Supplementary Material for "Deep End-to-End Alignment and Refinement for Time-of-Flight RGB-D Module"

Di Qiu[1,2*] Jiahao Pang[1*] Wenxiu Sun[1] Chengxi Yang[1]
[1] SenseTime Research [2] The Chinese University of Hong Kong
sylvesterqiu@gmail.com, jpang@connect.ust.hk, {sunwenxiu,yangchengxi}@sensetime.com

1. Introduction

In this supplementary material, we first provide more discussions on our deep end-to-end alignment and refinement (DEAR) framework in Section 2. We then provide more details about the generation of our synthetic dataset and our data pre-processing procedure in Section 3. We also showcase more experimental results of our proposal in Section 4.

2. More Details on Our DEAR Framework

In this section, we first derive the formulation of subproblem (2) in the paper via multi-view geometry. We then provide the detailed network architectures being used in our work.

2.1. Derivation of Subproblem (2)

The key of deriving subproblem (2) in the paper is to obtain the relationship of the pixel locations between the first image (the RGB image) and the second image (the ToF amplitude image), where the second image is taken at the viewpoint defined by the camera parameters $\{t_x, t_y, c_x, c_y\}$. We adopt the simple linear camera model [3] since we have assumed the weakly calibrated setting. In this regard, we let the world coordinate to be aligned with the first camera, so that the first camera matrix is of the form

$$\mathbf{P} = \mathbf{K}(\mathbf{I}\,|\,\mathbf{0}) = \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (1)$$

We choose the measuring unit to be in pixels. Thus if $\mathbf{x} = (x, y, z)^{\mathrm{T}}$ is a scene point in the world coordinate (hence z is the depth with respect to the first camera), its imaged position $(x_1, y_1)^{\mathrm{T}}$ by the first camera can be calculated by

$$[\mathbf{P}(\mathbf{x}; 1)] = \begin{pmatrix} f_x x/z \\ f_y y/z \\ 1 \end{pmatrix} \equiv \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}, \quad (2)$$

where $(\mathbf{x}; 1) = (x, y, z, 1)^{\mathrm{T}}$ and $[\cdot]$ denotes the homogeneous coordinate representation. The matrix for the second camera is

$$\mathbf{P}' = \mathbf{K}'(\mathbf{I}\,|\,\mathbf{t}) = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_t \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (3)$$

and accordingly $\mathbf{x}$ is imaged in the second camera at

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{f_x(x+t_x)}{z} + c_x \\ \frac{f_y(y+t_y)}{z} + c_y \end{pmatrix}. \quad (4)$$

With (2) and (4), coordinates of the correspondence between the two images can be related by

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \frac{f_x t_x}{z} + c_x \\ \frac{f_y t_y}{z} + c_y \end{pmatrix}. \quad (5)$$

The above equation naturally leads to the formulation of subproblem (2) in the paper, which aims at minimizing the squared difference between the rough flow W_{rough} and the flow converted from depth W_{convt}. Note that since f_x, f_y are assumed to be known, in subproblem (2) they are respectively absorbed into t_x and t_y for simplicity.

Notice that (5) also plays a crucial role in the data augmentation based on multi-view geometry (Section 4.3 in the paper). With (5) we can generate images taken by the second camera given the depth information from the perspective of the first camera. Specifically, given randomly sampled $\{t_x, t_y, c_x, c_y\}$, the right hand side of (5) defines the underlying optical flow from the first image to the second image, which is used to warp the color image into a novel view defined by those parameters.

2.2. Detailed Network Architectures

We used FlowNetC for the cross-modal optical flow estimation; therefore, we refer the readers to [1] for its detailed architecture. Since FlowNetC takes two three-channel images as inputs, we repeat the one-channel ToF amplitude for

*Both authors contributed equally. Jiahao Pang is the corresponding author, this work was done while he was with SenseTime.

three times before feeding it to FlowNetC. The detailed architectures of the flow fusion network and the backbone of the depth refinement network are provided in Table 1. Both of these networks are U-Nets with skip connections.

