Supplementary Material for 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

1. Introduction

In this supplementary material, we will present the details of our network architectures, the derivations and implementation details of our differentiable forward warping block. We will also present additional experimental results.

2. Network architectures

In this section, we present the details of our network architecture and all the other baseline networks proposed in our ablation study.

Deep shutter unrolling network: Fig. 1 demonstrates the overall architecture of our deep shutter unrolling network. 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 details of the image encoder network are shown in Fig. 2. It consists of three pyramid levels. Each level has a convolutional layer followed by a ReLU activation function and three ResNet blocks [1]. The convolutional layer of the first pyramid level has a kernel size 7×7, 32 filters and uses a stride size 1. The convolutional layers of both the second pyramid level and the third pyramid level have a kernel size 3×3 and use a stride size 2 to down-sample the learned feature representations. The number of filters are 64 and 128 respectively. The number of filters for the corresponding ResNet blocks are 32, 64 and 128 for the first, the second and the third pyramid levels, respectively.

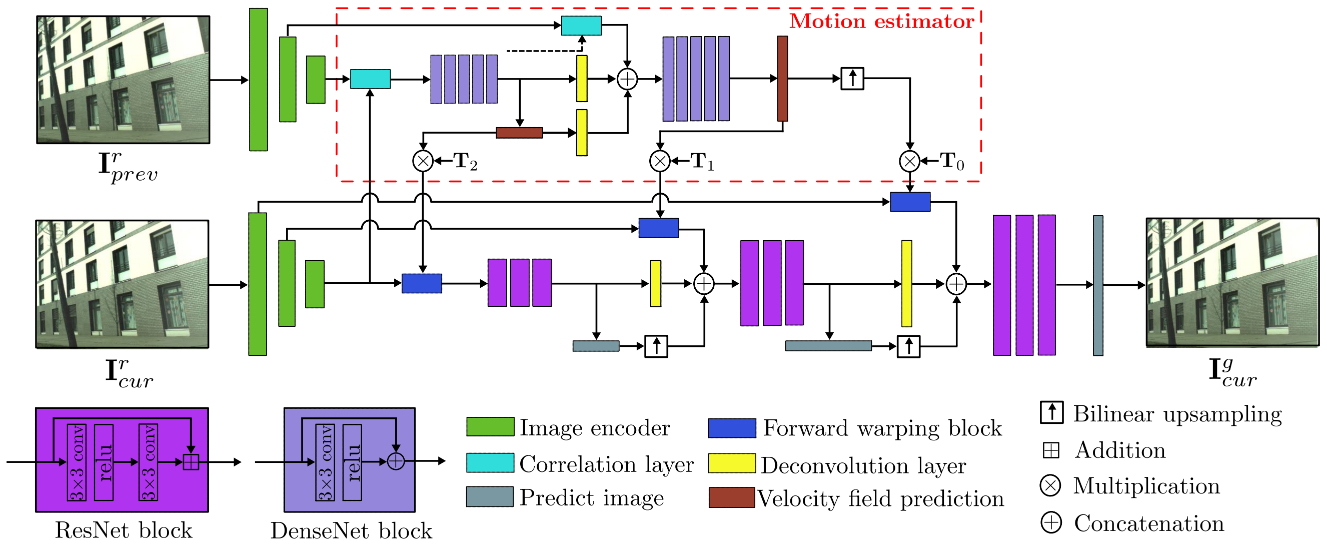

Figure 1: **Overall architecture of the deep shutter unrolling network.**

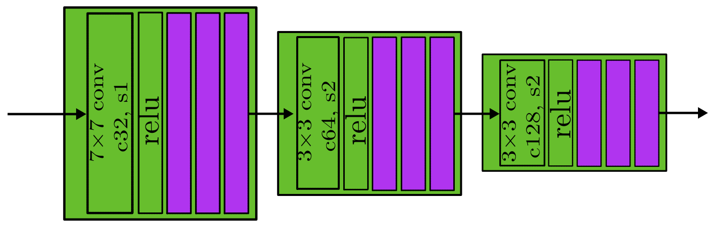

Figure 2: **Image encoder network.**

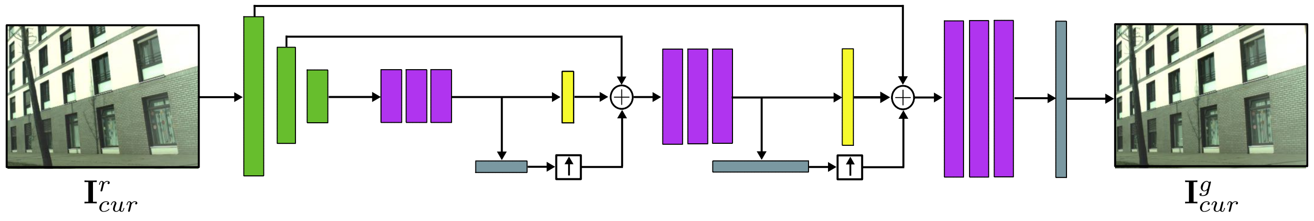

Figure 3: **Architecture of Net-autoenc-1 network.**

The motion estimation network estimates the dense displacement field $\mathbf{U}_{r\rightarrow g}$ from the rolling shutter image to its corresponding global shutter image. It computes the matching cost volumes based on the learned feature representations by the image encoder. The matching cost volumes are computed by a correlation layer, which is usually used in optical flow prediction networks [3, 5]. Five DenseNet blocks [2] are used to learn the dense velocity field for both the third pyramid level and the second pyramid