

# Minimal Solvers for Monocular Rolling Shutter Compensation under Ackermann Motion

Pulak Purkait, Christopher Zach  
Toshiba Research Europe, Cambridge, UK  
pulak.isi@gmail.com

## Abstract

*Modern automotive vehicles are often equipped with a budget commercial rolling shutter camera. These devices often produce distorted images due to the inter-row delay of the camera while capturing the image. Recent methods for monocular rolling shutter motion compensation utilize blur kernel and the straightness property of line segments. However, these methods are limited to handling rotational motion and also are not fast enough to operate in real time. In this paper, we propose a minimal solver for the rolling shutter motion compensation which assumes known vertical direction of the camera. Thanks to the Ackermann motion model of vehicles which consists of only two motion parameters, and two parameters for the simplified depth assumption that lead to a 4-line algorithm. The proposed minimal solver estimates the rolling shutter camera motion efficiently and accurately. The extensive experiments on real and simulated datasets demonstrate the benefits of our approach in terms of qualitative and quantitative results.*

## 1. Introduction

Recently, automotive driver-less vehicles have sparked a lot of vision research and unlocked the demands for the real time solutions for a number of unsolved problems in computer vision. While a commercial budget camera is an attractive choice of these vehicles, a significant distortion could be observed in the captured images. These cameras are generally built upon CMOS sensors, which possess a prevalent mechanism widely known as *rolling shutter* (RS). In contrast to *global shutter* (GS) camera, it captures the scene in a row-wise fashion from top to bottom with a constant inter-row delay. The RS imaging acquires apparent camera motion for different rows and violates the properties of the perspective camera model. This causes noticeable and prominent distortions.

In this work, we develop a minimal solver for RS compensation from a *single* distorted image. The monocular

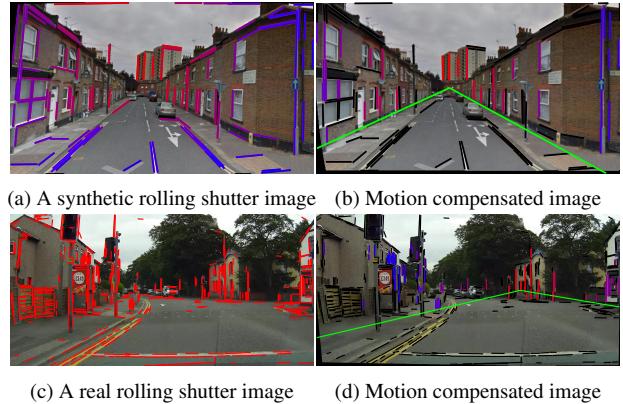


Figure 1: (a) A synthetic image is generated by simulating the Ackermann motion and (c) a real rolling shutter image captured by GS900 2.7" CMOS car camera. (b) and (d) are motion compensated image and the estimated scene depth by proposed method. Red line segments are at a high depth compared to the blue ones. Black line segments are the detected outliers, and green lines are the estimated road boundaries.

lar RS compensation has been addressed in recent methods [26, 21, 20], however, none of these methods incorporate the translational part of the ego-motion. Moreover, these methods are not fast enough to operate in real time. In contrast, we propose a fast minimal solver for rolling shutter motion compensation. The proposed algorithm is tailored for the *Ackermann motion*, under the known vertical direction. *This is a common scenario for automotive vehicles moving on a horizontal plane.* In conjunction with RANSAC, the proposed solver is executed in real time. In Figure 1, we display our result on a synthetic distorted image<sup>1</sup> and on a real rolling shutter image. The result indicates that the proposed method estimates the rolling shutter camera motion quite accurately along with the image depth and

<sup>1</sup>Image source: Google street view, Cambridge <https://www.google.com/streetview/>

the road boundaries.

## 1.1. Related Work

The existing RS compensation methods can be categorized into multi-frame based and single-frame based methods. External hardware devices have been utilized [14, 15, 19] to acquire camera motion directly in videos. However, these methods deal with the rotational motion only. The geometry of the scene has also been exploited [2, 5, 7, 10, 22, 18, 8] using multiple RS images. These methods detect a number of interest points and track over the frames. The tracked points are then exploited to estimate the camera motion. The camera poses for the other rows are then interpolated in order to compensate the RS motion. Nonetheless, none of these methods can directly be applied to Single-frame RS compensation.

Su *et al.* [26] utilize motion blur to extract information about the camera motion from a single RS frame. Rengarajan *et al.* [21] utilize the straightness property of the detected arc segments, the length constancy and the angle constancy to estimate the motion. However, [21, 26] do not incorporate the translation motion which cannot be disregarded in the Ackermann motion. Moreover, these methods require a nonlinear optimization which has a high runtime complexity.

In this work, we estimate the Ackermann motion under the known vertical direction. The proposed method also estimates an approximate scene depth while estimating the motion parameters. Moreover, our method is free from the aforementioned drawbacks. The contributions are summarized as follows:

- A minimal solver is developed by utilizing vertical line segments to compensate the rolling shutter camera motion. The proposed algorithm is tailored for the Ackermann motion principle which is the common scenario for automotive vehicles.
- Extensive experiments on simulated data show the effectiveness of the proposed approach—computationally and qualitatively.

## 2. Rolling Shutter Camera

Global shutter and rolling shutter cameras differ in how the image sensors are exposed while capturing the photographs. In the former case, the light strikes at all the rows of the image sensor simultaneously for a constant duration of time. In the latter case, each row is exposed for a regular interval of time, while the camera undergoes a (small) motion.

