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

Di Qiu\textsuperscript{1,2*} Jiahao Pang\textsuperscript{1*} Wenxiu Sun\textsuperscript{1} Chengxi Yang\textsuperscript{1}

\textsuperscript{1}SenseTime Research \textsuperscript{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.

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\cite{12, 18, 22, 29, 30}, ToF sensors provide higher precision depth values for short-range distance sensing\cite{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.

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\cite{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 \( t_x, t_y \)\cite{7, 37}; 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, \textit{e.g.}, with\cite{38}, 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 \textit{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. However, random perturbations to \( \{c_x, c_y, t_x, t_y\} \) lead to misalignment; so performing online 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 [25].

We call our Deep End-to-end Alignment and Refinement framework DEAR. Our paper is organized as follows. We review related works in Section 2. In Section 3 we elaborate our framework and in Section 4 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. [31] 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 [39] by combining the three channels of red, green, and blue optimally to mimic the infrared image. A very recent work [40] 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, 27], 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, 20], 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 [24] an auto-encoder (U-Net) is used to learn the MPI corrected depth directly, while [34] 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-
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 $I_{warped} = I \circ W$, that is,

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

where $I_{warped}(p)$ 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 [19]. 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 [23, 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 $W_{rough} \in \mathbb{R}^{h \times w \times 2}$ 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 [35], 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., $D_{RGB} = D_{ToF} \circ W_{rough}$. 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: \(^2\)

$$\{t^*_x, t^*_y, c^*_x, c^*_y\} = \arg \min_{t_x, t_y, c_x, c_y} \sum_p \left\| W_{rough}(p) - \left(\frac{t_x}{I_{RGB}(p)} + c_x\right) \right\|^2. \quad (2)$$

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, $W_{convt}$ (subscript $convt$ denotes it is converted from the depth image), given by

$$W_{convt} = \left(\frac{t^*_x}{D_{RGB}} + c^*_x\right). \quad (3)$$

Finally we concatenate $W_{rough}$ and $W_{convt}$ and feed them into a lightweight fusion U-Net, which outputs the refined flow $W_{ref}$. The architecture of this fusion CNN is illustrated in Figure 2b. Having computed the refined flow,
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, 26, 36]. 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),

$$D_{out}(p) = w_p^T \cdot \text{patch}(D(p)) + b(p),$$

where $D_{out}$ is the output depth and $D_{out}(p)$ 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 $w_p \in \mathbb{R}^{k^2}$ 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 kernel $w_p$ with very small magnitudes. In such cases, (4) degenerates to $D_{out} \approx b$ 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.,

$$\tilde{w}_p(i) = w_p(i) / \sum_{i=1}^{k^2} |w_p(i)|,$$

where $w_p(i)$ is the $i$-th entry of $w_p$.

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 $\tilde{w}_p$ for edge-aware filtering:

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

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 $I_{ToF} \circ W_{refn}$, 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 $\ell_1$ loss averaged over the image size for training.

Cross-modal optical flow estimation. $\ell_1$-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 $W_{gt}^{(s)}$, where $\Omega \in \{\text{rough, refn}\}$. Then given a training sample, its associated loss is

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

Here both $W_{\Omega}^{(s)}(p)$ and $W_{gt}^{(s)}(p)$ are $\mathbb{R}^2$ vectors, $N_s$ denotes the number of pixel of that scale. We use the same weighting factor $\alpha_s$ 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. $\ell_1$-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

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

where $N$ is the number of pixels, the gradient is computed with the discrete Sobel operator [33]. In our experiments,
A random set of instances for testing while the rest are used for training. An instance of real data is shown in the second row of 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 image 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, $L_{\text{rough}}, L_{\text{refn}}$ and $L_{\text{depth}}$, 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, 20]. 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, 34, 14, 24] have opt for synthesizing data using transient rendering from computer graphics [19, 32]. We learn from the experience of these previous works to synthesize our dataset.