level, from the computed matching cost volumes. We bi-linearly upsample the estimated dense velocity field from the second pyramid level for the first pyramid level as shown in Fig. 1. The DenseNet block consists of a convolutional layer followed by a ReLU activation function. The convolutional layer has a 3×3 kernel size and a stride size 1. Details can be found in Fig. 1. The number of filters for the five DenseNet blocks are 128, 128, 96, 64 and 32 for the third pyramid level. Similarly, we have 64, 64, 48, 32 and 16 for the second pyramid level. The deconvolutional layers have a kernel size 4×4, a stride size 2, a padding size 1 and 2 filters. The velocity field prediction layer has a 2D convolutional layer with a kernel size 3×3, a stride size 1, a padding size 1, and 2 filters.

We will present the details of the differentiable forward warping block in Section 3. The image decoder network has three pyramid levels. Each pyramid level has three ResNet blocks [1] followed by a deconvolutional layer and an image prediction layer. All the ResNet blocks have two convolutional layers with a kernel size 3×3, a stride size 1 and a padding size 1. The number of filters is equal to the input number of channels. In particular, the number of filters are 128, 64+3+3 and 32+3+3 respectively. The deconvolutional layers have a kernel size 4×4, a stride size 2, a padding size 1 and 3 filters. The image prediction layer has a 2D convolutional layer with a kernel size 3×3, a stride size 1, a padding size 1, and 3 filters.

Net-autoenc networks: Fig. 3 demonstrates the architecture of the Net-autoenc-1 network. It shares the same image encoder and decoder as our deep shutter unrolling network. The Net-autoenc-2 network is obtained by modifying the first convolutional layer of the image encoder network to accept two concatenated images as input. The rest layers are the same as that of the Net-autoenc-1 network.

Net-disp network: The Net-disp network has the same architecture as the deep shutter unrolling network. We simply set the time matrices to be $\mathbf{1}$, i.e., $\mathbf{T}_0 = \mathbf{1}$, $\mathbf{T}_1 = \mathbf{1}$ and $\mathbf{T}_2 = \mathbf{1}$.

