^{RS} \\ &= \mathbf{q}_1^{RS} - \frac{y_1^{RS}}{\frac{h}{\gamma} + (y_2^{RS} - y_1^{RS})}\mathbf{u}^{RS}.\end{aligned} \tag{10}$$

□

Note that given two consecutive RS images I_1^{RS} and I_2^{RS}, one can use pixel-wise pose-free RSC solver in Proposition 1 to correct the RS projection $\mathbf{q}_1^{RS}$ back to $\mathbf{q}_1^{GS}$, since y_1^{RS}, y_2^{RS} and $\mathbf{u}^{RS}$ are directly measured from I_1^{RS} and I_2^{RS}.

3.1.3 Pixel-wise pose-free RSC solver for uncalibrated perspective image

Notice that Proposition 1 models the correction procedure in the normalized image plane. For the uncalibrated perspective case, we assume the image measurements are projected by $\mathbf{m}_1^{GS} = [u_1^{GS}, v_1^{GS}, 1]^\top = \mathbf{K}\mathbf{q}_1^{GS}$, $\mathbf{m}_2^{GS} = [u_2^{GS}, v_2^{GS}, 1]^\top = \mathbf{K}\mathbf{q}_2^{GS}$, $\mathbf{m}_1^{RS} = [u_1^{RS}, v_1^{RS}, 1]^\top = \mathbf{K}\mathbf{q}_1^{RS}$, and $\mathbf{m}_2^{RS} = [u_2^{RS}, v_2^{RS}, 1]^\top = \mathbf{K}\mathbf{q}_2^{RS}$ with intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

where f_x and f_y are focal lengths, u_x, u_y and s are the principal point offset and skew coefficient. With such a setting, we present the following corollary of Proposition 1.

Corollary 1. *Given two consecutive RS images I_1^{RS} and I_2^{RS} with a perspective projection model, the RS image point $\mathbf{m}_1^{RS}$ can be corrected to the corresponding GS image measurement $\mathbf{m}_1^{GS}$ by the following transform:*

$$\mathbf{m}_1^{GS} = \mathbf{m}_1^{RS} - \frac{v_1^{RS}}{\frac{\tilde{h}}{\gamma} + (v_2^{RS} - v_1^{RS})}\tilde{\mathbf{u}}^{RS}, \tag{12}$$

where $\tilde{\mathbf{u}}^{RS}$ indicates the RS optical flow in the image pixel level system, and $\tilde{h} = f_y h$ is the image height.

Proof. By substituting the definition of $\mathbf{K}$ in Eq. (11) into Eq. (5) and Eq. (7), Lemma 1 and Lemma 2 still hold for the image measurement level as:

$$\tilde{\mathbf{u}}^{GS} = \frac{1}{1 + \frac{\gamma}{\tilde{h}}(v_2^{RS} - v_1^{RS})}\tilde{\mathbf{u}}^{RS}, \tag{5a}$$

$$\tilde{\mathbf{u}}^{corr} = -\frac{\gamma v_1^{RS}}{\tilde{h}}\tilde{\mathbf{u}}^{GS}, \tag{7a}$$

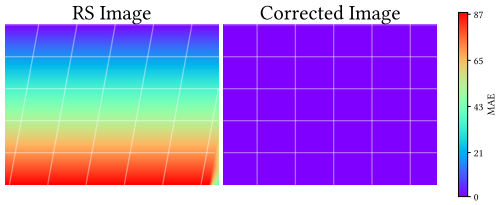

Fig. 7: Example of pixel-wise pose-free RSC with perspective camera model (Corollary 1). The perspective warp of the input RS image (left) has been removed, yielding the corrected image (right).

where the dimensionless scaling factors follow the same patterns as those in Eq. (5) and (7). By recursively substituting $\tilde{\mathbf{u}}^{\text{corr}}$ from (7a) and $\tilde{\mathbf{u}}^{\text{GS}}$ from (5a) into $\mathbf{m}_1^{\text{GS}} = \mathbf{m}_1^{\text{RS}} + \tilde{\mathbf{u}}^{\text{corr}}$, we obtain Corollary 1. □

Thus as shown in Fig. 7, which provides a visual example from Sec. 4.2.1, by giving two consecutive RS images with the perspective projection assumption, one can use the pixel-wise pose-free RSC solver in Corollary 1 to correct the RS projection $\mathbf{m}_1^{\text{RS}}$ back to $\mathbf{m}_1^{\text{GS}}$ using its optical flow vector. Note that the $\tilde{h}$ can be measured directly from the input images, while γ are obtained from pre-calibration or manually set as a fixed number (detailed discussion in Sec. 4.2.6).

3.1.4 Pixel-wise pose-free RSC Solver for Uncalibrated Non-perspective Image

To the best of our knowledge, RSC to uncalibrated non-perspective images has never been addressed in the existing literature. We point out that the proposed pixel-wise pose-free RSC solver also holds in such cases. We model the image measurements with the one parameter additive radial distortion [36], [37] as:

$$\begin{aligned} \mathbf{m}_1^{\text{RS}} &= \mathbf{K} \begin{bmatrix} x_1^{\text{RS}}(1+k_1 r_1^2) \\ y_1^{\text{RS}}(1+k_1 r_1^2) \\ 1 \end{bmatrix}, \\ \mathbf{m}_2^{\text{RS}} &= \mathbf{K} \begin{bmatrix} x_2^{\text{RS}}(1+k_1 r_2^2) \\ y_2^{\text{RS}}(1+k_1 r_2^2) \\ 1 \end{bmatrix}, \\ \text{with,} \quad r_1^2 &= (x_1^{\text{RS}})^2 + (y_1^{\text{RS}})^2, \quad r_2^2 = (x_2^{\text{RS}})^2 + (y_2^{\text{RS}})^2, \end{aligned} \tag{13}$$

where k_1 is a radial distortion coefficient of the lens. Note that we use a single coefficient of the quadratic term only. With such a setting, we present the following corollary of Proposition 1:

Corollary 2. *Given two consecutive RS images* I_1^{RS} and I_2^{RS} *with a non-perspective projection model, the RS image point* $\mathbf{m}_1^{\text{RS}}$ *can be corrected to the corresponding GS image measurement* $\mathbf{m}_1^{\text{GS}}$ *by the following transform:*

$$\mathbf{m}_1^{\text{GS}} = \mathbf{m}_1^{\text{RS}} - \frac{v_1^{\text{RS}}}{\frac{\tilde{h}}{\gamma} + (v_2^{\text{RS}} - v_1^{\text{RS}})} \tilde{\mathbf{u}}^{\text{RS}}, \tag{14}$$

where $\tilde{h} = f_y(h + k_1 h^2)$.

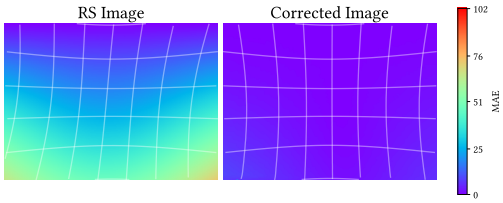

Fig. 8: Example of pixel-wise pose-free RSC with non-perspective camera model (Corollary 2). The distortion of the input RS image (left) is removed, yielding the corrected image (right).

Proof. By substituting the distortion model in Eq. (13) into Eq. (5) and Eq. (7), Lemma 1 and Lemma 2 still hold for the image measurement level as:

$$\tilde{\mathbf{u}}^{\text{GS}} = \frac{1}{1 + \frac{\gamma}{\tilde{h}}(v_2^{\text{RS}} - v_1^{\text{RS}})} \tilde{\mathbf{u}}^{\text{RS}}, \tag{5b}$$

$$\tilde{\mathbf{u}}^{\text{corr}} = -\frac{\gamma v_1^{\text{RS}}}{\tilde{h}} \tilde{\mathbf{u}}^{\text{GS}}, \tag{7b}$$

subject to two approximations $r_1^2 \approx r_2^2$ and $1 + k_1 h^2 \approx 1 + k_1 r_1^2$. The dimensionless scaling factors stay the same patterns as those in Eq. (5) and (7). Now, by recursively substituting $\tilde{\mathbf{u}}^{\text{corr}}$ from (7a) and $\tilde{\mathbf{u}}^{\text{GS}}$ from (5a) into $\mathbf{m}_1^{\text{GS}} = \mathbf{m}_1^{\text{RS}} + \tilde{\mathbf{u}}^{\text{corr}}$, we obtain Corollary 2. □

By following the experiment setup and evaluation in Sec. 4.1 and Eq. 15, we evaluated the performance of pixel-wise pose-free RSC solver in a simulated dynamic scene with lens distortion by increasing the distortion coefficient k1 (b) from 0 to 0.90, which is rare in real applications. As shown in Fig. 11(b), pixel-wise pose-free RSC solver can provide RSC errors below 10px even with 0.90 distortion coefficient k1. This strongly proves the effectiveness of the single-pixel pose-free solver correction when dealing with non-perspective projection. A visualized example of RSC on the non-perspective camera model is shown in Fig. 8, the pixel-wise pose-free RSC solver still holds with RS images with non-perspective projection assumption. However, once the camera has high values of high-order radial distortion terms or decentering distortion terms, the two approximations used in Corollary 2 may fail, leading to lower performance in RS correction.

3.1.5 Degeneracy Analysis

Note that some degenerate cases have been reported in existing optical flow-based works [18], [28], [35] that multiple combinations of camera motion and scene can explain a group of optical flows. Although such degeneracy in the optical flow-based methods is rarely raised in real applications, they still bring potential risks in recovering the camera motion and leading to the failure of RSC.

In contrast, the proposed pixel-wise pose-free RSC solver corrects the RS projections directly in the image without estimating camera motion or depth. Therefore, it has no known degenerate solution as optical flow-based motion recovery algorithms using epipolar [18] or homography [28] geometry.