Optical Flow Refinement Network						
Layer	**K**	**S**	**Channels**	**I**	**O**	**Input Channels**
conv0	3×3	1	4/64	1	1	$W_{\text{rough}}, W_{\text{convt}}$
conv1	3×3	2	64/64	1	2	conv0
conv1_1	3×3	1	64/128	2	2	conv1
conv2	3×3	2	128/128	2	4	conv1_1
conv2_1	3×3	1	128/128	4	4	conv2
$W^{(2)}_{\text{refn}}$	3×3	1	128/2	4	4	conv2_1
upconv1	4×4	2	130/128	4	2	conv2_1, $W^{(2)}_{\text{refn}}$
rconv1	3×3	1	256/64	2	2	upconv1, conv1_1
$W^{(1)}_{\text{refn}}$	3×3	1	64/2	2	2	rconv1
upconv0	4×4	2	66/64	2	1	rconv1, $W^{(1)}_{\text{refn}}$
rconv0	3×3	1	128/64	1	1	upconv0, conv0
W_{refn}	3×3	1	64/2	1	1	rconv0
Backbone U-Net of ToF-KPN						
conv0	3×3	1	5/64	1	1	$I_{\text{ToF}} \circ W_{\text{refn}}$, $D_{\text{ToF}} \circ W_{\text{refn}}$, I_{RGB}
conv0_1	3×3	1	64/64	1	1	conv0
conv1	3×3	2	64/128	1	2	conv0_1
conv1_1	3×3	1	128/128	2	2	conv1
conv2	3×3	2	128/128	2	4	conv1_1
conv2_1	3×3	1	128/128	4	4	conv2
conv3	3×3	2	128/256	8	8	conv2_1
conv3_1	3×3	1	256/256	8	8	conv3
upconv0	3×3	2	256/128	8	4	conv3_1
upconv0_1	4×4	1	128/128	4	4	upconv0
upconv1	3×3	2	256/128	4	2	conv2_1, upconv0_1
upconv1_1	4×4	1	128/128	2	2	upconv1
upconv2	3×3	2	256/64	2	1	upconv1, conv1_1
upconv2_1	4×4	1	64/64	1	1	upconv2
$\mathbf{w}, b$	3×3	1	64/10	1	1	upconv2

Table 1: Network architecture of the optical flow fusion network and the backbone of our ToF-KPN. $I_{\text{ToF}} \circ W_{\text{refn}}$ denotes the ToF amplitude image warped by the flow W_{refn} and $D_{\text{ToF}} \circ W_{\text{refn}}$ similarly denotes the warped ToF depth image. **K** means kernel size, **S** means stride, and **Channels** is the number of input and output channels. **I** and **O** are the input and output downsampling factor relative to the input. Separation by "," in the **Input Channels** means concatenation.

3. More Details on Data Generation and Pre-processing

This section briefly reviews the background of synthetic data generation and explains how to get simulated ToF depth from transient rendering. We also describe the data pre-processing procedure being used in our work.

3.1. More on Synthetic Data Generation

Transient rendering [6, 9] is a tool from computer graphics used to study the propagation of light in extremely short timescales. For ToF sensor with a single light source, transient rendering can be regarded as simulating the *temporal point spread function* (TPSF) of each pixel in the image that depends both on the camera and the scene. A TPSF encodes the temporal energy distribution of the homecoming light at its pixel. In case there is no MPI, the TPSF will be an impulse peaking at the true depth, otherwise the TPSF will have a scene-dependent tail. During rendering, we also adopt the assumption that the scenes contain mainly diffusive materials [10, 2, 7], which is valid for most real-life scenarios. Then each pixel of the raw ToF signal can be modeled as the integral over the exposure time of the temporal convolution between the modulated light and the TPSF. Since the TPSF captures the multi-path interference, it faithfully approximates the errors of ToF sensors in real life. We refer the readers to [4] for more mathematical details in this respect.