3. Differentiable forward warping block

In this section, we present the detailed derivations of our differentiable forward warping block. Since we need to incorporate it as part of our network, we thus also need the partial derivatives from the outputs to the inputs. Due to the varying number of neighboring pixels, i.e., the size of $\Omega(\mathbf{x})$ from Eq. (1), it is not trivial to take advantage of the automatic differentiation tools from PyTorch [4] for the derivative computation. We thus derive and implement all the analytical partial derivatives by ourselves.

Forward pass: Without loss of generality, we use a particular pixel to illustrate our formulation. As described in the main paper, we can approximate the intensity of a pixel $\mathbf{x}$ from the global shutter image, as a weighted average of its neighboring pixel intensities from the rolling shutter image. It can be formally defined as

$$\mathbf{I}^g(\mathbf{x}) = \frac{\sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} \mathbf{I}^r(\hat{\mathbf{x}})}{\sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}}, \tag{1}$$

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

$$\|\hat{\mathbf{x}} + \mathbf{u}_{r\to g} - \mathbf{x}\|_2 < r, \tag{2}$$

where r is a pre-defined threshold with unit in pixels. The weight is further defined as

$$\omega_{\hat{\mathbf{x}}} = e^{-\frac{d(\mathbf{x},\hat{\mathbf{x}})^2}{2\sigma^2}}, \tag{3}$$

where

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \|\hat{\mathbf{x}} + \mathbf{u}_{r\to g} - \mathbf{x}\|_2, \tag{4}$$

and σ is a pre-defined width of the kernel function.

Backward pass: From the above equations, we can find the inputs of the formation model are the rolling shutter image $\mathbf{I}^r(\hat{\mathbf{x}})$ and the optical flow $\mathbf{u}_{r\to g}$. The output is the global shutter image $\mathbf{I}^g(\mathbf{x})$. In order to incorporate it into our network, we thus need to have both $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{I}^r(\hat{\mathbf{x}})}$ and $\frac{\partial \mathbf{I}^g(\mathbf{x})}{\partial \mathbf{u}_{r\to g}}$. Without loss of generality, we take a particular channel of a particular pixel to derive the partial derivatives as follows.

$$\frac{\partial I_c^g(\mathbf{x})}{\partial I_c^r(\mathbf{x}')} = \frac{\omega_{\mathbf{x}'}}{\sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}}, \tag{5}$$

where $I_c^g(\mathbf{x})$ and $I_c^r(\mathbf{x}')$ are the c^{th} channel intensity of pixel $\mathbf{x}$ and $\mathbf{x}'$ respectively, $\mathbf{x}'$ is an element from $\Omega(\mathbf{x})$. Similarly, we can have

$$\frac{\partial I_c^g(\mathbf{x})}{\partial \mathbf{u}_{r\to g}} = \frac{\partial I_c^g(\mathbf{x})}{\partial \omega_{\mathbf{x}'}} \frac{\partial \omega_{\mathbf{x}'}}{\partial \mathbf{u}_{r\to g}}, \tag{6}$$

where $\mathbf{u}_{r\to g}$ is the displacement vector of pixel $\mathbf{x}'$. From Eq. (1), we can further get

$$\frac{\partial I_c^g(\mathbf{x})}{\partial \omega_{\mathbf{x}'}} = \frac{I_c^r(\mathbf{x}') \cdot \sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} - \sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}} I_c^r(\hat{\mathbf{x}})}{(\sum_{\hat{\mathbf{x}} \in \Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}})^2}. \tag{7}$$

Similarly, from Eq. (3), we can get

$$\frac{\partial \omega_{\mathbf{x}'}}{\partial \mathbf{u}_{r\to g}} = \frac{\partial \omega_{\mathbf{x}'}}{\partial d^2} \frac{\partial d^2}{\partial \mathbf{u}_{r\to g}}, \tag{8}$$

where

$$d^2 = \|\mathbf{x}' + \mathbf{u}_{r\to g} - \mathbf{x}\|_2^2 = (x' + u_x - x)^2 + (y' + u_y - y)^2, \tag{9}$$

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{10}$$

$$\mathbf{u}_{r\to g} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \tag{11}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{12}$$

