

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

Di Qiu^{1,2*} Jiahao Pang^{1*} Wenxiu Sun¹ Chengxi Yang¹

¹ SenseTime Research ² The Chinese University of Hong Kong

sylvesterqiu@gmail.com, jpang@connect.ust.hk, {sunwenxiu,yangchengxi}@sensetime.com

Abstract

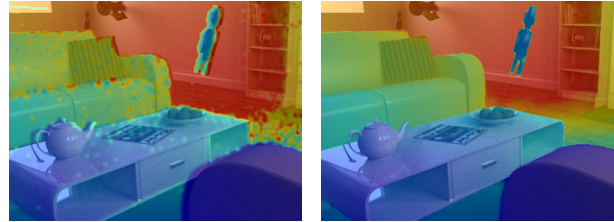
Recently, it is increasingly popular to equip mobile RGB cameras with Time-of-Flight (ToF) sensors for active depth sensing. However, for off-the-shelf ToF sensors, one must tackle two problems in order to obtain high-quality depth with respect to the RGB camera, namely 1) online calibration and alignment; and 2) complicated error correction for ToF depth sensing. In this work, we propose a framework for jointly alignment and refinement via deep learning. First, a cross-modal optical flow between the RGB image and the ToF amplitude image is estimated for alignment. The aligned depth is then refined via an improved kernel predicting network that performs kernel normalization and applies the bias prior to the dynamic convolution. To enrich our data for end-to-end training, we have also synthesized a dataset using tools from computer graphics. Experimental results demonstrate the effectiveness of our approach, achieving state-of-the-art for ToF refinement.

1. Introduction

Nowadays, RGB-D camera modules based on Time-of-Flight (ToF) sensors are becoming increasingly popular for mobile devices. At an affordable cost, it provides portable active depth measurements. In general, compared to monocular or stereo camera modules [12, 19, 25, 32, 33], ToF sensors provide higher precision depth values for short-range distance sensing [16]. However, off-the-shelf ToF RGB-D camera modules have two problems:

- (i) **Perspective difference:** The depth measurements are initially defined from the perspective of the ToF sensor, thus alignment between the depth images and RGB images is necessary;
- (ii) **Erroneous measurements:** depth measurements of ToF sensors suffer from different types of error such as multi-path interference, noise, *etc.*

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



(a) Unaligned erroneous depth image.

(b) Our result.

Figure 1: Proposed framework of alignment and refinement of ToF depth images for weakly calibrated ToF RGB-D module. The scene is chosen from our synthetic ToF-FlyingThings3D dataset.

These two problems hamper the direct usage of ToF RGB-D camera modules for applications such as computational photography, augmented reality and video entertainment.

Multi-view geometry sheds light on the first problem. In fact, pixel correspondences between the RGB image and the ToF amplitude image can be computed given the *true depth* from the perspective of either of the images accompanied with the *full set of camera parameters* [17]. However, under dynamic changes during deployment, mobile ToF RGB-D camera parameters can seldom be calibrated once and for all. In fact, modern RGB cameras are often equipped with optical image stabilization (OIS) systems which dynamically changes the principal points, alongside with other mild calibration degradation to the ToF RGB-D camera module. These impacts can be sufficiently modeled by the changes of the principal point c_x, c_y of the RGB camera, and the relative translation parameters t_x, t_y [7, 40]; while the rest of the parameters can be viewed as unchanged. Hence, it brings the need of performing online calibration and alignment for ToF RGB-D camera modules.

With the above practical setup, we assume the ToF sensor and the RGB camera have already been calibrated with standard procedure, *e.g.*, with [41], and therefore having known initial camera parameters. However, the set of parameters $\{c_x, c_y, t_x, t_y\}$ changes during deployment. We call such ToF RGB-D camera modules *weakly calibrated*. As a result, in the following we also assume both the ToF amplitude images and the ToF depth images pro-

vided to our framework *have already been rectified and warped* to the viewpoint of RGB camera according to the initial camera parameters,¹ However, random perturbations to $\{c_x, c_y, t_x, t_y\}$ lead to misalignment; so performing on-line alignment is a must.

Although a straightforward solution is to match their key points on the fly, this approach fails in practice because the imaging process of a ToF camera departs greatly from that of a standard RGB camera [16]. Above all, a ToF amplitude image is lightened up by a single light source located on the module. Moreover, since infra-red frequencies are used, the same material may have considerably different appearances in the ToF amplitude images and the color images.

To apply multi-view geometry directly, another difficulty is the second problem—erroneous measurements—as mentioned above. A ToF sensor approximates the true depth by estimating the phase shift of the received infra-red light, which is determined by the scene geometry, materials, the light source itself, *etc.* Apart from thermal noise which is common for electronic devices, a major source of error is the multi-path interference (MPI)—stems from the mechanisms of ToF sensor—making the depth measurements farther than the actual ones [16].

Given the coupled nature of the alignment and the refinement problems, it will be beneficial to solve them with the help from high-quality ToF RGB-D data. In this paper, we propose a novel end-to-end deep learning framework solving both the alignment and refinement tasks of depth images produced by off-the-shelf ToF RGB-D modules. Our key contributions include:

- (i) To address the alignment problem, we propose an effective two-stage method for estimating the cross-modal flow between the ToF amplitude and RGB image, utilizing the original depth measurements, and trained with dedicated data augmentation technique.
- (ii) For the ToF depth refinement problem, we propose an effective architecture, *ToF kernel prediction network* (ToF-KPN) which also employs the RGB images. With simple changes to the original KPN, we enable state-of-the-art performance in reducing MPI while enhancing the depth quality.
- (iii) It is difficult to collect sufficient real data with high-quality ground-truth for training. Hence, we *synthesize* a dataset for our problem with tools in computer graphics. We call our dataset ToF-FlyingThings3D, as we let various objects floating in the scenes similar to the FlyingThings3D dataset [28].

We call our *Deep End-to-end Alignment and Refinement* framework DEAR. Our paper is organized as follows. We

¹From the mechanisms of ToF sensor [16], we note that a ToF amplitude image and its corresponding ToF depth are essentially aligned.

review related works in Section 2. In Section A we elaborate our framework and in Section B we detail our data generation and collection strategy. Experimentation are presented in Section 5 and conclusions are provided in Section 6.

2. Related Work

To our best knowledge, we are the first in the literature to propose an end-to-end depth alignment and refinement framework for ToF RGB-D camera modules. Since none of the existing work has the same settings as ours, we briefly review works related to the two components of our framework, namely cross-modal correspondence matching and ToF depth image refinement.

Cross-modal correspondence matching. Our work performs online cross-modal dense correspondence matching, *i.e.*, optical flow estimation, between the ToF amplitude image and the RGB image, so as to address the alignment problem. In [5], the authors propose the Log-Gabor Histogram Descriptor (LGHD) which adopts multi-scale and multi-oriented Log-Gabor filters to extract feature descriptors from multi-spectrum image pairs, while Shen *et al.* [34] exploit the structure variation existing in multi-modal image sets. In [8], Chiu *et al.* propose cross-modal stereo for improving the accuracy of Microsoft Kinect [42] by combining the three channels of red, green, and blue optimally to mimic the infrared image. A very recent work [43] applies a deep neural network for solving the challenging problem of cross-spectral stereo matching using the rectified near infrared and RGB images, where a novel material-aware loss function is proposed specifically for applications in vehicle vision. None of the above works takes the ToF amplitude as the alternative modality nor matches correspondence under weakly calibrated stereos. Moreover, our method estimates the flow by exploiting the depth image obtained by the ToF sensor while the other works do not take it into account.

