From Bows to Arrows: Rolling Shutter Rectification of Urban Scenes

Vijay Rengarajan[1], A.N. Rajagopalan[2], R. Aravind[3]
Indian Institute of Technology Madras
[1]vijay.ap@ee.iitm.ac.in, [2]raju@ee.iitm.ac.in, [3]aravind@ee.iitm.ac.in

Abstract

The rule of perspectivity that 'straight-lines-must-remain-straight' is easily inflected in CMOS cameras by distortions introduced by motion. Lines can be rendered as curves due to the row-wise exposure mechanism known as rolling shutter (RS). We solve the problem of correcting distortions arising from handheld cameras due to RS effect from a single *image free from motion blur with special relevance to urban scenes. We develop a procedure to extract prominent curves from the RS image since this is essential for deciphering the varying row-wise motion. We pose an optimization problem with line desirability costs based on straightness, angle, and length, to resolve the geometric ambiguities while estimating the camera motion based on a rotation-only model assuming known camera intrinsic matrix. Finally, we rectify the RS image based on the estimated camera trajectory using inverse mapping. We show rectification results for RS images captured using mobile phone cameras. We also compare our single image method against existing video and nonblind RS rectification methods that typically require multiple images.*

1. Introduction

In recent years, inferring scene geometry has become possible from as little as a single image [13, 7, 18, 33]. The images of interest are often man-made structures (e.g. in the Manhattan world) which have predominant straight lines. From the knowledge of these lines, properties such as vanishing points, horizon, and zenith can be estimated [33] to aid in applications such as scene classification [14] and depth estimation [29]. For the single image scenario, camera motion is a hindrance to scene understanding. It introduces image distortions which are increasingly becoming commonplace. The CMOS mobile phone cameras employ a shutter mechanism in which the pixels on the sensor plane are exposed in a row-wise manner from top to bottom with a constant inter-row delay. Hence, even a small camera motion due to handshake can cause visible distortions in the captured image, a phenomenon referred to as the rolling shutter (RS) effect [19, 4, 26, 25, 31]. In a high exposure setting, motion blur further complicates the distortions, but in typical daylight scenarios, the blur is negligible and only the RS distortion is most perceivable.

(a) RS image (b) Our rectification (c) Lens rectification

Figure 1. (a) Image curved due to RS effect, (b) rectified image in which curves are corrected to lines respecting the orthogonality of vertical and horizontal edges using the proposed method, and (c) rectified image using the lens correction method [35]

The RS effect is more prominent in imaging urban scenes, since the row-varying camera motion alters the straightness of lines in the scene. This results in the manifestation of curves in the captured image in place of lines which can be problematic for scene inference methods that rely on line properties. The image shown in Fig. 1(a) is captured using a Motorola Moto G2 camera with handshake during exposure. The vertical lines in the 3D scene are affected by the row-varying camera motion and appear as curves.

Video rectification (i.e. correction of RS distortions) has been studied elaborately in RS imaging [19, 4, 26, 10, 32]; however, these methods are not applicable for the case of single image RS removal, since the RS motion is estimated by using the correspondences between successive video frames. While there are works for absolute RS camera pose estimation that use only a single image [1, 22, 2], they assume known correspondences between image and world points. It is also possible to use inertial sensor information to infer RS motion especially for videos [11, 16, 23, 24] and for long exposure motion blur images [30], but the low acquisition rate of inertial sensors prohibits usage on a single RS image. A combined RS and motion blur framework without the inertial information is proposed for change detection in [25]. Though the camera motion experienced by the RS image is actually estimated, it is carried out in a non-

blind fashion by assuming knowledge of a clean image with no RS artifacts. In [31], a single image deblurring algorithm is proposed for images affected by both RS and motion blur. The camera motion is inferred through the spatial variation of blur across rows by backprojecting the local point spread functions (PSFs) to a higher dimensional camera trajectory. However, it cannot handle the RS-only case (i.e. without motion blur), since the PSFs will all be identical impulses leading to no RS correction.

To the best of our knowledge, no work exists that corrects RS geometric distortions using the information from only a single observation. It is this challenging scenario that we tackle in this paper. We address the case where the exposure time of each row is short enough to ignore the presence of motion blur, while at the same time, the RS effect is unavoidable due to inter-row exposure delay. Without the availability of any other information (from additional images or from inertial sensors), the problem is very ill-posed. We do not have the liberty to use image correspondences as in a multi-image scenario; nor can we impose image priors such as gradient sparsity as in deblurring problems, since the RS image is already free of blur. Hence, we propose to exploit the presence of lines in man-made structures to impose constraints that enable camera motion estimation. Specifically, we use the 'straight-lines-must-be-straight rule' for perspective cameras [8] on RS curves. Even for small motion, vertical lines are affected by the RS mechanism. The resultant curves can potentially reveal the underlying camera motion.

Incidentally, studies on curve extraction exist in the realm of lens distortion. In one of the earliest works [8], curves are formed by linking spatially closer edge pixels. In a recent work [5], the distorted lines are modelled as circular arcs. In [3], small arcs are detected by using a modified Hough transform embedding the radial distortion parameter. An important distinction between methods extracting curves for lens and for RS distortions is that the former benefits from apriori knowledge of the lens distortion model which helps in tailoring the algorithm to treat curves that can be expected to occur in the captured image. In contrast, in the case of RS distortions, the extent and the properties of curves completely depend on the camera trajectory during the exposure and cannot be predicted beforehand. In addition, while it is well known that wide-angle lens defects can be handled with camera pre-calibration, this is not applicable here, since the RS effect depends on the amount of camera motion which essentially is unique for each capture. In [35], automatic calibration of intrinsic parameters of a camera including lens properties is performed exploiting low-rank textures from images. Correcting the curvature of the RS image in Fig. 1(a) using [35] results in partial curvature corrections as shown in Fig. 1(c), since the nature of the two distortions are completely different.

We develop a curve detection method that automatically links local line segments into curves based on spatial and angular proximities. Since we are interested only within a single image exposure, we model the row-wise variation of rotation-only RS motion as a polynomial similar to the one used in [31] which suits the camera motion adequately for short exposures. We propose an optimization problem with line, angle, and length constraints on the detected curves to estimate the camera motion thus enabling us to rectify the RS effect. We devise an inverse mapping procedure that assigns an intensity value to every pixel in the rectified image from its corresponding RS pixel based on the estimated camera motion. The rectified RS image using the proposed method is shown in Fig. 1(b). The curves are corrected to lines, and in addition, the orthogonality of the ceiling, wall, and ground planes is preserved.

1.1. Main Contributions

- This is the first work of its kind to address the problem of correcting geometric distortions due to the RS mechanism from a *single* image devoid of motion blur and lens distortions. The key idea is to exploit the information embedded in the curves of the RS image to reveal the underlying RS camera motion.
- This is also the first work to study geometric ambiguities while estimating row-wise rotation-only camera motion and suggest remedies to resolve them using line properties.

2. RS Rectification

After describing our RS motion model in Section 2.1, we proceed to describe curve detection procedure in Section 2.2, camera motion estimation in Section 2.3, and image rectification algorithm in Section 2.4.