We can further get

$$\frac{\partial \omega_{\mathbf{x}'}}{\partial d^2} = -\frac{1}{2\sigma^2} e^{-\frac{d^2}{2\sigma^2}}, \tag{13}$$

$$\frac{\partial d^2}{\partial \mathbf{u}_{r\to g}} = \begin{bmatrix} \frac{\partial d^2}{\partial u_x} & \frac{\partial d^2}{\partial u_y} \end{bmatrix} = \begin{bmatrix} 2(x' + u_x - x) & 2(y' + u_y - y) \end{bmatrix}. \tag{14}$$

Implementation details: From the above equations, we can find that we need to get $\Omega(\mathbf{x})$ before we compute both the forward pass and the backward pass. However, a direct implementation of the above equations is in-efficient, due to the varying size of $\Omega(\mathbf{x})$. We therefore propose an efficient implementation to solve the above challenge, such that it can fully take advantage of the computational capability of a graphic card.

Instead of pre-computing $\Omega(\mathbf{x})$ by checking every pixel of the global shutter image, we iterate over every pixel of the rolling shutter image. For each pixel $\mathbf{x}'$ of the rolling shutter image, we accumulate all the necessary computations related to pixel $\mathbf{x}$ from the global shutter image, which satisfies

$$\|\mathbf{x}' + \mathbf{u}_{r\to g} - \mathbf{x}\|_2 < r, \tag{15}$$

where r is the pre-defined threshold. Furtheremore, we can also re-use the already accumulated $\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}$ and $\sum_{\hat{\mathbf{x}}\in\Omega(\mathbf{x})} \omega_{\hat{\mathbf{x}}}\mathbf{I}^r(\hat{\mathbf{x}})$ from the forward pass, for the backward pass. By following this approach, all the pixels $\mathbf{x}'$ can be processed in parallel, which makes the implementation very efficient on a graphic card.

4. Additional qualitative experimental results

In this section, we present additional experimental results. Both Fig. 4 and Fig. 5 demonstrate the qualitative comparisons against the baseline methods on the Carla-RS and Fastec-RS datasets, respectively. We also evaluate the temporal consistency of our network. Fig. 6 presents the predicted global shutter images from a continuous image sequence. The image sequence is captured by a real rolling shutter camera and used to reconstruct the 3D model shown in Fig. 4 in our main paper. The experimental results demonstrate that our network is able to estimate temporally consistent global shutter images.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2

[2] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[3] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv.org*, 1612.01925, 2016. 2

[4] 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. 2

[5] 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. 2

[6] Bingbing Zhuang, Loong Fah Cheong, and Gim Hee Lee. Rolling shutter aware differential sfm and image rectification. In *ICCV*, 2017. 5, 6