Let  $\mathbf{P} \in \mathbb{R}^3$  be a 3D point in space w.r.t. the GS camera coordinates. Then the position of the point  $\mathbf{P}$  w.r.t. RS camera coordinates [Fig. 2(a)] is

$$\mathbf{P}_{\mathcal{A}} = R_{\mathcal{A}}^t(\mathbf{P} - \mathbf{T}_{\mathcal{A}}^t) \quad (1)$$

which can be written in the normalized pixel coordinates as follows [11, 21]

$$s_{\mathcal{A}}^{rs} K^{-1} \mathbf{p}^{rs} = R_{\mathcal{A}}^t(sK^{-1} \mathbf{p} - \mathbf{T}_{\mathcal{A}}^t), \quad (2)$$

where  $\mathbf{p} = [p_1, p_2, 1]^T$  and  $\mathbf{p}^{rs} = [p_1^{rs}, p_2^{rs}, 1]^T$  are the homogeneous pixel coordinates of the pixel  $(p_1, p_2)$  of the GS and RS cameras respectively.  $s$  and  $s_{\mathcal{A}}^{rs}$  are corresponding scene depths.  $K$  is the intrinsic camera matrix.  $R_{\mathcal{A}}^t$  and  $\mathbf{T}_{\mathcal{A}}^t$  are the rotation and translation of the vehicle at time  $t = \tau p_2^{rs}$  where  $\tau$  is the time delay between two successive rows.  $\mathcal{A}$  is the Ackermann motion parameters. For readability, in the rest of the paper, we consider  $\mathbf{p}$  and  $\mathbf{p}^{rs}$  are on the image plane, i.e., pre-multiplied by  $K^{-1}$ . Thus (2) becomes

$$s\mathbf{p} = s_{\mathcal{A}}^{rs} R_{\mathcal{A}}^{t^T} \mathbf{p}^{rs} + \mathbf{T}_{\mathcal{A}}^t. \quad (3)$$

where  $(R_{\mathcal{A}}^t)^T$  is the transpose of  $R_{\mathcal{A}}^t$ . Note that the scene depth  $s_{\mathcal{A}}^{rs}$  varies with the pixels and also with the camera motion.

## 3. Modeling Ackermann motion

The conventional Ackermann steering principle [23, 24, 25] is to have all the four wheels of a vehicle rolling around a common point during a turn. This principle holds for any automotive vehicle which ensures all the wheels exhibit a rolling motion. This common point is widely known as the Instantaneous Center of rotation (ICR) and is computed by intersecting all the roll axis of the wheels. In Figure 2, we demonstrate such motion. The radius of the circular motion goes to infinity under a pure forward motion.

### 3.1. Motion model

The vehicle is assumed to undertake a fixed translational and angular velocity while capturing the image. If it takes a pure circular motion with a rotation angle  $\theta^t$ , then, the vehicle must satisfy the circular motion constraints  $\phi^t = \frac{\theta^t}{2}$  [Fig. 2]. Note that here the  $x$ -axis is taken along the radius towards the ICR. The relative pose [12, 13] between the vehicles (previous and current position) can be written as

$$R_{\mathcal{A}}^t = \begin{bmatrix} \cos \theta^t & 0 & -\sin \theta^t \\ 0 & 1 & 0 \\ \sin \theta^t & 0 & \cos \theta^t \end{bmatrix}, \quad \mathbf{T}_{\mathcal{A}}^t = \begin{bmatrix} \rho^t \sin \frac{\theta^t}{2} \\ 0 \\ \rho^t \cos \frac{\theta^t}{2} \end{bmatrix} \quad (4)$$

where  $\theta^t$  is the relative yaw angle and  $\rho^t$  is the displacement of the vehicle at time  $t$ . Substituting in (3), we obtain the scene depth

$$s = s_{\mathcal{A}}^{rs} (\cos \theta^t - p_1^{rs} \sin \theta^t) + \rho^t \cos \frac{\theta^t}{2} \quad (5)$$

which can also be verified by Figure 2(a). Let the vehicle undergo a circular motion with an angular velocity  $\alpha$  and a

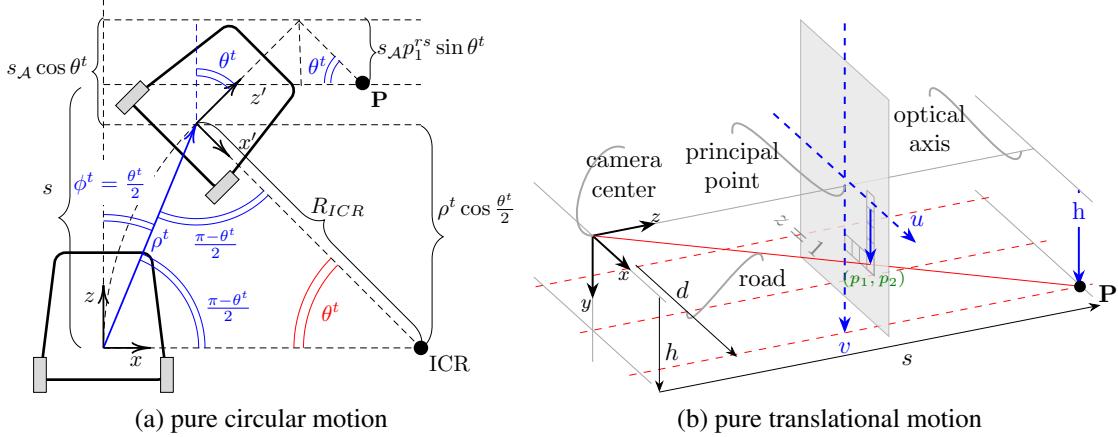


Figure 2: Ackermann motion: (a) pure circular motion and (b) pure translational motion—it also shows the relation between the scene depth and the pixel coordinates.

translational velocity  $\beta$  on the horizontal plane. Under the assumption,

$$R_{\mathcal{A}}^t = \begin{bmatrix} 1 - 2\alpha^2 t^2 & 0 & -2\alpha t \gamma_t \\ 0 & 1 & 0 \\ 2\alpha t \gamma_t & 0 & 1 - 2\alpha^2 t^2 \end{bmatrix}, \quad \mathbf{T}_{\mathcal{A}}^t = \beta t \begin{bmatrix} \alpha t \\ 0 \\ \gamma_t \end{bmatrix}, \quad (6)$$

where  $\alpha^2 t^2 + \gamma_t^2 = 1$  and  $\mathcal{A} = \{\alpha, \beta\}$  is the set of unknown rolling shutter parameters.  $\gamma_t$  was introduced to ensure the unit norm. The scene depth  $s_{\mathcal{A}}^{rs}$  is simplified into  $s_{\mathcal{A}}^{rs} = (s - \beta t) / (1 - 2p_1^{rs}\alpha t)$ . Notice that the terms involving the motion parameters  $\mathcal{A}$  of third and higher orders are ignored. Under small rotations, the rotation matrix can be linearly approximated [2]. Therefore, further dropping the second order terms from (6) (eventually,  $\gamma_t$  becomes 1) and substituting in (3)

$$s\mathbf{p} = \frac{s - \beta t}{1 - 2p_1^{rs}\alpha t} (I_3 + 2[\mathbf{r}_{\mathcal{A}}^t]_{\times}) \mathbf{p}^{rs} + \beta t s \mathbf{s}_{\mathcal{A}}^t \quad (7)$$

where  $[\cdot]_{\times}$  denotes the skew-symmetric cross-product matrix and  $I_3$  is the  $3 \times 3$  identity matrix.  $\mathbf{r}_{\mathcal{A}}^t = [0, \alpha t, 0]^T$  and  $\mathbf{s}_{\mathcal{A}}^t = [\alpha t, 0, 1]^T$  are the angular rotation and the contribution of the circular motion to the translation.