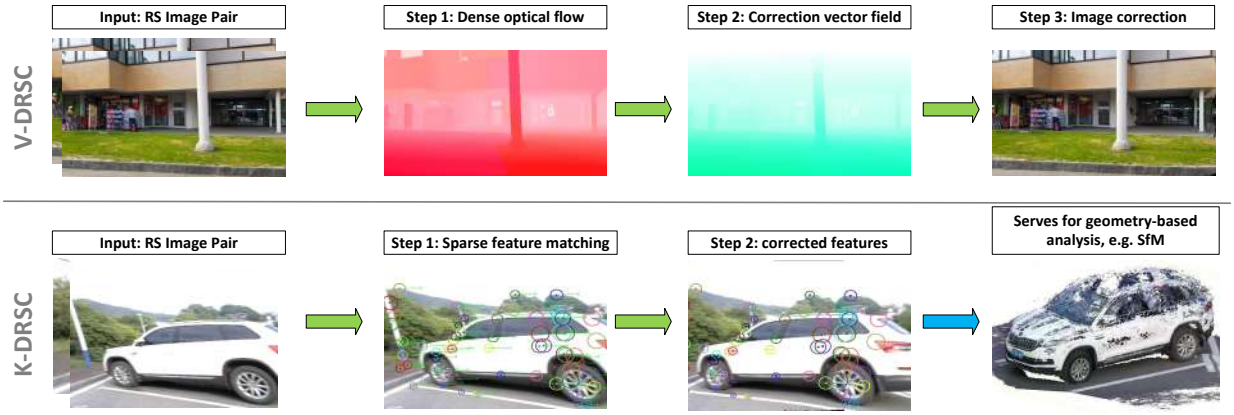

Fig. 9: Pipelines of the proposed two types of DRSC, namely, visual correction one as V-DRSC (top) and keypoint correction one as K-DRSC (bottom).

3.2 Direct RS Correction (DRSC)

As shown in Fig. 9, we present a direct RS correction (DRSC) framework for various applications using the proposed pixel-wise pose-free RSC solver:

- **Visual correction (V-DRSC).** By computing the dense optical flow over video, we can correct each frame with high quality and efficiency.
- **Keypoint correction (K-DRSC).** Towards 3D analysis, we only correct tracked keypoints instead of correcting the whole frame pixels, which allows argument GS SLAM/VO solutions to handle RS input in real-time.

3.2.1 V-DRSC

In this mode, we visually correct the distortion of input RS video in 3 steps:

Step 1: *Dense optical flow estimation.* To efficiently and robustly correct the whole RS frame with our pixel-wise pose-free RSC solver, the key is to obtain the accurate dense optical flow as input in a short time. Since dense optical flow estimation is a well-studied topic in the Computer Vision community, we choose the sparse pyramidal KLT to track features [38] and then interpolate the per-pixel optical flow using edge-preserving scheme [39] [1].

Step 2: *Correction vector field estimation.* With the obtained dense optical flow $\tilde{\mathbf{u}}^{\text{RS}}$, we can compute the correction vector field $\tilde{\mathbf{u}}^{\text{corr}}$ by applying Eq. (12) for both perspective and non-perspective camera based on Corollary 1 and 2.

Step 3: *Image correction.* In the final step, we apply a generic geometrical transformation to the current RS frame $\mathbf{I}^{\text{RS}}$ with correction vector field $\tilde{\mathbf{u}}^{\text{corr}}$ as displacement map [2]. Then a bilinear interpolation is applied to handle the corrected pixels with non-integer coordinates.

3.2.2 K-DRSC

In this mode, we correct sparse features only instead of every pixel as a pre-processing step for the downstream geometry-based 3D analysis, such as SfM, VO, and SLAM. In this paper, we use ORB-SLAM2 [33] as an example to illustrate how to use K-DRSC to argue classical GS VO to handle RS input. It is important to note that K-DRSC can also be easily applied to other 3D vision solutions. K-DRSC has two main steps:

Step 1: *Features matching.* Since the tracking thread in ORB-SLAM2 extracts ORB features [41] of every current frame and matches to the map points observed in the last frame. This mechanism directly provides sparse optical flow vectors $\tilde{\mathbf{u}}^{\text{corr}}$ between every two consecutive frames.

Step 2: *Output corrected features.* We update the coordinates of every matched feature on the current frame by performing pixel-wise pose-free RSC solvers independently based on Eq. (12). Note that the correction is directly on the image pixel level and is feasible for both perspective and non-perspective cases. Then the tracking thread continues the procedure to optimize the camera pose using the corrected 2D ORB feature. Thus we correct the detected features on every frame instead of processing the keyframes only to improve the quality of pose estimation and point reconstruction.

3.3 Limitation

The proposed method cannot be applied to all RSC scenarios. As with most other RSC works, it comes with some limitations:

- We focus on the RSC with video as input. Thus the proposed method can not rectify a single RS image alone. This limitation is shared with most state-of-the-art RSC solutions [16], [18], [22], [23], [24], [27], [28], [32]. But it is important to note that the existing single image base RSC works commonly suffer from strong assumptions on camera motion (*e.g.*, rotate-only [12], [20]) or 3D scene (*e.g.*, Manhattan world [11], [12], [29]) while the video-based approaches relax these restrictive assumptions.
- For visual correction, the proposed V-RSDC requires dense matching between two consecutive frames. However, we perform a backward-forward checking to filter the mismatching, but it still leads to risk in correcting close-depth RS projections with high-speed instantaneous motion. This drawback may be avoided by applying the Kalman filter over the whole RS video. Note that [18] also requires dense matching for depth map generation.
- For visual correction, V-RSDC does not handle the occlusion issue. However, with the report in [18] and our observations in the experiment that the correction vectors

[1] *calcOpticalFlowSparseToDense* function in opencv [40]
[2] *remap* function in opencv [40]

Fig. 10: (a) Input RS image has occlusion with ground-truth GS. (b) Corrected result by SOTA RSC method CVR [26] which can reason about occlusions. (c) Corrected result by proposed V-DRSC.

between two consecutive frames are small, and the ghosting artifacts caused by the mismatching can be negligible. Thus, we do not apply the gap-filling by copying the pixel from the next frame as [27]. As shown in Fig. 10, the occlusion leads to local blur after performing V-RSDC but is visually negligible and surprisingly better than the state-of-the-art RSC method CVR [26] which is designed to handle such occlusion.

4 EXPERIMENTS

Comparison methods. In our experiments, the proposed method *V-DRSC* and *K-DRSC* were compared to four state-of-the-art RS correction solutions and one famous commercial application:

- ***DSfM***: Differential RS-SfM based RSC method [18]. Specifically, we use the 4pt linear solver with the constant velocity model in synthetic experiments, and the 9pt linear solver with the constant acceleration model in real image experiment[3]. Note that *DSfM* requires pre-calibration. Thus we use the ground-truth information as input if it is available. Otherwise, we use rough estimates of intrinsic for the uncalibrated cases.
- ***Adobe Pr***: Premiere Pro (Pr) [25] is a well-known professional video editing software developed by Adobe. In the experiment, we use the RS repair function in *Adobe Pr*[4]. Specifically, we use the 'Pixel-Motion' method, which experimentally outperforms other selections.
- ***DSUN***: Learning-based two-view RSC method [22][5].
- ***JCD***: Learning-based two-view RSC and deblurring approach [23] [6].
- ***SUNet***: Learning-based two-view RSC and interpolation method [24][7].
- ***CVR***: Learning-based context-aware two-view RSC solution [26][8].

4.1 Validating Modeling Assumption

In this section, we analyze three main error sources of the proposed RSC method and experimentally evaluate the MAE error in Eq. (15) of DRSC with increasing rotation speed, distortion coefficient $k1$, rotation acceleration, and translation acceleration, respectively.

[3]https://github.com/ThomasZiegler/RS-aware-differential-SfM
[4]https://helpx.adobe.com/premiere-pro/using/rolling-shutter-repair.html
[5]https://github.com/ethliup/DeepUnrollNet
[6]https://github.com/zzh-tech/RSCD
[7]https://github.com/GitCVfb/SUNet
[8]https://github.com/GitCVfb/CVR

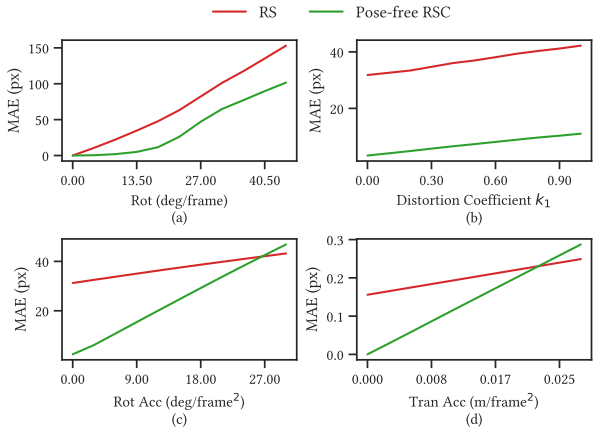

Fig. 11: Errors of the input RS image before and after correction by pixel-wise pose-free RSC with increasing rotation speed (a), distortion coefficient k_1 (b), rotational acceleration (c), and translation acceleration (d).

Simulation Setup: We generated a cube scene with 602 feature points, which was imaged in two RS consecutive frames under constant velocity with 480×640px image resolution and 320px focal length. Since we obtain matched features between two views, we can only evaluate *K-DRSC* to *DSfM* in this experiment. We compared the two approaches by varying the motion speed, depth of the 3D points, camera scanning speed, noise on image measurements, non-rigidity of the scene, and hyper-parameter setting. To simulate the noise, we directly added random Gaussian noise to the image measurements (pixel-level) in the experiments. The default setting is 15 deg/frame and 2.4 unit/frame for rotational and translational speed, 10m depth, 7.1510^{-5}seconds/row for scanning speed, 1.5px noise, 0% dynamic 3D feature point, and 0.9 for readout ratio.

