From two rolling shutters to one global shutter
Supplementary Material

Cenek Albl[1] Zuzana Kukelova[2] Viktor Larsson[1] Michal Polic[3] Tomas Pajdla[3] Konrad Schindler[1]

[1]ETH Zurich [2]VRG, FEE, CTU in Prague. [3]CIIRC, CTU in Prague

1. Detailed derivations

This section contains a more detailed derivation of the results presented in sections 3.4-3.6 in the main paper.

1.1. Translation in all axes

In this subsection, we present detailed derivation of equation (17) in the main paper, which describes the case of a general translational motion with constant velocity. In this case we have the translational velocity described by a vector $\mathtt{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\top$ and there is no rotational velocity, *i.e.* $\mathtt{R}_\omega(\alpha) = \mathtt{I}$. Substituting these in equation (4) in the main paper we obtain

$$\begin{aligned} \lambda_i \mathtt{u}_i &= \begin{bmatrix} \mathtt{I} \mid \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} v_i \end{bmatrix} \mathtt{X}_i \\ \lambda'_i \mathtt{u}'_i &= \begin{bmatrix} \mathtt{R}_r \mid \mathtt{R}_r \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} v'_i \end{bmatrix} \mathtt{X}_i \end{aligned} . \tag{1}$$

We are interested in a global shutter image that would be captured by a GS camera in a place of our camera pair, i.e., the first equation of (1) minus the translational motion

$$\lambda_i \mathtt{u}_{gi} = [\mathtt{I} \mid \mathtt{0}] \mathtt{X}_i. \tag{2}$$

By multiplying the second matrix equation in (1) from the left with $\mathtt{R}_r^\top = \mathrm{diag}([-1, -1, 1,])$ and by subtracting this equation from the first one, we obtain equation (17) in the main paper.

1.2. Translation in the xy-plane

Considering only the translation in the xy-plane, we have a translational velocity vector $\mathtt{t} = \begin{bmatrix} t_x & t_y & 0 \end{bmatrix}^\top$ and equations (17) in the main paper become

$$\begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} v_i - \begin{bmatrix} t_x \\ t_y \\ 0 \end{bmatrix} v'_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \lambda_i - \begin{bmatrix} -u'_i \\ -v'_i \\ 1 \end{bmatrix} \lambda'_i , \tag{3}$$

By subtracting the equation corresponding to the second row in (3) from the equation corresponding to the first we obtain a system of two equations in three unknowns t_x, t_y and λ_i

$$\begin{bmatrix} v_i - v'_i & 0 & -u_i - u'_i \\ 0 & v_i - v'_i & -v_i - v'_i \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ \lambda_i \end{bmatrix} = \mathtt{0} . \tag{4}$$

For a single correspondence this is a homogeneous system of linear equations of rank two (unless $v_i = v'_i$). Solving 4 brings us to equation (18) in the main paper.

1.3. Translation in the x-axis

For the motion only along the camera x-axis we have $\mathtt{t} = \begin{bmatrix} t_x & t_y & 0 \end{bmatrix}^\top$ and equations (17) in the main paper become

$$\begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix} v_i - \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix} v'_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \lambda_i - \begin{bmatrix} -u'_i \\ -v'_i \\ 1 \end{bmatrix} \lambda'_i. \tag{5}$$

From the third row we immediately see that $\lambda_i = \lambda'_i$, and thus the second row yields $v_i = -v'_i$. From the first row we see that

$$\begin{aligned} t_x v_i - t_x v'_i &= (u_i + u'_i)\lambda_i \\ t_x v_i + t_x v_i &= (u_i + u'_i)\lambda_i \\ t_x &= \tfrac{u_i + u'_i}{2v_i}\lambda_i, \end{aligned} \tag{6}$$

which gives us a direct relation between the perspective depth λ_i of the 3D point $\mathtt{X}_i$ and the translational velocity t_x. We are, again, interested in a global shutter image that would be captured by a GS camera in a place of our camera pair, i.e., the first equation of 1 minus the translational motion and we obtain

$$\lambda_i \mathtt{u}_{gi} = \lambda_i \mathtt{u}_i - \mathtt{t} v_i = \lambda_i \mathtt{u}_i - \begin{bmatrix} t_x \\ 0 \\ 0 \end{bmatrix} v_i . \tag{7}$$