To generate the synthetic ToF measurements, let $\{I_t\}_{t=1}^{T}$ be the transient images of a scene under the point light source of the ToF sensor. We also let $L^{(\omega)}_{\text{sin}}$, $L^{(\omega)}_{\text{cos}}$ be the sine and cosine light waves with frequency ω, respectively. Then, the ToF correlation images at pixel p are obtained by:

$$\begin{aligned} C_{\text{sin}}(p,\omega) &= \sum\nolimits_{t=1}^{T} I_t(p) \cdot L^{(\omega)}_{\text{sin}}(t), \\ C_{\text{cos}}(p,\omega) &= \sum\nolimits_{t=1}^{T} I_t(p) \cdot L^{(\omega)}_{\text{cos}}(t), \end{aligned} \tag{6}$$

where $\{I_t(p)\}_{t=1}^{T}$ is simply the TPSF at the pixel p. The phase angle at pixel p used for depth conversion can then be determined by, *e.g.*, taking the argument of the complex number $C_{\text{cos}}(p,\omega) + iC_{\text{sin}}(p,\omega)$.

Furthermore, note that the depth obtained above in fact measures the distance from the scene point to the *light source*, rather than to the *image plane*, where the latter is used in our work (recall Equation (2) in Section 2.1). Therefore, in our synthetic dataset we also perform standard plane correction [3] to the depth obtained above so as to convert point-to-point distance to point-to-plane distance.

3.2. Data Pre-processing

Since raw ToF amplitude images, ToF depth images and the captured RGB images initially have intensities of different scales, proper data pre-processing and normalization is helpful for the training of neural networks [5], especially when the Siamese network (*i.e.*, the FlowNetC) is used in our work [1]. We first present our pre-processing procedure for the ToF amplitude images. The ToF amplitude images often exhibit extremely high contrast between the foreground and the background of the captured scenes. Simply re-scaling a ToF amplitude image into the range $[0, 1]$

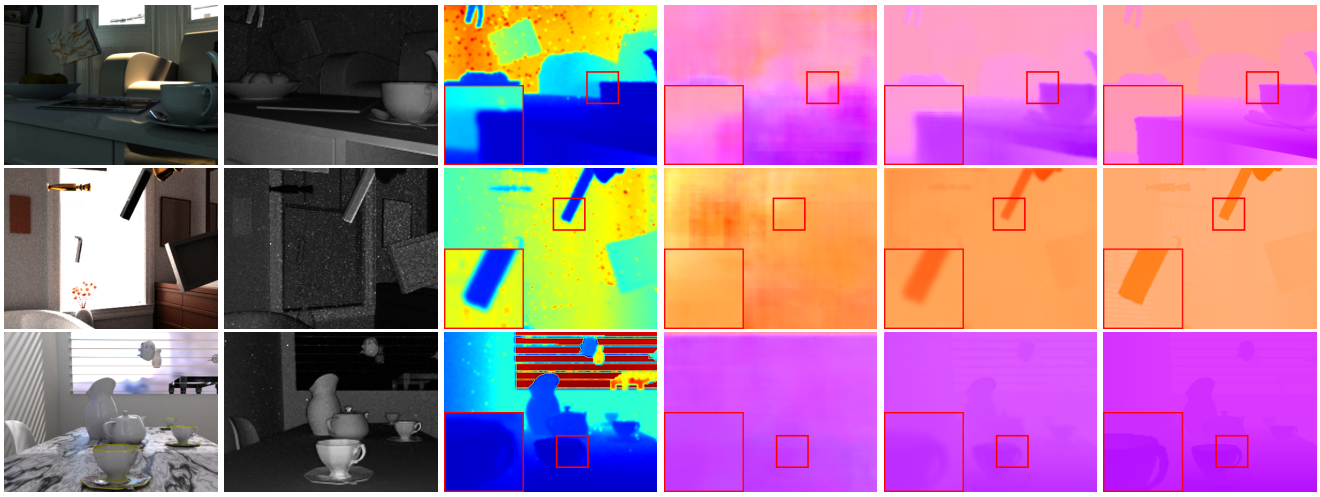


Figure 1: Visual comparisons before and after optical flow refinement. The refinement incorporates ToF depth image via a depth-to-flow conversion, which greatly enhances the accuracy of cross-modal optical flow estimation.