Evaluation Metrics: Since ground truths are available at each configuration, we evaluated RSC accuracy by running 100 trials and computing the mean absolute error (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{m}_{1,i}^{\text{GS}} - \bar{\mathbf{m}}_{1,i}^{\text{GS}}\right\|, \tag{15}$$

where n is the number of features. $\bar{\mathbf{m}}_{1,i}^{\text{GS}}$ is the coordinates of i^{th} corrected projection in the first image while $\mathbf{m}_{1,i}^{\text{GS}}$ is its corresponding ground truth GS projection.

4.1.1 *instantaneous motion Modelling*

The proposed RSC method is based on the optical flow modelling between two consecutive RS frames based on a small rotation approximation and constant velocity assumption.

1) Small rotation approximation. The rotation parametrization used in the works of [18], [35] and this paper is defined as $\mathbf{R} = \text{Exp}(\mathbf{r}) \approx \mathbf{I} + [\mathbf{r}]_\times$, which is a linearization of Rodrigues rotation formula by using first-order Taylor expansion around the rotation of I_1^{RS}. There is a gap between this approximation and a more accurate rotation such as SLERP [42]. However, note that such approximation holds when the relative rotation is small between two consecutive frames. This simplification

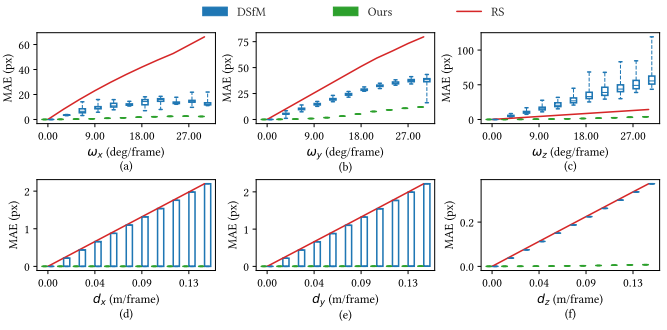

Fig. 12: Correction errors for *DSfM* [18] and *K-DRSC* with increasing rotational (a)(b)(c) and translation speed (d)(e)(f).

has been widely used in existing RS vision works [3], [18], [28], [29], [43], [44], and is a good compromise between precision and model complexity. As shown in Fig. 11(a), the proposed pixel-wise pose-free RSC solver with approximation can reduce the RS distortion caused by camera rotation with precise Rodrigues rotation from 0 deg/frame to 45 deg/frame speed.

2) Constant velocity vs acceleration. The works of [12], [18], [28] argue that the constant velocity assumption can be too restrictive for hand-held devices shaking and propose considering constant acceleration to enhance the generality of the motion model. As shown in Fig. 11(c)(d), the proposed DRSC solution can keep correcting RS projections back to the GS projection up to 25 deg/frame^2 and $0.022\text{m/frame}^2 \approx 17\text{m/s}^2$ (a scene with average 1m depth) which is rarely reached in real applications. We interpret this observation as the instantaneous motion with significant acceleration can be approximated as constant velocity motion between two consecutive frames.

4.1.2 Constant Depth Assumption

Notice that the optical flow models [18], [35] assume the scene depths are the same in the first and second frames, and so does the proposed pixel-wise pose-free RSC solver. This constant depth assumption may deviate from reality under certain instantaneous motion types. This section analyzes how this gap affects the accuracy of the proposed pixel-wise pose-free RSC solver. We point out that the error comes from the fact that the optical flow $\tilde{\mathbf{u}}^{\text{RS}}$ used in Proposition 1 and its corollaries are based on the constant depth assumption, while in the real application, the optical flow vectors are measured as $\mathbf{m}_2^{\text{RS}} - \mathbf{m}_1^{\text{RS}}$ with varying depth in two consecutive frames. Here, we model the error (in pixel) between ground truth GS projection $\bar{\mathbf{m}}_1^{\text{GS}}$ and $\mathbf{m}_1^{\text{GS}}$ corrected by our pixel-wise pose-free RSC solver using Eq. (12) under different types of instantaneous motion as:

$$\left\|\bar{\mathbf{m}}_1^{\text{GS}} - \mathbf{m}_1^{\text{GS}}\right\| = \left\|\bar{\mathbf{m}}_1^{\text{GS}} - \mathbf{m}_1^{\text{RS}} + \frac{v_1^{\text{RS}}(\mathbf{m}_2^{\text{RS}} - \mathbf{m}_1^{\text{RS}})}{\frac{h}{\gamma} + (v_2^{\text{RS}} - v_1^{\text{RS}})}\right\|, \quad (16)$$

$$\bar{\mathbf{m}}_1^{\text{GS}} = \frac{1}{Z_1^{\text{GS}}}\mathbf{K}(\mathbf{IP} + \mathbf{0}), \quad (16a)$$

$$\mathbf{m}_1^{\text{RS}} = \frac{1}{Z_1^{\text{RS}}}\mathbf{K}((\mathbf{I} + v_1^{\text{RS}}[\boldsymbol{\omega}]_\times)\mathbf{P} + v_1^{\text{RS}}\mathbf{d}), \quad (16b)$$

where Z_1^{GS} and Z_1^{RS} respectively denote the depth of GS and RS pixel. The instantaneous motion $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$ and $\mathbf{d} = [d, d_y, d_z]^\top$ has 6 types of atomic motion, namely ω_x, ω_y, ω_z, d_x, d_y and d_z. With these atomic motions, the correction error becomes:

- Substituting $\omega_z \neq 0$ and the others $= 0$, $d_x \neq 0$ and the others $= 0$ or $d_y \neq 0$ and the others $= 0$ into Eq. (16a) and (16b), the correction error becomes 0 (proof in the supplemental material), which means that the instantaneous motion constituted by these three types introduce no error in proposed pixel-wise pose-free RSC solver.
- Substituting $\omega_x \neq 0$, $\omega_y \neq 0$ and $d_z \neq 0$ into Eq. (16a) and (16b) leads to non-zero error (proof in the supplemental material). Thus, the correction accuracy of DRSC will decrease with increasing speed. However, existing work [35] and our experiments show that the depth variation between two consecutive frames is small and can be ignored. Thus, the correction error caused by deviation from constant depth is negligible.

The experiment results shown in Fig. 12 verify our above theoretical analysis that pixel-wise pose-free RSC solver leads to no error with the atomic motions along ω_z, d_x, d_y. While the correction error of the pose-free RSC solver grows with the increasing speed along ω_x, ω_y, d_z but is still able to reduce 90% RS effect error even up to 25 deg/frame and 3.5 m/frame.

4.2 Synthetic Data

4.2.1 Effect of instantaneous motion

We evaluated the robustness of the two methods against six atomic kinematics types, namely ω_x, ω_y, ω_z, d_x, d_y, d_z, with increasing rotational and translation speed from 0 to 30 deg/frame and 4 units/frame gradually. The results in Fig. 12 show *DSfM* provides satisfying corrections with slow kinematics but drop dramatically beyond 9 deg/frame or 1 unit/frame. We attribute the performance gap between *DRSC* and *DSfM* to the fact that DRSC is a pixel-wise pose-free method that warps each pixel directly on the image, while DSfM requires an accurate estimation of the camera motion and depth for each pixel before correction. Thus, even small recovered errors among the mid-product $\mathbf{r}$, $\mathbf{t}$ and $\mathbf{Z}$ by *DSfM* could lead to bad corrections. For instance, we observe that *DSfM* often predicts a large depth for each feature with pure translation and leads to almost no correction (Fig. 12 (d)(e)(f)). In contrast, *K-DRSC* can provide stable and accurate corrections under all configurations.

4.2.2 Effect of Depth

In this experiment, we vary the average depth of 3D features from 1 to 100 m. The results in Fig. 13(a) show that both *DSfM* and *K-DRSC* perform better with large depth. However, *K-DRSC* outperforms *DSfM* significantly under small depth. This is easily understood because any movement generally induces an optical flow of magnitude inversely proportional to the depth (except for pure rotations).

4.2.3 Effect of Scanning Speed

We evaluate the robustness of the two methods with increasing camera scanning speed. The results in Fig. 13(b)

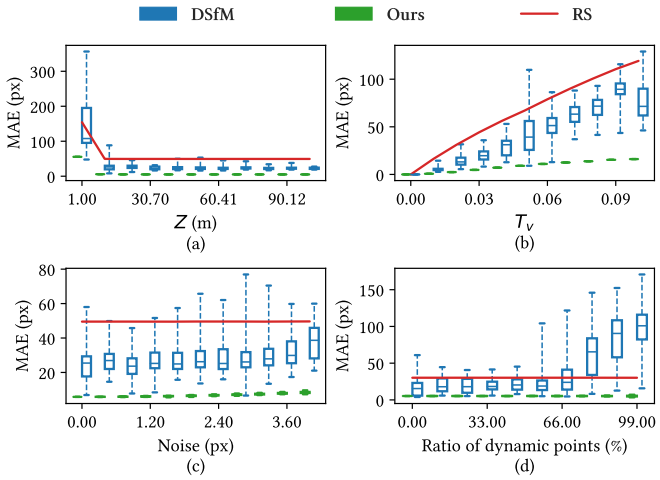

Fig. 13: Correction errors for *DSfM* [18] and *K-DRSC* under different feature depth (a), camera scanning period T_v (b), noise levels in the image (c), and the ratio of dynamic 3D feature (d).

show that the RSC errors of *DSfM* and *K-DRSC* increase linearly. Therefore, the slower the scanning speed (t_v), the larger RSC error due to the violation of constant depth and small motion assumptions. However, the RSC errors of *K-DRSC* rise slightly from 0 to 16px in the interval from 0 to 0.1 seconds/frame, while *DSfM* increases dramatically and closing RS projection errors.

4.2.4 Effect of Image Noise

In Fig. 13(c), we observe that the errors for all methods increase linearly when noise varies from 0 to 4 pixels. However, *K-DRSC* shows a significantly better tolerance to noise than *DSfM*.