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

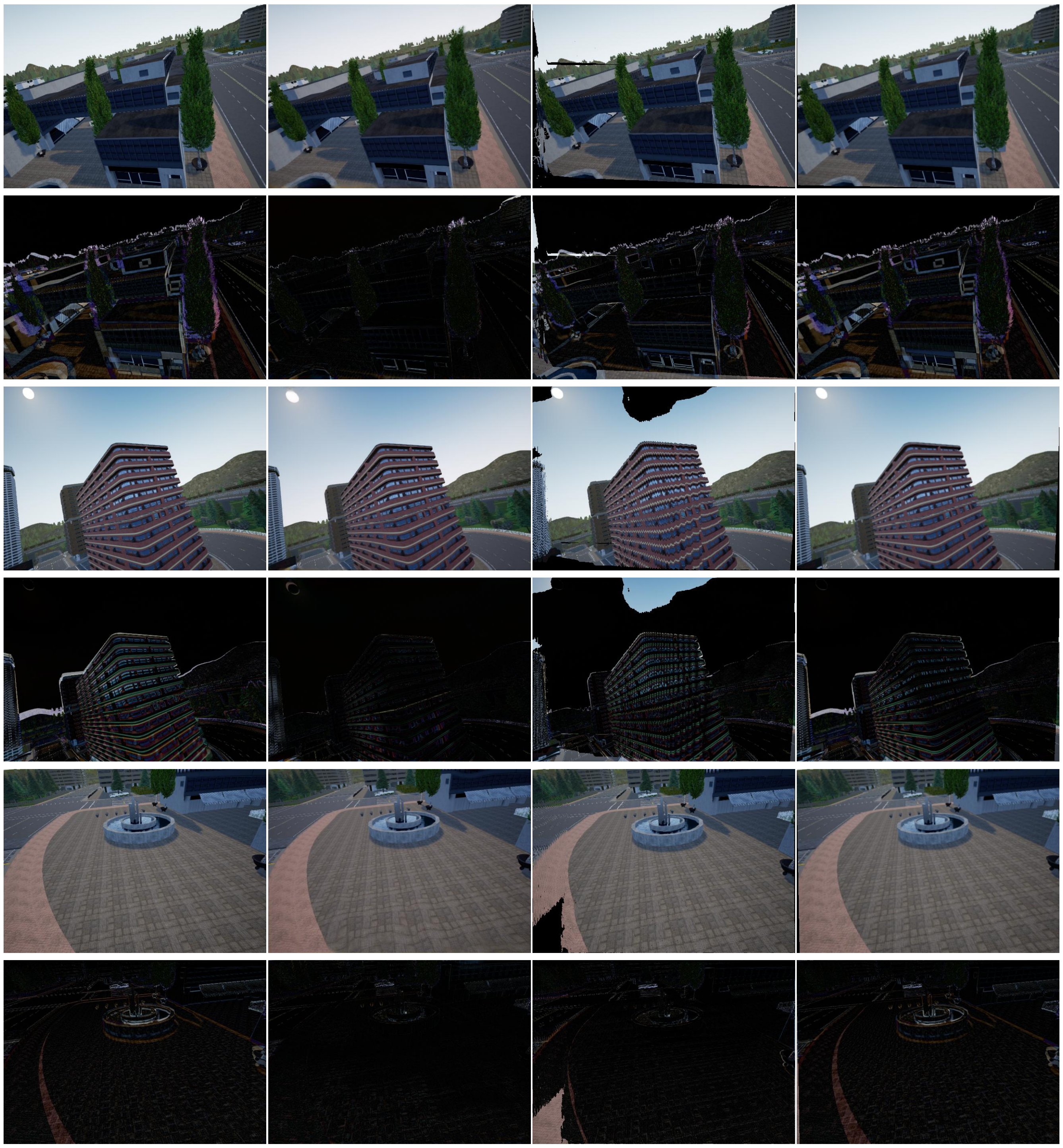

Figure 4: **Qualitative comparisons against baseline methods with the Carla-RS dataset. Even rows:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}_{gt}^{g}$. **First column:** Input rolling shutter image. **Second column:** Predicted global shutter image by our network. **Third column:** Predicted global shutter image by Zhuang et al. [6]. **Fourth column:** Predicted global shutter image by Zhuang et al. [7].