(*i.e.*, divide the image by its maximum intensity), or truncation (*i.e.*, set all values above certain threshold to be a same value) may result in overly dark or overly bright regions. Such an unbalanced intensity difference brings negative impact for the rough flow estimation as well as depth refinement. We have thus adopted a simple pre-processing to the raw ToF amplitude images to mitigate such effects.

Our approach is based on the fact that, the intensity of the ToF amplitude obeys an inverse square relationship to the distance. Specifically, we found the following simple pixel-wise transform to work well in practice:

$$I_{\text{ToF}} = I_{\text{ToF}}^{(\text{raw})} \odot D_{\text{ToF}}^2, \tag{7}$$

where $\odot$ denotes the *Hadamard product*, D_{ToF}^2 denotes the pixel-wise squaring of the ToF depth image D_{ToF}, and $I_{\text{ToF}}^{(\text{raw})}$ denotes the raw capture of ToF amplitude. The above normalization scheme can not only mitigate the huge contrast difference, but also make the overall brightness of different scenes roughly similar. After that, the obtained ToF amplitude images are further normalized to the range $[0, 1]$. All of the ToF amplitude images in our work, synthetic or real, used in training or testing, have been pre-processed with the above normalization scheme.

Data normalization is also performed for the RGB images and the depth images (both the ToF depth images and the ground-truth depth images). Specifically, for both of the training and testing, the RGB images and the depth images are all normalized to the interval $[0, 1]$.

4. More Experimental Results

We provide more experimental results in this section. We first present more results on our optical flow refinement via the ToF depth images. We then showcase more results demonstrating the effectiveness of our ToF-KPN. Finally, more results on our overall DEAR framework are presented.

4.1. More Visual Results on Flow Refinement

More visual results of our optical flow refinement module on the ToF-FlyingThings3D dataset are presented in Figure 1. We can see that, the quality of the optical flow is substantially improved by our flow refinement module.