4.2.5 Effect of Non-rigidity

We evaluated the performance of *DSfM* and *K-DRSC* in a simulated dynamic scene with varying non-rigidity by increasing the ratio of randomly moving 3D features from 0% to 100%. As shown in Fig. 13(d) that *DSfM*, which assumes a static scene, fails with a large ratio of dynamic features. Thanks to the pixel-wise pose-free RSC solver, the proposed *K-DRSC* corrects each feature independently without rigidity assumption. Therefore, *K-DRSC* can provide RSC errors below $1px$ even with 100% dynamic features.

4.2.6 Hyper-parameter γ Sensitivity

Notice that Proposition 1 and its corollaries require readout ratio γ as input. Hence, we evaluate the sensitivities of the proposed method and DSfM to hyper-parameter γ. In the first experiment, we set the value of γ as ground truth in all configurations. As shown in Fig. 14(a), *K-DRSC* provides accurate correction with increasing γ and outperforms *DSfM* by a significant margin. In the second experiment, we evaluate the influence on the accuracy of the proposed pixel-wise pose-free RSC solver caused by the deviation between the setting value and the ground truth one. The value of γ is fixed to 0.9 in all configurations. The results in Fig. 14(b) show that both *DSfM* and *K-DRSC* remain stable with varying different camera readout ratios.

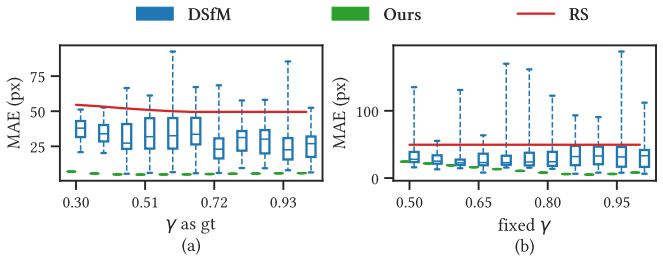

Fig. 14: Correction errors for *DSfM* [18] and *K-DRSC* with increasing camera readout ratio γ. As a hyper-parameter, we set γ as the ground truth value in (a), and the fixed value as 0.9 in (b).

K-DRSC provides MAE of RSC lower than 13px even with significant difference between ground truth $\gamma = 0.5$ and the fixed setting $\gamma = 0.9$.

4.3 RS Image Correction

Datasets. We compare all the RSC methods in the following publicly available RS datasets:

- ***Carla-RS and Fastec-RS***: Both datasets were published in [22] and contain ground truth GS images. Carla-RS dataset[9] is synthesized from a virtual 3D scene while Fastec-RS[10] is synthesized by high frame rate GS video.
- ***WHU***: WHU dataset[11] was published in [45] and provides ground truth GS images and camera poses.
- ***TUM***: The TUM RS dataset[12] was published in [46] and contains time-synchronized global-shutter and rolling-shutter images captured by non-perspective cameras and provides ground-truth poses for ten RS video sequences.
- ***GPark***: Strong RS video sequences from [7].
- ***Seq77***: Synthetic indoor RS video and provides ground-truth poses of each frame [47][13].
- ***3GS and House***: Rolling shutter rectification dataset in [19][14]. which provides synthetic RS images with ground truth GS images (House sequences), and real RS video captured by iPhone 3GS (3GS sequences).
- ***BS-RSC***: Real-world RSC dataset with various motions collected by a beam-splitter acquisition system [48][15].
- ***UCR***: We contribute the first synthetic RS dataset for dynamic scene UCR (Urban Crossroads Rolling Shutter Dataset) based on Kubric [49] that contains seven challenging sequences with various strong dynamic objects and camera motions[16].
- ***YouTube***: We collect a number of RS sequences, which contain non-perspective (fisheye camera) videos captured in highly dynamic scenes from YouTube.

[9]https://drive.google.com/file/d/15vXSX3g_STd6RPDWLg2sIn11mKH0sXxg/view
[10]https://drive.google.com/file/d/1gJoI7PSv7KEm2qb9-bt6hiyZ3PPlwEpd/view
[11]http://aric.whu.edu.cn/caolike/2019/11/05/the-whu-rsvi-dataset
[12]https://vision.in.tum.de/data/datasets/rolling-shutter-dataset
[13]https://cs.adelaide.edu.au/~jaehak/data/rolling_shutter
[14]https://www.cvl.isy.liu.se/research/datasets/rs-dataset
[15]https://github.com/ljzycmd/BSRSC
[16]https://github.com/DelinQu/Urban-Crossroads-Rolling-Shutter-Dataset

TABLE 2: Quantitative performance comparisons between our approach and the state-of-the-art methods. The Best and second results are shown in green and blue.

Method	PSNR ↑							SSIM ↑						
	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC
Fastec [22]	24.36	23.70	24.59	24.55	**29.66**	**30.43**	26.98	0.77	0.72	0.77	0.75	**0.87**	**0.88**	0.82
BS-RSC [48]	**32.24**	30.12	19.67	24.53	15.37	28.41	**34.53**	**0.93**	0.91	0.71	0.80	0.56	0.90	**0.95**
TUM [46]	25.28	**28.24**	24.36	25.81	22.59	27.22	**28.96**	0.86	**0.90**	0.85	0.85	0.80	**0.90**	**0.92**
UCR	**28.02**	24.92	21.92	22.85	17.70	24.86	**32.39**	**0.88**	**0.88**	0.81	0.81	0.67	**0.88**	**0.97**

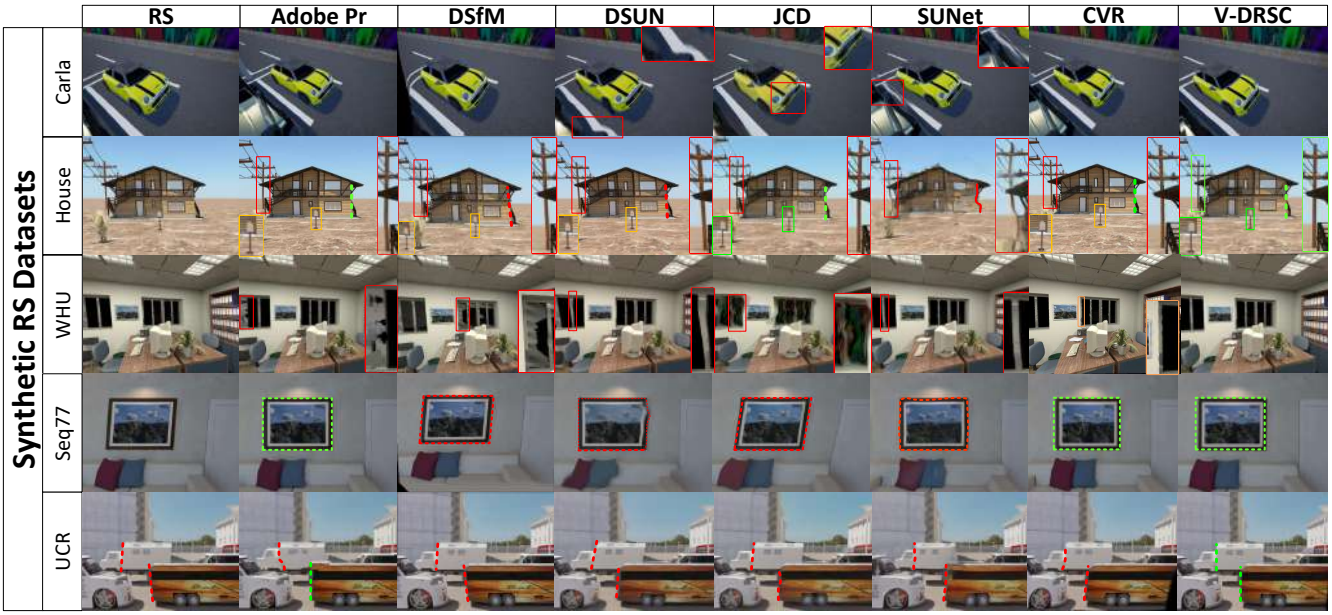

Fig. 15: Visual comparisons of our *V-DRSC* with existing RSC algorithms *Adobe Pr* [25], *DSfM* [18], *DSUN* [22], *JCD* [23], *SUNet* [24] in synthetic RS image datasets.

4.3.1 Quantitative Analysis

Evaluation Metrics. We use signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) metrics to report the quantitative results of our method: The larger PSNR and SSIM score, the higher quality of the corrected GS image.

Four datasets that provide both RS and ground truth GS images are used: Fastec, BS-RSC, TUM and UCR. We reported the median of PSNR and SSIM against the state-of-the-art methods in Tab. 2. Except for Fastec dataset *V-DRSC* achieves third performance. In the rest datasets, *V-DRSC* outperforms the other five methods by a significant margin. An interesting observation is that all the learning-based methods *DSUN*, *JCD*, *SUNet* and *CVR* fail in handling the distortion caused by RS effect and lens distortion simultaneously and provide obvious lower scores than geometry-based methods *DSfM* and *V-DRSC* in TUM dataset.

4.3.2 Visual Comparisons

We report the visual comparisons against the previous RSC methods in this experiment. For a more comprehensive evaluation, we divide the datasets into four categories: synthetic RS image scene, general scene, dynamic scene, and non-perspective scene.

Synthetic RS image. We first evaluate the correction results by all the methods on synthetic RS image datasets *Carla*, *House*, *WHU*, *Seq77* and *UCR*. As shown in Fig. 15, the learning-based approaches often produce artifacts, *e.g.*, significant local blur and distortion, while the geometry-based methods *DSfM* and *Adobe Pr* achieve better performance but fail in handling the depth-dependent RS distortions. For example, in *House* dataset, the electric pole and house facade have different depths and thus lead to distinct slopes on the image. However, all the previous methods cannot correct them back to vertical. In contrast, the proposed *V-DRSC* provides more clean and visually pleasant corrections in all sequences.

