Unsupervised Domain Adaptation of Deep Networks for ToF Depth Refinement

Gianluca Agresti, Henrik Schäfer, Piergiorgio Sartor, Yalcin Incesu, and Pietro Zanuttigh, *Member, IEEE*

Abstract—Depth maps acquired with ToF cameras have a limited accuracy due to the high noise level and to the multi-path interference. Deep networks can be used for refining ToF depth, but their training requires real world acquisitions with ground truth, which is complex and expensive to collect. A possible workaround is to train networks on synthetic data, but the domain shift between the real and synthetic data reduces the performances. In this paper, we propose three approaches to perform unsupervised domain adaptation of a depth denoising network from synthetic to real data. These approaches are respectively acting at the input, at the feature and at the output level of the network. The first approach uses domain translation networks to transform labeled synthetic ToF data into a representation closer to real data, that is then used to train the denoiser. The second approach tries to align the network internal features related to synthetic and real data. The third approach uses an adversarial loss, implemented with a discriminator trained to recognize the ground truth statistic, to train the denoiser on unlabeled real data. Experimental results show that the considered approaches are able to outperform other state-of-the-art techniques and achieve superior denoising performances.

Index Terms—Time-of-flight, depth, denoising, deep learning, unsupervised domain adaptation, adversarial learning

1 Introduction

Time-of-flight (ToF) sensors can acquire depth data at interactive frame rates. They measure the depth by illuminating a scene with a periodic, amplitude modulated light signal and by estimating the time taken by the signal to reach the scene points and come back [1]. Even if the performances of ToF sensors have improved a lot since their introduction, they are still affected by critical issues, including the limited spatial resolution, relatively high noise levels (especially on low reflective surfaces) and artifacts on the edges due to the mixed pixel effect. A particularly critical issue is the Multi-Path Interference (MPI), due to the fact that the emitted light can bounce multiple times in the scene before getting back to the sensor, causing a depth overestimation. The MPI distortion is related to the modulation frequency of the ToF signal and multi-frequency ToF (MF-ToF) sensors have been used to extract useful clues to deal with this issue. However, the complete removal of MPI remains an open challenge.

Recently, deep learning techniques have been exploited for this task [2], [3], [4], [5], but even if these approaches have outperformed previous analytical approaches, they as well struggle to completely remove MPI. A challenging issue of deep learning techniques is their limited generalization property. This is specially due to the small amount of training data for supervised learning. Indeed, there are no large public datasets, mostly because while ToF depth data can be easily acquired, obtaining the corresponding ground truth information requires 3D acquisition with highly accurate scanning equipment and the registration of this data with the ToF measurements is a time consuming process.

A possible workaround is to train the deep network with synthetic data, produced by a Time-of-Flight camera simulator. The approach of [2] exploits this idea and obtains impressive performance on synthetic data. But the differences between real world and simulated data reduce the performance in the real domain.

This paper introduces a set of novel unsupervised domain adaptation methods, allowing to more efficiently denoise real world ToF depth data avoiding artifacts caused by the domain shift with respect to the synthetic data. A Coarse-Fine CNN exploiting MF-ToF data derived from [2] is used for depth denoising. The network is trained in a supervised way on synthetic datasets and adapted to the real world setting, exploiting unlabeled real world ToF acquisitions using three different approaches. Each of these applies unsupervised adaptation using adversarial learning at three possible stages of the processing: at the network input; on the internal network features; or at the output.

The first approach works at input level and uses a domain translation network to transform the synthetic ToF data into a representation with statistical properties matching the ones of real data. The translated data are then used to train the denoising architecture.