### 3.2. Minimal Solver with known vertical direction

We assume that the vertical direction is known, which is readily available from an external hardware source, e.g., inertial measurement unit (IMU). The accelerometers and gyroscopes of the IMUs provide a precise measure of the orientation of the gravitational acceleration (roll and pitch angles) in the camera frame [3], from which one can easily derive the vertical direction  $\mathbf{e}_y$ . Without loss of generality, we can assume further that the known vertical direction of the camera is also vertical w.r.t. the scene coordinates, i.e.,  $\mathbf{e}_y = [0, 1, 0]$ .

Let  $\mathbf{u}_i = [u_{i1}, u_{i2}, u_{i3}]^T$  and  $\mathbf{v}_i = [v_{i1}, v_{i2}, v_{i3}]^T$  be the motion compensated end points (7) of a line segment  $l_i$ . Then, the normal of the interpretation plane of  $l_i$  is  $\mathbf{u}_i \times \mathbf{v}_i$ . If the  $l_i$  is vertical, then the interpretation plane must pass through the vertical axis, which leads to  $(\mathbf{u}_i \times \mathbf{v}_i)^T \mathbf{e}_y = 0$ , i.e.,

$$u_{i3}v_{i1} - u_{i1}v_{i3} = 0 \quad (8)$$

Substituting in terms of rolling shutter pixel coordinates, the above (8) leads to a polynomial equation with unknown motion parameters  $\mathcal{A}$  and unknown scene depth  $s$ . Apparently, the rolling shutter compensation requires an estimation of the scene depth  $s$  at every individual pixel. In the following section, we employ a simplified parametric representation of the scene depth  $s$ , which is approximately valid for vehicles driving in urban areas.

### 3.3. Parametrization of the depth

The scene is assumed to be composed of two vertical planes (buildings, etc.) and one horizontal plane (road). In Figure 3(a)-(b), we illustrate our scene depth assumptions. The two sides of the roads are approximated by two vertical planes which intersect at the line at infinity. The road is considered as the horizontal ground plane. Then the horizon must pass through the principal point of the camera as the camera is assumed to be vertical. The scene depth at any pixel is considered as the minimum depth of the scene surrounded by the planes.

Let  $(p_1, p_2)$  be the normalized pixel coordinates of a 3D point  $\mathbf{P} \in \mathbb{R}^3$  on the ground plane at a depth  $s$ . By the pinhole camera model [11],  $\frac{p_2}{p_1} = \frac{h}{s}$ , i.e.,  $s^{-1} = h^{-1}p_2$  [Fig. 2(b)], where  $h$  is the height of the camera from the ground plane. Similarly, for the points on any of the vertical planes,  $s^{-1} = d^{-1}p_1$ , where  $d$  is the distance of the vehicle from the vertical plane. In general, for the vehicles facing the vertical

planes sideways,  $s^{-1} = d^{-1}(p_1 - \delta)$  where  $\delta$  is the column corresponds to line at infinity [Fig. 3(b)]. Thus, the inverse of the scene depth  $s^{-1}$  is linear in column number of the pixel in normalized image coordinates.

By our assumptions,  $s^{-1} = \max\{\lambda'(p_1 - \delta), \lambda(p_1 - \delta), \lambda^* p_2\}$  is the inverse depth at the pixel  $(p_1, p_2)$ . The column  $\delta$  corresponds to the line at infinity.  $\lambda', \lambda$  are the inverse of the distances of the camera from the vertical planes.  $\lambda^*$  is the inverse of the height of the camera which is assumed to be known. By the construction, the vertical line segments must lie on any of the vertical planes in 3D represented by  $\lambda', \lambda$ . Thus, (7) becomes

$$\begin{aligned} \mathbf{p} &= \frac{(1-\beta ts^{-1})}{(1-2p_1^{rs}\alpha t)}(I_3 + 2[\mathbf{r}_A^t]_{\times})\mathbf{p}^{rs} + \beta ts^{-1}\mathbf{s}_A^t \\ s^{-1} &= \lambda'[p_1 - \delta]_- + \lambda[p_1 - \delta]_+ \end{aligned} \quad (9)$$

where  $[x]_- = -\min\{0, x\}$  and  $[x]_+ = \max\{0, x\}$  are non-zeros only on the respective vertical planes. Clearly, the scale of the depth parameter  $\lambda', \lambda$  and the scale of the translational velocity  $\beta$  introduce a gauge freedom. Thus, knowing one parameter, other two can be estimated explicitly. In this work, we fix the gauge by choosing  $\lambda' = 1$ , i.e., assume that the position of the vehicle w.r.t. one of the vertical plane is known, and the translational velocity  $\beta$  of the vehicle and the parameter  $\lambda$  of the other side of the vertical plane are estimated. On the other hand, if the velocity of the vehicle  $\beta$  is known, estimation of the parameters of the vertical planes  $\lambda'$  and  $\lambda$ , leads to the same solver. Further, if the detected vertical line segments touch the ground plane whose pixel coordinates are known, then the velocity of the vehicle as well as the location can be computed from  $\lambda^*$ . Nevertheless, the distorted image can always be compensated without knowing the scale (velocity, depth etc.).

To solve two motion parameters  $\{\alpha, \beta\}$  and two depth parameter  $\{\lambda, \delta\}$ , we require four line segments to estimate the model as well as the depth of the scene. Notice that in above,  $s^{-1} = [p_1 - \delta]_- + \lambda[p_1 - \delta]_+$  is computed w.r.t. GS camera coordinates. Substituting (9) into (8) leads to a system of eight polynomial equations of degree five with 29 monomials. The Gröbner basis solution [4, 16] for this system involves eliminating a  $467 \times 478$  matrix, which takes  $\approx 0.1$  second for each problem instance. Note that in conjunction with RANSAC, the whole procedure for robustly estimating the model parameters would take  $0.1N$  where  $N$  is the number of RANSAC iterations.