General scene. Then, we evaluate the correction results by all the methods on real RS image datasets captured in a general scene. As shown in Fig. 16, existing works fail in handling strong rotation (*e.g. 3GS*, *GPark*) and strong translation (*e.g. YouTube* and *BS-RSC*). Learning-based approaches produce local blur in *3GS* and *YouTube* sequences. In contrast, obvious skew, curvature, and depth-dependent distortions in these varied datasets are properly removed by our method.

Dynamic scene. Particularly, we collect a series of challenging RS videos that capture various moving objects in *Youtube* dataset to evaluate the performances of compared approaches in the dynamic scene. In the 'Cycling' and 'Bus' sequences, the static camera captured moving objects (*e.g.*, advancing bicycle or bus). As shown in Fig. 17 (1st and 2nd row), only the proposed method *V-DRSC* can rectify the moving bus back to the right position while the others either fail in correction or produce artifacts. More challenging scenes are in the 'Train at speed' sequences, where the camera moves while the captured objects are also under

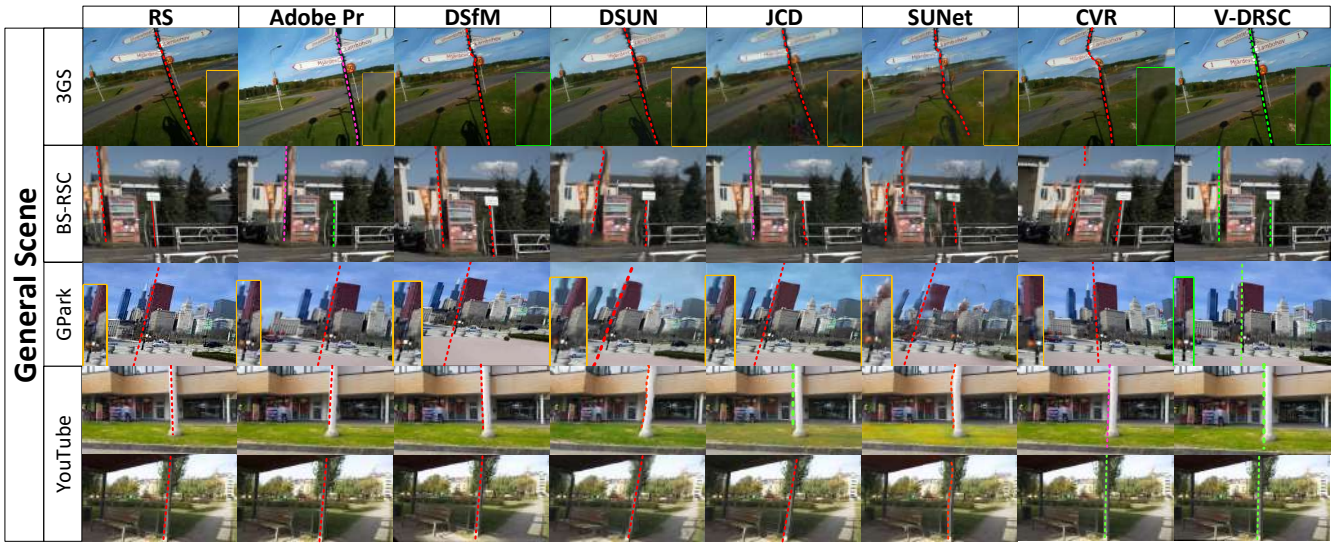

Fig. 16: Visual comparisons of our *V-DRSC* with existing RSC algorithms *Adobe Pr* [25], *DSfM* [18], *DSUN* [22], *JCD* [23], *SUNet* [24] in general scenes.

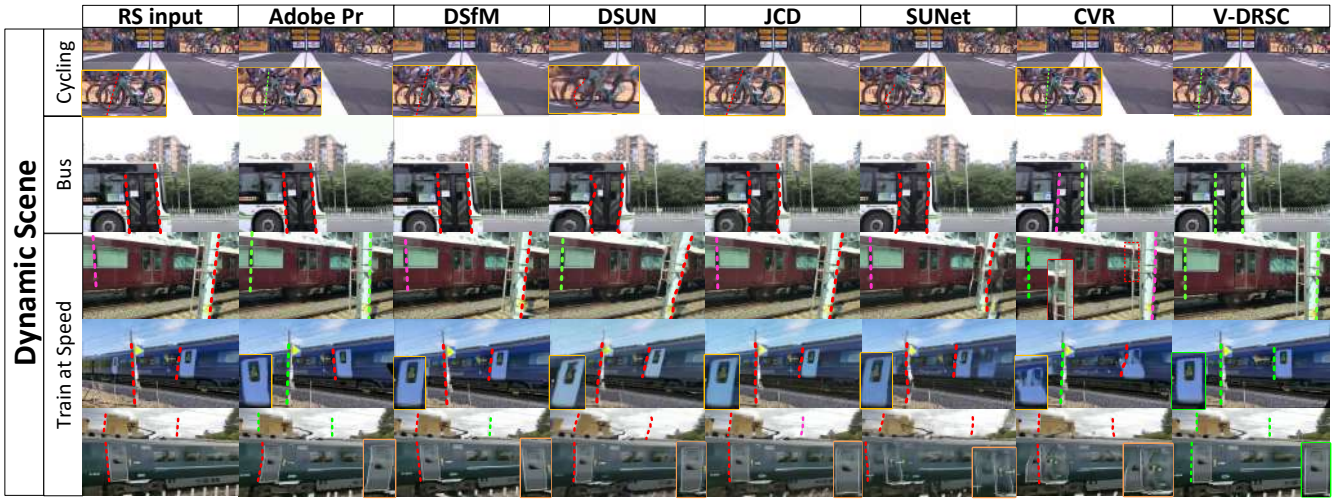

Fig. 17: Visual comparisons of our *V-DRSC* with existing RSC algorithms *Adobe Pr* [25], *DSfM* [18], *DSUN* [22], *JCD* [23], *SUNet* [24] in dynamic scenes.

other motions. For example, Fig. 17 (3rd row) shows that all the existing state-of-the-art approaches are unable to correct the train and pole (lean to the left and right, respectively) to the vertical simultaneously. While *Adobe Pr* and *V-DRSC* correct distortions to a good extent. However, in the next subsequence, where the camera was rotating rapidly to film the forwarding train, the corrections of *Adobe Pr* are imperfect among the pole and building (blur and skew). Similarly, all existing RSC solutions fail in such a dynamic scene. In contrast, *V-DRSC* corrects distortions properly in all these varied examples.

Non-perspective camera scene. Fig. 18 (1st column) shows RS images with both heavy lens distortions and RS effect captured with moving fisheye RS camera in *TUM* datasets. Fig. 18 (3rd to 8th column) shows that the commercial software *V-DRSC* and state-of-the-art learning-based methods *DSUN* [22], *JCD* [23], *SUNet* [24] and *CVR* [26] provide significant artifacts (blur and split) among the outputs. While geometry-based method *DSfM* and *V-DRSC* achieve the best corrections that are close to the ground-truth GS image in Fig. 18 (2nd column). However, it is important to realize that we provide the ground truth intrinsic information ($\mathbf{K}$ together with the lens distortion parameters) to *DSfM* while *V-DRSC* is calibration-free without requiring pre-knowledge about the camera intrinsic.

Human perception rating. Further, We questioned 56 users to provide preferences for corrected frames by *Adobe Pr*, *DSfM*, *DSUN*, *JCD*, *SUNet*, *CVR* and the proposed method *V-DRSC* based on their visual perception. Corrected images from all the datasets mentioned above are distributed to users without information about the corresponding generation method for specific images. As shown in Fig. 19, our outputs are rated equal to or better than comparison methods at least 80% of the time across all datasets.

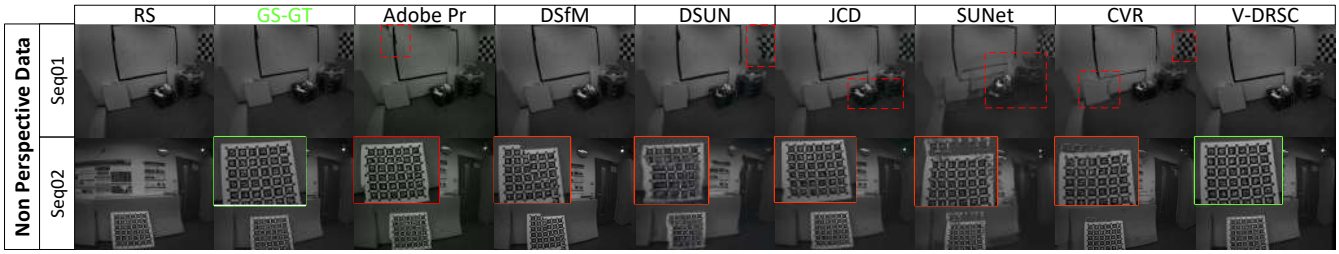

Fig. 18: Visual comparisons of our *V-DRSC* with existing RSC algorithms *Adobe Pr* [25], *DSfM* [18], *DSUN* [22], *JCD* [23], *SUNet* [24] in correcting RS image with lens distortion.

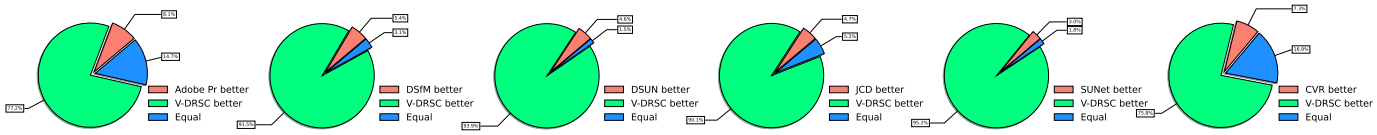

Fig. 19: User survey rating for proposed method *V-DRSC* against *Adobe Pr* [25], *DSfM* [18], *DSUN* [22], *JCD* [23], *SUNet* [24].

TABLE 3: Comparing runtime across nHD (480p), HD (720p), and FHD (1080p) input RS image resolutions.