ToF depth image refinement. There exist a notable number of works on mitigating errors of continuous-wave ToF depth images. Early works, such as [13, 11, 10, 30], often adopt simplified assumptions such as two-path formulation of MPI, leading to closed-form solutions or costly optimization. Another stream of works focus on the acquisition side, for example using signals in the GHz band instead of the MHz band to mitigate MPI in diffusive environment [15, 23], or exploiting epipolar geometry of light paths [3] at the expense of sequential multiple captures. These methods can produce physically accurate results but are not yet ready for the markets. Closely related to our methods are the recent works based on deep learning which utilizes physically accurate synthetic data. In [27] an auto-encoder (U-Net) is used to learn the MPI corrected depth directly, while [37] starts instead from raw correlation measurements aiming for an end-to-end ToF imaging pipeline. Guo *et al.* [14] propose deep learning methods that tackle artifacts from multi-

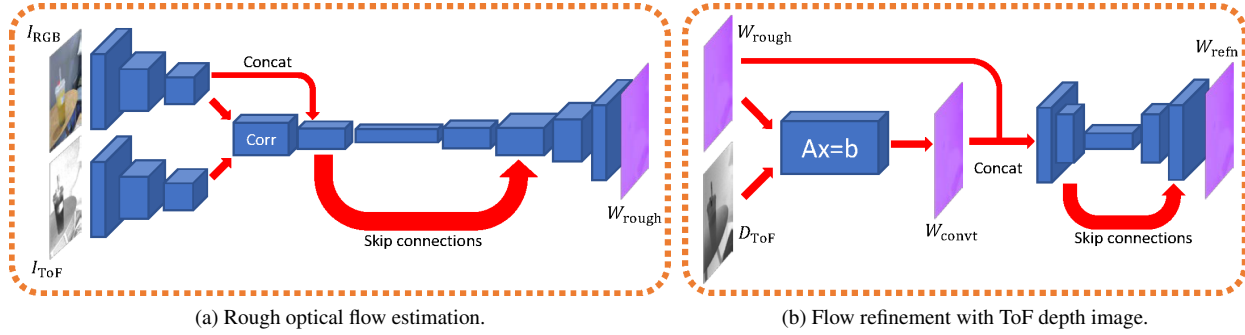


Figure 2: Architecture overview of the cross-modal flow estimation. A rough optical flow is first estimated via FlowNetC. It is then refined by incorporating the depth measurements of the ToF sensor. For flow refinement, we make a depth-flow conversion by estimating the perturbed camera parameters. The converted flow and the rough flow are fed to a small fusion network to obtain the refined flow.

frame fusion as well. All these works are targeted for purely refining the depth images of ToF sensors, so they do not take the corresponding color images into account.

3. Alignment and Refinement

This section illustrates our end-to-end framework for joint alignment and refinement. Particularly, we first estimate the cross-modal dense optical flow for image alignment, then a novel architecture—ToF kernel prediction network (ToF-KPN)—is proposed for depth refinement.

3.1. Cross-Modal Dense Flow Estimation

We solve the alignment problem by estimating a flow (denoted as $W \in \mathbb{R}^{h \times w \times 2}$) where the RGB image (denoted by I_{RGB}) and the ToF amplitude image (denoted by I_{ToF}) are regarded as the first and the second images, respectively. We denote the operation of warping of a one-channel $h \times w$ image I by the flow (a warp field) W as $I_{\text{warped}} = I \circ W$, that is,

$$I_{\text{warped}}(p) = I(m + W_x(p), n + W_y(p)), \quad (1)$$

where $I_{\text{warped}}(p)$ denotes the $p = (m, n)$ -th pixel of image I , similarly for $I(p)$; and $W_x, W_y \in \mathbb{R}^{h \times w}$ are the x - and y - components of the estimated optical flow. The warping operation as in (1) is differentiable with respect to the warp field [21]. Compared to the classic optical flow estimation approaches, recent approaches via convolutional neural networks (CNNs) not only have strong learning/adaptation power, but are also better at exploiting spatial and non-local information across multiple scales [26, 9]. Therefore, we cast the matching task as the estimation of cross-modal dense optical flow with CNNs. We divide the estimation task into two stages: 1) rough optical flow $W_{\text{rough}} \in \mathbb{R}^{h \times w \times 2}$ estimation, and 2) flow refinement. In the first stage we compute a flow solely based on the I_{RGB} and I_{ToF} , while in the second we make use of the depth image of the ToF sensor to refine the flow details.

To compute the rough flow, we have adopted a representative architecture, *FlowNetC* [9], though more advanced choices, *e.g.*, PWC-Net [38], are also applicable. FlowNetC is an U-Net with skip connections, where the encoder part contains a Siamese tower followed by a correlation layer computing a cost volume. This rough flow estimation module is illustrated in Figure 2a.

In the second stage, we refine the flow by incorporating the depth images obtained by the ToF sensor using a lightweight fusion CNN. Particularly, we first warp the depth image from the perspective of the ToF camera, denoted by D_{ToF} , to the perspective of the RGB camera, D_{RGB} , *i.e.*, $D_{\text{RGB}} = D_{\text{ToF}} \circ W_{\text{rough}}$. For the weakly-calibrated module, we can readily estimate a new set of camera parameters $\{t_x^*, t_y^*, c_x^*, c_y^*\}$ between the ToF amplitude image (after initial rectification) and the RGB image by solving the following least-square problem,²

$$\{t_x^*, t_y^*, c_x^*, c_y^*\} = \arg \min_{t_x, t_y, c_x, c_y} \sum_p \left\| W_{\text{rough}}(p) - \begin{pmatrix} \frac{t_x}{D_{\text{RGB}}(p)} + c_x \\ \frac{t_y}{D_{\text{RGB}}(p)} + c_y \end{pmatrix} \right\|^2. \quad (2)$$

Solving this problem is equivalent to solving a linear system, which is differentiable. Hence, it is embedded as a component in our refinement network. Then we can *convert* D_{RGB} to another estimated flow, W_{convnt} (subscript *convnt* denotes it is converted from the depth image), given by

$$W_{\text{convnt}} = \begin{pmatrix} \frac{t_x^*}{D_{\text{RGB}}} + c_x^* \\ \frac{t_y^*}{D_{\text{RGB}}} + c_y^* \end{pmatrix}. \quad (3)$$

Finally we concatenate W_{rough} and W_{convnt} and feed them into a lightweight fusion U-Net, which outputs the refined flow W_{refin} . The architecture of this fusion CNN is illustrated in Figure 2b. Having computed the refined flow

²A detailed derivation of this formulation is presented in the supplementary material.

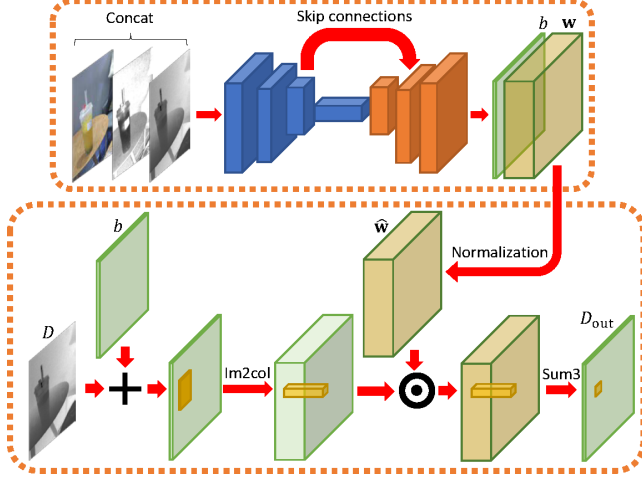


Figure 3: Architecture overview of the depth refinement using the proposed ToF kernel prediction network (ToF-KPN). Here “Im2col” rearranges each patch along the channel dimension while “Sum3” sums along the channel dimension.

W_{refn} , it is applied on the input depth for later depth refinement, *i.e.*, $D_{\text{ToF}} \circ W_{\text{refn}}$. For convenience, we simply use D to denote the final warped depth, $D_{\text{ToF}} \circ W_{\text{refn}}$, in the rest of the paper.