Substituting t_x from equation (6) leads to

$$\begin{aligned} \lambda_i \mathtt{u}_{gi} &= \lambda_i \mathtt{u}_i - \begin{bmatrix} \frac{u_i+u_i'}{2v_i}\lambda_i \\ 0 \\ 0 \end{bmatrix} v_i \\ \mathtt{u}_{gi} &= \begin{bmatrix} \frac{u_i+u_i'}{2} \\ v_i \\ 1 \end{bmatrix} \end{aligned} , \tag{8}$$

which is the result presented in the paper, suggesting that we can simply interpolate the x-coordinate of the corresponding projections to obtain a GS equivalent one.

2. 6DOF solver with known baseline

As promised, we present an extension of the solution in section 3.1 of the main paper, which solves the case when there is a known, fixed baseline between the cameras. Although we did not use this solution for the results in the main paper, since the baseline between the cameras was negligible compared to the scene distance, it could be useful for systems with larger baselines such as stereo setups on robots, cars, etc. We can augment equations (5) in the main paper by a known baseline $\mathtt{b}$ as

$$\begin{aligned} \mathtt{P}(v_i) &= [\mathtt{R}_\omega(v_i) \mid \mathtt{t}\, v_i] \\ \mathtt{P}'(v_i') &= [\mathtt{R}_r \mathtt{R}_\omega(v_i') \mid v_i' \mathtt{R}_r \mathtt{t} + \mathtt{R}_r \mathtt{b}] \end{aligned} \tag{9}$$

and, analogously to the solution in the main paper, transform them so the first camera is identified with the world coordinate system, obtaining

$$\begin{aligned} \mathtt{P} &= [\mathtt{I} \mid \mathtt{0}] \\ \mathtt{P}'(v_i, v_i') &= [\mathtt{R}(v_i, v_i') \mid v_i' \mathtt{R}_r \mathtt{t} + \mathtt{R}_r \mathtt{b} - v_i \mathtt{R}(v_i, v_i') \mathtt{t}], \end{aligned} \tag{10}$$

where $\mathtt{R}(v_i, v_i') = \mathtt{R}_r \mathtt{R}_\omega(v_i') \mathtt{R}_\omega(v_i)^\top$. The essential matrix in equation (7) of the main paper will now become

$$\mathtt{E}(v_i, v_i') = [\mathtt{t}(v_i, v_i')]_\times \mathtt{R}(v_i, v_i') \ , \tag{11}$$

where $\mathtt{t}(v_i, v_i') = v_i' \mathtt{R}_r \mathtt{t} + \mathtt{R}_r \mathtt{b} - v_i \mathtt{R}(v_i, v_i') \mathtt{t}$. Note that we are now also solving for the scale of $\mathtt{t}$ with respect to the baseline, cannot use the parameterization $\mathtt{t} = \begin{bmatrix} 1-x & x & y \end{bmatrix}$ and therefore need six correspondences to solve for the six unknowns.

Similarly to the no-baseline case, we use the hidden variable trick. Hiding the translation $\mathtt{t}$, we get equations in only the rotation parameters. Applying the generator from [2] we generate a solver for this system as well. The solver performs Gaussian elimination on a 15×15 matrix and solves a 20×20 eigenvalue problem.

In Fig. 1 we compare the solvers presented in the main paper on data where cameras have an increasing baseline to the solution with known baseline provided here. One can see that the performance of the baseline solver provides stable performance over the increasing baseline.