2.1. Camera Model

For a static CMOS camera located at the world origin, the scene point $\mathbf{X}$ is related to the image point through the camera intrinsic matrix $\mathbf{K}$ [12]. We refer this image as the global shutter (GS) image. The image point in the homogeneous representation is given by

$$\widetilde{\mathbf{x}}_{GS} = \mathbf{K}\mathbf{X}. \tag{1}$$

When the CMOS camera moves during exposure, each row of the image sensor plane experiences its own camera pose due to the row-wise sequential acquisition. This results in distortions in the captured image which is referred to as the RS image. For images captured with hand-held cameras, only the rotations play a major role, as noted in works dealing with camera motion in conventional global exposure cameras [34] as well as in RS cameras [26, 31].

Hence, we employ a rotation-only model for the camera motion. Let the number of rows in the RS image be M, and let the row number be indexed by y. The camera pose for the y^{th} row is given by the 3D rotation angles $r_x(y)$, $r_y(y)$, and $r_z(y)$, and the equivalent orthogonal rotational matrix is denoted by $\mathbf{R}(y)$. Thus, the series of rotation matrices $\{\mathbf{R}(y)\}_{y=1}^{M}$ represents the camera motion during exposure. During RS acquisition, the scene point $\mathbf{X}$ is mapped on the image plane due to the camera motion $\mathbf{R}(y)$ as follows [12]:

$$\widetilde{\mathbf{x}}_{RS} = \mathbf{K}\mathbf{R}(y)\mathbf{X}, \tag{2}$$

where $\widetilde{\mathbf{x}}_{RS}$ is the homogeneous representation of the RS image point and $\widetilde{\mathbf{x}}_{RS}(2)/\widetilde{\mathbf{x}}_{RS}(3) = y$.

For a same scene point $\mathbf{X}$, the relationship between the image points in the GS and RS images is written using (1) and (2) as

$$\widetilde{\mathbf{x}}_{GS} = \mathbf{K}\mathbf{R}^{-1}(y)\mathbf{K}^{-1}\widetilde{\mathbf{x}}_{RS}. \tag{3}$$

Given an RS image distorted by camera motion, our aim is to recover the undistorted GS image. This is not possible without estimating the underlying camera trajectory.

Recovering the motion of each row independently is very ill-posed given only a single image. The number of unknowns to be estimated for 3D camera motion is $3M$ which is very high. To alleviate this problem, we adopt a polynomial model for camera motion during exposure. Using a simpler model such as a spherical linear interpolation model [26] restricts the trajectory to be piecewise-linear, while higher order camera models such as B-splines [24] are more suitable for modelling video frames, but complex and unnecessary for a single RS image.

We model the rotation trajectory along each axis $r_i(y)$ where $i \in \{x, y, z\}$ as a polynomial of degree n. Hence, we have

$$r_i(y) = \begin{cases} \alpha_{i0}, & y = 1 \\ \sum_{j=0}^{n} \alpha_{ij} \left(\frac{y-1}{M}\right)^j, & 2 \leq y \leq M \end{cases} \tag{4}$$

where α_{ij} is the j^{th} polynomial coefficient for the i^{th} axis motion. These coefficients are denoted by $\boldsymbol{\alpha}$, and equivalently, we obtain $\mathbf{R}(y)$ for all y. In this model, the number of unknowns to be estimated reduces to $3(n+1)$ (with known camera matrix $\mathbf{K}$). We use $n = 3$, and hence, the number of unknowns is 12. The validity of this model is discussed further in the experiments section. While a similar model was used earlier for the RS deblurring problem in [31], unlike that work, our model does not require the knowledge of the inter-row delay time of the RS camera. For a high exposure time setting, a polynomial model may not suffice due to the complex camera motion. However, this setting would also lead to motion blur which is beyond the scope of this paper. We also do not handle depth-dependent RS effect [28] caused by translations of fast moving cameras (e.g. mounted on vehicles) in this work.

2.2. Curve Detection

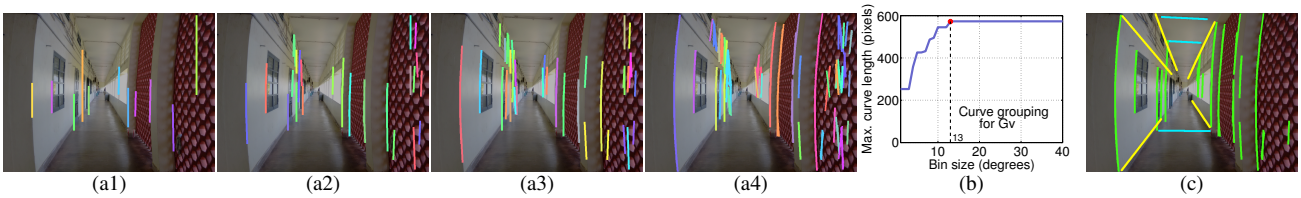
In this section, we develop a procedure to detect both curves and lines which serve as features for camera motion estimation.

Line segment extraction We first extract edges from the RS image using the Canny detector [6]. Let θ denote the angle of a line with respect to the horizontal axis. To detect line segments, we discretize the angle space $(-90^\circ, 90^\circ]$ in steps of 1° and denote it by $\mathbf{S}$. For each angle $\theta \in \mathbf{S}$, we map edge pixels to the Hough domain [15, 27] and detect lines from the peaks of the Hough accumulator matrix [9]. We then extract local line segments based on the edge pixels situated near the Hough lines. We fix a minimum length for line segments to avoid trivial edges. Each edge pixel is assigned to one of the line segments based on its Euclidean distance. Pixels that are at a distance greater than a threshold to all line segments are ignored.

Curve grouping Our next step attempts to link these line segments into curves (where possible) based on their spatial proximity. In addition, we classify the RS curves into three groups corresponding to horizontal, vertical, and slanted lines in the unknown GS image. These groups are denoted by $\mathbf{G}_h$, $\mathbf{G}_v$ and $\mathbf{G}_s$, respectively. Instead of manually binning the angle space to group curves, we follow a greedy approach in which we start with a seed angle and progressively increase the size of the bin to link more and more line segments that have proximal angles. We continue this process until a stopping point which is determined by the longest curve length along the vertical axis.

To create the vertical group $\mathbf{G}_v$, we consider a bin $\mathbf{B}_{\theta_s} \subset \mathbf{S}$ of angles around the seed angle $\theta_s = 90^\circ$. Let the bin size be B. We consider only the line segments which belong to $\mathbf{B}_{\theta_s}$, and join them into curves based on their endpoint and angle proximities. As the bin size B is increased to include more line segments, longer curves are formed (see Fig. 2(a1) to (a4)). This continues until the point after which there are no more line segments to link to get longer curves. This stopping point is determined by the saturation in the maximum curve length with increasing bin size, which is shown as a red point in Fig. 2(b). To detect curves for the horizontal group $\mathbf{G}_h$, we repeat this process with seed angle $\theta_s = 0^\circ$.

For scenes which lack long edges, the stopping point is difficult to determine. Hence, we fix the maximum bin size as 30° for $\mathbf{G}_v$ and a smaller value of 10° for $\mathbf{G}_h$, since lines closer to horizontal are least affected (as they are exposed to just one homography), and lines closer to vertical are most affected (as they are acted upon by many homographies). Once a subset of line segments are linked and classified to form the vertical and horizontal groups, $\mathbf{G}_v$ and $\mathbf{G}_h$, the remaining ones are considered for the slanted group $\mathbf{G}_s$. These are linked to form curves corresponding to GS lines at angles other than 90° and 0°.



