Deep Shutter Unrolling Network

Peidong Liu[1] Zhaopeng Cui[1] Viktor Larsson[1] Marc Pollefeys[1,2]
[1]Computer Vision and Geometry Group, ETH Zürich, Switzerland
[2]Microsoft Artificial Intelligence and Mixed Reality Lab, Zürich, Switzerland
{peidong.liu, zhaopeng.cui, vlarsson, marc.pollefeys}@inf.ethz.ch
mapoll@microsoft.com

Abstract

We present a novel network for rolling shutter effect correction. Our network takes two consecutive rolling shutter images and estimates the corresponding global shutter image of the latest frame. The dense displacement field from a rolling shutter image to its corresponding global shutter image is estimated via a motion estimation network. The learned feature representation of a rolling shutter image is then warped, via the displacement field, to its global shutter representation by a differentiable forward warping block. An image decoder recovers the global shutter image based on the warped feature representation. Our network can be trained end-to-end and only requires the global shutter image for supervision. Since there is no public dataset available, we also propose two large datasets: the Carla-RS dataset and the Fastec-RS dataset. Experimental results demonstrate that our network outperforms the state-of-the-art methods. We make both our code and datasets available at https://github.com/ethliup/DeepUnrollNet.

1. Introduction

CMOS imaging sensors are widely used in many consumer products. Most CMOS sensors capture images with a rolling shutter mechanism. In contrast to a global shutter camera, which captures all pixels at the same time, a rolling shutter camera sequentially captures the image pixels row by row. Therefore, different types of distortions, e.g. skew, smear or wobble, will appear if the camera is moving during the image capture. It is well known that many vision tasks (e.g. structure from motion, visual odometry, pose estimation or depth prediction) suffer from rolling shutter distortions [1, 11, 15, 16, 26, 27]. The rolling shutter effect correction problem has thus received considerable attention in the past [24, 30, 32].

Existing works on rolling shutter effect correction can be categorized into classical approaches and single image based deep learning approaches. The classical approaches can be further categorized into single image based methods and methods which use multiple images. Single image based rolling shutter effect correction is an ill-posed problem and relies heavily on prior assumptions (e.g. straight lines must remain straight), either formulated explicitly or learned implicitly by a deep network, which limit their applicability to real scenarios. Classical multi-image based approaches are more general and instead rely on geometric constraints from multiple views to perform the rectification. However, they usually formulate it as a computationally expensive optimization problem for 6 DoF camera motions, which prevents the algorithm from being used in time constrained applications.

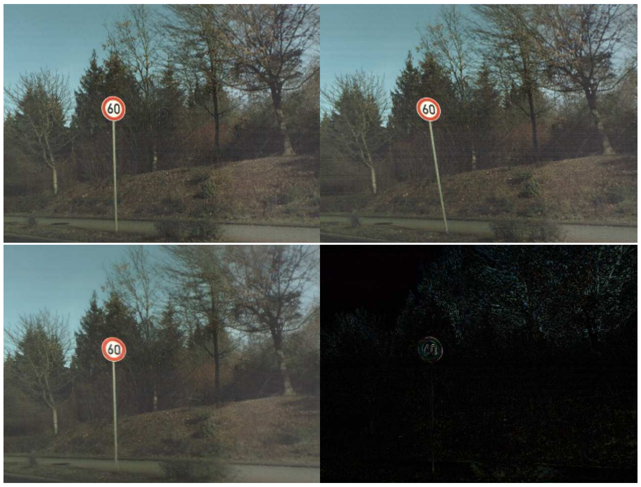

Figure 1: **Deep shutter unrolling network. Top left:** Ground truth global shutter image. **Top right:** Input rolling shutter image. **Bottom left:** Predicted global shutter image by our network. **Bottom right:** Absolute difference between our predicted image and the ground truth global shutter image.

Inspired by the recent success of deep neural networks on image-to-image translation problems, such as optical flow estimation [29], dense depth prediction [6], motion deblurring [21] and image super-resolution [19], we propose an efficient end-to-end deep neural network for rolling shutter effect correction. Our method solves a generic rectification problem from two consecutive frames. It is able to take advantage of the parallel computational power of a graphic card and runs in near real time. Furthermore, benefiting from the representational power of a deep network, our network is also able to learn good image priors to further boost the quality of the rectified image. Different from the above image-to-image translation problems, in which the estimations usually rely on its local neighborhood pixels of the input image, the rolling shutter effect correction problem is more challenging. A pixel of the rectified image might lie far away from its corresponding pixel of the input rolling shutter image, depending on the types of motion, 3D scene structure as well as its capturing time. To resolve these challenges, we propose a novel network architecture for rolling shutter image correction.

Our network takes two consecutive rolling shutter images as input and predicts the corresponding global shutter image of the latest frame. It consists of four main parts: an image encoder, a motion estimator, a differentiable forward warping block and an image decoder. The motion estimator estimates the dense per-pixel displacement field from a rolling shutter image to its corresponding global shutter image, given the learned feature representation from the image encoder. The differentiable forward warping block warps the learned feature representation to its corresponding global shutter representation, given the estimated displacement field. The global shutter image is then recovered by the image decoder from the warped feature representation. Our network can be trained end-to-end and only requires the ground truth global shutter image for supervision, which is easy to obtain by using a high-speed camera to synthesize the training data. Experimental results demonstrate that our method outperforms the state-of-the-art methods [31,32]. Fig. 1 presents qualitative results from our network.