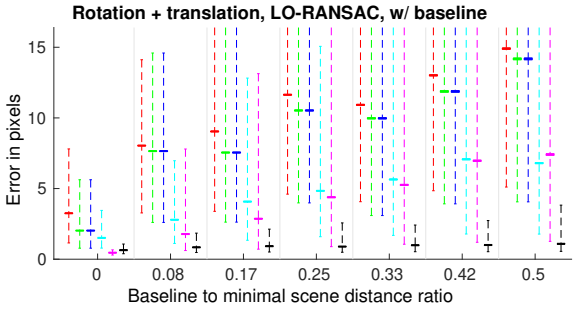

Figure 1: The solvers that do not consider a baseline (interpolation, txy, txyz, ω, ωt) gracefuly degrade in performance as the actual baseline in the data increases, whereas the 6DOF solver that considers a baseline (black) provides stable performance in terms of the pixel error of the undistorted correspondences with respect to the global shutter equivalent correspondences.

3. Combining the undistorted images

As described in section 5 of the main paper, when performing the undistortion of images distorted by rotational motion we actually obtain two undistorted images by warping each of the inputs to the virtual global shutter image plane. Each input image covers a part of a scene content that is not visible in the other image. Therefore, it makes sense to combine both undistorted images to obtain a more complete image. In Fig. 2 you can see several examples of the distorted inputs, the undistorted images and the combined output. Note that the images were only warped to the same image plane using the rotational velocity computed by ω solver and optimized using non-linear least squares refinement step minimizing equation (20) in the main paper, no other image processing was performed. One can notice small discontinuities at the boundaries, which could be improved by additional image processing software suited for image blending.

4. Correcting images distorted by translation

Pixel-wise correspondences

First, the pixel-wise correspondences must be found for both images. We found PWC-net [3] to work the best for these purposes. We compare the flow from the first image to the second as well as the flow from the second image to the first and filter out the flow vectors that are not consistent, see Fig.3. Since we know that the flow is caused by the difference in time of capture, that is zero for the middle row and increases towards the top and bottom of the image we can also filter the flow further by introducing a treshold on the maximum flow based on the distance from the middle row of the image.