3.2. Refinement via ToF Kernel Prediction Network

It is well-known that the ToF depth measurements suffer from error such as the MPI, the “flying pixel” artifact, and also thermal noise [16]. Moreover, the warped depth D does not guarantee to be tightly aligned with the RGB image. Consequently, a post-processing procedure for depth refinement is indispensable. Kernel prediction network (KPN) is a recently proposed model which performs edge-aware adaptive filtering to images in a data-driven manner [6, 29, 39]. Given depth image D , a vanilla (original) KPN uses an U-Net with skip connections to predict for each pixel a kernel operating only on its surrounding patch. Specifically, for a KPN with output kernel size k ($k = 3$ is used in our work),

$$D_{\text{out}}(p) = \mathbf{w}_p^T \cdot \text{patch}(D(p)) + b(p), \quad (4)$$

where D_{out} is the output depth and $D_{\text{out}}(p)$ is its p -th pixel, $\text{patch}(D(p)) \in \mathbb{R}^{k^2}$ denotes the vectorized patch of D centered at pixel p . The pixel-wise kernel $\mathbf{w}_p \in \mathbb{R}^{k^2}$ and the bias $b \in \mathbb{R}^{h \times w}$ are outputs of the KPN. In other words, the KPN output is a 3-D volume of size $h \times w \times (k^2 + 1)$. We will present an improved KPN for ToF depth image refinement, which differs from (4) in two major perspectives.

First, we empirically find that, in the depth refinement task the vanilla KPN inclines to produce kernel \mathbf{w}_p with very small magnitudes. In such cases, (4) degenerates to $D_{\text{out}} \approx b$ and the KPN behaves like an U-Net. To make full use of the filtering of KPN, we normalize the kernel weights

by their sum of absolute values, *i.e.*,

$$\hat{\mathbf{w}}_p(i) = \mathbf{w}_p(i) / \sum_{i=1}^{k^2} |\mathbf{w}_p(i)|, \quad (5)$$

where $\mathbf{w}_p(i)$ is the i -th entry of \mathbf{w}_p .

Secondly, resolving MPI is challenging, since it introduces gross error almost uniformly in large area and can hardly be resolved by filtering. Consequently, we propose to add the bias term $b(p)$ firstly aiming at correcting the MPI, then use the kernel $\hat{\mathbf{w}}_p$ for edge-aware filtering:

$$D_{\text{out}}(p) = \hat{\mathbf{w}}_p^T \cdot \text{patch}([D + b](p)), \quad (6)$$

where $\text{patch}([D + b](p))$ denotes the patch on $D + b$ centered at pixel p . We call our improved KPN as *ToF-KPN* since it is designed for ToF depth image refinement. It takes as inputs the RGB image I_{RGB} , the warped ToF amplitude image $I_{\text{ToF}} \circ W_{\text{refn}}$, and the warped depth D , and outputs the parameters for elementwise filtering on D . Its filtering scheme is illustrated in Figure 3. We have performed extensive ablation studies and will discuss the effects of our modifications in Section 5.2. These simple changes can boost the results over the vanilla KPN by a *significant* margin.

3.3. Loss Functions

In our work, the training data consists of both the synthetic data with perfect ground-truth and the real data. To achieve robustness in both flow estimation and depth refinement, we apply ℓ_1 loss averaged over the image size for training.

Cross-modal optical flow estimation. ℓ_1 -loss across multiple scales is used in this module. Particularly, we denote the network output at scale s by $W_{\Omega}^{(s)}$ and the corresponding ground-truth by $W_{\text{gt}}^{(s)}$, where $\Omega \in \{\text{rough}, \text{refn}\}$. Then given a training sample, its associated loss is

$$L_{\Omega} = \sum_{s,p} \frac{\alpha_s}{N_s} \left\| W_{\Omega}^{(s)}(p) - W_{\text{gt}}^{(s)}(p) \right\|_1. \quad (7)$$

Here both $W_{\Omega}^{(s)}(p)$ and $W_{\text{gt}}^{(s)}(p)$ are \mathbb{R}^2 vectors, N_s denotes the number of pixel of that scale. We use the same weighting factor α_s as that of FlowNetC [9].

Depth refinement. Choosing proper loss functions are crucial for learning correct geometry without MPI and irrelevant textures from the RGB image. ℓ_1 losses on the output depth and its gradients are used in this module. Particularly, given the output depth D_{out} and the corresponding ground-truth depth D_{gt} , its associated loss is

$$L_{\text{depth}} = \frac{1}{N} \sum_p \|D_{\text{out}}(p) - D_{\text{gt}}(p)\|_1 + \lambda \|\nabla D_{\text{out}}(p) - \nabla D_{\text{gt}}(p)\|_1, \quad (8)$$

where N is the number of pixels, the gradient is computed with the discrete Sobel operator [36]. In our experiments,

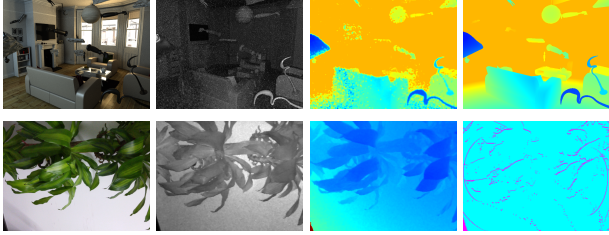


Figure 4: Examples of our datasets. The first row shows an instance of our synthetic dataset, from left to right are the RGB image, the ToF amplitude, the ToF depth image and the ground-truth depth respectively. The second row shows an instance of our real dataset, from left to right are the RGB image, the ToF amplitude, the ToF depth image and the confidence mask, respectively. We use the cyan color to indicate available pixels on the mask.

we set $\lambda = 10$ to let the ToF-KPN learn correct geometry with minimal MPI while preserving details. We summed up the three loss functions, L_{rough} , L_{refn} and L_{depth} for overall end-to-end training.

4. Datasets and Augmentation

4.1. Synthetic Data Generation

Due to the mechanisms of ToF depth sensing, it is uneasy to mitigate the error of ToF depth measurements, *e.g.*, by using a longer exposure time or a higher modulation frequency [15, 23]. As a result, collecting a large amount of ground-truth depth images for ToF cameras is very challenging. Previous works on ToF signal processing [4, 37, 14, 27] have opt for synthesizing data using *transient rendering* from computer graphics [21, 35]. We learn from the experience of these previous works to synthesize our dataset.

Technically, we follow the approach provided by Su *et al.* [37] in synthetic data generation. Additionally, we randomly place diverse kinds of objects with various sizes into the publicly available Blender scenes, totalling 6250 different views for training our framework. We place our objects in a way similar to the FlyingThings3D dataset [28] designed for optical flow estimation. Hence, we call our dataset *ToF-FlyingThings3D*. We also render the corresponding RGB images using *Cycles* in Blender. These together form the {ToF amplitude, RGB, ToF depth} triplets mimicking the outputs of an off-the-shelf ToF RGB-D camera module. The corresponding ground-truth depths are obtained from Blender’s *Z-pass*. Each data sample consists of the ToF amplitude, RGB image, ToF depth image, and the ground-truth depth image, all of size 640×480 and generated at the *same* view point. We randomly set aside 20% of the data instances for testing while the rest are used for training. An example of our synthesized data is shown in the first row of Figure 4. More details about the synthetic dataset can be

found in the supplementary material.

4.2. Real Data Collection

We have also collected a real dataset with several smartphones equipped with both an RGB camera and a Panasonic ToF depth sensor [1]. Each data sample consists of an RGB image, a ToF amplitude image, a depth image, and a binary mask all of size 640×480 . The binary mask indicates the locations of the depth measurements of high confidence. Only depth measurements with high confidence are considered as ground-truth during training. By carefully calibration during the collection of each data sample, we align the depth image, the ToF amplitude image, the binary mask, and the RGB image to the *same* view point by warping. Our real dataset includes 400 scenes collected under different illumination, in which there are 42% of the samples belonging to indoor and the rest belonging to outdoor. These data samples complement the aforementioned synthetic dataset. Again, 20% of the real data are reserved for testing while the rest are used for training. An instance of real data is shown in the second row of Figure 4.