Since there is no public dataset available, we also propose two novel datasets: the Fastec-RS dataset and the Carla-RS dataset, as our second contribution. The Fastec-RS dataset has 2584 image pairs. It is generated via a professional high-speed camera (with a framerate of 2400 FPS) and captured in real environments. Since the camera is mounted on a ground vehicle which undergoes limited motion, we also create the Carla-RS dataset with general six degree of freedom (DoF) motions. This dataset is generated from a virtual 3D environment and has 2500 image pairs. To further research in this area we make both our code and the datasets public.

2. Related Work

We categorize the related work into classical approaches and deep learning based approaches. The classical approaches can be further classified into single image based and multiple image based approaches.

Classical single image based approaches: Rengarajan et al. [25] proposes to take advantage of the "straight lines must remain straight" assumption to rectify a single rolling shutter image. The camera motion is assumed to be purely rotational. Curves are extracted and the motion is iteratively estimated by enforcing the transformed curves to be straight. Purkait et al. [23] assumes the 3D scene captured by the camera obeys Manhattan world assumption [4]. The distortion is corrected by jointly aligning the vanishing directions. Lao and Ait-Aider [17] propose a minimal solver to estimate the camera motion based on four straight lines from a single image. The motion is parameterized by pure rotations. The RANSAC algorithm [7] is used to remove outliers such that the camera motion can be estimated robustly. The rolling shutter effects can then be removed given the estimated motion.

Classical multiple image based approaches: Liang et al. [18] estimates per-pixel motion vector to rectify a rolling shutter image. Block matching is used to find correspondences between two consecutive frames, such that the motion can be estimated. Forssén and Ringaby [8] assumes the camera has either pure rotation or in-plane translational motion. The camera motion is estimated by minimizing the re-projection errors between sparse corresponding points. A KLT tracker [20] is used to establish the sparse correspondences. Karpenko et al. [14] extends the work of [8] by using inertial measurements. They model the camera motion by pure rotations. The rolling shutter effect is removed by solving an optimization problem, which jointly stabilizes the video and calibrates the gyroscope. Baker et al. [2] estimates the per-pixel motion vector from a video sequence to correct the rolling shutter distortion. The motion is estimated via a constant affine or translational distortion model, and is estimated in up to 30 row blocks. Grundmann et al. [10] relaxes the constraints that a calibrated camera is required for rolling shutter effect removal. They model the motion between two neighbouring frames as a mixture of homography matrices. The mixture of homographies is estimated by minimizing the re-projection errors of corresponding points. Zhuang et al. [31] proposes to solve a dense SfM problem given two consecutive rolling shutter images. They estimate both the camera motion and dense depth map from dense correspondences. A minimal solver is proposed to estimate the camera motion. Both the depth map and motion are further estimated/refined by minimizing the re-projection errors. Vasu et al. [30] propose to solve occlusion aware rolling shutter correction problem using multiple consecutive frames. They model the 3D geometry as a layer of planar scenes. The depth, camera motion, latent layer mask and latent layer intensities are jointly estimated. The global shutter image is recovered by the pro-

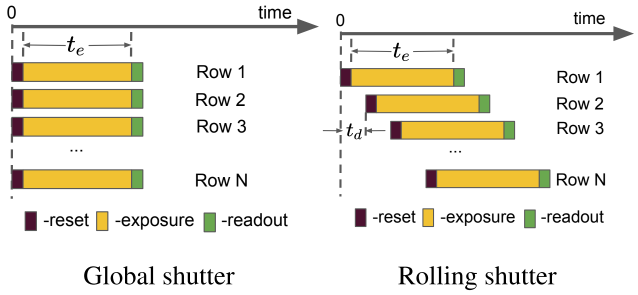

Figure 2: **Image formation models.** The difference between a rolling shutter camera and a global shutter camera is that the rows of a rolling shutter image are captured at different timestamps with a constant time offset t_d. For simplicity, we assume the exposure time t_e is infinitesimal throughout the paper.

posed image formation model given all above estimations.

Deep learning based approaches: Rengarajan et al. [24] propose to estimate the camera motion from a single rolling shutter image by using a deep network. They assume a simple affine motion model. The global shutter image is then recovered given the estimated motion. They train the network with synthetic data, which is generated by using the proposed motion model. Zhuang et al. [32] extends [24] for depth aware rolling shutter effect correction from a single image. Two independent networks are used to predict the dense depth map and camera motion respectively. The global shutter image is then recovered as a post-processing step, given the estimated dense depth map and camera motion.

3. Method

The main concept of our method is to learn a dense per-pixel displacement field, which is used to warp the learned features from the rolling shutter image to its global shutter counterpart. The global shutter image is then recovered by an image decoder which decodes the warped features to an image. Our network can be trained end-to-end and only requires the global shutter image for supervision. Fig. 3 presents the details of our network architecture.

Rolling shutter image formation model: The difference between a rolling shutter camera and a global shutter camera is that every scanline of the rolling shutter camera is exposed at different timestamps, as shown in Fig. 2. Without loss of generality, we assume the read-out direction is from top to bottom. We further assume all pixels from the same row are captured at the same timestamp. We can thus obtain the image formation model of a rolling shutter image as follows:

$$[\mathbf{I}^r(\mathbf{x})]_i = [\mathbf{I}_i^g(\mathbf{x})]_i, \tag{1}$$

where $\mathbf{I}_i^g(\mathbf{x})$ is the virtual global shutter image captured at timestamp $i \cdot t_d$, t_d is the time to read out a single row, $[\mathbf{I}_i^g(\mathbf{x})]_i$ is an operator to extract the i^{th} row from an image $\mathbf{I}_i^g(\mathbf{x})$.