(a1) (a2) (a3) (a4) (b) (c)

Figure 2. *Curve detection* (best viewed in PDF). (a1-a4) Curve linking by increasing the bin size for vertical group ($\mathbf{G}_v$) detection. Line segments grouped into curves are shown in same color. (b) Maximum curve length vs. bin size. (c) Final curve detection for all groups $\mathbf{G}_v$, $\mathbf{G}_h$, and $\mathbf{G}_s$ after outlier removal.

Outlier removal There is a possibility that some of the edge pixels have been wrongly grouped as curves, while in reality they are not. As this will affect our camera motion estimation, we follow a curve rejection procedure. We fit a third degree polynomial using the edge pixels corresponding to each curve independently, and the ones which result in a large error are removed from further consideration. Detected curves post outlier removal are shown in Fig. 2(c). Note that many of the small edge groups on the rightmost red wall have been discarded.

The final set of curves is denoted by $\{\mathbf{c}_i\}_{i=1}^{N_c}$, where N_c is the total number of curves. Each $\mathbf{c}_i$ is a set of edge pixels, and it belongs to one of the groups $\mathbf{G}_v$, $\mathbf{G}_h$, or $\mathbf{G}_s$.

2.3. Camera Motion Estimation

The edge pixel locations belonging to the RS curve $\mathbf{c}_i$ are denoted by $\mathbf{x}_{RS} = \{(x_{ij}, y_{ij})\}_{j=1}^{N_i}$, where N_i is the number of pixels in $\mathbf{c}_i$, x_{ij} is the column index and y_{ij} is the row index. Let $\widehat{\boldsymbol{\alpha}}$ (equivalently, $\widehat{\mathbf{R}}(y)$ or $[\widehat{r}_x(y), \widehat{r}_y(y), \widehat{r}_z(y)]$) be a camera motion estimate. Considering $\widetilde{\mathbf{x}}_{RS} = [x_{ij}, y_{ij}, 1]^T$, applying (3) would result in the estimated RS-corrected point, $\widetilde{\mathbf{x}}_{GS}$. Let (x'_{ij}, y'_{ij}) be the corresponding 2D point where $x'_{ij} = \widetilde{x}_{GS}(1)/\widetilde{x}_{GS}(3)$ and $y'_{ij} = \widetilde{x}_{GS}(2)/\widetilde{x}_{GS}(3)$. If $\boldsymbol{\alpha}$ is the ground-truth motion, then the points (x'_{ij}, y'_{ij}) will lie on a line at a particular angle. Hence, to evaluate the goodness of an estimate $\widehat{\boldsymbol{\alpha}}$, we develop a line desirability cost that penalizes higher curvatures.

Line Desirability Cost We fit a line through $\{(x'_{ij}, y'_{ij})\}_{j=1}^{N_i}$ using least squares which results in a line with parameters $(\overline{\rho}_i, \overline{\theta}_i)$, where $\overline{\rho}_i$ is the orthogonal distance of the line from the origin, and $\overline{\theta}_i$ is the angle of the line with respect to the horizontal axis. We formulate the error due to this line fitting for curve $\mathbf{c}_i$ as

$$e_i^{\text{line}} = \frac{1}{N_i} \sum_{j=1}^{N_i} (x'_{ij} \sin\overline{\theta}_i + y'_{ij} \cos\overline{\theta}_i - \overline{\rho}_i)^2. \quad (5)$$

The total line cost for this camera motion estimate is the sum of the above cost value for all the curves. We make an important note that, since the camera exposure sequence is row-wise, the motion information is better embedded in curves that span longer along the vertical axis. Hence, we weight the individual curve costs by a factor based on the length in pixels that the original curve spans along the vertical axis. Thus, the total cost based on the line constraint for all curves is given by

$$E^{\text{line}} = \frac{1}{N_c} \sum_{i=1}^{N_c} w_i e_i^{\text{line}}, \quad (6)$$

where w_i is the normalized weight of $\mathbf{c}_i$ calculated based on its length along the vertical axis. The length of the row span of $\mathbf{c}_i$ is given by $u_i = \max_j(y_{ij}) - \min_j(y_{ij}) + 1$. We then calculate the normalized weight as $w_i = u_i / \sum_{k=1}^{N_c} u_k$.

Geometric Ambiguities Minimizing the line desirability cost (6) would correct curves into lines, but there can be multiple solutions as shown in Fig. 3. It is straightforward to observe that multiple solutions can result from a global rotation r_z since it preserves the straightness of lines (first column in Fig. 3). Due to the row-wise variation, ambiguities arise in motion estimation that are unique to RS images. Since a global r_y corresponds to horizontal translation, a row-wise increase or decrease in r_y causes horizontal shearing (third column in Fig. 3). Correspondingly, a global r_x translates image up or down, and hence a row-wise change in r_x causes vertical stretching or shrinking (rows in Fig. 3). Therefore, all these transformations do not disrupt straightness of lines and potentially pose problems in minimizing the line cost. To arrive at a preferred solution, we additionally impose constraints involving the angles and the lengths.

Angle Desirability Cost To control arbitrary inplane rotation and horizontal shearing, we introduce a desirability cost to control the angles of lines. Ideally, we want the points on the curve $\mathbf{c}_i$ to be corrected to a GS line with an angle that it actually exhibits in the scene. Though it is not possible to know the exact angles of all GS lines, we note that the confidence of assuring a vertically (or horizontally) oriented RS curve as a vertical (or horizontal) line in the scene is presumably higher as compared to lines at any other angles. Hence, we add an angle cost for curves only in the groups $\mathbf{G}_v$ and $\mathbf{G}_h$. The angle desirability cost for a curve is defined as the squared error between the angle of the least squares fit line $(\overline{\rho}_i, \overline{\theta}_i)$ and the angle corresponding

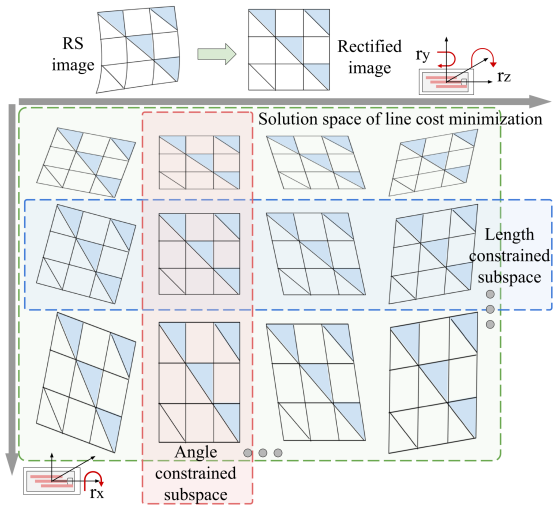


Figure 3. Visualization of our optimization problem.

to the curve. This cost for the curve $\mathbf{c}_i$ is given by

$$e_i^{\text{ang}} = b_i(\bar{\theta}_i - \theta_i)^2, \tag{7}$$

where $b_i = 1$ for $\mathbf{c}_i \in \{\mathbf{G}_v, \mathbf{G}_h\}$, and 0 otherwise, and $\theta_i = 90^\circ$ for $\mathbf{c}_i \in \mathbf{G}_v$, and $\theta_i = 0^\circ$ for $\mathbf{c}_i \in \mathbf{G}_h$. The total angle desirability cost is thus written as

$$E^{\text{ang}} = \frac{1}{\sum_{k=1}^{N_c} b_k} \sum_{i=1}^{N_c} e_i^{\text{ang}}. \tag{8}$$

Minimizing the line cost (6) subject to a low value for the angle desirability cost (8) will ensure that the solution lies in the angle constrained subspace shown in Fig. 3.