Resolution	Time cost						
	GPU					CPU	
	Adobe Pr	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	DSfM [18]	V-DRSC
480P	0.072	0.122	0.211	0.059	0.093	206.052	0.025
720P	0.362	0.381	0.687	0.147	0.199	1469.020	0.073
1080P	0.721	0.909	—	0.348	0.314	1525.880	0.153

4.3.3 Runtime Comparison

Our method is implemented in C++ and runs on a desktop with a Core i9 CPU. On average, our method uses 73ms to correct a frame of HD resolution (1280×720). The runtime breakdown is 62ms for dense image matching, 3ms for correction field computation, and 8ms for frame warping. In Tab. 3, we compare runtime for processing nHD (640×360), HD (1280×720), and FHD (1920×1080). The results demonstrate that our method is nearly an order of magnitude faster than the commercial software *Adobe Pr*, and nearly 4 orders of magnitude faster than the geometrical-based method *DSfM*. Besides, without GPU acceleration, *V-DRSC* is faster than the learning approaches, specifically, nearly 5 times faster than *DSUN*, 8 times faster than *JCD*, and 2 times faster than *SUNet* and *CVR*. Also, note that our method runs at a real-time speed for nHD video without attempting to optimize the implementation. We believe that the speed of our pipeline can be further improved by using parallel computing.

4.4 RSC for 3D Analysis

As discussed in Sec. 1.2, RSC should visually be more appealing and rectified to be geometrically more meaningful. Thus, we evaluate the effectiveness of all the RSC methods on different datasets by applying the correction results to various downstream 3D analyses, namely, relative pose estimation, visual odometry, and SfM. We use *K-DRSC* to the RSC for 3D analysis tasks instead of *V-DRSC* for higher efficiency.

4.4.1 Relative Pose Estimation

Recovering relative pose between two given images is vital in 3D vision. This experiment applies all RSC methods to RS sequences frame-by-frame and is followed by a conventional relative pose estimation.

Metrics. Once the RS frames are well rectified, the corrected images in the pair are related by homography or epipolar geometry. Thus, we compute the homography or fundamental matrix between every two consecutive images and use the median value of found inliers $|R_F|$ and the ratio of inlier out of rough matches $|R_F|$ for evaluation. We apply the Homography-RANSAC procedure to *3GS*, *GPark*, *YouTube* sequences where the cameras were under approximate pure rotation, while fundamental-RANSAC to *Seq77*, *Fastect* and *TUM* sequences where cameras were under significant translation. Specifically, the RANSAC procedure was applied 100 times on each image pair. The inlier threshold is chosen as 1 pixel in all the cases.

The experiment results in Fig. 20 and Tab. 4 demonstrate that the proposed method *K-DRSC* outperforms all the existing approaches with a significant margin in both homography and epipolar estimation with a high $|R_F|$ and $|R_F|$ scores.

4.4.2 Visual Odometry

In this experiment, we compare the abilities of different RSC methods to support VO. Specifically, we used *Sequence77*, 6 sequences in *WHU* and 10 sequences in *TUM* for comparison since they all provide both RS video sequence and ground truth camera poses.

Note that *Adobe Pr*, *DSUN*, *JCD*, *SUNet* and *CVR* perform RSC to each frame and then use the corrected sequence as input to ORB-SLAM2 [33]. In contrast, as discussed in Sec. 3.2, the geometry-based approaches *DSfM* and *K-DRSC* can plug-in GSVO system by correcting the keypoint only instead of rectifying the whole image. Despite this, we also report the results of *V-DRSC* to demonstrate its effectiveness in geometrically correcting the whole frame. However, it is

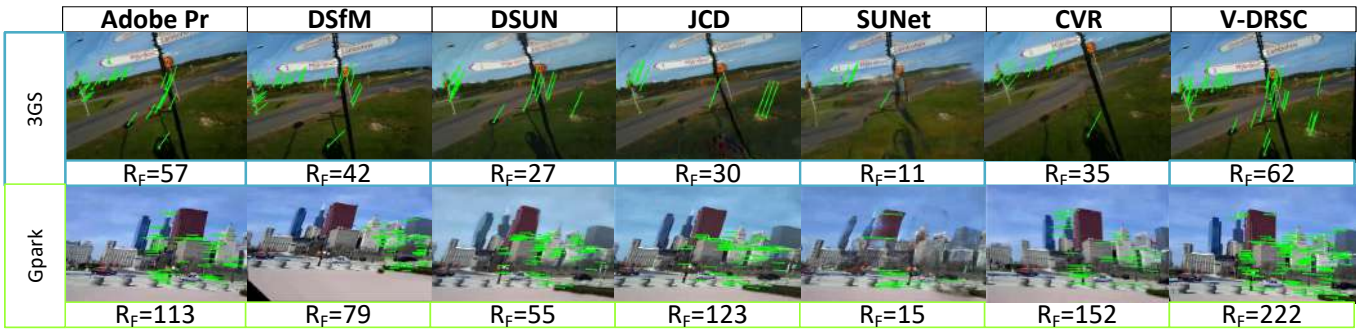

Fig. 20: Two examples of image matching comparison on '3GS' (top) and 'GPark' sequences (bottom). A selected image pair from each sequence are displayed in separate rows for better qualitative comparison. The inliers are displayed along with the median of the number of inliers R_F. Details of quantitative comparison are reported in Tab. 4.

TABLE 4: Inlier Feature Numbers and Rate of corrected frame-pairs on vast RS datasets. The best results are shown in green.

Method	$\lvert R_F\rvert \uparrow$							$\lvert R_F\rvert$ *Rate* $\uparrow$						
	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC
3GS [19]	57.0	42.0	26.5	30.0	11.0	35.0	62.0	0.41	0.35	0.30	0.32	0.35	0.38	0.44
BS-RSC [48]	259.0	198.0	30.0	127.0	12.0	110.0	294.0	0.64	0.49	0.26	0.44	0.21	0.44	0.70
Gpark [7]	113.0	78.5	55.0	123.0	15.0	152.5	222.0	0.47	0.39	0.33	0.48	0.33	0.56	0.64
Fastec [22]	26.0	18.0	24.0	24.0	24.0	26.0	26.0	0.38	0.31	0.38	0.38	0.39	0.41	0.40
Seq77 [47]	45.0	36.0	34.0	40.0	21.0	45.0	49.0	0.61	0.47	0.57	0.60	0.42	0.62	0.62
TUM [46]	66.0	57.0	18.0	37.0	10.0	34.0	73.0	0.40	0.35	0.32	0.37	0.28	0.34	0.41
YouTub	810.5	660.0	601.0	720.5	576.0	816.0	843.0	0.62	0.54	0.56	0.32	0.56	0.64	0.64

TABLE 5: The ratio of successfully tracked frames divided by the total frames ***DUR*** of different RSC methods. The best and second best results are shown in green and blue, respectively.

Method		**DUR**									
		GS	RS	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC	K-DRSC
Seq77 [47]	FULL	—	94.96	80.26	**95.11**	94.89	94.44	0.00	90.53	92.67	**95.56**
WHU [45]	Traj 1 FC	99.94	99.72	99.70	99.54	50.34	99.35	99.74	99.56	**99.80**	**99.76**
	Traj 2 FC	99.34	98.73	99.53	98.73	99.63	99.47	99.01	**99.67**	**99.73**	99.32
TUM [46]	Seq 01	99.67	50.55	48.73	37.36	24.33	48.32	12.36	37.41	**99.71**	**99.61**
	Seq 02	99.50	82.19	29.20	28.88	9.58	77.45	1.68	26.44	**99.72**	**99.72**
	Seq 06	99.85	83.74	64.31	58.92	5.75	60.88	5.37	58.12	**99.91**	**99.89**

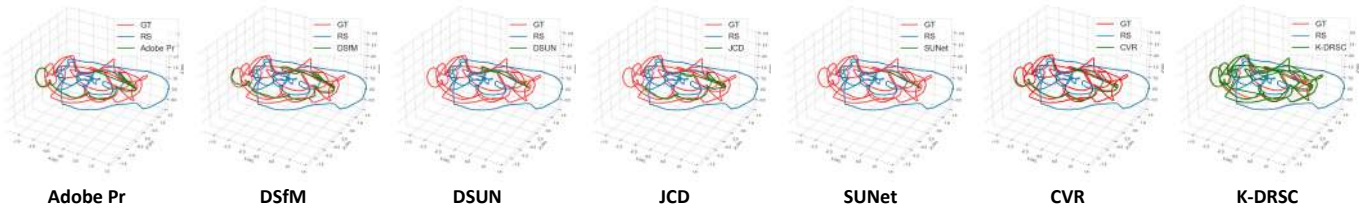

Fig. 21: Ground truth camera trajectory (red), the trajectory of GS method on RS data (convention ORB-SLAM2 with RS sequences input, blue), and trajectories estimated by ORB-SLAM2 augmented by different RSC methods (green).

important to realize that we always recommend *K-DRSC* for supporting 3D analysis over *V-DRSC* due to its significantly higher efficiency and to boost the number of inliers potentially.

Metrics. We use the absolute trajectory error (***ATE***) [46] to evaluate the VO results quantitatively. Given ground truth frame positions $\bar{\mathbf{c}}_i \in \mathbb{R}^3$ and corresponding ORB-SLAM2 [33] tracking results $\mathbf{c}_i \in \mathbb{R}^3$ using corrected sequence by each RSC method. It is defined as

$$e_{\text{ate}} = \min_{\mathbf{T}\in \text{Sim}(3)} \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{T}(\mathbf{c}_i) - \bar{\mathbf{c}}_i\|}, \tag{17}$$

where $\mathbf{T} \in \text{Sim}(3)$ is a 7D similarity transform that aligns the estimated trajectory with the ground truth one since the scale is not observable for monocular methods. We run each method 20 times on each sequence to obtain the APE e_{ate}. Besides, we found out that some RSC solutions provide results of the corrections that are even worse than the original input RS frames. This leads to failure in tracking and makes ORB-SLAM2 interrupt before the last frame. Therefore, we use the ratio of the successfully tracked frames out of the total frames ***DUR*** as an evaluation metric.