As the whole image readout time (i.e., Nt_d where N is the height of the image) is typically small (<50 ms), we can assume that during the time of capture the image content is primarily affected by image motion and not by other changes like object appearance or illumination. We can thus model the virtual global shutter image $\mathbf{I}_i^g(\mathbf{x})$ as the result of the first virtual global shutter image $\mathbf{I}_0^g(\mathbf{x})$ warped by a displacement vector $\mathbf{u}_{i\rightarrow 0}$:

$$\mathbf{I}_i^g(\mathbf{x}) = \mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{i\rightarrow 0}), \tag{2}$$

where $\mathbf{u}_{i\rightarrow 0} \in \mathbb{R}^2$ denotes the displacement vector of pixel $\mathbf{x}$ from the i^{th} virtual global shutter image $\mathbf{I}_i^g$ to the reference image $\mathbf{I}_0^g$, which corresponds to the virtual global shutter image captured at timestamp 0. Thus, we can reformulate Eq. (1) to

$$[\mathbf{I}^r(\mathbf{x})]_i = [\mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{i\rightarrow 0})]_i. \tag{3}$$

We can further have

$$\mathbf{I}^r(\mathbf{x}) = \mathbf{I}_0^g(\mathbf{x} + \mathbf{u}_{r\rightarrow g}), \tag{4}$$

where $\mathbf{u}_{r\rightarrow g} \in \mathbb{R}^2$ denotes the displacement vector of pixel $\mathbf{x}$ from the rolling shutter image to the first virtual global shutter image. If we stack $\mathbf{u}_{r\rightarrow g}$ for all pixels, it has following form

$$[\mathbf{U}_{r\rightarrow g}]_i = [\mathbf{U}_{i\rightarrow 0}]_i, \tag{5}$$

where both $\mathbf{U}_{r\rightarrow g}$ and $\mathbf{U}_{i\rightarrow 0}$ are the dense displacement field for all pixels, in matrix form.

As a special case, if the rolling shutter camera is stationary during image capture, the displacement field $\mathbf{U}_{r\rightarrow g}$ is zero. The captured rolling shutter image equals to the global shutter image.

Rolling shutter effect removal: Rolling shutter effect removal is an operation to reverse the above image formation model, i.e., Eq. (4). In particular, it is to estimate the global shutter image $\mathbf{I}_0^g(\mathbf{x})$ given the captured rolling shutter image $\mathbf{I}^r(\mathbf{x})$. It is an ill-posed problem for single image rolling shutter effect removal, since the displacement field $\mathbf{U}_{r\rightarrow g}$ is difficult to recover from a single image. Existing works typically take advantage of prior assumptions (e.g. straight lines should remain straight) to estimate the displacement field [25, 32]. The prior assumption can be either explicitly formulated [25] or implicitly learned by a deep network [32]. Thus, single image rolling shutter correction methods cannot generalize to scenarios where the prior assumption is not satisfied. Therefore, we propose to use two frames to solve a more general rectification problem.

To recover $\mathbf{I}_0^g$ from $\mathbf{I}^r$, it is more convenient to have displacement field $\mathbf{U}_{g\rightarrow r}$ instead of $\mathbf{U}_{r\rightarrow g}$. The global shutter image can then be simply recovered by

$$\mathbf{I}_0^g(\mathbf{x}) = \mathbf{I}^r(\mathbf{x} + \mathbf{u}_{g\rightarrow r}), \tag{6}$$

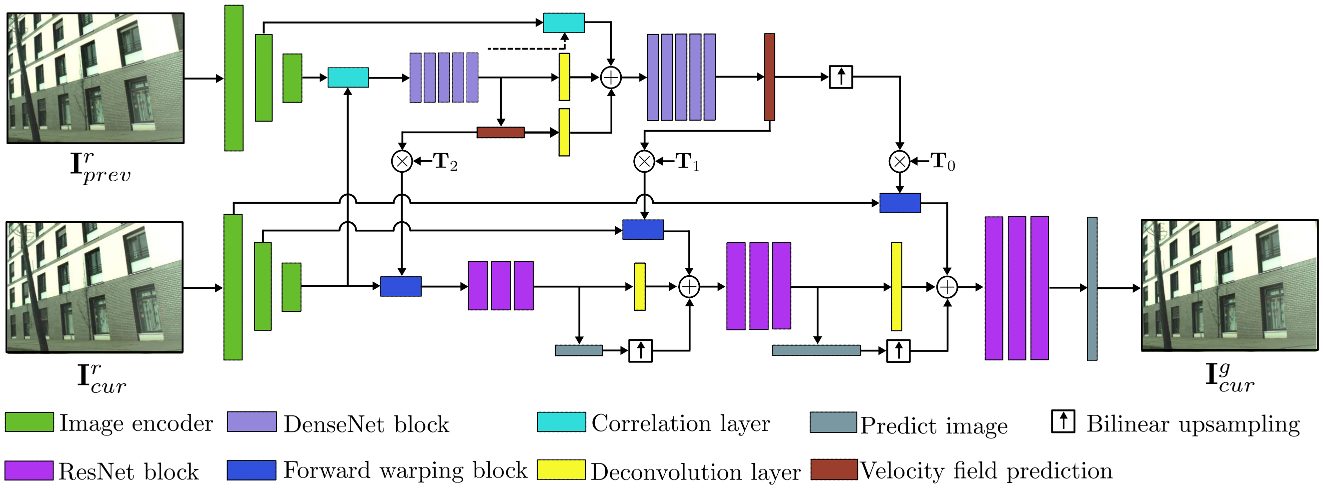

Figure 3: **Deep shutter unrolling network.** Our network takes two consecutive rolling shutter images as input and predicts a global shutter image. It consists of four main parts: an image encoder network, a motion estimation network, a differentiable forward warping block and an image decoder network. The motion estimation network estimates the dense pixel-wise velocity field from the rolling shutter image to the global shutter image. The displacement field is recovered by multiplying the velocity field with the respective time offset of each pixel to that of the estimated global shutter image (i.e., $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$). The forward warping block warps the learned feature representation of the current rolling shutter image to its global shutter representation, given the estimated displacement field. The image decoder network then transforms the warped feature representation to a global shutter image. The dashed arrow represents the corresponding feature representation from the current input image $\mathbf{I}^r_{cur}$.