Length Desirability Cost To avoid the vertical scaling ambiguity, we add a cost to control the length of the RS corrected curves due to only r_x. Even though r_z also affects the length of the row-span of lines, it does not lead to complete vertical shrinkage of lines, since r_z is controlled by angle cost. Let (x_{ij}^r, y_{ij}^r) be the RS corrected point of (x_{ij}, y_{ij}) using only $\widehat{r}_x(y)$ considering $\widehat{r}_y(y)$ and $\widehat{r}_z(y)$ to be 0. Then, the length of the row span of the $\widehat{r}_x$-corrected $\mathbf{c}_i$ is given by $u_i^r = \max_j(y_{ij}^r) - \min_j(y_{ij}^r) + 1$. The length desirability cost is thus written as

$$E^{\text{len}} = \frac{1}{N_c} \sum_{i=1}^{N_c} (u_i - u_i^r)^2, \tag{9}$$

where u_i is the length of the RS curve $\mathbf{c}_i$ as defined earlier. A low value of this cost limits arbitrary vertical scaling, thus restricting the solution to lie within the length constrained subspace indicated in Fig. 3.

Optimization We estimate the polynomial coefficients of the camera motion in an iterative manner by minimizing the line desirability cost (6) subject to the constraints that the angle cost (8) and the length cost (9) be smaller than preset thresholds. In Fig. 3, this is equivalent to locating the intersection of the angle and length constrained subspaces. Thus, we have

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \left\{E^{\text{line}}\right\} \text{ subject to } E^{\text{ang}} < \epsilon_1, E^{\text{len}} < \epsilon_2, \tag{10}$$

where we set $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 1$. Initialized with zero camera motion, the constrained nonlinear least squares problem (10) is solved iteratively which converges to a final solution $\boldsymbol{\alpha}^*$. We use the `fmincon` function in MATLAB to solve (10). Corresponding to the polynomial coefficient vector $\boldsymbol{\alpha}^*$, the resultant rotation matrix $\mathbf{R}^*(y)$ can be obtained for any y using (4).

To avoid global perspective correction in certain images, zeroth degree motion parameters (global r_x and r_y rotation) can be left out during optimization, and only the coefficients from first degree could be estimated for these two rotations. Employing global r_z in optimization will derotate slanted lines to be vertical (which is visually pleasing) even if the user captures with z-rotation. Not estimating r_x at all would leave small curvature in slanted lines, though the result might be visually acceptable in most scenarios. We estimate only nonlinear r_x trajectory; linear r_x motion (which does not cause curvature) cannot be estimated without very strong priors (e.g. building windows are square).

2.4. Image Rectification

Once the camera motion is estimated, our task is to estimate an intensity for every pixel of the GS image (i.e. the rectified image). The size of the GS image is assumed to be the same as that of the RS image. A forward mapping procedure could be applied using (3) to map intensities from RS pixel locations to GS pixel locations. The drawback of this approach is that due to the row-wise camera motion, not all pixels in the GS image might get an intensity value. Hence, we follow an inverse mapping procedure in which we map every GS pixel to an RS coordinate through the polynomial camera motion.

For every pixel $\mathbf{x}_{GS} = (x', y')$, we find a real number y^* such that applying the motion corresponding to y^* on $\mathbf{x}_{GS}$ (using the estimated motion $\mathbf{R}^*(y^*)$ obtained from $\boldsymbol{\alpha}^*$) would result in vertical RS coordinate y^* after warping. This is a nonlinear least squares problem of a continuous variable, and we use Levenberg-Marquardt algorithm [20] to solve for y^*. This algorithm is iterative, and is initialized with y' for faster convergence. Let this estimated RS coordinate location be (x^*, y^*). We then assign the intensity $I_{RS}(x^*, y^*)$ from the RS image as the intensity $\widehat{I}_{GS}(x', y')$ in the estimated GS image. We bilinearly interpolate the intensity values of four nearest integer locations while arriving at $I_{RS}(x^*, y^*)$. The rectification steps are summarized in Algorithm 1.

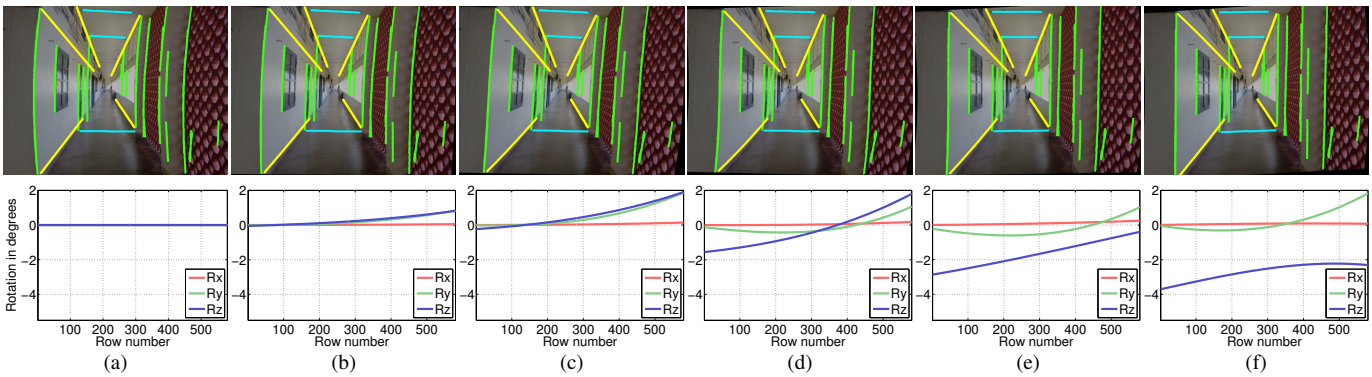


Figure 4. *Results of our optimization over iterations* (best viewed in PDF). (a) first iteration, (b-e) intermediate iterations, and (f) final iteration. *Top row.* Rectified images using the current motion estimate overlaid with vertical (green), horizontal (blue), and slanted (yellow) edges. *Bottom row.* Estimated camera trajectories.

Algorithm 1 RS rectification using inverse mapping.