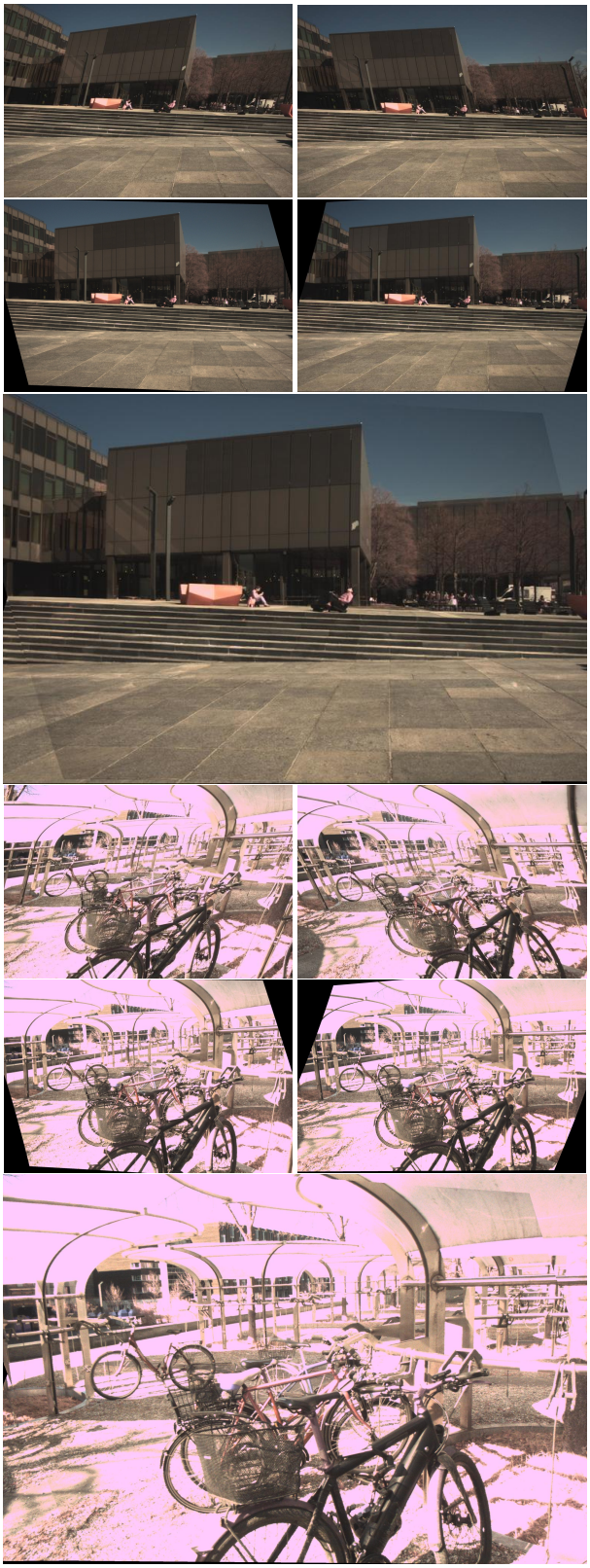

Figure 2: Combining the two undistorted outputs. Displayed are the input images (top), warped results (middle) and the combined result (bottom).

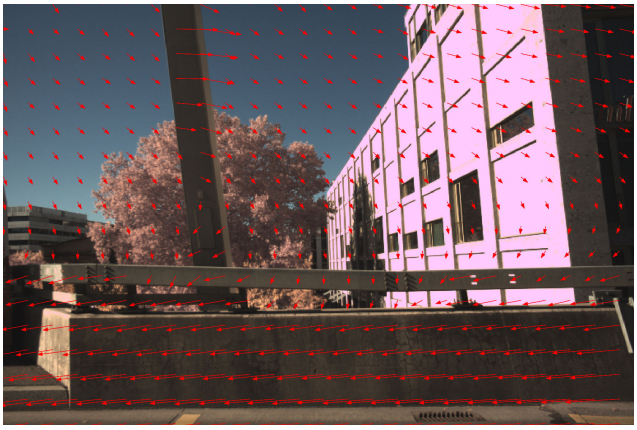

Figure 3: Flow estimated using PWC-net [3].

Motion estimation

Second step is computing the 6DOF motion parameters. Although, in principle we could use just the txy or even tx solver to solve for the translation if we knew the motion was purely translational we did not resort to such constraint, since our data was coming from a handheld camera travelling in a car and rotation could be present as well.

We have two choices of correspondences to use - sparse matches coming from, e.g., SIFT features or dense matches coming from the optical flow. Although sparse correspondences are usually more reliable, because they come from distinct image features and are usually sub-pixel precise, we found that they are in some cases not sufficient for correct motion estimation, since the parts of the image that determine the motion - e.g. the pole in Fig. 3 might contain only little texture and provide only few features. The correspondences coming from the flow are therefore a better candidate for a robust solution.

Choosing good correspondences for RANSAC

RANSAC is necessary for filtering mismatches. We found that to estimate correct motion parameters, the pixel-wise correspondences obtained from optical flow should not be used without proper pre-selection. To determine the motion, some correspondences carry more information than others. E.g. in our case, the correspondences around the middle row (where the temporal displacement is small) carry only little information and have low signal to noise ratio.

Another issue is that some types of motion cause very similar distortions to others, e.g. translation in x causes the same distortions as rotation around y if the scene is planar. Therefore, our motion estimation is prone to the presence of a dominant plane, similar to the estimation of epipolar geometry [1]. In our scenario, we found that for a general scene it is important to choose a balanced set of correspondences that contain both large displacements on the

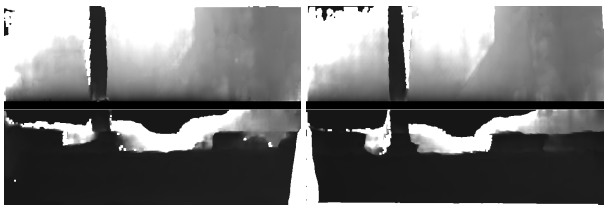

Figure 4: Depth maps projected in a virtual GS image plane, each created using one direction of the flow. Darker means closer. Notice the wrongly estimated flow in the bottom half of the image creates errors in the depth estimation.