where bilinear interpolation can be used for pixels which have non-integer positions. However, we are only given rolling shutter images as input. It is more difficult for us to estimate $\mathbf{U}_{g\rightarrow r}$ compared to $\mathbf{U}_{r\rightarrow g}$. Thus, we design a motion estimation network to estimate $\mathbf{U}_{r\rightarrow g}$ for rolling shutter image rectification. It is not trivial to recover the global shutter image given the displacement field $\mathbf{U}_{r\rightarrow g}$ and the rolling shutter image, since we cannot find the pixel correspondences from the global shutter image to the rolling shutter image. Thus, we propose to employ a forward warping block [8] to resolve this challenge. We derive and implement the derivatives of the forward warping block, to make it differentiable such that we can incorporate it into our deep network for end-to-end training. For compactness, we denote $\mathbf{I}^g_0$ as $\mathbf{I}^g$ for future sections.

Differentiable forward warping block: As shown in Fig. 4, we can approximate the intensity of a particular pixel from the global shutter image, as a weighted average of its neighboring pixel intensities from the rolling shutter image, which was previously used in [8]. Formally, this can be defined as

$$\mathbf{I}^g(\mathbf{x}) = \frac{\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} \mathbf{I}^r(\hat{\mathbf{x}})}{\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}}, \tag{7}$$

where $\Omega(\mathbf{x})$ is the set of all pixels $\hat{\mathbf{x}}$ from the rolling shutter image, which satisfy

$$\left\|\hat{\mathbf{x}} + \mathbf{u}_{r\rightarrow g} - \mathbf{x}\right\|_2 < r, \tag{8}$$

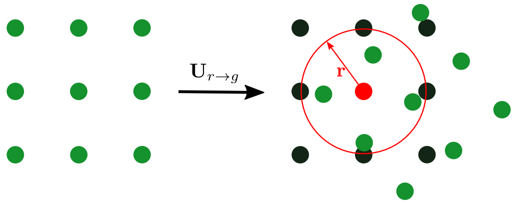

Figure 4: **Differentiable forward warping.** The rolling shutter image (i.e., green pixels) is warped to the image grid of the global shutter image (i.e., black pixels and the red pixel) by the estimated displacement field $\mathbf{U}_{r\rightarrow g}$. To recover the intensities of the red pixel, we can compute the weighted average of its four neighboring pixels (i.e., the green pixels covered by the red circle with a radius r) from the rolling shutter image.

where r is a pre-defined threshold with unit in pixels. We can further define

$$\omega_{\hat{\mathbf{x}}} = e^{-\frac{d(\mathbf{x},\hat{\mathbf{x}})^2}{2\sigma^2}}, \tag{9}$$

where

$$d(\mathbf{x},\hat{\mathbf{x}}) = \left\|\hat{\mathbf{x}} + \mathbf{u}_{r\rightarrow g} - \mathbf{x}\right\|_2, \tag{10}$$

and σ is a pre-defined width of the kernel function. We derive all the derivatives (i.e., $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{I}^r(\mathbf{x})}$ and $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{u}_{r\rightarrow g}}$) that are required for gradient back-propagation, which is necessary for network training. Both the forward and backward pass can be implemented efficiently by parallelizing the computations on a graphic card. The derivations and implementation details can be found in our supplementary material.

Network architecture: In this section, we explain how to design a deep network to estimate $\mathbf{U}_{r\rightarrow g}$ and recover the global shutter image. Everything presented in the previous sections can be directly generalized from pixels to learned feature representations. Fig. 3 presents the architecture of our network.

Our network accepts two consecutive rolling shutter images and outputs a global shutter image corresponding to the latest frame. It consists of four main parts, i.e., an encoder network, a motion estimation network, a differentiable forward warping block and an image decoder network. The encoder network consists of three pyramid levels. Each level has a convolutional layer followed by three residual blocks. To recover the latent global shutter image, it would be easier for the image decoder network to operate on the feature representation which corresponds to the global shutter image. We therefore use the differentiable forward warping block to transform the learned feature representation of the latest rolling shutter image to its global shutter counterpart. The displacement field $\mathbf{U}_{r\rightarrow g}$ used by the forward warping block is estimated by the motion estimation network.

Besides the camera motion and 3D scene geometry, $\mathbf{U}_{r\rightarrow g}$ also depends on the time when a particular pixel is being captured, i.e., the displacement vector of a pixel nearer to the first row is usually smaller than those are further away. During our ablation study, we find that the motion estimation network has difficulty to learn/model this implicitly. We thus model this dependency explicitly and design the network to learn the dense velocity field instead, as shown in Fig. 3. Our motion estimation network computes the cost volumes between both frames by a correlation layer [12], based on the learned feature representation. The velocity field is then estimated by a dense network block, given the computed cost volumes as input. To recover the displacement field $\mathbf{U}_{r\rightarrow g}$, we multiply the estimated velocity field with the time offset (i.e., $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ as shown in Fig. 3) between the captured pixel and that of the first row. $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ represent the time-offset for different pyramid levels and they have the same resolutions as the feature representations of the corresponding pyramid levels. Without calibrating the camera, we simply set the row read-out time (i.e., t_d as shown in Fig. 2) as 1 for simplicity. The image decoder then predicts the global shutter image given the warped feature representation. The decoder network also consists of three pyramid levels. Each level has three residual blocks followed by a deconvolution layer. The details of the network architecture can be found in our supplementary material.