The results demonstrate that the proposed *V-DRSC* and *K-DRSC* outperform existing RSC methods in camera pose estimation. Qualitatively, this is clearly visible in Fig. 21 that

TABLE 6: Absolute trajectory error (ATE) of different RSC methods after Sim(3) alignment to ground truth. The best and second results are shown in green and blue. Since some methods will lose tracking without processing the whole frame, thus we highlight the background of each cell with different colours depending on its corresponding ***DUR*** value. Specifically, $\boldsymbol{DUR} > 0.9$, $0.5 < \boldsymbol{DUR} \leqslant 0.9$ and $\boldsymbol{DUR} \leqslant 0.5$ are highlighted in light green, cyan, and orange.

Method		APE									
		GS	RS	Adobe Pr	DSfM [18]	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	V-DRSC	K-DRSC
Seq77 [47]	FULL	—	0.2253	0.2439	0.2196	0.2304	0.2449	—	0.2260	**0.2180**	**0.2074**
WHU [45]	Traj 1 FC	0.0313	0.2822	0.2303	0.3659	—	0.2198	0.0868	**0.0804**	0.1116	**0.0760**
	Traj 2 FC	0.0077	0.0404	0.0331	0.0509	0.0328	0.0873	**0.0107**	**0.0106**	**0.0107**	0.0129
TUM [46]	Seq 01	0.0056	—	—	—	—	—	—	—	**0.0118**	**0.0149**
	Seq 02	0.0049	0.0962	—	—	—	0.0881	—	—	**0.0077**	**0.0143**
	Seq 06	0.0040	0.1177	0.0742	—	—	0.1665	—	0.0233	**0.0095**	**0.0093**

TABLE 7: Realtime factor ϵ is computed by the duration of the successfully tracked frames in the sequence divided by the processing time of different RSC augmented ORB-SLAM2. The best results are shown in green.

Method	Real time factor $\epsilon \uparrow$									
	GPU					CPU				
	Adobe Pr	DSUN [22]	JCD [23]	SUNet [24]	CVR [26]	GS	RS	DSfM [18]	V-DRSC	K-DRSC
Seq77 [47]	0.58481	0.37021	0.22596	—	0.62330	—	3.78200	0.00025	**1.30707**	**1.64976**
WHU [45]	0.57471	0.36231	0.22154	0.66984	0.46373	3.26613	3.30405	0.00024	**1.25023**	**1.46146**
TUM [46]	0.16118	0.08600	0.04680	0.20400	0.21530	1.84048	1.90404	0.00002	**0.37123**	**1.55851**

Adobe Pr, *DSfM*,*DSUN*, *JCD*, *SUNet* and *CVR* lost tracking and provide incomplete trajectories with obvious systematic drift to the ground truth.

More quantitative comparisons are given in Tab. 5 and 6. On the sequences in *Sequence* and *WHU* where RS effects are limited, ORB-SLAM2 successfully tracks over all the frames by using the corrected frame by all the RSC methods. However, surprisingly, the APEs of *Adobe Pr*, *DSfM* and *JCD* are even larger than the GS method on RS data (convention ORB-SLAM2 with RS sequences input) *RS*. While *DSUN*, *SUNet*, *CVR* and proposed methods *V-DRSC* and *K-DRSC* provide significantly smaller APEs than *RS*.

However, as shown in the Tab. 5, for the 10 sequences in *TUM* where the camera is under rapid movement and has significant RS effect, *Adobe Pr*, *DSfM*, *DSUN*, *SUNet*, and *CVR* can only track less than 50% of the frames while *JCD* achieves 60% of DUR. Note that it is meaningless to evaluate the APE of these methods with short trajectories. In contrast, the proposed *V-DRSC* and *K-DRSC* are capable of tracking 100% of the frames. Moreover, as shown in the Tab. 6, the APEs of *V-DRSC* and *K-DRSC* are smaller than *RS* by one order of magnitude in most cases. Particularly, we observe that a comparison of the GS method on GS data (convention ORB-SLAM2 with GS sequences input) *GS* with the proposed *V-DRSC* and *K-DRSC* on RS data does not yield a clear preference for all sequences in all datasets.

Runtime. Tab. 7 reports the efficiencies of *Adobe Pr*, *DSfM*,*DSUN*, *JCD*, *SUNet*, *CVR*, *V-DRSC* and *K-DRSC*. The real-time factor ϵ is calculated as the real duration of the sequence divided by the processing time of the algorithm. The results show that the learning-based methods are faster than geometry-based approaches *Adobe Pr* and *DSfM* by one order of magnitude. However, even with GPU acceleration, the fastest learning-based method *CVR* achieves $\epsilon \approx 0.5$. In contrast, without GPU, the proposed *K-DRSC* achieves average $\epsilon > 1.5$ performance in three datasets, indicating it is a real-time capable method.

4.4.3 Structure-from-Motion

This experiment shows the results of applying the proposed RSC algorithm to serve downstream SfM applications. First, we present the results on pairs of consecutive images from the sequences on *House* dataset. As shown in Fig. 22 that the reconstructed point clouds by the corrected frames of existing RSC methods *Adobe Pr*, *DSfM*,*DSUN*, *JCD*, *SUNet* and *CVR* are uncompleted, noisy with observable distortions. In contrast, the proposed method *K-DRSC* provides a much more complete and clean reconstruction. Besides, we recorded three own RS sequences for larger-scale SfM evaluation. The results in Fig. 22 demonstrate that the frames corrected by the proposed RSC method *K-DRSC* augment the classical GS-SfM pipeline to handle the large-scale 3D reconstruction by using image sequences even with significant RS effect. Nevertheless, we can observe that the state-of-the-art RSC methods' reconstructions of the corrected images suffer from fragmented reconstructions or apparent deformations.

5 CONCLUSIONS

In this paper, we propose a geometry-based fast RSC solution. We draw the mathematical link between the natural optical flow and its corresponding corrected flow based on epipolar geometry. As a result, we then propose a novel RSC solution called *DRSC* based on a pixel-wise pose-free RSC solver. Given two consecutive RS images, it can effectively and efficiently estimate the corresponding GS image without GPU acceleration. The proposed method brings five primary advantages over the previous works: *1)* It can handle the depth-dependent RS effect. *2)* It is feasible to highly dynamic scenes. *3)* It is a calibration-free method without requiring pre-knowledge of camera intrinsics. *4)* It

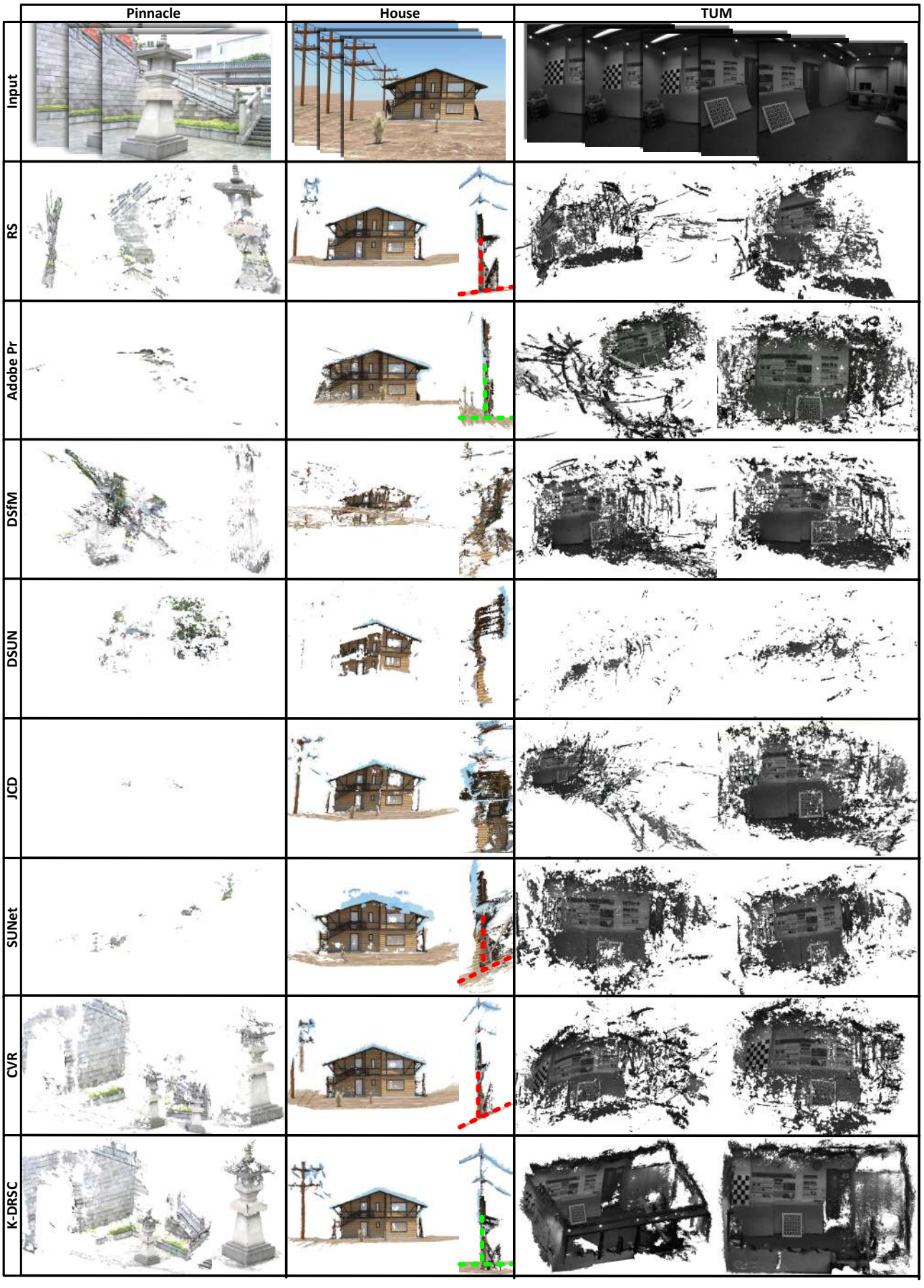

Fig. 22: Reconstruction results of SfM using corrected results by various RSC methods in Pinnacle (our self-collected dataset), House [19] and TUM [46] dataset.

performs the RSC in real-time without GPU assistance. *5)* It is capable of the plugin and augmenting GS-SfM and GS-SLAM to handle RS images as input. Extensive experiments demonstrate that the proposed *DRSC* solution achieves state-of-the-art RS correction performances in both visual corrections and supports downstream 3D analysis tasks. We are currently concerned that the proposed method does not fully use the constraints between continuous frames over the whole sequence in RS video processing. In the future, we will explore further optimizing the video-based RSC by modeling the camera trajectory to increase accuracy and stability.