Figure 5: Occlusion masks for both input images computed based on the flow. Based on those we decide from which image we take pixels in areas that are occluded. White areas mean we do not fill the pixels at this location in the final image from the respective input image.

foreground objects as well as the small displacements that appear towards the top or bottom of the image (around the middle row of the image all displacements will be small), because only using those we can distinguish between translational and rotational motion.

If, e.g. in the example in Fig. 3 we would select a uniform representation of correspondences accros the entire image, the correspondences on the slanted pole would be dominated by correspondences from other parts of the image in RANSAC, even though they are very important for determining the correct motion parameters.

Depth maps and occlusion masks

After computing the motion parameters, we can proceed to computing the pixel-wise depth from the flow. We compute two depth maps, one for each flow, see Fig. 4. The depth is computed in the following way. Using the motion parameters ω and $\mathtt{t}$ we create for each correspondence $\mathtt{u}_i \leftrightarrow \mathtt{u}'_i$ the camera matrices corresponding to their rows, i.e. $\mathtt{P}(v_i)$ and $\mathtt{P}(v'_i)$, using equation (4) in the main paper. We then use these projection matrices and the corresponding image points to triangulate a 3D point $\mathtt{X}_i$, which we project into a virtual GS camera coordinate system by $\mathtt{PX} = \begin{bmatrix}\mathtt{I} & \mathtt{0}\end{bmatrix} = \mathtt{X} = \begin{bmatrix}X_1 & X_2 & X_3\end{bmatrix}^\top$. The depth X_3 is then projected at the corresponding location in the image plane, i.e. $\begin{bmatrix}X_1/X_3 & X_2/X_3\end{bmatrix}$. A small number of rows around the middle of the image is filled with zeros as we consider the depth there to be too unreliable and we just use interpolation of the input images in this region.

By comparing the depth assigned to each pixel from the first and from the second flow we determine the occlusion mask, which tells us which image has pixel value that should be assigned to this location, see Fig. 5. This is important, since the flow is also estimated for pixels that lie in occluded regions and therefore don't actually have a correspondence and those should be filled with values from the image in which the occluding object was not present. If the mask is zero for given location, it means that this location in the final image should not be filled with pixels coming from the corresponding input image. White areas in figure 5 depict the areas with zeros.

Figure 6: Final depth map fused from the two in Fig.4. Notice that in the lower half of the image, errors in the estimated flow cause errors in the depth estimation. In the middle region the depth is undetermined.

Before the final undistortion we fuse the two depth maps, taking the closer values from each, see figure 6.

Creating the final image

Final image is created by traversing the depth map pixel by pixel, recovering the corresponding 3D point $\mathtt{X}$ and finding the coordinates where this point projects into either of the input images based on the occlusion mask. The pixel value is then taken from this location. The resulting image is the left column of Fig. 7.

Note that this approach is very generic and does not assume any properties of the motion, scene or image content and the results occasionaly contain artefacts due to errors in the input flow. In the places where good correspondences are provided, our methods are able to correct large RS distortion of very challenging scenes, see Fig. 7 right column.

The visual quality of the result could be improved by assuming scene properties such as piecewise planarity [4], segmenting the scene, applying more advanced filtering of the flow and depth maps and further post-processing steps. This was, however, not the purpose of this work.