Inputs: Estimated motion $\mathbf{R}^*$, RS image $\mathbf{I}_{RS}$

for all $\mathbf{x}_{GS} = (x', y')$ in GS image **do**

$\widetilde{\mathbf{x}}_{GS} \leftarrow [x', y', 1]^T$

$y^* = \arg\min_y (y - \overline{y})^2$, where

$$\begin{cases} \widetilde{\mathbf{x}}_{RS} \leftarrow \mathbf{K}\mathbf{R}^*(y)\mathbf{K}^{-1}\widetilde{\mathbf{x}}_{GS}, \\ \mathbf{x}_{RS} = (\overline{x}, \overline{y}) \leftarrow [\widetilde{x}_{RS}(1), \widetilde{x}_{RS}(2)]/\widetilde{x}_{RS}(3) \end{cases}$$

$\widetilde{\mathbf{x}}_{RS} \leftarrow \mathbf{K}\mathbf{R}^*(y^*)\mathbf{K}^{-1}\widetilde{\mathbf{x}}_{GS}$, and

$\mathbf{x}_{RS} = (x^*, y^*) \leftarrow [\widetilde{x}_{RS}(1), \widetilde{x}_{RS}(2)]/\widetilde{x}_{RS}(3)$

$\widehat{I}_{GS}(x', y') \leftarrow I_{RS}(x^*, y^*)$

end for

Output: GS image $\widehat{\mathbf{I}}_{GS}$

3. Experiments

In this section, we demonstrate our motion estimation and image rectification results. We also evaluate the performance of the polynomial model against other existing models for an RS image affected by camera motion in the publicly available handshake dataset [17]. Finally, we compare the performance of our single image method against video as well as nonblind RS rectification methods.

Motion Estimation and Rectification The RS image in Fig. 1(a) was taken with a MotoG2 mobile phone. Fig. 4(a) shows the detected curves in vertical (green), horizontal (blue), and slanted (yellow) directions. The vertically oriented edges are curved the most, followed by slanted curves, while the horizontal curves are the least affected. The blue lines are not exactly horizontal in this case. We solve (10) using the edge pixels along the curves to estimate camera motion that straightens the curves while respecting angle and length constraints. The RS image is rectified using the final motion estimate as described in Algorithm 1. The rectified image, that was shown in Fig. 1(b), is reproduced in Fig. 4(f) but with overlaid edges. Observe that the RS curves are successfully corrected with green lines aligned to vertical, blue lines aligned to horizontal, and yellow lines retaining their straightness, without disrupting global vertical scaling.

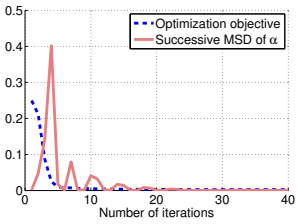


Figure 5. The line cost reduces in each iteration. Initially, the mean squared difference between $\boldsymbol{\alpha}$ of successive iterations is large when the optimization takes large steps, and it decreases when approaching the final solution.

In Fig. 4, we also show the motion trajectory estimates and the corresponding rectified images over iterations while solving (10). The algorithm initially corrects the curvatures by varying the rotational motion trajectories. It then proceeds to vary the inplane rotation r_z to impose orthogonality between vertical and horizontal lines. The blue lines in Fig. 4(f) are closer to horizontal than in Fig. 4(a). The estimated r_x motion is minimal without affecting vertical scale, and at the same time, suffices to correct the curvature of slanted lines. It is therefore clear that our method is able to rectify the distortions due to RS effect. The variation of the objective value in (10) and the estimated $\boldsymbol{\alpha}^*$ over iterations is shown in Fig. 5. The importance of RS rectification for geometric analysis based on vanishing points is provided in the supplementary material.

Comparison of Motion Models We have modelled the camera trajectory during sequential exposure of rows by a polynomial. This is unlike other works that use weighted row-wise motion based on the motion of certain fixed rows (known as key rows). To study the suitability of the polynomial model, we compare its performance against two other RS models, namely, spherical linear interpolation model [26] (with five equally spaced key rows) and Gaussian interpolation model [10] (with ten equally spaced key rows). We generate the RS image using a camera trajectory

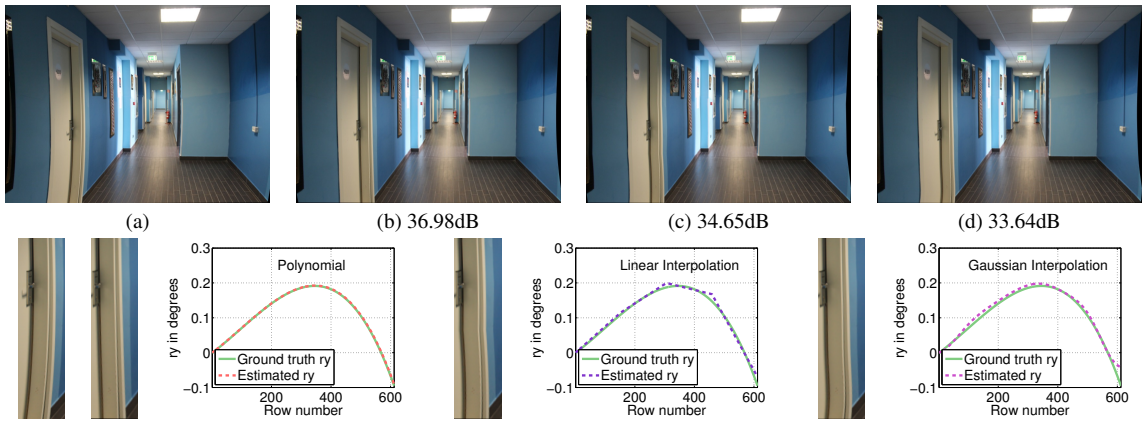


Figure 6. *Motion model analysis.* Top row. (a) RS image simulated using handshake motion from the dataset of [17], rectified images using (b) polynomial, (c) linear interpolation [26], and (d) Gaussian interpolation [10] models. Bottom row shows zoomed-in patches and the plots of ground truth and estimated trajectories.

picked from the handshake camera motion dataset of [17]. We employ the same cost function for all the three models to estimate camera motion.

The RS image, and the rectified images using polynomial, linear, and Gaussian interpolation models are shown, respectively, in Figs. 6(a), (b), (c), and (d). The estimated motion trajectories are shown in Figs. 6(b2), (c2), and (d2), in which the continuous green path denotes the ground truth trajectory and the dotted paths denote estimated trajectories. Note that the polynomial model closely follows the handshake dataset motion, while the linear interpolation model uses piecewise approximation, and the Gaussian interpolation model exhibits a residual throughout the trajectory. The average angular error for $r_y \in (-0.1, 0.2)^\circ$ between the ground truth and estimated trajectories for the three models are 0.0012°, 0.0050°, 0.0069°, respectively. The polynomial model exhibits the least estimation error. Though the errors for the other two models are small in value, the residual effect is strikingly visible in the rectified image. The zoomed-in patch in Fig. 6(a1) shows a high curvature region of the RS image. While the rectified patch (b1) corresponding to the polynomial model is visually better, the other two patches (c1) and (d1) show wavy artifacts.

Comparison with Video and Nonblind RS Rectification To the best of our knowledge, there are no existing works that deal with RS rectification given a single image. While [31] uses a single image, it works only in the presence of motion blur. Nevertheless, we compare our method against two contemporary video RS rectification and stabilization methods [26, 10] and a non-blind GS-RS registration method [25]. For [26] and [10], we use the RS video dataset (captured using mobile phones) as well as the outputs provided by them. For [25], the authors sent us their estimated camera motion between the GS and RS images that we captured using a Google Nexus 4 phone and made available to them. Subsequently, we rectified the RS image using our Algorithm 1 (instead of the polynomial model, we directly used their estimated row-wise motion). We must mention here that the aim of these three comparisons is to only gauge whether our blind RS image rectification measures up to the performance of these video and non-blind methods, and not whether our proposed method outperforms them. While [26] and [10] use an RS-affected video sequence, [25] needs the original GS image. In contrast, our method is blind and works on a single RS image.