4.3. Data Augmentation via Multi-view Geometry

We are now equipped with both synthetic data (Section 4.1) and real data (Section 4.2) in which every data sample is well aligned. During training for the alignment module and end-to-end training, we generate unaligned training samples from the aligned ones *on the fly*. In this way we enhance the robustness, by making sure that the unaligned ToF and RGB training data cover as much as possible the permissible perturbations of camera parameters.

The perturbation range is determined from the devices used. Specifically, for each sample, we uniformly sample c_x, c_y within $\pm 2.5\%$ of the input image size. For images of size 640×480 , these perturbations can cause the true alignment to deviate from initial calibration by 20 pixels or more. Among all the initial t_x ’s of our ToF RGB-D camera modules, we denote the one with largest absolute value be t'_x , similarly for t'_y . Then we uniformly sample t_x and t_y within $\pm 30\%$ of t'_x and t'_y , respectively. With multi-view geometry, we use the generated $\{t_x, t_y, c_x, c_y\}$ to compute the forward optical flow from the view of the ToF sensor to a virtual RGB camera. With this flow, we warp both the ground-truth depth and the RGB image to the view of the virtual RGB camera, leading to the ground-truth depth and the RGB image for training. We also compute the ground-truth inverse flow regarding the RGB image as the first image and the ToF amplitude image as the second image. This inverse optical flow is used as the supervising signal for training the alignment module. Note that we also update the confidence masks that indicate both the occlusion pixels or invalid depth values due to warping. These masks are used in the optimization (2) and calculation of losses, where

the contributions by the invalid pixels are not considered.

5. Experimentation

5.1. Training Specifications

We have adopted a pre-training strategy for both the alignment and refinement modules. During pre-training, the alignment module is trained in a stage-wise manner, that is, we first trained the FlowNetC only for rough flow estimation, then we included the flow refinement module, both for 20 epochs. In parallel, we pre-trained the ToF-KPN for 40 epochs. We finally stack the alignment and refinement modules together for overall end-to-end fine-tuning for 10 epochs. For all the training, we used the ADAM optimizer [24] with a batch size of 3, where images are randomly cropped into size 384×512 . When training from scratch, the learning rates are set to be 4×10^{-4} , while during overall fine-tuning, the learning rates are set to be 1×10^{-5} . In both cases, we adopt a staircase decay rate of 0.7 to the learning rates after every two epochs. Our implementation is based on the *TensorFlow* framework [2]. All the models are trained on an Nvidia GTX1080 Ti GPU. The results reported in Section 5.2 and Section 5.3 are based on the separately trained alignment and refinement modules, and in Section 5.4 the jointly fine-tuned DEAR framework.

5.2. Ablation Studies

Flow refinement with fusion network. Camera parameter estimation in Figure 2b acts as an intermediate step bringing raw depth information into flow estimation, together with the fusion network it refines the rough optical flow. We herein quantitatively evaluate the flow estimation results before and after adding the optical flow refinement, as well as directly using depth as fusion network’s input, on both the real and synthetic datasets. Average end-point error (AEPE) is used as the metric for objective evaluation.

We first validate the accuracy of our alignment module using both the synthetic data and the real data. Specifically, we apply the method described in Section 4.2 to generate test data from randomly sampled camera parameters. To model different levels of perturbations, we generate 6 groups of data, each containing 1000 {ToF amplitude, RGB, ToF depth} triplets accompanied with the ground-truth flow, where perturbations are sampled from normal distributions with increasing standard deviations. Our experiments found that the flow refinement module consistently leads to improved accuracy (Table 1). We also qualitatively demonstrate the effect of flow refinement in Figure 5.

Depth refinement with ToF-KPN. Recall that for depth refinement, we aim to not only enhance the depth details by exploiting the RGB image, but also reduce the ToF depth sensing error such as the MPI and the sensor noises. This

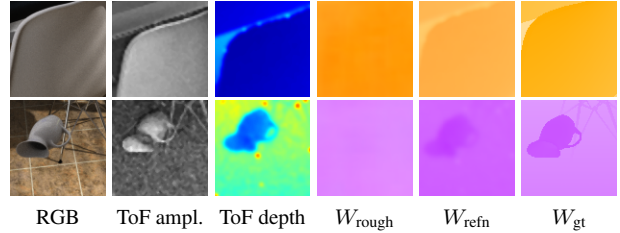


Figure 5: Optical flow refinement incorporating the raw ToF depth measurements greatly refines flow quality.

Standard Deviation σ	2.00		4.00		6.00	
Datasets	Real	Syn.	Real	Syn.	Real	Syn.
Before Refinement	1.28	1.52	1.48	2.10	1.59	2.70
Direct Fusion	1.29	1.55	1.50	2.16	1.63	2.79
After Refinement	1.15	1.34	1.31	1.87	1.36	2.45

Table 1: Average end-point error before and after flow refinement.

experiment shows that superior refinement quality can be achieved with our proposed ToF-KPN architecture. Specifically, we validate the performance of our refinement module, denoted by TOF-KPN, against several networks and hyper-parameter variations, they are:

- U-NET: A U-Net with the same structure as the backbone of our TOF-KPN, but instead it directly regresses the depth. It is supervised using the same loss function (8) as the TOF-KPN.
- NOGRAD: The same with TOF-KPN except is trained using no additional gradient loss as compared to (8) of TOF-KPN.
- NONORM: The same with TOF-KPN except the kernel normalization step (5) is not performed.
- AFTBIAS: The same with TOF-KPN except the bias is added after applying the kernel.
- NONORMAFTBIAS: The same with NONORM except the bias is added after applying the kernel, *i.e.*, the vanilla KPN as in (4).
- NONORMNOBIAS: The same with NONORM except that no bias term is added.

We follow the experimentation approach as in [4, 37] to analyze the model behaviors. Specifically, we sort the pixel-wise errors between the input depth and the ground-truth depth within range of 4 meters in ascending order and divide them into four quantiles, by which the pixels are classified. The first quantile (0 ~ 25%) consists of the pixels that are identified as having low-error, while the second (25 ~ 50%) and the third (50 ~ 75%) quantiles are mid- and high-error pixels. Errors in the last quantile are treated as outliers. On the test split of our synthetic ToF-FlyingThings3D dataset, we compute the overall MAE as well as the MAEs of individual classes, and report them in Table 2.

We first observe that our TOF-KPN provides the best MAE across all error levels. By comparing TOF-KPN

³Adopted from [27], please refer to the text for details.

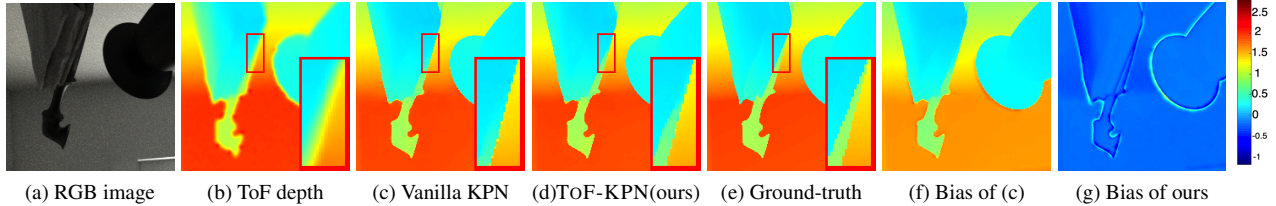


Figure 6: Depth refinement results of an image fragment. The vanilla KPN, *i.e.*, NONORMAFTBIAS in (c), produces dominating bias term and diminishing kernels, which behaves very close to a simple U-Net. As shown in (f), the bias image is very similar to the depth itself. In contrast, our approach produces well-behaved bias image (g).