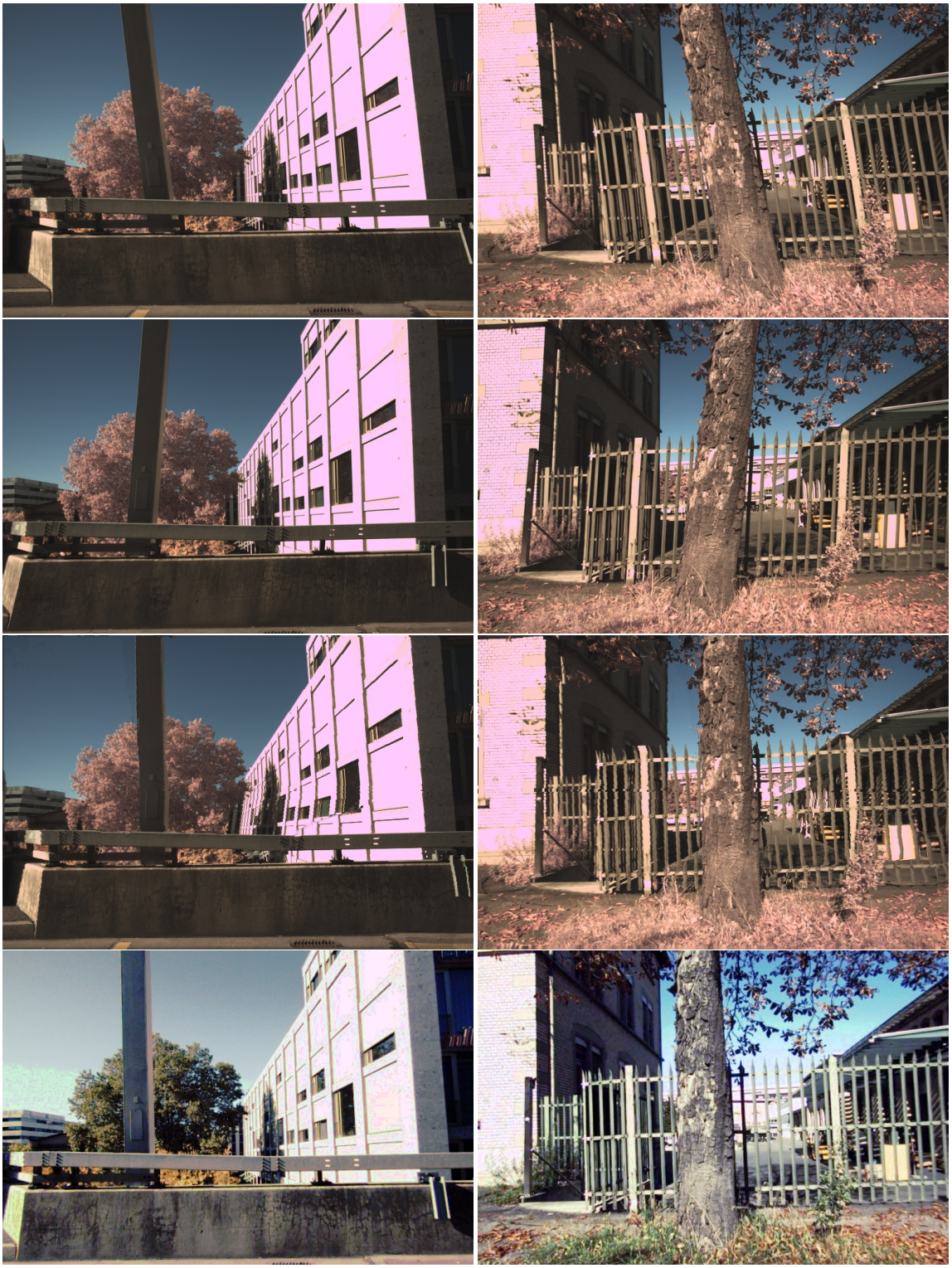

Figure 7: Undistortion examples on two image pairs, one in each column. From top to bottom - the two input images, the resulting undistorted image and the corresponding image from a GS camera.

References

[1] Ondrej Chum, Tomás Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, 2005.

[2] V. Larsson, Kalle Åström, and Magnus Oskarsson. Efficient Solvers for Minimal Problems by Syzygy-Based Reduction. In *CVPR*, 2017.

[3] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *CVPR*, 2017.

[4] Subeesh Vasu, Mahesh Mohan M. R., and A. N. Rajagopalan. Occlusion-Aware Rolling Shutter Rectification of 3d Scenes. In *CVPR*, 2018.