Loss functions: To train our network, only the corresponding ground truth global shutter image $\mathbf{I}^g_{gt}$ is required. Empirically, we find a linear combination of the pixel-wise $\mathcal{L}_1$ loss, the perceptual loss $\mathcal{L}_p$ [13] and a total variation loss $\mathcal{L}_{tv}$ to encourage piecewise smoothness in the estimated displacement field $\mathbf{U}_{r\rightarrow g}$, can give satisfactory performance. If the perceptual loss is omitted, the estimated image tends to be blurry. In summary, our loss function can be formulated as

$$\mathcal{L} = \mathcal{L}_p(\mathbf{I}^g, \mathbf{I}^g_{gt}) + \lambda_1 \mathcal{L}_1(\mathbf{I}^g, \mathbf{I}^g_{gt}) + \lambda_2 \mathcal{L}_{tv}(\mathbf{U}_{r\rightarrow g}), \quad (11)$$

where both λ_1 and λ_2 are hyper-parameters and determined empirically, $\mathbf{I}^g$ is the estimated global shutter image and $\mathbf{I}^g_{gt}$ is the ground truth global shutter image.

4. Datasets

Both Rengarajan et al. [24] and Zhuang et al. [32] propose individual datasets to train their networks respectively. However, their datasets are not released to the community. Furthermore, both datasets simplify the real formation process of a rolling shutter image. For example, Rengarajan et al. [24] generate the synthetic image by applying simple affine image warping and does not consider the 3D geometry. Zhuang et al. [32] do consider the effect of the 3D geometry. They warp a single global shutter image from the KITTI dataset [9] to generate a synthetic rolling shutter image, given the corresponding dense depth map and camera motion. The dense depth map is estimated from a stereo camera by a depth prediction network [3]. The 6 DoF camera motion is randomly sampled from a pre-defined interval. However, it still simplifies the real image formation process, e.g. their dataset does not model occlusions, which are common in real-world scenarios. Furthermore, the estimated dense depth map is not the true 3D scene geometry either.

Thus, we propose two datasets: the Carla-RS dataset and the Fastec-RS dataset. Our datasets are synthesized via high speed cameras and simulate the real image formation process. The Carla-RS dataset is generated from a virtual 3D environment provided by the Carla simulator [5]. Carla simulator is an open-source platform for autonomous driving research and it provides seven photorealistic 3D virtual towns. We implement a rolling shutter camera model since the original simulator does not support it. We also relax the constraint that the camera is mounted on a ground vehicle, such that we can freely move our rolling shutter camera in six DoF. 250 sequences are randomly sampled and each sequence has 10 consecutive frames. Both a constant translational velocity model and a constant angular rate model are used for the sequence generation, which is typically hold in real scenarios due to the short time interval (i.e., <50ms) between two consecutive frames. In total, we generate 2500 rolling shutter images at a resolution of 640×448 pixels.

Since the Carla-RS dataset is generated from a virtual environment, we also propose another dataset, the Fastec-RS dataset which is created using real images in the wild. The Fastec-RS dataset is synthesized using a professional Fastec TS5[1] high speed global shutter camera with a framerate of 2400 FPS. We mount the camera on a ground vehicle and collect 76 image sequences at a resolution of 640 × 480 pixels in mainly urban environment. Each sequence synthesizes 34 rolling shutter images. In total, we have 2584 image pairs. The rolling shutter image is synthesized by sequentially copying a row of pixels from the captured global shutter images.

5. Experimental Evaluation

Datasets: We evaluate our algorithm with both the Carla-RS dataset and Fastec-RS dataset. We split the Carla-RS dataset into training data and test data. The training data has 210 sequences and the test data has 40 sequences. Similarly, we split the Fastec-RS dataset into 56 sequences for training and 20 sequences for test. Both the training data and test data have no overlapping scenes. Since the ground truth occlusion masks can be obtained and are also provided by the Carla-RS dataset, we thus compute two quantitative metrics for better evaluation, i.e., one without using the occlusion mask and the other one using the occlusion mask. For compactness, we denote the Carla-RS dataset with masks, the Carla-RS dataset without masks and the Fastec-RS dataset as **CRM**, **CR** and **FR** respectively for quantitative evaluations.

Implementation details: We implemented our network in PyTorch [22]. The differentiable forward warping is imeplemted in CUDA with PyTorch wrappers. The hyper-parameters are set empirically to $r = 2$, $\sigma = 0.5$, $\lambda_1 = 10$ and $\lambda_2 = 0.1$ unless stated otherwise. For better convergence, we train our network in three pyramid levels. The network is trained in 200 epochs with a learning rate 10^{-4}. We use a batch size of 3 and use uniform random crop at a resolution of 320 × 256 pixels for data augmentation.

State-of-the-art methods: We compare our network against two state-of-the-art methods from [31] and [32], which are the two most related works to our approach. The method from [31] is a classical two image based approach and we use the implementation provided by the authors. The method from [32] is a single image based deep learning approach. Since the authors did not release their implementations, we reimplemented their network and trained it on our datasets for fair comparisons. To ensure our implementation is correct, we generated the same dataset as described by [32] and trained our implemented network with it. In our experiments, the test performance is similar to what was reported in [32], in terms of both quantitative and qualitative metrics on their dataset. Since the dataset from [32] is for single image based method, we cannot evaluate our algorithm with that dataset.