The efficiency is improved with the following approximations. The inverse of depth  $s^{-1}$  is assumed to be linear w.r.t. RS pixel coordinates, i.e.,  $s^{-1} = [p_1^{rs} - \delta]_- + \lambda[p_1^{rs} - \delta]_+$ . The experiments demonstrate that even with such approximation, we can still estimate the parameters very accurately. The problem is further simplified by considering three line segments in one of the vertical planes which are solved by Gröbner basis method. It computes  $\alpha, \beta$  and  $\delta$ . These are then substituted in the constraint (8)

of the other line segment—forms a quadratic equation in  $\lambda$  of which least absolute solution is chosen. The Gröbner basis method generates an elimination template matrix of size  $12 \times 15$  on which a G-J elimination method is performed. It produces 3 solutions and takes  $\approx 0.001s$  for each problem instance. Among the solutions, only the potential ones are considered by discarding unrealistic solutions (e.g., solutions corresponding to the absolute translation velocity above 200km/h and absolute angular velocity over 90deg/s).

In the following, we furnish the pseudo-code of the proposed algorithm (Algo. 1). Note that proposed algorithm is a minimal solver that can estimate the rolling shutter motion robustly in conjunction with RANSAC (described in Algo. 2). An interested reader can continue to the following sections else can move safely to the experiment section (sec. 4).

---

#### Algorithm 1 Rolling Shutter Motion Compensation

---

```

1: procedure COMPENSATEMOTION(Input image)
2:   detect line segments  $l_i := (\mathbf{u}^{rs}, \mathbf{v}^{rs})$ 
3:   prune line segments /* see sec. 4.1 */
4:   estimate( $\alpha, \beta$ ) and depth( $\delta, \lambda$ ) algo. 2
5:   warp compensated image /* see sec. 4.2 */
6:   return motion compensated image
7: end procedure

```

---



---

#### Algorithm 2 RANSAC robust estimation of $(\alpha, \beta, \delta, \lambda)$

---

```

1: procedure RANSACACKERMANN( $l_i := (\mathbf{u}^{rs}, \mathbf{v}^{rs})$ )
2:   while count  $\neq$  maximum iterations do
3:      $(\mathbf{u}^{rs}, \mathbf{v}^{rs}) \leftarrow$  get 4 random  $(\mathbf{u}^{rs}, \mathbf{v}^{rs})$ 
4:      $(\mathbf{u}^{rs}, \mathbf{v}^{rs}) \leftarrow$  pick 3 leftmost  $(\mathbf{u}^{rs}, \mathbf{v}^{rs})$ 
5:      $(\alpha, \beta, \delta) \leftarrow$  Gröbner Basis  $(\mathbf{u}^{rs}, \mathbf{v}^{rs})$ 
6:      $\lambda \leftarrow$  solve quadratic rightmost
      /* substituting  $(\alpha, \beta, \delta)$  in (8) */
7:     if out of range  $(\alpha, \beta, \delta, \lambda)$  then
8:       continue /* see sec. 4.3 */
9:     end if
10:    count inliers  $(\alpha, \beta, \delta, \lambda)$ 
11:    update best-found-so-far  $(\alpha, \beta, \delta, \lambda)$ 
12:    count  $\leftarrow$  count +1
13:  end while
14:  return best-found-so-far  $(\alpha, \beta, \delta, \lambda)$ 
15: end procedure

```

---

### 3.4. A more general case

Let us consider a more general case where the camera is installed with a known angle with vertical given by an orthogonal matrix  $R_\omega \in \text{SO}(3)$ . Then the pixel coordinate of a 3D point  $\mathbf{P} \in \mathbb{R}^3$  in (1) w.r.t. the GS and RS camera coordinates become  $s'R_\omega \mathbf{p} = \mathbf{P}$  and  $s'_A R_\omega \mathbf{p}^{rs} = \mathbf{P}_A$  where

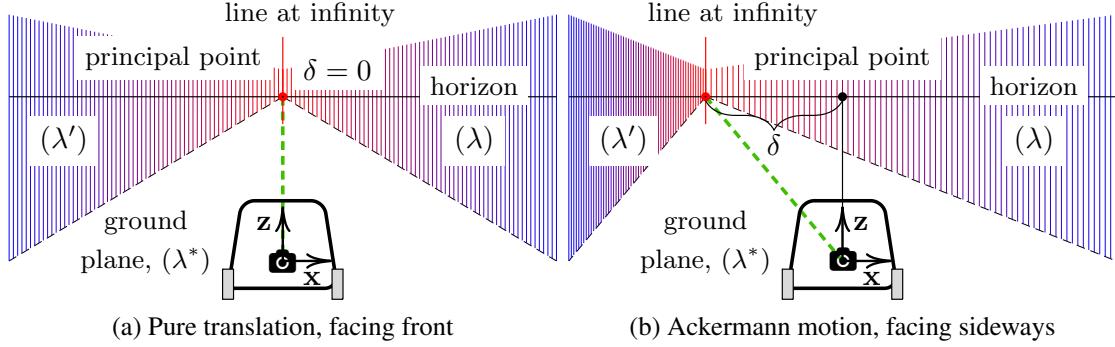


Figure 3: The inverse of the scene depth is assumed to be zero for the column corresponding to the line at infinity and then increases linearly along the sides of the road. The vertical lines corresponding to red lines are at a large depth compared to the blue ones.

$s'$  and  $s'_{\mathcal{A}}$  are proportional to the scene depth. Thus, (1) becomes

$$s'_{\mathcal{A}} R_{\omega} \mathbf{p}^{rs} = R_{\mathcal{A}}^t (\mathbf{P} - \mathbf{T}_{\mathcal{A}}^t). \quad (10)$$