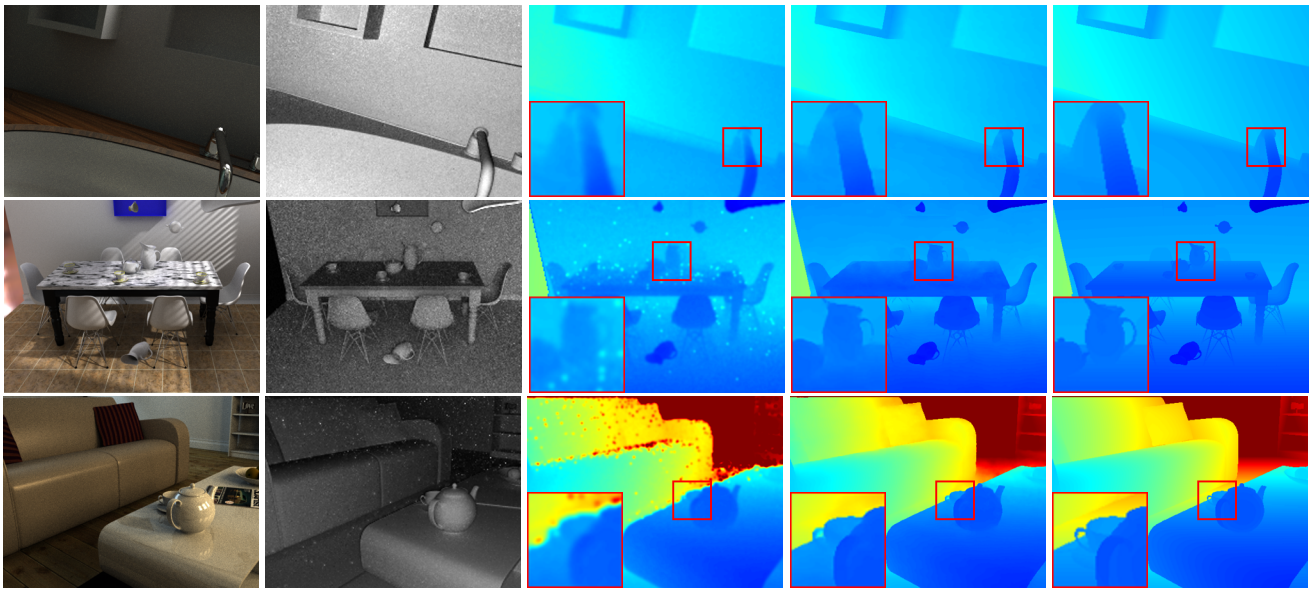
4.2. More Results on ToF-KPN

We hereby show more depth image refinement results of ToF-KPN. Specifically, we apply it onto our ToF-FlyingThings3D dataset where each data instance are already aligned. Some of the results are shown in Figure 2. It can be seen that our refinement results are very close to the ground-truth depth images.

We further demonstrate the MPI reduction of our ToF-KPN. Specifically, we present more comparisons to DEEPTOF [8] and the method of Su *et al.* [10] in Figure 3. We follow the identical settings as in Section 5.3 of the paper, and plot the depth values along scan-lines of four different scenes. We clearly see that, our ToF-KPN has greatly suppressed the MPI effects (compare to the original ToF depth images) while provides very high depth accuracies (compared to DEEPTOF [8] and Su *et al.* [10]).