Model	Mean Absolute Error (MAE) in cm			
	Low Err.	Mid Err.	High Err.	All
U-NET	1.71	1.42	1.52	1.79
NOGRAD	2.19	1.78	1.96	2.43
NONORM	1.60	1.37	1.51	1.73
AFTBIAS	1.52	1.29	1.39	1.62
NONORMAFTBIAS	1.64	1.38	1.52	1.76
NONORMNOBIAS	1.63	1.37	1.50	1.74
TOF-KPN (ours)	1.44	1.19	1.29	1.51

Table 2: Quantitative study of model design for the depth refinement module on the ToF-FlyingThings3D dataset.

Model	Mean Absolute Error (MAE) in cm				No. of Param.
	Low Err.	Mid Err.	High Err.	All	
DEEPTOF ³ [27]	4.31	3.52	4.08	4.69	2.6M
Su <i>et al.</i> [37]	4.58	4.14	4.57	4.90	24.3M
TOF-KPN w/o RGB	2.21	1.93	2.21	2.44	2.6M

Table 3: Quantitative comparison with competitive ToF depth image refinement methods on the ToF-FlyingThings3D dataset. Note that in this comparison no color images are used as inputs.

and NOGRAD, we note that the greatest gain comes from the weighted gradient loss, without which it results in at least 60.9% increase in MAE. With the same loss functions, different model architectures also result in different performances. The worst behaving KPN variant is NONORMAFTBIAS, *i.e.*, the vanilla KPN (4), which neither have kernel normalization nor add the bias first. For this model, we empirically find that the bias quickly dominates while the kernels degenerates to zeros during training. Hence, the network behave very similar to U-NET, as mentioned in Section 3.2. To mitigate this phenomenon and fully utilize the power of KPN, one may either use kernel normalization or applying the bias beforehand, leading to slightly smaller MSE (AFTBIAS and NONORM). However, we furthermore note that for NONORM, the bias term has little contribution since its performance is similar to the one without bias term, *i.e.*, NONORMNOBIAS. Performing both kernel normalization and adding bias in the first place as our TOF-KPN leads to the best performance with a substantial margin of 6.8% over the second best model, AFTBIAS. A subjective comparison between NONORMAFTBIAS and TOF-

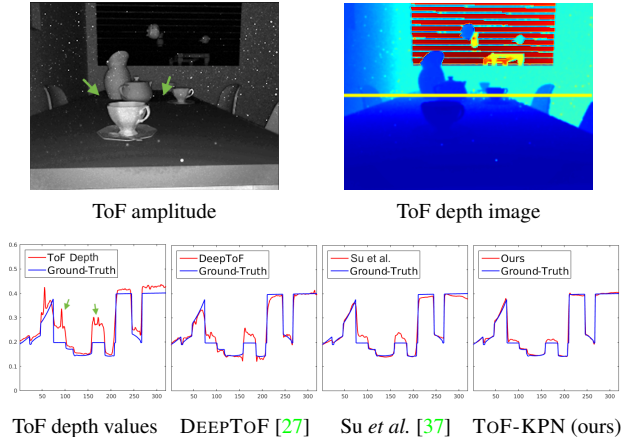


Figure 7: Depth values of different approaches on a scan-line are shown, alongside with the ground-truth. The green arrows indicate the locations that suffer from severe MPI effect.

KPN is also shown in Figure 6, where NONORMAFTBIAS has dominating bias while our TOF-KPN gives more faithful results.

5.3. Comparisons on ToF Depth Image Refinement

We compare our proposed ToF-KPN with the state-of-the-art ToF depth image refinement approaches based on deep neural networks.

Experiments on ToF-FlyingThings3D. We compare our proposal with two other representative approaches. The first one is a deep end-to-end ToF pipeline proposed by Su *et al.* [37] which takes the raw correlation measurements as inputs. In the experiment, we directly use their released model because our ToF-FlyingThings3D dataset is generated using the same scenes and settings as [37]. The second competing method is the DEEPTOF framework based on an auto-encoder which processes off-the-shelf ToF depth images directly [27]. The original DEEPTOF employs a model smaller than ours and it is trained on their real dataset. For fair comparison, we replace their model by our U-NET backbone and train it on our synthetic dataset. We also apply the Euclidean norm as the loss function as indicated in [27]. Note that these two methods takes as inputs the ToF depth image and the ToF amplitude, *i.e.*, they do not use the RGB image. For fairness, we train a version of our TOF-

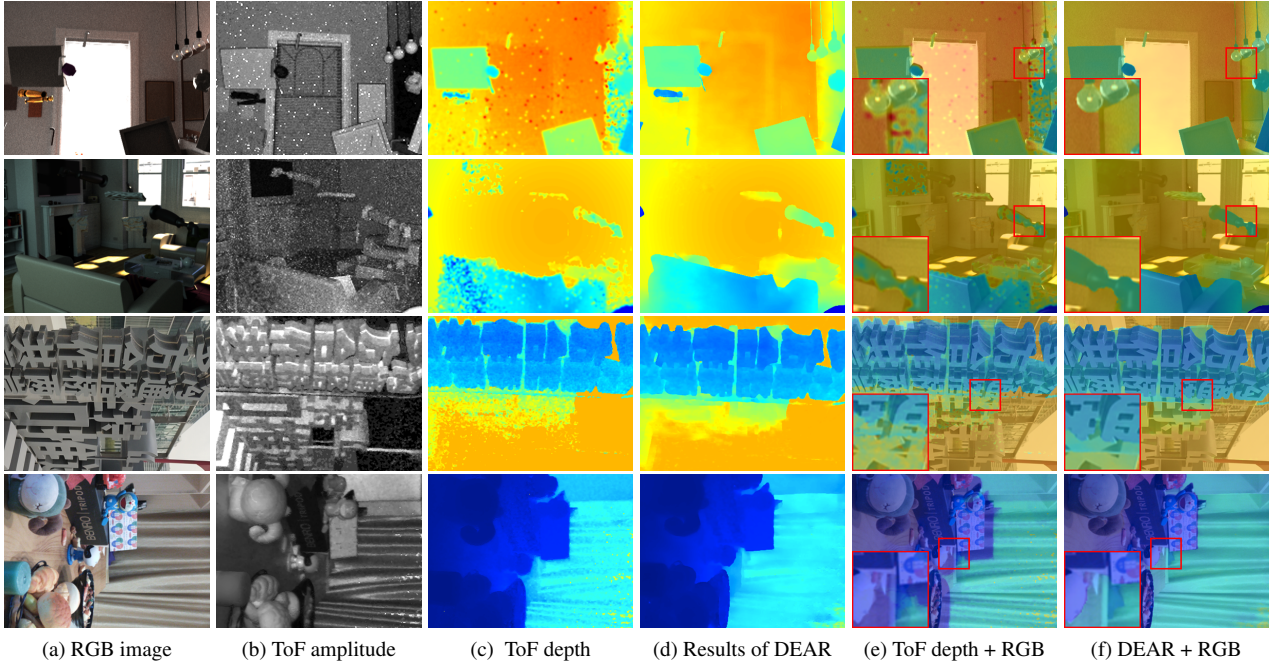


Figure 8: Visual results of our deep end-to-end alignment and refinement framework. In the first two rows we show the results on synthetic data, while last two rows for real data taken by weakly calibrated ToF RGB-D camera modules.

KPN which does not take the RGB image as input.

The objective results, in terms of MAE, are presented in Table 3. We see that our approach, TOF-KPN, achieves the best performance with minimal amount of model parameters. In Figure 7, we demonstrate our capability of reducing MPI by plotting the depth values along a scan-line.

Experiments on FLAT [14]. We compare our refinement with the multi-reflection module (MRM) in FLAT on 120 static test images provided in the FLAT dataset. The MRM uses a KPN architecture but performs filtering on the raw correlation measurements. We fine-tune our model on the static training dataset in FLAT, using the depths obtained from the default de-aliasing algorithm used in libfreenect2 [31] as input. Note that we do not train nor test on the images of objects without complete background environment, which have little MPI error but takes up about half of the entire FLAT dataset. In testing, we achieve an MAE of 0.68 cm while that of MRM is 3.88 cm.