The above could also be written in terms of GS co-ordinate frame as follows

$$s' R_{\omega} \mathbf{p} = s'_{\mathcal{A}} (I_3 + 2[\mathbf{r}_{\mathcal{A}}^t] \times) R_{\omega} \mathbf{p}^{rs} + \mathbf{s}_{\mathcal{A}}^t. \quad (11)$$

where  $\mathbf{r}_{\mathcal{A}}^t = [0, \alpha t, 0]^T$  is the angular rotation and  $\mathbf{s}_{\mathcal{A}}^t = [\alpha t, 0, 1]^T$  is the contribution of the circular motion to the translation. Note that  $(I_3 + 2[\mathbf{r}_{\mathcal{A}}^t] \times) R_{\omega}$  could also be observed as the linearization of the rotation matrix along  $R_{\omega}$  where  $s' = s/(R_{\omega}^3 \mathbf{p})$ ,  $s'_{\mathcal{A}} = s_{\mathcal{A}}^{rs}/(R_{\omega}^3 \mathbf{p}^{rs})$ ,  $s$  is the scene depth corresponding to the pixel  $\mathbf{p}$  and  $R_{\omega}^3$  is the third row of  $R_{\omega}$ . Substituting,  $\mathbf{p}^{rs'} = \frac{1}{R_{\omega}^3 \mathbf{p}} R_{\omega} \mathbf{p}^{rs}$  and  $\mathbf{p}' = \frac{1}{R_{\omega}^3 \mathbf{p}} R_{\omega} \mathbf{p}$  along with  $s'_{\mathcal{A}} = (s - \beta t)/(1 - 2p_1^{rs'} \alpha t)$  in above, however, we reach again to the similar set of equations as follows

$$\begin{aligned} \mathbf{p}' &= \frac{(1 - \beta t s'^{-1})}{(1 - 2p_1^{rs'} \alpha t)} (I_3 + 2[\mathbf{r}_{\mathcal{A}}^t] \times) \mathbf{p}^{rs'} + \beta t s'^{-1} \mathbf{s}_{\mathcal{A}}^t \\ s'^{-1} &= \lambda' [p'_1 - \delta]_- + \lambda [p'_1 - \delta]_+ \end{aligned} \quad (12)$$

where  $t$  could be replaced by  $p_2^{rs}$ . Note that substituting above in (8), we arrive at a set of equations of degree five and solve them by Gröbner basis method. It generates an elimination template matrix of size  $37 \times 43$  on which a G-J elimination method is performed. It produces 6 solutions for each problem instance which were fed into the RANSAC iterations.

### 3.5. Specific Motions:

**Pure translation motion** This is a very common scenario for which the vehicle undergoes a pure translation motion, i.e.,  $\alpha = 0$ . Thus, for the left side of the plane, (9) becomes

$$\mathbf{p} = \frac{(1 - \beta p_2^{rs}(p_1 - \delta))}{(1 - 2p_1^{rs} \alpha t)} \mathbf{p}^{rs} + \beta p_2^{rs}(p_1 - \delta) \mathbf{s}_{\mathcal{A}}^t \quad (13)$$

where  $\mathbf{s}_{\mathcal{A}}^t = [0, 0, 1]^T$ . In this scenario, a 3-line algorithm will be sufficient to have a finite solution—two of the line segments lie on one plane and the other line segment on the other plane. We can directly compute  $p_1 - \delta$  in terms of GS pixel coordinates as follows

$$p_1 = p_1^{rs} (1 + \beta \delta p_2^{rs}) / (1 + \beta p_1^{rs} p_2^{rs}) \quad (14)$$

$$\text{i.e., } p_1 - \delta = (p_1^{rs} - \delta) / (1 + \beta p_1^{rs} p_2^{rs}) \quad (15)$$

Substituting the pixels in terms of rolling shutter pixel coordinates in (8) and then simplifying the equations, we obtain

$$(v_{i1}^{rs} u_{i1}^{rs} u_{i2}^{rs} - u_{i1}^{rs} v_{i1}^{rs} v_{i2}^{rs}) \beta - (u_{i1}^{rs} u_{i2}^{rs} - v_{i1}^{rs} v_{i2}^{rs}) \beta \delta = u_{i1}^{rs} - v_{i2}^{rs} \quad (16)$$

Note that  $u_{i3}^{rs} = 1$  and  $v_{i3}^{rs} = 1$  were substituted during the simplification. Two such constraints (16) for two line segments lead to a unique solution. Thus, three line segments are sufficient to have a finite solution for both of the vertical planes—two of the line segments lie on one plane from which we uniquely compute  $\beta$  and  $\delta$ . The substitution of the estimated parameters to the other constraint, corresponding to the other line segment, leads to a linear constraint in  $\lambda$  which again has a unique solution. Since this closed form solution is obtained without an elimination template, it leads to an extremely fast solver.

**Pure rotational motion** If the vehicle undergoes a pure rotational motion, i.e.,  $\beta = 0$ . The change of depth  $s'_{\mathcal{A}}$  will not affect the vertical direction (8). In this scenario, the geometric relation between the GS and RS image frames is become

$$\mathbf{p} \propto (I_3 + [\mathbf{r}_{\mathcal{A}}^t] \times) \mathbf{p}^{rs} \quad (17)$$

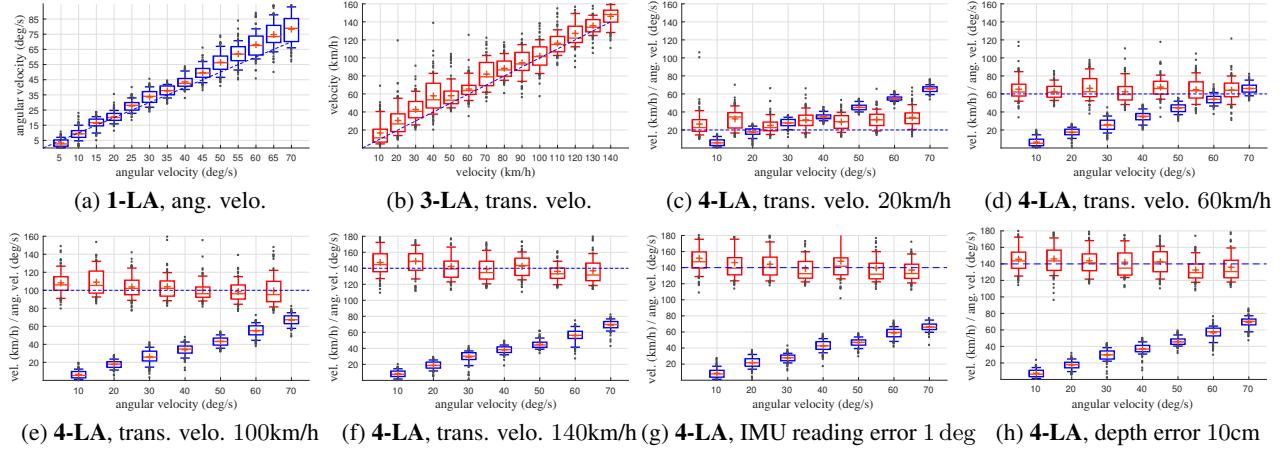


Figure 4: (a)-(f) Evaluation of the proposed method **1-LA**, **3-LA** and **4-LA** on a synthetic data. (g) Results under IMU reading error with normal distribution for standard deviation 1deg [17] (vertical assumption) and (h) results under depth error 10cm (depth assumption). The blue dotted lines represent the simulated velocity. Along  $x$ -axis and  $y$ -axis, we display the simulated and the predicted motion. The estimated translational and the angular velocities are represented by red and blue boxes respectively, where the 75% of estimated values lie within the box. '+' and '-' are the mean and the median of the estimated motions.