Fig. 7(a) shows a frame of an RS video from the dataset of [26]; this video is chosen as it has heavy RS distortions. The video suffers mainly from inplane rotations. The vertical posts in the scene appear bent during the capture. The method of [26] rectifies and stabilizes the video, and the output frame corresponding to (a) is shown in Fig. 7(b), in which the bent poles are correctly straightened. The global shift is due to video stabilization. In our proposed method, we use only the image (a) to detect curves and to estimate motion. Our single image RS rectification output is shown in Fig. 7(c). It can be seen that the performance of our method is comparable to [26] in correcting the RS curves.

A skew-distorted frame from an RS video of the dataset from [10] is shown in Fig. 7(d). The corresponding frame from the rectified video [10] is shown in Fig. 7(e), and the result of our single image rectification is shown in Fig. 7(f). Note that our method corrects the distortions very well; slanted pillars are correctly rendered as vertical.

Finally, we compare our method with the nonblind rectification of [25]. We captured two images of the same scene, one without motion (GS image in Fig. 7(g)) and one with motion (RS image in Fig. 7(h)). We then estimated the camera motion using [25], and corrected the RS effect using Algorithm 1. The nonblind rectified image is shown in Fig. 7(i). We then use our proposed method to esti-

(a) (b) (c) (g) (h)

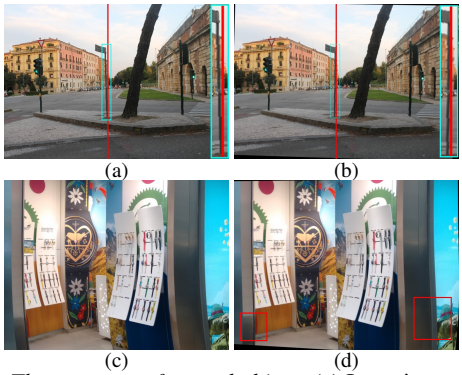
(d) (e) (f) (i) 34.92dB (j) 32.90dB

Figure 7. *Comparison with video rectification.* (a) RS video frame, (b) rectified video frame [26], (c) our single image rectification output, (d) RS video frame, (e) rectified video frame [10], and (f) our single image rectification output. *Comparison with non-blind rectification.* (g) GS reference image, (h) RS image, (i) rectified output through the camera motion estimated between (g) and (h) using [25], and (j) our single image RS rectification output of (h).

mate motion from only the RS image, and our blind rectification output is shown in Fig. 7(j). The performance of our method is clearly on par with the two-image nonblind method. To quantify the rectification, we calculated the error during global homography estimation between the GS image and the other three images using 4-point RANSAC estimation based on SIFT correspondences [21]. The errors for the RS, nonblind, and blind rectified images are 1.4226, 0.3031, and 0.3854, respectively. Our blind rectified output matches the GS image through a global homography (since the row-wise variations are rectified) as good as the nonblind rectified image.

Presence of Curved Objects Although our method assumes the scene to contain man-made structures with straight lines, it can handle the presence of few natural objects too. Fig. 8(a) shows an image of a scene with a tree that is naturally curved and jagged. The RS effect is visually less in this image, but there is a vertical misalignment as can be seen from the marked red line. The presence of the tree affects our cost minimization only marginally, since there are a number of other lines that exert a greater influence on the cost. The rectification rotates the image to closely align with the vertical as shown in Fig. 8(b). Fig. 8(c) shows an RS image with many natural curves (white boards) and our method leaves some RS residuals in the rectified image as shown in Fig. 8(d). We have further discussed the effect of the presence of curves and curve breaks in the supplementary material.

Run-time We implemented our method in MATLAB with warping operations accelerated by C-mex. Our method takes approximately 25 seconds for curve detection, 10 seconds to estimate camera motion, and 5 seconds for RS rectification, for an 816x612 image on a 3.4GHz machine with 8GB RAM. More RS rectification examples are provided in the supplementary material.



(a) (b) (c) (d)

Figure 8. *The presence of curved objects* (a) Input image containing a tree that is misaligned with the vertical axis, and (b) rectified image. (c) An RS image with many natural curves (white boards), and (d) rectified image with incomplete rectification.

Acknowledgements We thank the reviewers for their valuable comments. The first author thanks Subeesh Vasu for his help in running some comparison methods.

4. Conclusions

In this paper, we handled the challenging task of correcting RS distortions from a single image of urban scenes. Using the proposed curve detection procedure, we automatically picked good lines and curves as features. We then formulated an optimization problem based on line, angle, and length desirability costs on these features to solve for the underlying (rotation-only) camera motion using which we finally rectify the RS image through inverse mapping. With video and nonblind RS rectification methods being the state-of-the-art, our work opens up new vistas for tackling the RS effect in single images. Experiments reveal that our method, despite being single-image based, performs commendably against existing multi-image methods.

References

[1] O. Ait-Aider, N. Andreff, J. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Computer Vision - ECCV 2006*, volume 3952 of LNCS, pages 56–68. Springer Berlin Heidelberg, 2006.

[2] C. Albl, Z. Kukelova, and T. Pajdla. R6P-rolling shutter absolute camera pose. In *Computer Vision and Pattern Recognition*, pages 2292–2300, 2015.

[3] M. Alemán-Flores, L. Alvarez, L. Gomez, and D. Santana-Cedrés. Line detection in images showing significant lens distortion and application to distortion correction. *Pattern Recognition Letters*, 36:261–271, 2014.

[4] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *Computer Vision and Pattern Recognition*, pages 2392–2399. IEEE, 2010.

[5] F. Bukhari and M. N. Dailey. Automatic radial distortion estimation from a single image. *Journal of mathematical imaging and vision*, 45(1):31–45, 2013.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[7] E. Delage, H. Lee, and A. Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *Robotics Research*, pages 305–321. Springer, 2007.

[8] F. Devernay and O. Faugeras. Straight lines have to be straight. *Machine Vision and Applns.*, 13(1):14–24, 2001.

[9] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[10] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Calibration-free rolling shutter removal. In *International Conference on Computational Photography*, pages 1–8. IEEE, 2012.

[11] G. Hanning, N. Forslow, P.-E. Forssén, E. Ringaby, D. Tornqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *International Conference on Computer Vision Workshops*, pages 1–8. IEEE, 2011.

[12] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[13] D. Hoiem, A. Efros, M. Hebert, et al. Geometric context from a single image. In *International Conference on Computer Vision*, volume 1, pages 654–661. IEEE, 2005.

[14] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. on Graphics*, 24(3):577–584, 2005.

[15] V. Hough and C. Paul. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654.

[16] C. Jia and B. L. Evans. Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *International Workshop on Multimedia Signal Processing*, pages 203–208, 2012.

[17] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *Computer Vision–ECCV 2012*, pages 27–40. Springer, 2012.

[18] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition*, pages 2136–2143. IEEE, 2009.

[19] C.-K. Liang, L.-W. Chang, and H. H. Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330, 2008.

[20] M. Lourakis. levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++. In *[web page] http://www.ics.forth.gr/∼lourakis/levmar/*.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] 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 *Computer Vision–ECCV 2012*, pages 456–469. Springer, 2012.