5.4. Evaluation of Deep End-to-End Alignment and Refinement Framework

In this last experiment, we evaluate the overall performance of our deep end-to-end alignment and refinement (DEAR) framework on both the synthetic and real datasets. For this purpose we generate 150 extra misaligned {ToF amplitude, RGB, ToF depth} triplets (accompanied with the ground-truth depth) for testing. They are rendered at novel views defined by randomly sampled camera param-

eters. The visual results are demonstrated in Figure 8, where the first two rows show results of the synthetic data while the rest show results of our real data. To visualize the alignment quality, in the last two columns of Figure 8, we blend the RGB images with the corresponding input depth D_{ToF} and the output depth D_{out} , respectively.

Quantitatively, by assembling the separately trained alignment and refinement modules then applying them to the synthetic data, the average depth MAE reduces from 14.61 cm to 2.90 cm. By jointly fine-tuning the overall DEAR framework, the average MAE further reduces to 2.81 cm. This demonstrates that our proposal is capable of producing high-quality refined depths that are also well aligned with the corresponding RGB images. More results can be found in the appendix.

6. Conclusion

We have proposed DEAR, a deep end-to-end alignment and refinement framework for weakly calibrated ToF RGB-D camera module. Our alignment module estimates cross modal optical flow, integrating information from the ToF depth; our refinement module, based on a specifically designed kernel prediction network, tackles the erroneous ToF depth measurements. To obtain high-quality data for training we have synthesized a dataset, ToF-FlyingThings3D, with tools from computer graphics. Comprehensive experiments have been conducted to demonstrate the effectiveness of our proposal.

Appendices

A. More Details on 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.

A.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 [17] 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

$$\mathbf{P} = \mathbf{K}(\mathbf{I} | \mathbf{0}) = \begin{pmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (9)$$

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

$$[\mathbf{P}(\mathbf{x}; 1)] = \begin{pmatrix} f_x x/z \\ f_y y/z \\ 1 \end{pmatrix} \equiv \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}, \quad (10)$$

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

$$\mathbf{P}' = \mathbf{K}'(\mathbf{I} | \mathbf{t}) = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_t \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (11)$$

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

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{f_x(x+t_x)}{z} + c_x \\ \frac{f_y(y+t_y)}{z} + c_y \end{pmatrix}. \quad (12)$$

With (10) and (12), coordinates of the correspondence between the two images can be related by

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \frac{f_x t_x}{z} + c_x \\ \frac{f_y t_y}{z} + c_y \end{pmatrix}. \quad (13)$$

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 (13) also plays a crucial role in the data augmentation based on multi-view geometry (Section 4.3 in the paper). With (13) 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 (13) 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.

A.2. Detailed Network Architectures

We used FlowNetC for the cross-modal optical flow estimation; therefore, we refer the readers to [9] for its detailed architecture. Since FlowNetC takes two three-channel images as inputs, we repeat the one-channel ToF amplitude for 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 4. Both of these networks are U-Nets with skip connections.

B. 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.

B.1. More on Synthetic Data Generation

Transient rendering [22, 35] 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 [37, 15, 23], 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 [18] 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

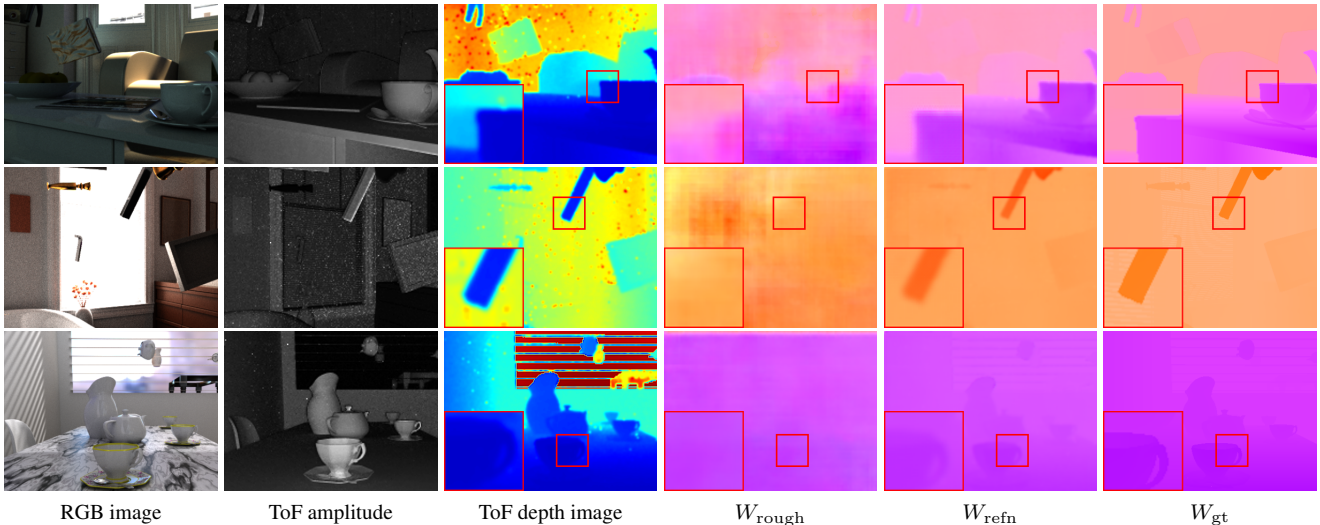


Figure 9: Visual comparisons before and after optical flow refinement. The refinement incorporates ToF depth image via a depth-to-flow conversion, which greatly enhances the accuracy of cross-modal optical flow estimation.

source of the ToF sensor. We also let $L_{\sin}^{(\omega)}$, $L_{\cos}^{(\omega)}$ be the sine and cosine light waves with frequency ω , respectively. Then, the ToF correlation images at pixel p are obtained by:

$$\begin{aligned} C_{\sin}(p, \omega) &= \sum_{t=1}^T I_t(p) \cdot L_{\sin}^{(\omega)}(t), \\ C_{\cos}(p, \omega) &= \sum_{t=1}^T I_t(p) \cdot L_{\cos}^{(\omega)}(t), \end{aligned} \quad (14)$$

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 $C_{\cos}(p, \omega) + iC_{\sin}(p, \omega)$.

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 (10) in Section A.1). Therefore, in our synthetic dataset we also perform standard plane correction [17] to the depth obtained above so as to convert point-to-point distance to point-to-plane distance.

B.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 [20], especially when the Siamese network (*i.e.*, the FlowNetC) is used in our work [9]. 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 $[0, 1]$ (*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 re-

gions. 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:

$$I_{\text{ToF}} = I_{\text{ToF}}^{(\text{raw})} \odot D_{\text{ToF}}^2, \quad (15)$$

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 $[0, 1]$.