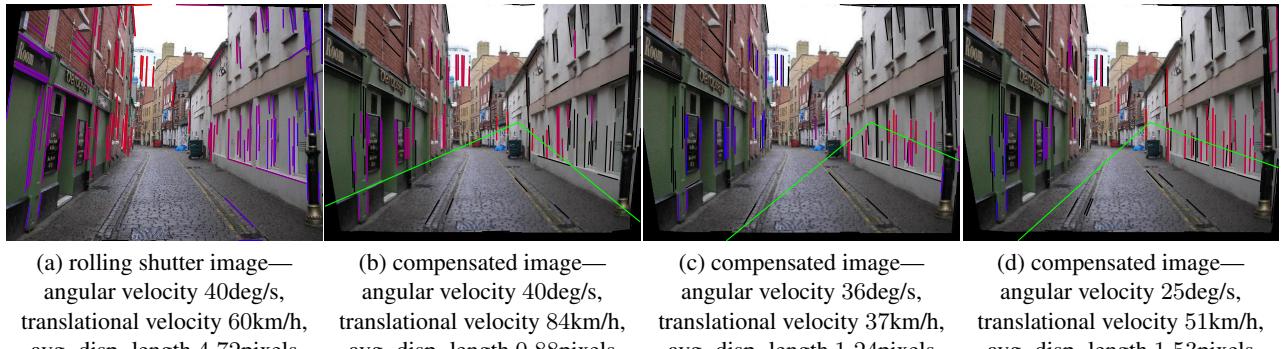


Figure 5: Three different extreme results (executed three times) on one of the instances of the synthetic data of Figure 4. Although, the estimated angular velocities and the translational velocities are not very accurate in these chosen cases, proposed method still reduces the pixel displacements occurred during rolling shutter motion. The average pixel displacement is computed by robust matching of patches from the original image and the compensated image.

where  $\mathbf{r}_A^t = [0, \alpha t, 0]^\top$ . After simplification, the constraint (8) is simplified to

$$(u_{i1}^{rs} u_{i2}^{rs} v_{i2}^{rs} - v_{i1}^{rs} v_{i2}^{rs} u_{i2}^{rs}) \alpha^2 - (u_{i1}^{rs} v_{i1}^{rs} v_{i2}^{rs} - v_{i1}^{rs} u_{i1}^{rs} u_{i2}^{rs} + u_{i2}^{rs} - v_{i2}^{rs}) \alpha + u_{i1}^{rs} - v_{i1}^{rs} = 0 \quad (18)$$

which is a quadratic in  $\alpha$  and only the solution with the least absolute value is considered. Thus, a single line segment is sufficient to estimate  $\alpha$ , this leads to an efficient and a much faster 1-line algorithm. However, this scenario is not an Ackermann motion (also different from motion demonstrated in Figure 2(a)) and the solution is presented here for completeness. Note that the existing accurate nonlinear op-

timization techniques, e.g., [21] can be utilized in this case which is, nevertheless, much slower than the proposed minimal solver.

## 4. Experiment

We conduct experiments on synthetic and real datasets to verify the effectiveness of the proposed minimal solver. Synthetic experiments were aimed to analyze the efficiency and stability of the solver. In particular, the degenerate cases where the camera undergoes a pure translation or rotational velocity. We list our solver as follows



Figure 6: (a) A random instance of Figure 4 with estimated translational velocity 59km/h and the angular velocity 43deg/s, and (b) under the estimation error of the height of the camera as 10cm. We observe very small image quality degradation under the violation of the assumption.

- **4-LA:** The full 4-line solver for Ackermann motion [Gröbner method, Eq. (9)]. /\* steps-3-6, algo. 2 \*/
- **3-LA:** The 3-line solver for pure translational motion [Eq. (17)]. /\* steps-3-6, algo. 2 modified accordingly \*/
- **1-LA:** The 1-line solver for pure rotational motion [Eq. (18)]. /\* steps-3-6, algo. 2 modified accordingly \*/

#### 4.1. Parameter settings

We evaluate the proposed solver in conjunction with the RANSAC schedule [6]. In all of our experiments, the inlier threshold is chosen to be 0.5 pixel and the number of iterations is chosen at a confidence level 0.99. We utilize the line segment detector `lsd` [9] with all the default parameters. The line segments of size less than 35 pixels are discarded. The line segments are pre-filtered with the algebraic error  $|(\mathbf{u}_i^{rs} \times \mathbf{v}_i^{rs})^\top \mathbf{e}_y| > 0.5$ . This increases the efficiency of RANSAC significantly, since it removes clear outliers.

#### 4.2. Image Rectification

After estimating the parameters, the undistorted image is obtained by the following forward mapping procedure of the RS pixels into the global frame (3)

$$\begin{aligned} \mathbf{p} &= \frac{(1-\beta p_2^{rs}s^{-1})}{(1-2p_1^{rs}\delta)} (I_3 + 2[\mathbf{r}_A^t]_\times) \mathbf{p}^{rs} + \beta p_2^{rs}s^{-1} \mathbf{s}_A^t \\ s^{-1} &= \max([p_1^{rs} - \delta]_-, \lambda[p_1^{rs} - \delta]_+, p_2^{rs} \lambda^*) \end{aligned} \quad (19)$$

where  $\mathbf{r}_A^t = [0, \alpha p_2^{rs}, 0]^\top$ ,  $\mathbf{s}_A^t = [\alpha p_2^{rs}, 0, 1]^\top$ . Note that  $\lambda^*$  is the inverse of the height of the camera from the ground plane (assumed to be known). Note that the boundaries of the ground plane and the vertical planes are the pixels for which  $p_1^{rs} - \delta = p_2^{rs} \lambda^*$ ,  $p_2^{rs} > 0$  or  $\lambda(p_1^{rs} - \delta) = p_2^{rs} \lambda^*$ ,  $p_2^{rs} > 0$ . After the forward map, the unknown pixels are linearly interpolated. Pixels located outside of the compensated frame are placed with intensity 0.