	PSNR↑ (dB)			SSIM↑	
Networks	CRM	CR	FR	CR	FR
Net-autoenc-1	19.04	18.96	23.41	0.60	0.70
Net-autoenc-2	21.39	21.33	26.07	0.67	0.75
Net-disp	21.24	21.10	25.74	0.67	0.73
Net-vel-self	27.31	26.87	26.77	0.82	0.76
Ours	**27.78**	**27.30**	**27.04**	**0.84**	**0.77**

Table 1: **Ablation study on the network architectures and loss function.**

Evaluation metrics: We use the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) for quantitative comparisons. Both PSNR and SSIM metrics are commonly used to measure the similarity between images (see e.g. [19, 21]). Larger PSNR/SSIM values indicate better image quality.

Ablation study on the network architectures: We implemented several baseline networks to justify the design of our network architecture. We remove the motion estimation network and differentiable forward warping block to have a vanilla auto-encoder network. We also modify the image encoder such that it can accept a single rolling shutter image as input. In total, we have two auto-encoder networks for comparison. We denote them with **Net-autoenc-1** and **Net-autoenc-2** respectively. We also study the performance if we learn the displacement field directly, instead of the velocity field as described in Section 3. It is achieved by setting $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ all equal to $\mathbf{1}$, such that the estimated velocity field equals to the displacement field. We denote the network as **Net-disp**.

Table 1 presents the quantitative performances of the networks. It demonstrates that a vanilla auto-encoder network has difficulties to learn a good representation for rolling shutter effect removal. A possible reason is that the rectification problem involves non-local operations, which challenge the representation power of a vanilla auto-encoder network. Besides the dependencies of $\mathbf{U}_{r\rightarrow g}$ on the camera motion and 3D scene geometry, it also depends on the capture time of a particular pixel. We find it challenges the motion estimation network to estimate the displacement field directly. It is well explained by our experimental results, i.e., our network outperforms **Net-disp**. Furthermore, the experimental results also demonstrate that **Net-autoenc-2** network performs better than **Net-autoenc-1** network with around 2.37 dB improvement on the Carla-RS dataset and 2.66 dB improvement on the Fastec-RS dataset. It demonstrates that it is more difficult to learn a good representation with a single rolling shutter image compared to multiple images, due to the ill-posed nature of single image based method.

[1] https://www.fastecimaging.com/fastec-high-speed-cameras-ts-series/

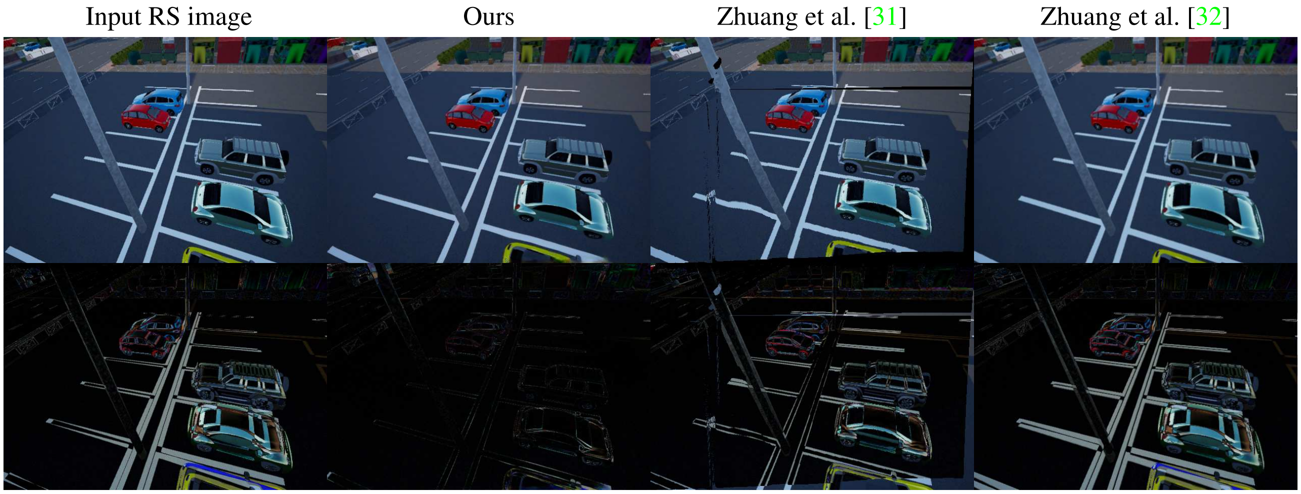

Figure 5: **Qualitative comparisons against state-of-the-art methods on the Carla-RS dataset. Second row:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}^g_{gt}$.

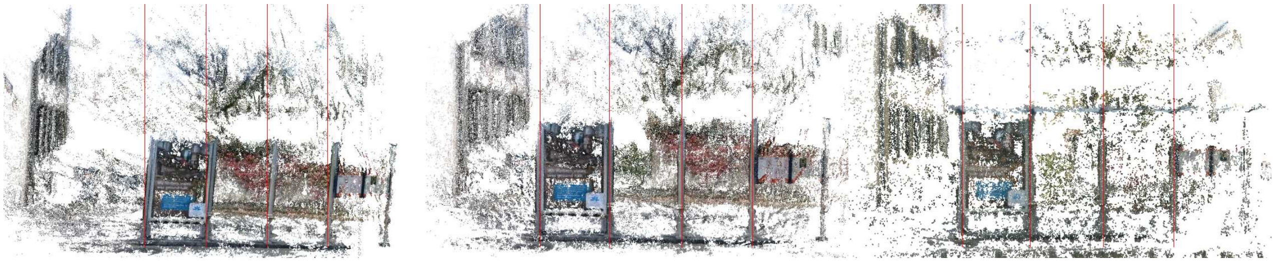

Figure 6: **Generalization performance on real data. Left:** Reconstructed 3D model with input rolling shutter images. **Middle:** Reconstructed 3D model with predicted global shutter images. **Right:** Reconstructed 3D model with real global shutter images.