C. 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.

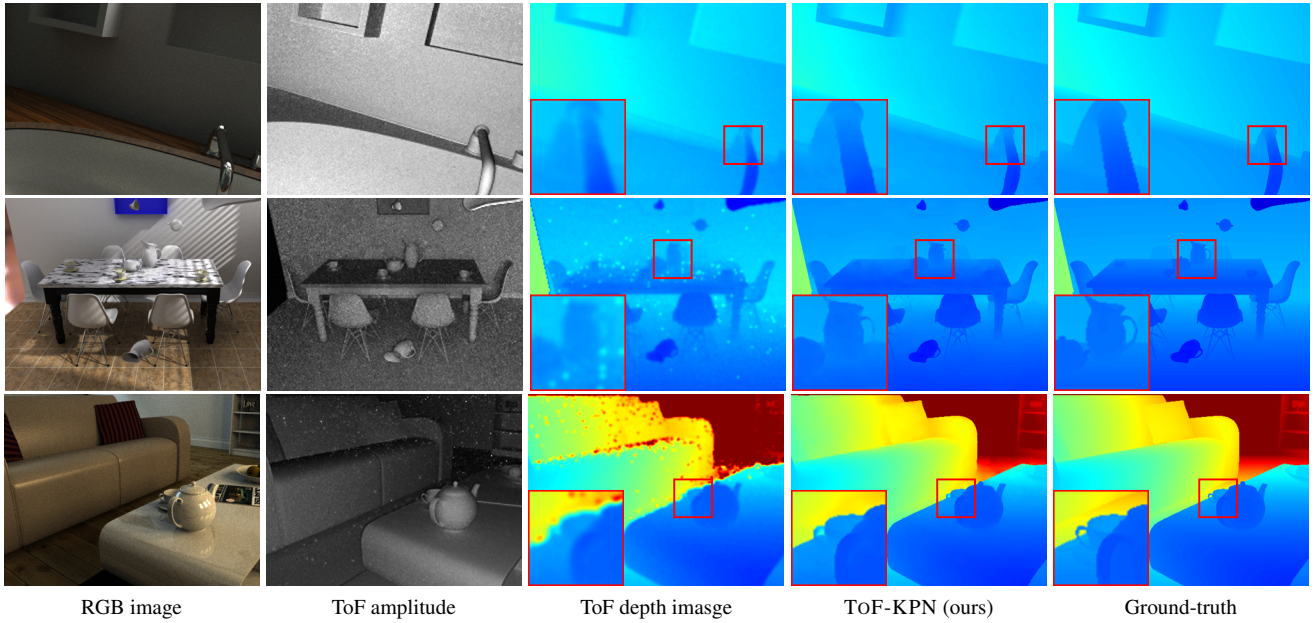


Figure 10: Visual results of the ToF-KPN module for depth image refinement on the ToF-FlyingThings3D dataset.

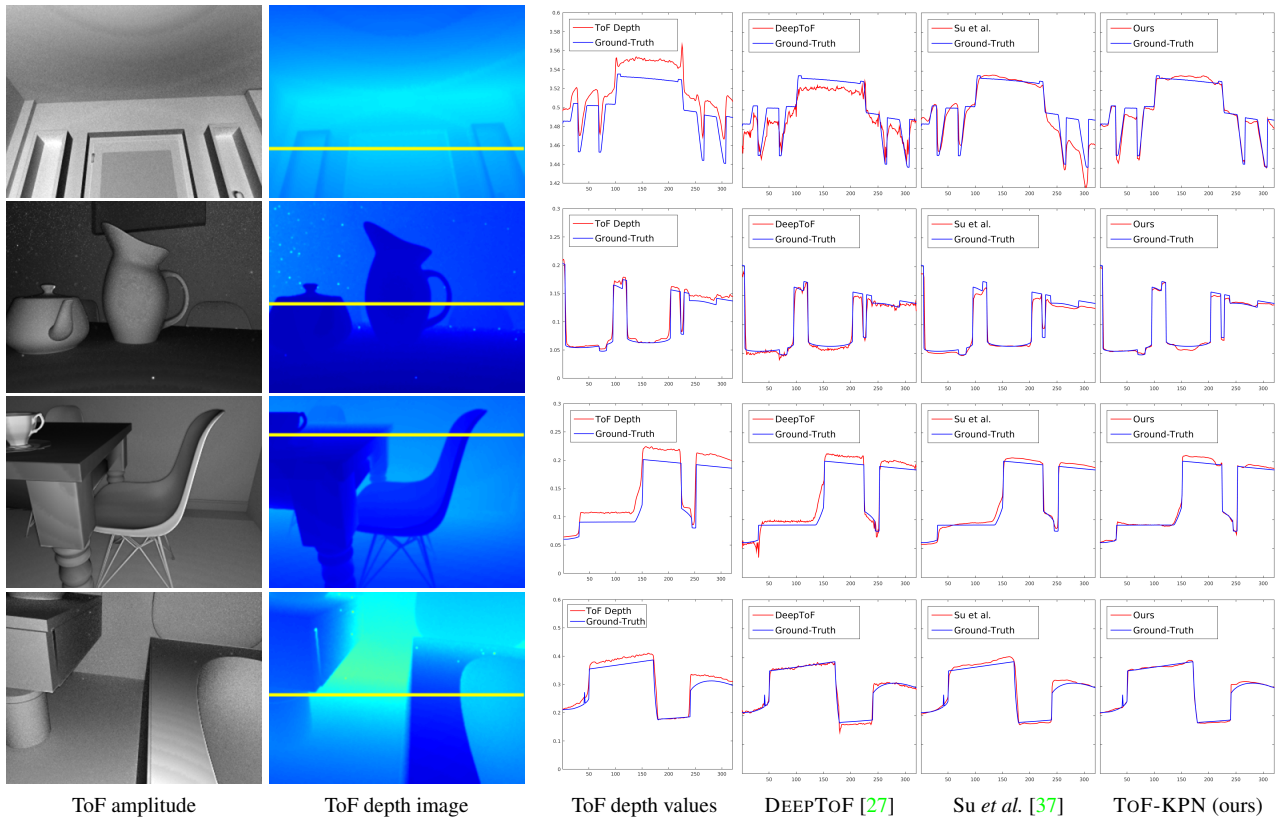


Figure 11: Depth values of different approaches on a scan-line are shown, alongside with the ground-truth. Note that no color images are used in this experiment.