[23] S. H. Park and M. Levoy. Gyro-based multi-image deconvolution for removing handshake blur. In *Computer Vision and Pattern Recognition*, pages 3366–3373. IEEE, 2014.

[24] A. Patron-Perez, S. Lovegrove, and G. Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, pages 1–12, 2015.

[25] V. Pichaikuppan, R. Narayanan, and A. Rangarajan. Change detection in the presence of motion blur and rolling shutter effect. In *Computer Vision - ECCV 2014*, volume 8695 of *LNCS*, pages 123–137. Springer, 2014.

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

[27] A. Rosenfeld. Picture processing by computer. *ACM Computing Surveys*, 1(3):147–176, 1969.

[28] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *International Conference on Computer Vision*, pages 465–472. IEEE, 2013.

[29] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[30] O. Sindelar, F. Sroubek, and P. Milanfar. A smartphone application for removing handshake blur and compensating rolling shutter. In *International Conference on Image Processing*, pages 2160–2162. IEEE, 2014.

[31] S. Su and W. Heidrich. Rolling shutter motion deblurring. In *Computer Vision and Pattern Recognition*, pages 1529–1537, 2015.

[32] Y. Sun and G. Liu. Rolling shutter distortion removal based on curve interpolation. *IEEE Transactions on Consumer Electronics*, 58(3):1045–1050, 2012.

[33] E. Tretyak, O. Barinova, P. Kohli, and V. Lempitsky. Geometric image parsing in man-made environments. *International Journal of Computer Vision*, 97(3):305–321, 2012.

[34] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International journal of computer vision*, 98(2):168–186, 2012.

[35] Z. Zhang, Y. Matsushita, and Y. Ma. Camera calibration with lens distortion from low-rank textures. In *Computer Vision and Pattern Recognition*, pages 2321–2328. IEEE, 2011.

Supplementary Material

From Bows to Arrows: Rolling Shutter Rectification of Urban Scenes

In this supplementary document, we show how RS distortions could affect geometric inference through vanishing point analysis. We then provide an analysis of the performance of our method in the presence of curve breaks and in the presence of natural curves as well. We also provide additional insights for a rectification example that we showed in the main paper. To show the effectiveness of our method, we also show additional RS rectification examples.

1. Geometric Analysis

To show how the RS distortions could affect geometric inference, we show an analysis based on the vanishing point. Parallel lines in a scene should converge in the image to a common vanishing point [12]. The RS effect disturbs the straightness of lines, and hence affects vanishing point estimation. We use this estimation error to quantify rectification. To avoid making all the lines parallel, we removed the zeroth degree parameter (i.e. the DC value) in (4) during r_x and r_y estimation. To estimate the vanishing point, we first fit lines to vertically oriented curves (of RS and rectified images, separately) and determine the point which minimizes the orthogonal distance to all these lines. As there is no ground truth image, we use this minimum distance as our vanishing point estimation error. The lower this distance, the better is the rectification.

(a) RS image (b) Rectified image

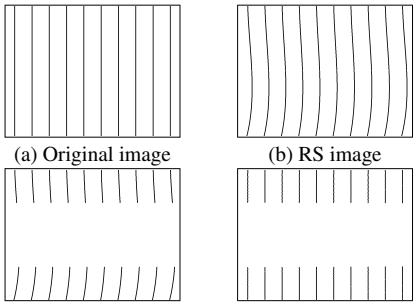
Figure 9. Vanishing point estimation.

Fig. 9(a) shows an RS image of a building with curved vertical edges. The fitted lines to these curves are marked in red. Correspondingly, Fig. 9(b) shows our rectification output with rectified curves and fitted lines. The fitted lines after rectification align more closely with the vertical edges as compared to those before rectification (compare zoomed-in patches). This fitting error due to curvature leads to a resultant error distance of 12.1 pixels for the RS image, but a lower value of 5.1 pixels for the rectified image during vanishing point estimation. This shows that the RS effect must be handled correctly to study scene geometry, and an RS rectification method such as ours will be pivotal.

2. Presence of Curve Breaks

Since our motion estimation method uses curves as features, we conduct an experiment in which we break the curves to create discontinuities and test the performance.

For this experiment, we consider an image of size 816x612 in which there are ten vertical lines spaced equally apart (Fig. 10(a)). We then introduced an r_y motion to transform these lines into curves (Fig. 10(b)). In those curves, we introduced breaks by removing pixels from a block of rows of length b in the middle (Fig. 10(c)). After this process, the number of RS curves becomes 20; the top and bottom broken parts of each of the original curves are treated as two curves. Using these discontinuous curves, we estimate the camera motion r_y (fixing r_y of the top row as 0) and then rectify the broken curves. The rectified image is shown in Fig. 10(d)). It is clear that these additional discontinuities in the curves do not significantly affect our RS rectification method.



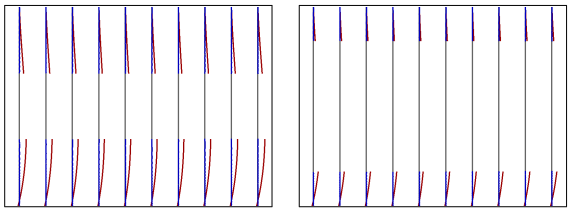
(a) Original image (b) RS image

(c) RS image with curve breaks (d) Rectified image

Figure 10. RS rectification in the presence of curve breaks.

Further, we vary the break-length b from 0 to 550 (total curve length is 612), and study the performance both in terms of the motion estimate and the rectified output. In addition, this experiment will reveal how well our method identifies the association between the top and bottom broken parts of curves. Fig. 11 shows our rectification for break lengths $b = 200$ and 400 (corresponding to 67.3% and 34.6% of total information, respectively, for 612 rows). The RS curves are shown in red and the rectified curves are shown in blue. The rectification of curves as lines is correct, and there is no association-disconnect between the broken parts too; both the top and bottom curve segments are corrected back to the same original line. The estimated motion trajectories for different break-lengths (for $b = 0$ to 550)

are shown as dotted paths in Fig. 12. The ground truth trajectory is shown as continuous green path in Fig. 12. The estimated trajectory closely follows the ground-truth trajectory till $b = 400$. This indicates that our method performs very well, identifying correct curve segment associations, even when only 34.6% of the curve information is present.



$b = 200$ $b = 400$

Figure 11. *RS rectification in the presence of curve breaks* (best viewed in PDF). The gray color denotes original lines, the red color denotes RS curves, the blue color denotes rectified curves. Here, b is the break-length of curves.

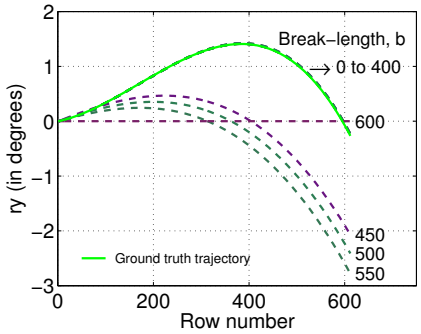
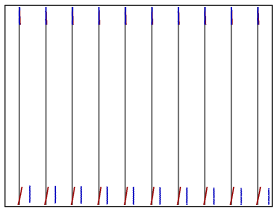


Figure 12. Motion estimation in the presence of curve breaks.