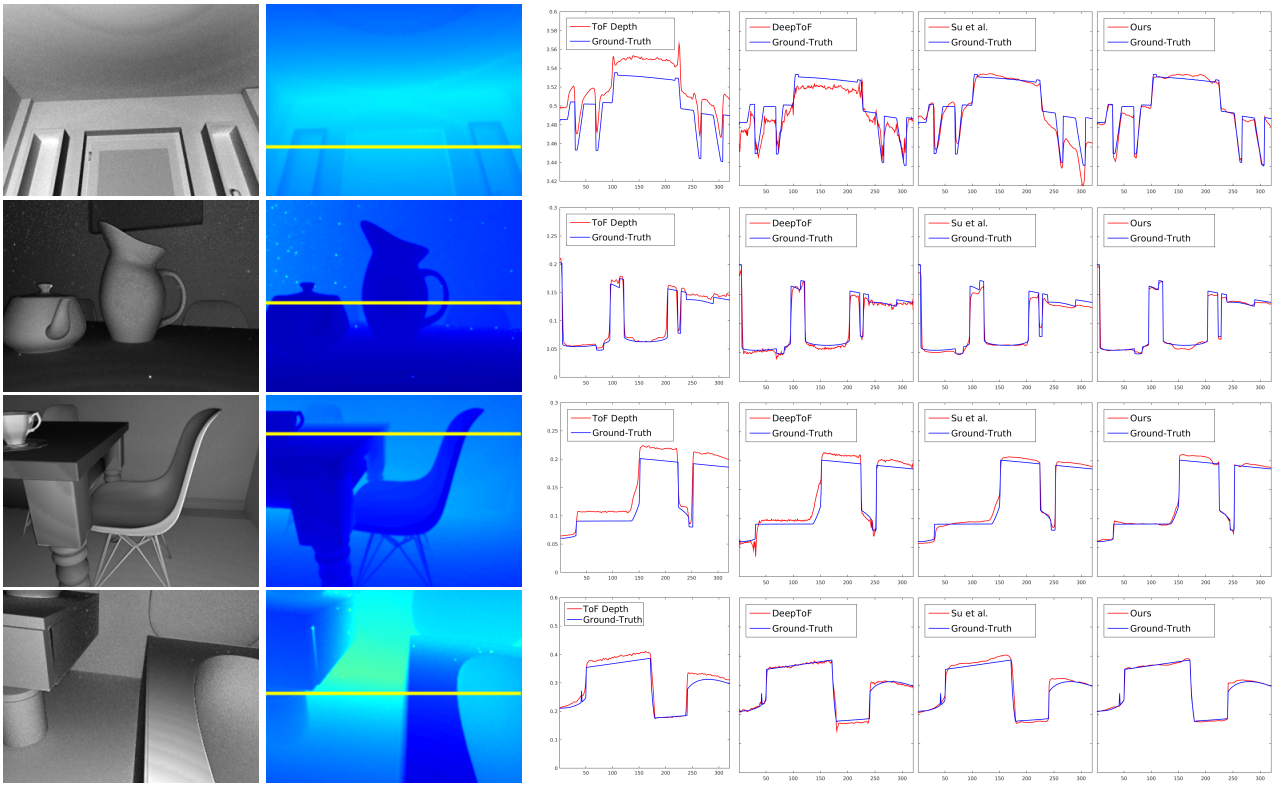
4.3. More Visual Results of DEAR

More results of our DEAR framework are shown in Figure 4, following the same settings of Section 5.4 of the paper. In Figure 4, the first three rows show the results on the synthetic data while the rest show results of our real data. It can be seen that, our DEAR framework provides visu-



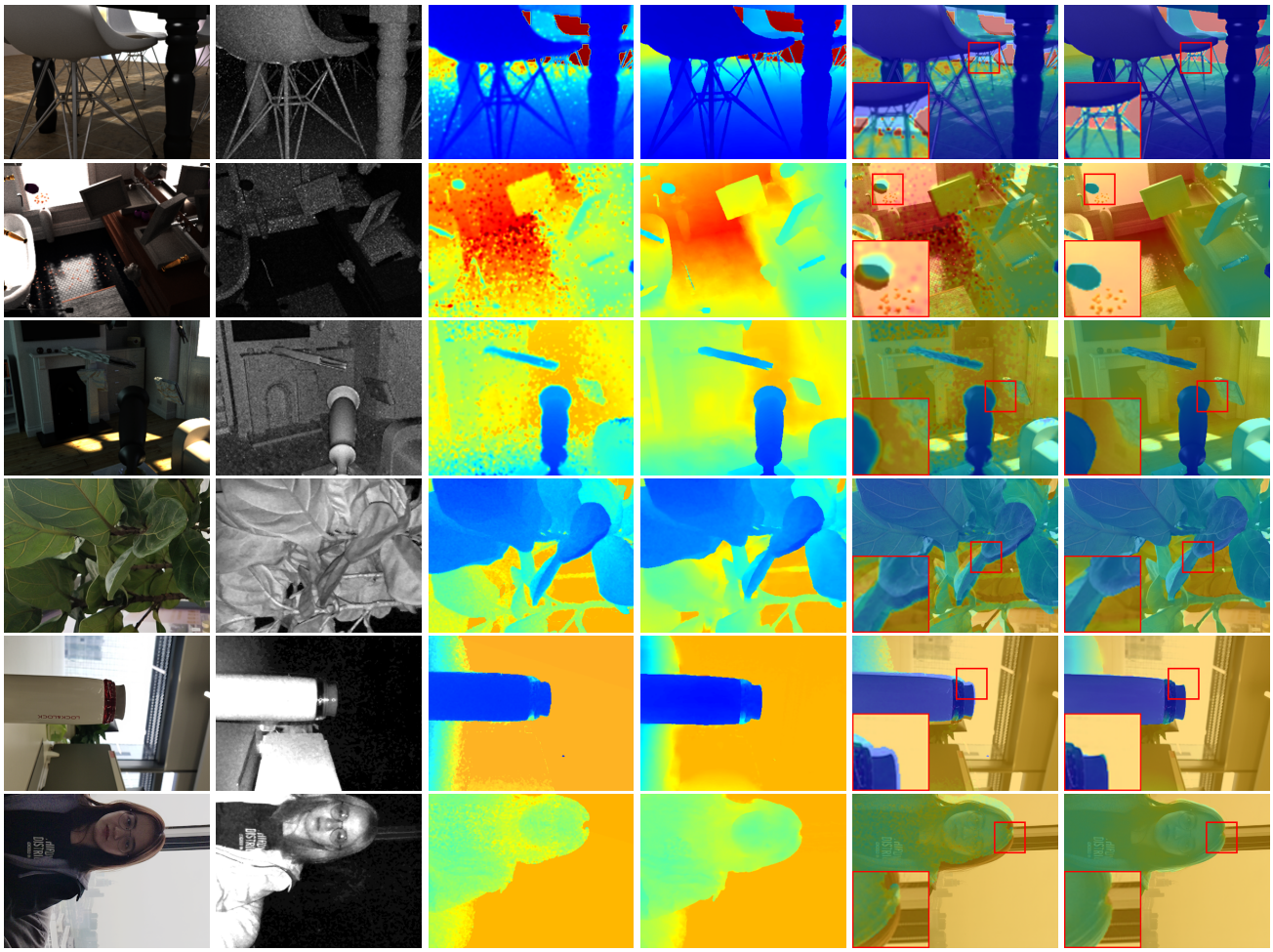
RGB image | ToF amplitude | ToF depth imasge | ToF-KPN (ours) | Ground-truth