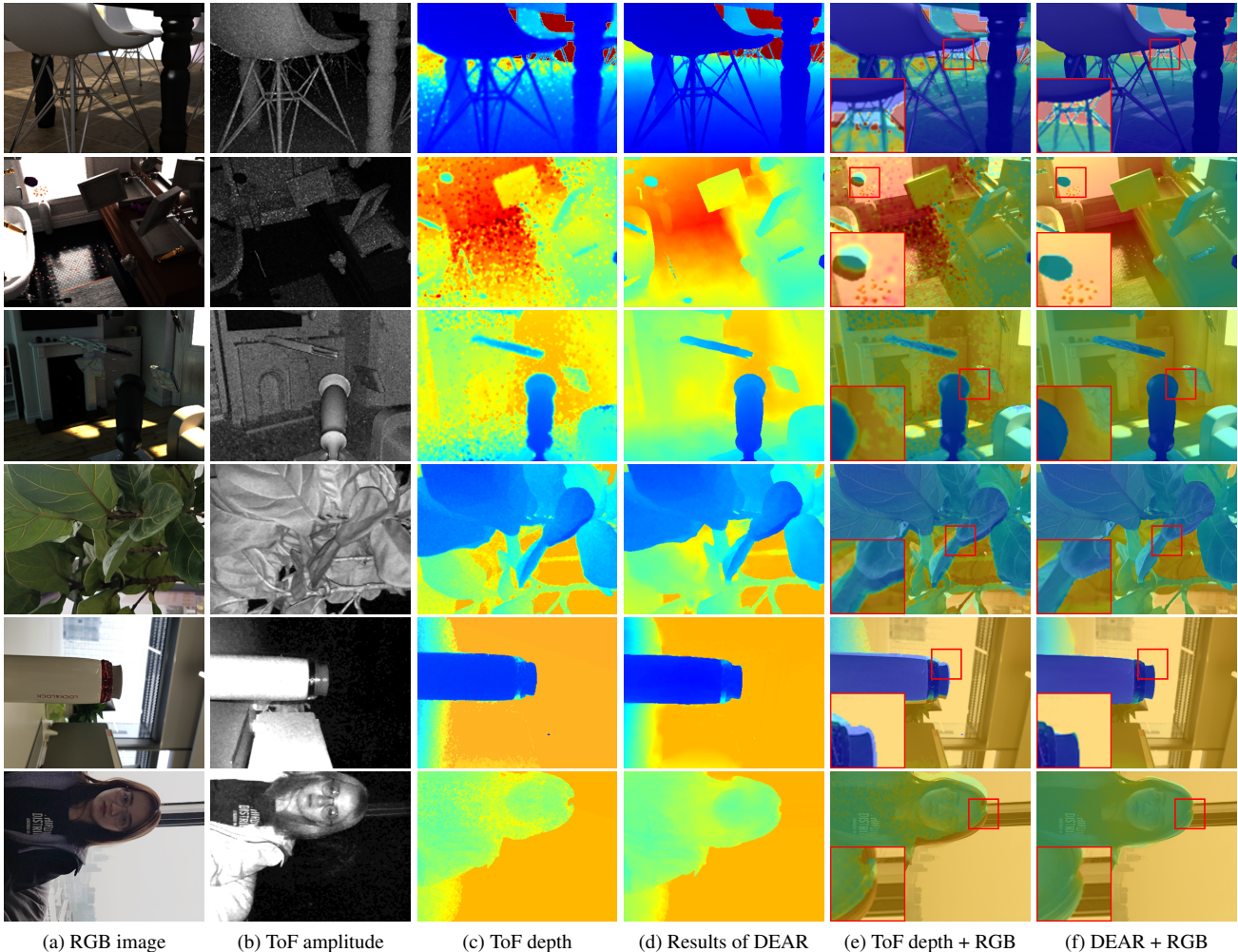


Figure 12: Visual results of our deep end-to-end alignment and refinement framework. In the first three rows we show the results on synthetic data, while last three rows for real data taken by weakly calibrated ToF RGB-D camera modules.

C.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 9. We can see that, the quality of the optical flow is substantially improved by our flow refinement module.

C.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 10. 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 [27] and the method of Su *et al.* [37] in Figure 11. 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 [27] and Su *et al.* [37]).

C.3. More Visual Results of DEAR

More results of our DEAR framework are shown in Figure 12, following the same settings of Section 5.4 of the paper. In Figure 12, 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 visually 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.

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
$W_{\text{refn}}^{(2)}$	3×3	1	128/2	4	4	conv2_1
upconv1	4×4	2	130/128	4	2	conv2_1, $W_{\text{refn}}^{(2)}$
rconv1	3×3	1	256/64	2	2	upconv1, conv1_1
$W_{\text{refn}}^{(1)}$	3×3	1	64/2	2	2	rconv1
upconv0	4×4	2	66/64	2	1	rconv1, $W_{\text{refn}}^{(1)}$
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	$I_{\text{ToF}} \circ W_{\text{refn}},$ $D_{\text{ToF}} \circ W_{\text{refn}}, I_{\text{RGB}}$
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
w, b	3×3	1	64/10	1	1	upconv2

Table 4: 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.

References

- [1] Panasonic 3D sensing solution. <https://b2bsol.panasonic.biz/semi-spt/apl/en/3d-tof/>. 5
- [2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016. 6
- [3] Supreeth Achar, Joseph R Bartels, William L Whittaker, Kiriakos N Kutulakos, and Srinivasa G Narasimhan. Epipolar time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(4):37, 2017. 2
- [4] Amit Adam, Christoph Dann, Omer Yair, Shai Mazor, and Sebastian Nowozin. Bayesian time-of-flight for realtime shape, illumination and albedo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):851–864, 2017. 5, 6
- [5] Cristhian A Aguilera, Angel D Sappa, and Ricardo Toledo. LGHD: A feature descriptor for matching across non-linear intensity variations. In *IEEE International Conference on Image Processing (ICIP)*, pages 178–181, 2015. 2
- [6] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony Derosé, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics (TOG)*, 36(4):97–1, 2017. 4
- [7] Brent Cardani. Optical image stabilization for digital cameras. *IEEE Control Systems Magazine*, 26(2):21–22, 2006. 1
- [8] Wei-Chen Chiu, Ulf Blanke, and Mario Fritz. Improving the kinect by cross-modal stereo. In *British Machine Vision Conference*, volume 1, page 3, 2011. 2
- [9] 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. 3, 4, 9, 10
- [10] Daniel Freedman, Yoni Smolin, Eyal Krupka, Ido Leichter, and Mirko Schmidt. SRA: Fast removal of general multipath for tof sensors. In *European Conference on Computer Vision (ECCV)*, pages 234–249. Springer, 2014. 2
- [11] Stefan Fuchs. Multipath interference compensation in time-of-flight camera images. In *20th International Conference on Pattern Recognition*, pages 3583–3586. IEEE, 2010. 2
- [12] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017. 1
- [13] John P Godbaz, Michael J Cree, and Adrian A Dorrington. Closed-form inverses for the mixed pixel/multipath interference problem in amcw lidar. In *Computational Imaging X*, volume 8296, page 829618, 2012. 2
- [14] Qi Guo, Iuri Frosio, Orazio Gallo, Todd Zickler, and Jan Kautz. Tackling 3D ToF artifacts through learning and the flat dataset. In *European Conference on Computer Vision (ECCV)*, pages 381–396. Springer, 2018. 2, 5, 8
- [15] 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, 5, 9
- [16] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Patrice Horaud. *Time-of-flight cameras: Principles, methods and applications*. Springer Science & Business Media, 2012. 1, 2, 4
- [17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 9, 10

- [18] 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. [9](#)
- [19] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. [1](#)
- [20] 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. [10](#)
- [21] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in neural information processing systems (NIPS)*, pages 2017–2025, 2015. [3](#), [5](#)
- [22] 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. [9](#)
- [23] Achuta Kadambi and Ramesh Raskar. Rethinking machine vision time of flight with GHz heterodyning. *IEEE Access*, 5:26211–26223, 2017. [2](#), [5](#), [9](#)
- [24] Diederick P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. [6](#)
- [25] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 155–163, 2018. [1](#)
- [26] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [3](#)
- [27] 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 interference in time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 36(6):219, 2017. [2](#), [5](#), [7](#), [11](#), [12](#)
- [28] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. [2](#), [5](#)
- [29] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2502–2510, 2018. [4](#)
- [30] Nikhil Naik, Achuta Kadambi, Christoph Rhemann, Shahram Izadi, Ramesh Raskar, and Sing Bing Kang. A light transport model for mitigating multipath interference in time-of-flight sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 73–81, 2015. [2](#)
- [31] OpenKinect. Openkinect/libfreenect, Jun 2019. [8](#)
- [32] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV Workshop on Geometry Meets Deep Learning*, Oct 2017. [1](#)
- [33] Jiahao Pang, Wenxiu Sun, Chengxi Yang, Jimmy Ren, Ruichao Xiao, Jin Zeng, and Liang Lin. Zoom and learn: Generalizing deep stereo matching to novel domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2070–2079, 2018. [1](#)
- [34] Xiaoyong Shen, Li Xu, Qi Zhang, and Jiaya Jia. Multi-modal and multi-spectral registration for natural images. In *European Conference on Computer Vision (ECCV)*, pages 309–324. Springer, 2014. [2](#)
- [35] Adam Smith, James Skorupski, and James Davis. Transient rendering. Technical report, 2008. [5](#), [9](#)
- [36] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *Stanford Artificial Intelligence Laboratory*, pages 271–272, 1968. [5](#)
- [37] 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](#), [5](#), [6](#), [7](#), [9](#), [11](#), [12](#)
- [38] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018. [3](#)
- [39] Thijs Vogels, Fabrice Rousselle, Brian McWilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (TOG)*, 37(4):124, 2018. [4](#)
- [40] Ruichao Xiao, Wenxiu Sun, Jiahao Pang, Qiong Yan, and Jimmy Ren. DSR: Direct self-rectification for uncalibrated dual-lens cameras. In *2018 International Conference on 3D Vision (3DV)*, pages 561–569, 2018. [1](#)
- [41] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 2000. [1](#)
- [42] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10, 2012. [2](#)
- [43] Tiancheng Zhi, Bernardo R Pires, Martial Hebert, and Srinivasa G Narasimhan. Deep material-aware cross-spectral stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1916–1925, 2018. [2](#)