Ablation study on the loss function: We did an ablation study to justify the loss functions that we used for network training, i.e., Eq. (11). We focus our attention on the explicitly supervision of the dense displacement field estimation. To achieve this, we introduce an additional loss function $\mathcal{L}_d$

$$\mathcal{L}_d = \left\|\mathbf{I}^r - \mathcal{W}_{g\to r} \circ \mathbf{I}^g_{gt}\right\|_1 , \tag{12}$$

where $\mathbf{I}^r$ is the latest input rolling shutter image, $\mathbf{I}^g_{gt}$ is the corresponding ground truth global shutter image, $\mathcal{W}_{g\to r}$ is an operator which warpes the global shutter image to its corresponding rolling shutter image and it depends on the estimated dense displacement field $\mathbf{U}_{r\to g}$. The warping is achieved by bilinear interpolations. The final loss function used to train the network can be formualted as

$$\mathcal{L}_f = \mathcal{L} + \lambda_1 \mathcal{L}_d, \tag{13}$$

where $\mathcal{L}$ represents the original loss function as shown in Eq. (11), λ_1 is a hyper-parameter and is empiracally selected as 10. We represent the network trained with the loss function $\mathcal{L}_f$ as **Net-vel-self**.

We train **Net-vel-self** with the same parameter configurations as other networks. Experimental results presented in Table 1 demonstrate that our network which is trained with only $\mathcal{L}$ performs better than **Net-vel-self**. The introduction of the self-supervision loss $\mathcal{L}_d$ for the dense displacement field $\mathbf{U}_{r\to g}$ does not help improve the performance of global shutter image estimation. A possible explanation is that the occlusions between $\mathbf{I}^r$ and $\mathbf{I}^g_{gt}$, which are used to supervise the learning of $\mathbf{U}_{r\to g}$, degrades the prediction of $\mathbf{U}_{r\to g}$ since we cannot do bi-directional occlusion detection. The forward feature warping for global shutter image recovery would thus be affected by the degraded $\mathbf{U}_{r\to g}$ and it further affects the final global shutter image prediction. It demonstrates that the loss function $\mathcal{L}$ presented in Eq. (11) is sufficient to implicitly supervise the learning of $\mathbf{U}_{r\to g}$.

Quantitative and qualitative evaluations against baseline methods: We compare our network with two state-of-the-art baseline methods. Both the quantitative and qualitative comparisons are presented in Table 2 and Fig. 5 respectively. The experimental results demonstrate that our method performs better than the other two state-of-the-art approaches. The work from Zhuang et al. [32] is a single image learning based approach. We find that it has limited generalization performance on our test data. The reason is that the scene content of our test data is quite different from

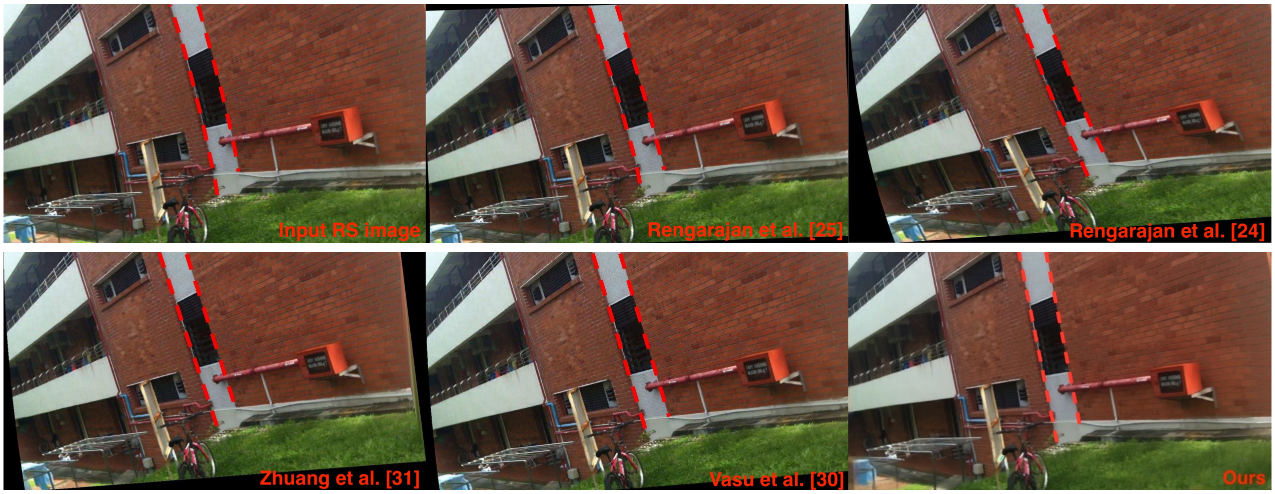

Figure 7: **Qualitative comparisons against conventional methods with dataset of [31].** It demonstrates that our network predicts a plausible rectification and also inpaints the occluded regions with the learned image priors.

that of our training data. The learned geometric priors from training data do not hold to the test data. In contrast, our network can be generalized well as we solve a generic rectification problem with two input frames. Zhuang et al. [31] is a classical approach with two input frames. We find it can work well if the input images have good textures. However, as shown in Fig. 5, it does not perform well for input frames with poorly textured regions, which results in visually unpleasing global shutter images. In contrast, our network predicts a plausible rectification with better image quality. Furthermore, our network can also take advantage of the learned image priors to fill in the occluded regions, from which the classical approach (i.e., Zhuang et al. [31]) is unable to reconstruct. This is also visible in the quantitative results, shown in Table 2, i.e., the PSNR metrics for Zhuang et al. [31] on Carla-RS dataset are 25.93 dB and 22.88 dB for the evaluations with occlusion masks and without masks respectively; The difference is 3.05 dB, however, ours is only 0.48 dB. It demonstrates our method handles occlusion better, since our network is able to inpaint the occluded regions from the learned image priors. Furthermore, our network is also orders of magnitude faster than Zhuang et al. [31]. It takes around 0.43 second to process a VGA resolution image (i.e., 640 × 480 pixels) with an Nvidia GTX 1080Ti graphic card, while Zhuang et al. [31] takes around 467.26 seconds on an Intel Core i7-7700K CPU. More qualitative results can be found from our supplementary material.