The second uses an adversarial learning framework to statistically align the internal CNN features between the

- *Gianluca Agresti, Henrik Schäfer, Piergiorgio Sartor, and Yalcin Incesu are with Sony R&D Center Europe Stuttgart Laboratory 1, 70327 Stuttgart, Germany. E-mail: {gianluca.agresti, henrik.schaefer, piergiorgio.sartor, yalcin.incesu}@sony.com.*
- *Pietro Zanuttigh is with the University of Padova, 35122 Padova, Italy. E-mail: zanuttigh@dei.unipd.it.*

Manuscript received 13 January 2021; revised 15 October 2021; accepted 18 October 2021. Date of publication 29 October 2021; date of current version 3 November 2022.
The research for this paper was funded by Sony Europe B.V. Gianluca Agresti, Henrik Schaefer, Piergiorgio Sartor and Yalcin Incesu are Sony Europe B.V. employees. Moreover, part of the work was carried out by Gianluca Agresti during his PhD, funded by Sony Europe B.V.
(Corresponding author: Gianluca Agresti.)
Recommended for acceptance by M. Salzmann.
Digital Object Identifier no. 10.1109/TPAMI.2021.3123843

synthetic and real world domains, forcing the network to have the same behavior on both domains and thus improving performance on real world data.

The third, derived from [6], of which this work is the journal extension, uses an adversarial loss at the output level of the depth refinement network. The network performing ToF data denoising is trained along with a discriminator network capturing the joint statistics of the noisy data and of the denoised maps, thus implementing the adversarial model used for the unsupervised domain adaptation.

This work contains several novel contributions. First of all, it introduces a novel adversarial learning framework for domain adaptation in regression problems and it is the first to apply this technique to the denoising of ToF depth data. While the conference version of the work [6] was focusing only the unsupervised adaptation at the output level (i.e., the third variant of the approach), here we present multiple strategies to perform the adaptation at different stages of the denoising network. Finally a more detailed experimental evaluation and ablation study have been performed.

The paper is organized as follows: Section 2 discusses the related work, then the deep learning model used for ToF denoising is presented in Section 3, while the main contribution of this work, i.e., the unsupervised domain adaptation techniques, are detailed in Section 4. Section 5 introduces the datests used for the training and the evaluation of the considered methods. The experimental results are discussed in Sections 6 and 7 draws the conclusions.

2 Related Works

Noise and distortions affecting ToF sensor data have different typologies depending on the architecture of its pixels and on the working principle of the various devices [1], [7], [8], [9]. Related to the sensor itself, the depth is distorted by different effects. Among these, we have: the thermal noise due to the electronics of the sensor; the lens scattering, due to the light inter-reflection between the lens and the sensor [10]; the so called *harmonic distortion* or *cyclic error*, due to the not ideal sinusoidal shape of the projected light signal; the *photon shot noise* (PSN), due to the finite probability of a photon conversion inside the sensor. Considering instead sources external to the sensor, the main distortion is the multi-path interference (MPI) due to the multiple reflections of the projected light before coming back to the ToF sensor.

ToF Depth Refinement: Model Based Approaches. The zero-mean errors which do not have a systematic impact such as the PSN and the thermal noise can be handled with well known denoising methods as the bilateral filter or total variation techniques [2], [11], [12]. Instead, when the distortion is systematic and scene dependent as in the case of MPI, the correction becomes more challenging. Ideally, the perfect correction of MPI would require the complete description of the scene geometry and the materials composing it. Different methods proposed in literature [13] try to leverage on the structure of ToF data to collect relevant information to correct this distortion but MPI correction remains an open problem.

Some methods [14], [15], [16] try to infer the correct scene geometry starting from single frequency ToF acquisitions. These methods rely on various light reflection models and try to fit the captured ToF data on them to reconstruct the scene depth. They usually assume that the scene is composed only by diffuse reflective objects. However this is a limiting hypothesis in real case scenarios where other types of reflection can arise. These methods are also computationally expensive because they solve the problem iteratively.

Other methods leverage on the *frequency diversity* of the MPI distortion and try to correct it by using multi-frequency ToF data. Many of these methods assume a particular structure for the back-scattered light, e.g., assuming it is composed by few specular rays. Freedman *et al.* [17] imposes a linear minimization problem where the employed solution model is a back-scattering vector composed by few spikes, related to specular reflections. Bhandari *et al.* [18] presented a closed form solution to correct MPI due to K interfering rays using $2K + 1$ modulation frequencies.