Technically, we follow the approach provided by Su et al. [34] 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 [25] 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 $640 \times 480$ 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 $640 \times 480$. 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 $640 \times 480$, these perturbations can cause the true alignment to deviate from initial calibration by 20 pixels or more. Among all the initial $t_z$’s of our ToF RGB-D camera modules, we denote the one with largest absolute value be $t_z'$, 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 [21] with a batch size of 3, where images are randomly cropped into size $384 \times 512$. When training from scratch, the learning rates are set to be $4 \times 10^{-4}$, while during overall fine-tuning, the learning rates are set to be $1 \times 10^{-5}$. 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).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$\sigma$</th>
<th>$\sigma$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before Refinement</td>
<td>1.28</td>
<td>1.52</td>
<td>1.48</td>
</tr>
<tr>
<td>Direct Fusion</td>
<td>1.29</td>
<td>1.55</td>
<td>1.50</td>
</tr>
<tr>
<td>After Refinement</td>
<td>1.15</td>
<td>1.34</td>
<td>1.31</td>
</tr>
</tbody>
</table>

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

We follow the experimentation approach as in [4, 34] 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 ($0 \sim 25\%$) consists of the pixels that are identified as having low-error, while the second ($25 \sim 50\%$) and the third ($50 \sim 75\%$) quantiles are mid- and high-error pixels. Errors in the last quantile are treated as outliers.

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

---

3Adopted from [24], please refer to the text for details.
Table 2: Quantitative study of model design for the depth refinement module on the ToF-FlyingThings3D dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Absolute Error (MAE) in cm</th>
<th>No. of Param.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Err.</td>
<td>Mid Err.</td>
</tr>
<tr>
<td>U-NET</td>
<td>1.71</td>
<td>1.42</td>
</tr>
<tr>
<td>NOGRAD</td>
<td>2.19</td>
<td>1.78</td>
</tr>
<tr>
<td>NOB NORM</td>
<td>1.60</td>
<td>1.37</td>
</tr>
<tr>
<td>AFT BIAS</td>
<td>1.52</td>
<td>1.29</td>
</tr>
<tr>
<td>NORM AFT BIAS</td>
<td>1.64</td>
<td>1.38</td>
</tr>
<tr>
<td>NO NORM NO BIAS</td>
<td>1.63</td>
<td>1.37</td>
</tr>
<tr>
<td>ToF-KPN (ours)</td>
<td>1.44</td>
<td>1.19</td>
</tr>
</tbody>
</table>

Table 3: Quantitative comparison with competitive ToF depth image refinement methods on the ToF-FlyingThings3D dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Absolute Error (MAE) in cm</th>
<th>No. of Param.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Err.</td>
<td>Mid Err.</td>
</tr>
<tr>
<td>DEEPToF [24]</td>
<td>4.31</td>
<td>3.52</td>
</tr>
<tr>
<td>Su et al. [34]</td>
<td>4.58</td>
<td>4.14</td>
</tr>
<tr>
<td>ToF-KPN w/o RGB</td>
<td>2.21</td>
<td>1.93</td>
</tr>
</tbody>
</table>

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 NO NORM AFT BIAS, 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 (AFT BIAS and NO NORM). However, we furthermore note that for NO NORM, the bias term has little contribution since its performance is similar to the one without bias term, i.e., NO NORM NO BIAS. 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, AFT BIAS. A subjective comparison between NO NORM AFT BIAS and ToF-KPN is also shown in Figure 6, where NO NORM AFT BIAS 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. [34] 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 [34]. The second competing method is the DEEPToF framework based on an auto-encoder which processes off-the-shelf ToF depth images directly [24]. 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 [24]. 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-
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* [28] 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 0.69 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 parameters. 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 $D_{\text{ToF}}$ and the output depth $D_{\text{out}}$, 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 supplementary material.

**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.
References


[33] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. Stanford Artificial Intelligence Laboratory, pages 271–272, 1968. 4