Figure 2: Visual results of the ToF-KPN module for depth image refinement on the ToF-FlyingThings3D dataset.



ToF amplitude | ToF depth image | ToF depth values | DeepToF [8] | Su *et al.* [10] | ToF-KPN (ours)

Figure 3: Depth values of different approaches on a scan-line are shown, alongside with the ground-truth. Note that no color images are used in this experiment.



(a) RGB image (b) ToF amplitude (c) ToF depth (d) Results of DEAR (e) ToF depth + RGB (f) DEAR + RGB

Figure 4: Visual results of our deep end-to-end alignment and refinement framework. In the first three rows we show the results on synthetic data, while last three rows for real data taken by weakly calibrated ToF RGB-D camera modules.

ally pleasant depth results, which are not only well-aligned with the corresponding RGB images but also largely refined compared to the original ToF depth images.

References

[1] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2758–2766, 2015. 1, 2

[2] Mohit Gupta, Shree K Nayar, Matthias B Hullin, and Jaime Martin. Phasor imaging: A generalization of correlation-based time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 34(5):156, 2015. 2

[3] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 2

[4] Felix Heide, Matthias B Hullin, James Gregson, and Wolfgang Heidrich. Low-budget transient imaging using photonic mixer devices. *ACM Transactions on Graphics (ToG)*, 32(4):45, 2013. 2

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 2

[6] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. A framework for transient rendering. *ACM Transactions on Graphics (ToG)*, 33(6):177, 2014. 2

[7] Achuta Kadambi and Ramesh Raskar. Rethinking machine vision time of flight with GHz heterodyning. *IEEE Access*, 5:26211–26223, 2017. 2

[8] Julio Marco, Quercus Hernandez, Adolfo Munoz, Yue Dong, Adrian Jarabo, Min H Kim, Xin Tong, and Diego Gutierrez. DeepToF: Off-the-shelf real-time correction of multipath in-

terference in time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(6):219, 2017. 3, 4

[9] Adam Smith, James Skorupski, and James Davis. Transient rendering. Technical report, 2008. 2

[10] Shuochen Su, Felix Heide, Gordon Wetzstein, and Wolfgang Heidrich. Deep end-to-end time-of-flight imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6383–6392, 2018. 2, 3, 4