#### 4.3. Experiment on synthetic data

We choose a clean GS image (size  $640 \times 380$ , focal length = 816pixel and principle point at the center of the image) where the camera is placed at 1.2m height from the ground plane and 2.5m from the vertical plane on the left side of the road. We synthesize RS images with 30 frames/(40% readout time) by simulating the Ackermann motion randomly at a discrete interval of the translational velocity 10 – 140km/h and the angular velocity 10 – 70deg/s. A random noise is further added with standard deviation of 2km/h and 2deg/s while simulating the motion.

The proposed minimal solvers **1-LA**, **3-LA** and **4-LA** in conjunction with RANSAC are executed on each of the instances of the synthetic RS image. The statistics of the estimated motions are plotted in Figure 4. We observe that overall the motions are estimated accurately. Moreover, the joint estimations of the velocities are as accurate as individual estimations. Note that for a small velocity, the estimation of the line at infinity (and therefore the scene depth) is not accurate, thus, the estimation is not very accurate for small velocity. However, in almost all the cases, the proposed minimal solvers produce visually plausible images. In Figure 5, we display few results of such instances. We observe that in some cases our method does not estimate the rolling shutter motion accurately nonetheless it improves the image.

#### 4.4. Robustness against our assumptions

**Known verticle angle:** In Figure 4(g), we run experiments to show the robustness of our method under the known vertical angle. In this experiment, we have generated 100 random vertical direction with normal distribution for maximal deviation of 1deg to simulate industry quality IMU reading error [17] and observe minimal effect on the results.

**Known scale:** In Figure 4(h), we show the motion prediction under the scale (distance of one of the vertical plane from the camera) error. However, as the scale is used only to compute the car motion in units and there will not be any effect on the output image quality.

**Known height of the camera:** Moreover, we also observe minimal effect on the output image quality (as shown in Fig. 6), if the height of the camera from the ground plane is not given accurately.

#### 4.5. Experiment on real data

We employ the proposed method **4-LA** on a number of rolling shutter distorted image sequences, downloaded from YouTube<sup>2</sup>, and the results are displayed in Figure 7. The only qualitative evaluation is conducted as no ground-truths are available. The dataset is uncalibrated, thus, we estimated the focal length as 0.9 times the maximal image di-

<sup>2</sup><https://www.youtube.com>

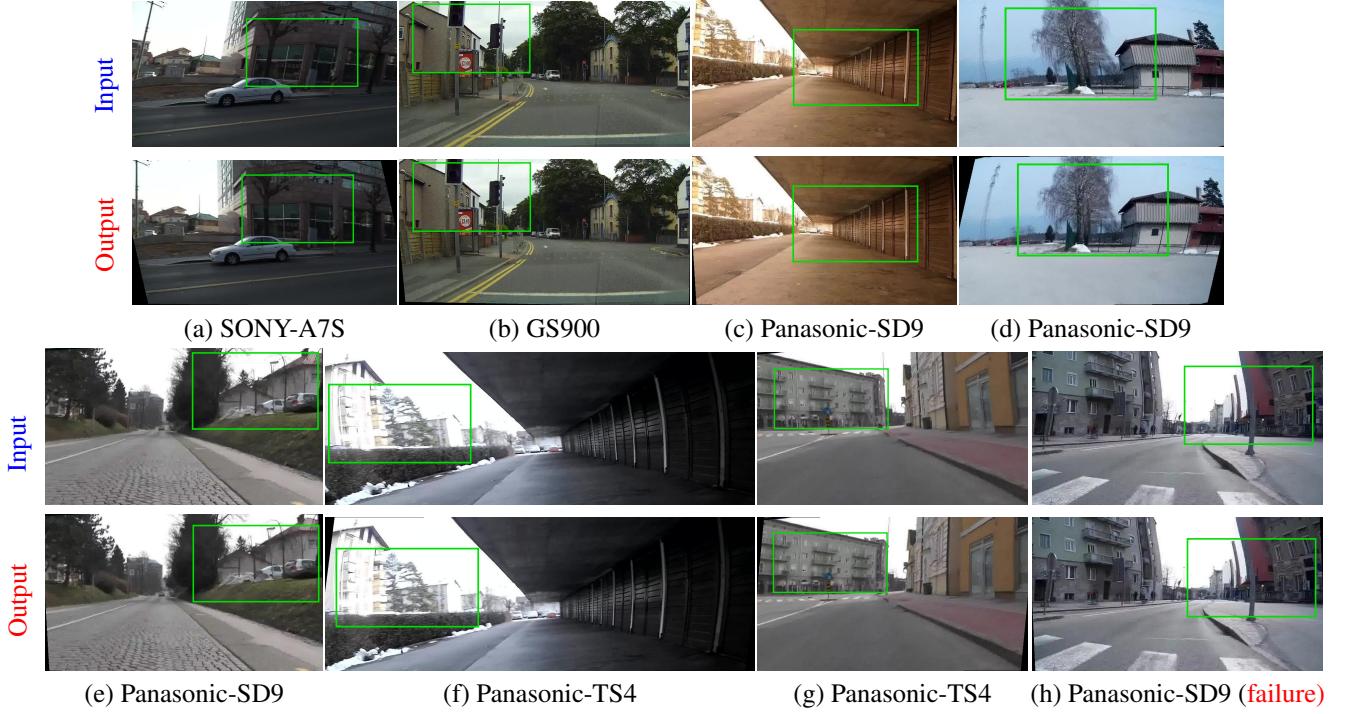


Figure 7: Evaluation of the proposed method **4-LA** on real datasets captured by a number of cameras mentioned below. Aesthetically pleasing areas, original and output correspondences, are marked by green rectangular boxes. For visual inspection the readers are encouraged to view the soft copy of the manuscript.

mension. The principal point was chosen to be the center of the image. Overall, proposed method performs quite well for all the images. However, the car was shaking very rapidly for Figure 7(h) and proposed method could not perform very well as in this scenario the car motion do not quite follow the assumed Ackemann motion.

The proposed method was also evaluated frame-by-frame on the image sequences from the datasets [10]<sup>3</sup>. The results are displayed in Figure 8. Note that [10] do not consider the translational motion while compensating the motion and the resulted image were cropped for visually pleasant output. This dataset was chosen for evaluation as it consists of corridor scene required by the proposed algorithm, despite captured by an handheld camera and there is little rolling shutter effect. Readers are requested to look at the zoomed version of the draft.