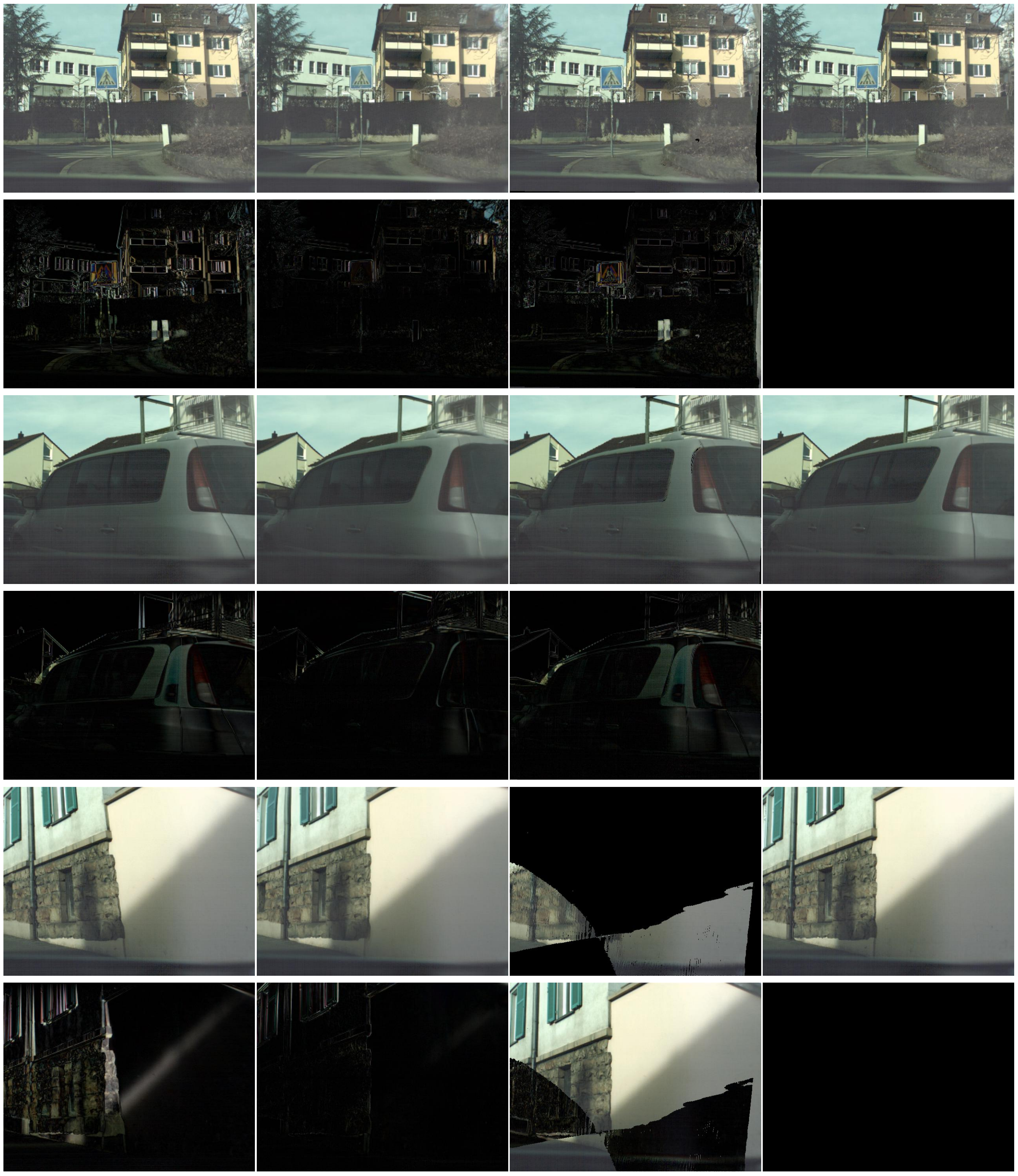

Figure 5: **Qualitative comparisons against baseline methods with the Fastec-RS dataset. Even rows:** Residual image, which is defined as the absolute difference between the corresponding image and the ground truth global shutter image $\mathbf{I}^{g}_{gt}$. **First column:** Input rolling shutter image. **Second column:** Predicted global shutter image by our network. **Third column:** Predicted global shutter image by Zhuang et al. [6]. **Fourth column:** Ground truth global shutter image.