Generalization performance to real data: To evaluate the generalization performance of our network, we also collect a sequence of real rolling shutter images with a Logitech C210 webcam. The camera is mounted on the side of a ground vehicle which moves forward. For comparison, we also collect global shutter images with the same camera (i.e., the camera is stationary while capture). The rolling shutter images are then rectified with our pretrained network. We run a SfM pipeline (i.e., COLMAP [28]) to process the rolling shutter images, the rectified rolling shutter images, and the global shutter images respectively. Fig. 6 demonstrates that our pretrained network corrects the distortion and results in a more accurate 3D model as the ground truth model. We also evaluate the generalization performance of our network against conventional methods with the dataset from Zhuang et al. [31]. The results presented in Fig. 7 demonstrate that our network predicts a plausible rectification and also inpaints the occluded regions with the learned image priors.

	PSNR↑ (dB)			SSIM↑	
Methods	CRM	CR	FR	CR	FR
Zhuang et al. [31]	25.93	22.88	21.44	0.77	0.71
Zhuang et al. [32]	18.70	18.47	N.A.	0.58	N.A.
Ours	**27.78**	**27.30**	**27.04**	**0.84**	**0.77**

Table 2: **Quantitative comparisons against the state-of-the-art methods.** Since the Fastec-RS dataset does not have ground truth depth and motion, we cannot evaluate Zhuang et al. [32] with it.

6. Conclusion

We propose an efficient end-to-end deep neural network for generic rolling shutter image correction. Our network takes two consecutive frames and estimates the global shutter image corresponding to the latest frame. It is able to take advantage of the representational power of a deep network and outperforms existing state-of-the-art methods. We also present two large datasets, which simulate the real image formation process of a rolling shutter image.

References

[1] Cenek Albl, Zuzana Kukelova, Viktor Larsson, and Tomas Pajdla. Rolling shutter camera absolute pose. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2019. 1

[2] Simon Baker, Eric Bennett, Sing Bing Kang, and Richard Szeliski. Removing rolling shutter wobble. In *CVPR*, 2010. 2

[3] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018. 5

[4] James M. Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *Advances in Neural Information Processing Systems (NIPS)*, 2000. 2

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. 5

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 1

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981. 2

[8] Per Erik Forssen and Erik Ringaby. Rectifying rolling shutter video from hand-held devices. In *CVPR*, 2010. 2, 4

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 5

[10] Matthias Grundmann, Vivek Kwatra, Daniel Castro, and Irfan Essa. Calibration free rolling shutter removal. In *Proc. of the IEEE International Conf. on Computational Photography (ICCP)*, 2012. 2

[11] Johan Hedborg, Per-Erik Forssén, Michael Felsberg, and Erik Ringaby. Rolling shutter bundle adjustment. In *CVPR*, 2012. 1

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 5

[14] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. In *Technical report, Stanford*, 2011. 2

[15] Jae-Hak Kim, Yasir Latif, and Ian Reid. Rrd-slam: Radial-distorted rolling-shutter direct slam. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017. 1

[16] Bryan Klingner, David Martin, and James Roseborough. Street view motion-from-structure-from-motion. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 1

[17] Yizhen Lao and Omar Ait-Aider. A robust method for strong rolling shutter effects correction using lines with automatic feature selection. In *CVPR*, 2018. 2

[18] Chia-Kai Liang, Li-Wen Chang, and Homer H. Chen. Analysis and compensation of rolling shutter effect. In *Proc. of the IEEE Transactions on Image Processing (ITIP)*, 2008. 2

[19] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 6

[20] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the International Joint Conf. on Artificial Intelligence (IJCAI)*, 1981. 2

[21] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 6

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 6

[23] Pulak Purkait, Christopher Zach, and Ales Leonardis. Rolling shutter correction in manhattan world. In *ICCV*, 2017. 2

[24] Vijay Rengarajan, Yogesh Balaji, and A.N. Rajagopalan. Unrolling the shutter: Cnn to correct motion distortions. In *CVPR*, 2017. 1, 3, 5

[25] Vijay Rengarajan, A.N. Rajagopalan, and R. Aravind. From bows to arrows: rolling shutter rectification of urban scenes. In *CVPR*, 2016. 2, 3

[26] Olivier Saurer, Kevin Köser, Jean-Yves Bouguet, and Marc Pollefeys. Rolling shutter stereo. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2013. 1

[27] Olivier Saurer, Marc Pollefeys, and Gim Hee Lee. Sparse to dense 3d reconstruction from rolling shutter images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[28] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[29] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[30] Subeesh Vasu, Mahesh Mohan, and A.N. Rajagopalan. Occlusion aware rolling shutter rectification of 3d scenes. In *CVPR*, 2018. 1, 2

[31] Bingbing Zhuang, Loong Fah Cheong, and Gim Hee Lee. Rolling shutter aware differential sfm and image rectification. In *ICCV*, 2017. 2, 6, 7, 8

[32] Bingbing Zhuang, Quoc-Huy Tran, Pan Ji, Loong-Fah Cheong, and Manmohan Chandraker. Learning structure-and-motion-aware rolling shutter correction. In *CVPR*, 2019. 1, 2, 3, 5, 6, 7, 8