In a different settings, we choose the real dataset [1] where the dataset was captured from a fixed location only moving the camera horizontally. Thus this is the scenario of a specific motion where translation is negligible. Here, we evaluate our algorithm **1-LA** with baselines [15, 21] and the results are displayed in Figure 9. Note that [15] compensates the camera motion by the gyroscope readings. [21]

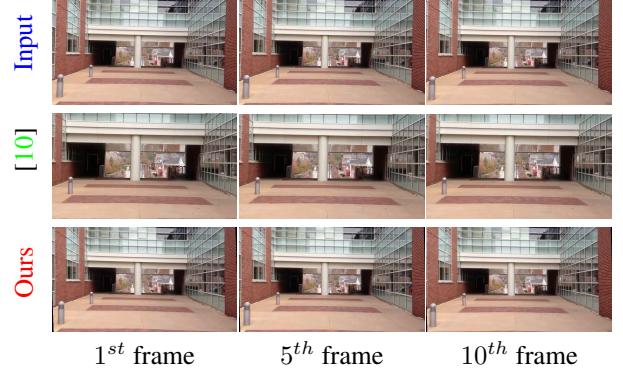


Figure 8: Evaluation of the proposed method **4-LA** on a real data sequence *i4s\_depth\_3\_.mp4* from the dataset [10] captured by iPhone4. We display 1st, 5th and 10th images of the sequence.

estimates the yaw and roll angular velocities where as our solver estimates only the yaw angular velocity. However, **1-LA** produces equivalent qualitative results on the selected images while three orders of magnitude faster than the baseline [21].

<sup>3</sup><http://www.cc.gatech.edu/cpl/projects/rollingshutter/>



Figure 9: Comparison of proposed **1-LA** on a real video sequence. Note that proposed method estimates only *yaw* rotation in contrast [21] estimates full rotation with the cost of high computation time mentioned in Table 1

#### 4.6. Runtime comparison

The proposed method is implemented in Matlab and the runtimes are computed on an *i7* 2.8GHz CPU utilizing only a single core. On average it takes around 0.087 second to rectify a  $360 \times 520$  image, excluding LS detection and the rectification (also exclude from [10], [21] and [26]), which is two-three orders of magnitude faster over the most recent methods. See Table 1 for the detailed comparison. Note that we rescale the runtime comparison according to the image size reported in the paper.

Table 1: Average runtime of different methods in seconds.

Methods	<b>1-LA</b>	<b>3-LA</b>	<b>4-LA</b>	[10]	[21]	[26]
Runtime	0.001s	0.003s	0.087	0.1s	8s	249.6s

## 5. Conclusion

We proposed a minimal solver for rolling shutter camera motion compensation under an Ackermann motion from a single view. A 4-line algorithm is developed to estimate the simplified depth of the scene along with the motion parameters. The proposed method is also the first of its kind to exploit the minimal solvers for monocular rolling shutter motion compensation which allows to have an extremely fast compensation method. Extensive experiments were performed on the simulated and real datasets. The results demonstrate the computational efficiency and the effectiveness of the proposed approach. During the motion compensation in a video, individual frames were compensated separately; however, a nonlinear optimization method (viz., rolling shutter bundle adjustment) can be developed for more accurate estimation with proposed fast method as an initialization and hence there lies a potential future extension.

## Acknowledgements

We thank V. Rengarajan for running his method [21] and providing the results on selected datasets [Fig. 9].

## References

- [1] Probabilistic 3-d motion estimation for rolling shutter video rectification. <http://users.ece.utexas.edu/~bevans/papers/2012/rolling/>. 8
- [2] C. Albl, Z. Kukelova, and T. Pajdla. R6p-rolling shutter absolute camera pose. In *CVPR*, pages 2292–2300, 2015. 2, 3
- [3] C. Albl, Z. Kukelova, and T. Pajdla. Rolling shutter absolute pose problem with known vertical direction. In *CVPR*, pages 3355–3363, 2016. 3
- [4] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *CVPR*, pages 1–8. IEEE, 2008. 4
- [5] Y. Dai, H. Li, and L. Kneip. Rolling shutter camera relative pose: Generalized epipolar geometry. In *CVPR*, June 2016. 2
- [6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 7
- [7] P.-E. Forssén and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *CVPR*, pages 507–514, 2010. 2
- [8] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. 2
- [9] v. G. R. Grompone, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: a fast line segment detector with a false detection control. *IEEE TPAMI*, 32(4):722–732, 2010. 7
- [10] M. Grundmann, V. Kwatra, D. Castro, and I. Essa. Calibration-free rolling shutter removal. In *ICCP*, pages 1–8, 2012. 2, 8, 9
- [11] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 3
- [12] G. Hee Lee, F. Faundorfer, and M. Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *CVPR*, pages 2746–2753, 2013. 2
- [13] G. Hee Lee, M. Pollefeys, and F. Fraundorfer. Relative pose estimation for a multi-camera system with known vertical direction. In *CVPR*, pages 540–547, 2014. 2
- [14] S. Hee Park and M. Levoy. Gyro-based multi-image deconvolution for removing handshake blur. In *CVPR*, pages 3366–3373, 2014. 2
- [15] C. Jia and B. L. Evans. Probabilistic 3-d motion estimation for rolling shutter video rectification from visual and inertial measurements. In *MMSP*, 2012. 2, 8, 9
- [16] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. *ECCV*, pages 302–315, 2008. 4
- [17] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *ACCV*, pages 216–229. Springer, 2010. 6, 7
- [18] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *BMVC*, 2013. 2

- [19] A. Patron-Perez, S. Lovegrove, and G. Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *IJCV*, 113(3):208–219, 2015. [2](#)
- [20] P. Purkait, C. Zach, and A. Leonardis. Rolling shutter correction in manhattan world. In *ICCV*, pages 882–890, 2017. [1](#)
- [21] V. Rengarajan, A. N. Rajagopalan, and R. Aravind. From bows to arrows: Rolling shutter rectification of urban scenes. In *CVPR*, pages 2773–2781, 2016. [1](#), [2](#), [6](#), [8](#), [9](#)
- [22] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *ICCV*, pages 465–472, 2013. [2](#)
- [23] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *ICCV*, pages 1413–1419. IEEE, 2009. [2](#)
- [24] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *ICRA*, pages 4293–4299. Ieee, 2009. [2](#)
- [25] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011. [2](#)
- [26] S. Su and W. Heidrich. Rolling shutter motion deblurring. In *CVPR*, pages 1529–1537, June 2015. [1](#), [2](#), [9](#)