A further increase in the break length (which may not be observable in real scenarios though) causes our method to disassociate the top and bottom curve segments. Fig. 13 shows our rectification for break length $b = 500$. The available amount of information is only 18.3%. The rectification is proper in which all the curve segments are transformed as lines. With no availability of information in a large number of rows in the middle, our estimated motion deviates from the ground-truth for $b > 400$ as can be seen in Fig. 12. Since we fix the r_y of the top row to be 0, the estimated path follows the ground-truth for the first few rows rectifying the curve segments at the top as same lines as in the original image. It is important to note that the curvature of the estimated trajectory for the last few rows is very similar to that of the ground-truth trajectory. This is, in fact, the reason that the curve segments at the bottom (red curves in Fig. 13) do get rectified as lines (blue lines in Fig. 13). Due to the unavailability of information in most of the middle rows, our method fits a different camera path from that of the ground-truth, and hence the rectified lines have shifted a little from the original lines (gray lines in Fig. 13).



Break length, $b = 500$

Figure 13. *RS rectification in the presence of curve breaks* (best viewed in PDF).

Though long curves are favored as inputs for our motion estimation method, as we have discussed in this section, our method performs very well even in the presence of curve breaks correcting them as lines and associates the broken curve parts correctly even when only 34.6% of the information is available.

3. Presence of Natural Curves

A scene naturally containing many curves might pose issues during optimization, since our cost formulation primarily assumes that the RS curves correspond to lines in the original scene. In the main paper, we showed an example of a scene containing a tree. Our method performed very well, since there were a number of other straight lines that influenced the cost. In this section, we analyze the effect of such natural curves in our method under synthetic conditions.

We consider a total of ten lines and curves in the scene. The natural curves are generated without conforming to any property mutually. The RS effect causes all the ten structures to become curves. We then rectify the RS image and test its performance based on the error in the rectified curves that correspond to straight lines in the original scene. We study this behavior by increasing the number of natural curves $n = 0$ to 9. The first case $n = 0$ is the no-natural-curve case, and the last case $n = 9$ is the all-but-one-are-curves case. The RS and rectified images (for $n = 0, 2, 4, 6$) are shown in Fig. 14. As n increases, the performance of RS rectification decreases. In Fig. 14(a), there are no natural curves, and hence the rectification is perfect. The rectification performance of valid lines is visually good till $n = 4$ (Fig. 14(c)). In (d), there is a clear visual deviation of the rectified line from the original line.

Fig. 15 shows a quantitative analysis. In the left, we show the variation of average absolute angular error during motion estimation with respect to n, and in the right, we show the variation of mean squared error deviation of the rectified curves from original lines. Both the angular error and the rectification error exhibit an increasing behavior with increasing n. Note that the rectification error is under three pixels till $n = 4$. This shows that our method is robust to the presence of natural curves to a good extent.

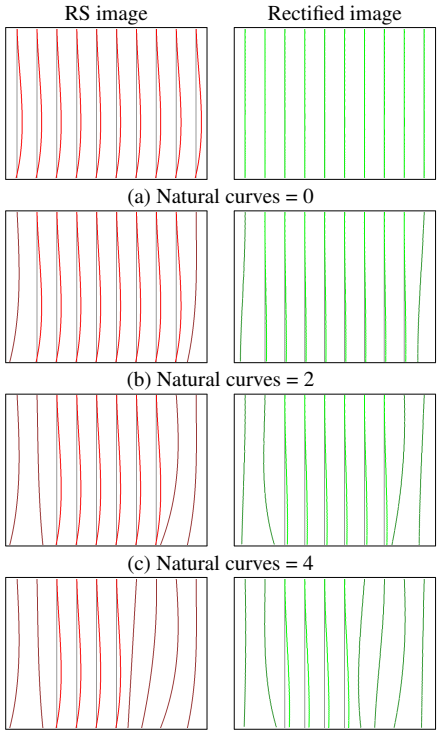


Figure 14. *RS rectification in the presence of natural curves* (best viewed in PDF). The red color denotes RS curves, and the green color denotes rectified curves. Dark and light color variations correspond to natural curve and straight line, respectively. The gray color indicates vertical lines in the original undistorted image.

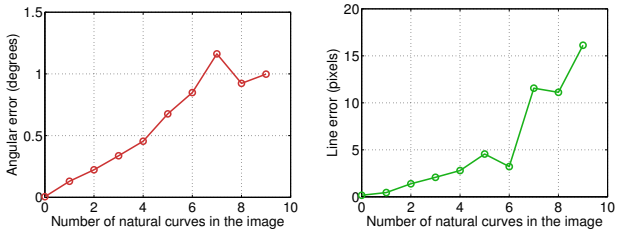
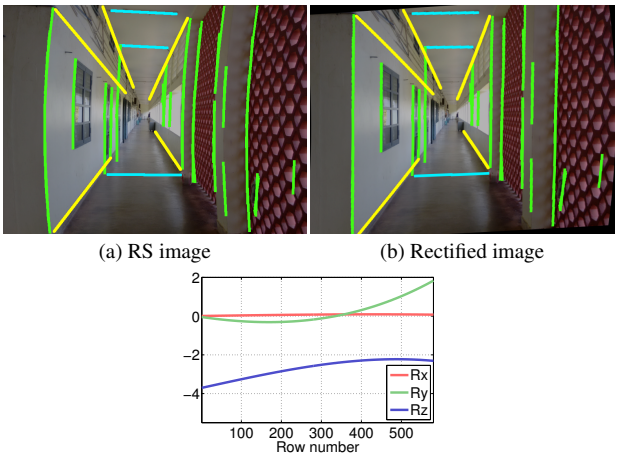


Figure 15. Motion estimation in the presence of natural curves.

4. Additional Insights

Due to space constraints, we were not able to discuss more about our motion estimation in the main paper. In Fig. 16, we show the RS and rectified images overlaid with curves and lines (repeated from Fig. 4 of the main paper). As already noted, all curves are transformed into lines. In addition, we would like to point out that the lines that are already straight in the RS image are not disturbed during rectification. The green lines closer to the image center in Fig. 16(a) are, in fact, not curved significantly due to camera motion; only the long green lines near the periphery are bent. Our cost formulation implicitly considers these aspects and estimates a camera motion that straightens only the curves leaving the already-straight lines in the RS image intact. In this example, the camera motion consists predominantly of inplane rotation r_z, and the effect of which increases when going away from the center. Adhering to this behavior, our optimization results in the correct motion trajectory which transforms both curves and lines in the RS image as lines in the rectified image. Hence, it is important to detect and exploit lines also (in addition to curves) in the RS image, without which an incorrect motion might be estimated that could disturb lines that are already straight in the RS image. The rectified outputs for this RS image over iterations during optimization is included as a video.



(a) RS image (b) Rectified image

(c) Estimated camera trajectory

Figure 16. RS rectification and motion estimation.

5. Additional Examples

In this section, we show some more single image RS rectification outputs (in addition to the examples in the main paper). In the examples shown in Fig. 17, we can observe that the RS effect has been rectified by our method successfully. In the last example in Fig. 17, the RS effect of the pillars are rectified correctly, but a small residual RS effect is visible along the gray wall edge in the right-part of the image. This may be due to the presence of small translatory motion (which is depth-dependent) that we do not account for in this work.

Figure 17. More examples of our RS rectification.