REFERENCES

[1] J. Janesick, J. Pinter, R. Potter, T. Elliott, J. Andrews, J. Tower, J. Cheng, and J. Bishop, "Fundamental performance differences between CMOS and CCD imagers: part III," in *Astronomical and Space Optical Systems*, P. G. Warren, C. J. Marshall, J. B. Heaney, E. T. Kvamme, R. K. Tyson, and M. Hart, Eds., vol. 7439, International Society for Optics and Photonics. SPIE, 2009, pp. 47 – 72. [Online]. Available: https://doi.org/10.1117/12.831203

[2] M. Meingast, C. Geyer, and S. Sastry, "Geometric models of rolling-shutter cameras," in *OMNIVIS*, 2005.

[3] C. Albl, Z. Kukelova, and T. Pajdla, "R6p-rolling shutter absolute pose problem," in *CVPR*. IEEE, 2015, pp. 2292–2300.

[4] C. Wu *et al.*, "Visualsfm: A visual structure from motion system," 2011.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *T-RO*, 2015.

[6] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," *CSTR*, vol. 1, no. 2, p. 13, 2011.

[7] C. Jia and B. L. Evans, "Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements." in *MMSP*, 2012, pp. 203–208.

[8] S. Hee Park and M. Levoy, "Gyro-based multi-image deconvolution for removing handshake blur," in *CVPR*, 2014.

[9] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *IJCV*, 2015.

[10] J. Mo, M. J. Islam, and J. Sattar, "Imu-assisted learning of single-view rolling shutter correction," in *Conference on Robot Learning*. PMLR, 2022, pp. 861–870.

[11] V. Rengarajan, A. N. Rajagopalan, and R. Aravind, "From bows to arrows: Rolling shutter rectification of urban scenes," in *CVPR*, 2016.

[12] P. Purkait, C. Zach, and A. Leonardis, "Rolling shutter correction in manhattan world," in *ICCV*, 2017, pp. 882–890.

[13] P. Purkait and C. Zach, "Minimal solvers for monocular rolling shutter compensation under ackermann motion," in *WACV*, 2017.

[14] Y. Lao, O. Ait-Aider, and H. Araujo, "Robustified structure from motion with rolling-shutter camera using straightness constraint," *Pattern Recognition Letters*, 2018.

[15] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski, "Removing rolling shutter wobble," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2392–2399.

[16] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Calibration-free rolling shutter removal," in *ICCP*, 2012.

[17] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust l1 optimal camera paths," in *CVPR*. IEEE, 2011, pp. 225–232.

[18] B. Zhuang, L.-F. Cheong, and G. H. Lee, "Rolling-shutter-aware differential sfm and image rectification," in *ICCV*, 2017.

[19] P.-E. Forssén and E. Ringaby, "Rectifying rolling shutter video from hand-held devices," in *CVPR*, 2010.

[20] V. Rengarajan, Y. Balaji, and A. Rajagopalan, "Unrolling the shutter: Cnn to correct motion distortions," in *CVPR*, 2017.

[21] B. Zhuang, Q.-H. Tran, P. Ji, L.-F. Cheong, and M. Chandraker, "Learning structure-and-motion-aware rolling shutter correction," in *CVPR*, 2019, pp. 4551–4560.

[22] P. Liu, Z. Cui, V. Larsson, and M. Pollefeys, "Deep shutter unrolling network," in *CVPR*, 2020, pp. 5941–5949.

[23] Z. Zhong, Y. Zheng, and I. Sato, "Towards rolling shutter correction and deblurring in dynamic scenes," in *CVPR*, 2021, pp. 9219–9228.

[24] B. Fan, Y. Dai, and M. He, "Sunet: symmetric undistortion network for rolling shutter correction," in *ICCV*, 2021, pp. 4541–4550.

[25] Adobe, "Professional video editing software — adobe premiere pro," 2022.

[26] B. Fan, Y. Dai, Z. Zhang, Q. Liu, and M. He, "Context-aware video reconstruction for rolling shutter cameras," in *CVPR*, 2022.

[27] S. Vasu, M. M. MR, and A. Rajagopalan, "Occlusion-aware rolling shutter rectification of 3d scenes," in *CVPR*, 2018.

[28] B. Zhuang and Q.-H. Tran, "Image stitching and rectification for hand-held cameras," in *ECCV*. Springer, 2020.

[29] Y. Lao and O. Ait-Aider, "A robust method for strong rolling shutter effects correction using lines with automatic feature selection," in *CVPR*, 2018.

[30] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *CVPR*, 2018.

[31] T. P. Bapat, Akash and J.-M. Frahm, "Rolling shutter and radial distortion are features for high frame rate multi-camera tracking," in *CVPR*, 2018.

[32] Y. Lao and O. Ait-Aider, "Rolling shutter homography and its applications," *TPAMI*, vol. 43, no. 8, pp. 2780–2793, 2021.

[33] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[34] D. J. Heeger and A. D. Jepson, "Subspace methods for recovering rigid motion i: Algorithm and implementation," *International Journal of Computer Vision*, vol. 7, pp. 95–117, 1992.

[35] Y. Ma, J. Košecká, and S. Sastry, "Linear differential algorithm for motion recovery: A geometric approach," *IJCV*, vol. 36, no. 1, pp. 71–89, 2000.

[36] J. Weng, P. Cohen, M. Herniou *et al.*, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[37] D. C. Brown, "Decentering distortion of lenses," *PE*, vol. 32, no. 3, pp. 444–462, 1966.

[38] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, 2001.

[39] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *CVPR*, 2015.

[40] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, 2000.

[41] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV*, 2011.

[42] K. Shoemake, "Animating rotation with quaternion curves," in *SIGGRAPH*, 1985.

[43] L. Magerand, A. Bartoli, O. Ait-Aider, and D. Pizarro, "Global optimization of object pose and motion from a single rolling shutter image with automatic 2d-3d matching," in *ECCV*, 2012.

[44] Y. Dai, H. Li, and L. Kneip, "Rolling shutter camera relative pose: generalized epipolar geometry," in *CVPR*, 2016.

[45] L. Cao, J. Ling, and X. Xiao, "The whu rolling shutter visual-inertial dataset," *IEEE Access*, vol. 8, pp. 50 771–50 779, 2020.

[46] D. Schubert, N. Demmel, L. von Stumberg, V. Usenko, and D. Cremers, "Rolling-shutter modelling for visual-inertial odometry," in *IROS*, November 2019.

[47] J. H. Kim, C. Cadena, and I. Reid, "Direct semi-dense slam for rolling shutter cameras," in *ICRA*, 2016.

[48] M. Cao, Z. Zhong, J. Wang, Y. Zheng, and Y. Yang, "Learning adaptive warping for real-world rolling shutter correction," in *CVPR*, 2022.

[49] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, T. Kipf, A. Kundu, D. Lagun, I. Laradji, H.-T. D. Liu, H. Meyer, Y. Miao, D. Nowrouzezahrai, C. Oztireli, E. Pot, N. Radwan, D. Rebain, S. Sabour, M. S. M. Sajjadi, M. Sela, V. Sitzmann, A. Stone, D. Sun, S. Vora, Z. Wang, T. Wu, K. M. Yi, F. Zhong, and A. Tagliasacchi, "Kubric: a scalable dataset generator," 2022.

Delin Qu Delin Qu received a Bachelor's degree from the College of Computer Science and Electronic Engineering at Hunan University. He is pursuing his Ph.D. in computer vision at the School of Computer Science, Fudan University, and interning as a researcher at Shanghai AI Laboratory. His research interests are in the area of computer vision and computational imaging.

Bangyan Liao BangYan Liao is pursuing his Bachelor's degree in Electrical and Information Engineering from Hunan University. His research interests include visual odometry/simultaneous localization and mapping, computer vision, and Rolling Shutter cameras.

Huiqing Zhang Huiqing Zhang is a master's candidate at Hunan University, advised by Prof. Yizhen Lao, where she works in Computational Photography and Computer Vision. She focuses on Camera Synchronization and Neural Rendering. Before that, She got Bachelor's Degree in Computer Science and Technology from Jiangxi Normal University.

Omar Ait Aider Omar Ait Aider is an Associate Professor and the Head of the Bachelor's degree in Mechatronics at the University of Clermont-Auvergne. He received his Master's Degree on "Autonomous Robotics" at Pierre et Marie Curie University in Paris, then his PhD on "Computer Vision for Robotics" at the University of Evry Val d'Essonne in France.Since September 2006, he is a member of the "Image, Perception Systems and Robotics" group within the Institut Pascal-CNRS. His research is focused on geometrical camera modelling and 3D vision. He received the Honorable Mention Best Paper Award at the European Conference on Computer Vision ECCV'2006.

Yizhen Lao Yizhen Lao is an Associate Professor at the College of Computer Science and Electronic Engineering, at Hunan University. He obtained his Ph.D. degree from the University of Clermont-Auvergne, Master's degree in Geo-information Science and Earth Observation from the University of Twente, and Bachelor's degree in Spatial-Informatics and Digitalized Technology from Wuhan University. His research is focused on 3D computer vision and computational photography.