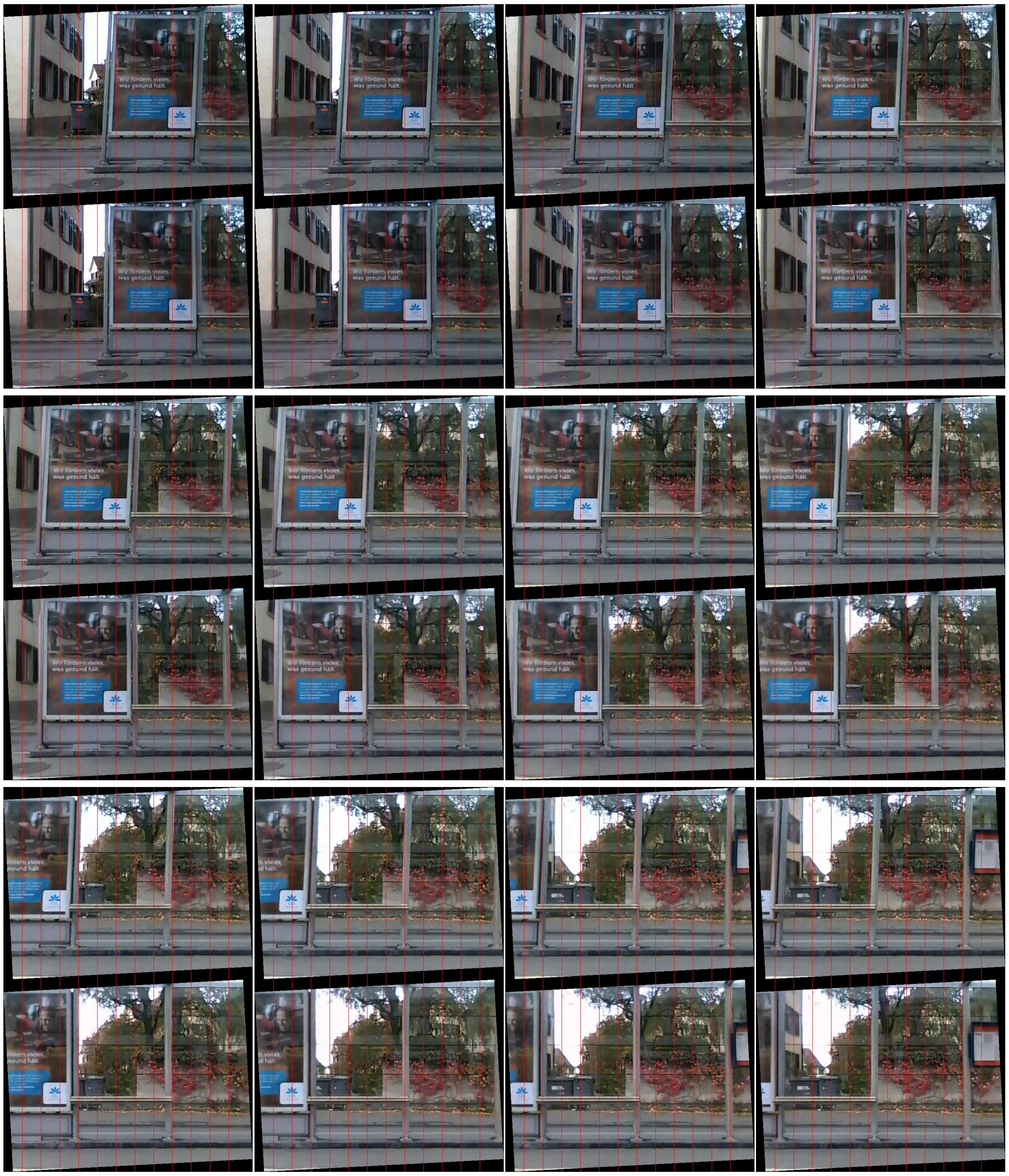

Figure 6: **Qualitative evaluation with a real rolling shutter image sequence. Odd rows:** Input rolling shutter image. **Even rows:** Predicted global shutter image by our pretrained model. For better visualization, the images are rotated by a constant offset to compensate the misalignment between the camera and the gravity direction. It demonstrates that our network is able to estimate temporally consistent global shutter images.