Another family of approaches is based on hardware modifications, mainly changing the ToF sensor light projector [19], [20], [21]. These try to separate the direct component of the light, the one informative about the true geometry of the scene from the global one, causing the MPI. Not related to MPI, but of interest for noise reduction are the Hamiltonian codes for ToF acquisitions [22], [23] and the use of spatial patterns for phase disambiguation [24].

ToF Depth Refinement: Data Driven Approaches. Recently data driven approaches for ToF depth refinements have been introduced. The methods from Son *et al.* [25] and Marco *et al.* [3] exploit deep learning and single frequency ToF data. In particular, the latter introduces an encoder-decoder CNN for ToF depth refinement that is initially pre-trained in an unsupervised way on real world depth maps. Then, only the decoder part is trained in a supervised way on a synthetic ToF dataset. In [26], the authors propose a neural network able to refine ToF data even in conditions of low signal, as when the used ToF integration time or illumination power are small.

Some works exploit also the integration of ToF data with other sources of information as the related color images [27], [28] or stereo vision data [29], [30], [31], [32].

The most recent deep learning methods for ToF refinement instead use multi-fequency ToF data [2], [4], [5], [33]. Su *et al.* [4] introduce an end-to-end approach where a CNN takes the raw ToF correlation measurements as input and it outputs the refined depth map. Guo *et al.* [5] use a similar approach, but they focus additionally on the correction of motion blur. Finally, in our previous work [2], we proposed a CNN for MPI correction trained on multi-frequency synthetic data and we quantitatively evaluated the performance when it is tested on real data. In order to avoid the complexity and the high cost of collecting real data with depth ground truth, these methods are trained on synthetic recordings. This simplifies the data collection, however it can implicitly affect the performances on real acquisitions in a negative way, due to the *domain shift* issue, as we evaluated in [2]. By consequence, domain adaptation is required.

Unsupervised Domain Adaptation. Unsupervised Domain Adaptation has not been applied for ToF depth refinement, but it has been widely explored in other fields, starting from image classification. In this field, some methods focus on reducing the first order [34], [35] and the second order [36], [37] statistic discrepancy of the network internal features on the different domains. Also variants of the *batch normalization*

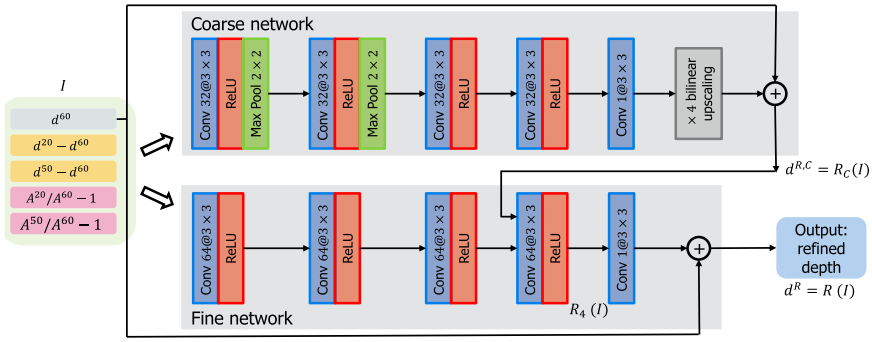

Fig. 1. Representation of the ToF depth refinement network R. The upper part is the coarse branch. It is able to capture a wide receptive field at the cost of a lower output resolution. The lower part is the fine branch. This takes as input the raw ToF data and the output of the coarse branch in order to estimate an accurate depth map of the scene.

layers have been used for the feature statistical alignment [38], [39]. A different approach is proposed by Ganin *et al.* [40]. In this work, the authors reduced the domain shift of a classifier in an unsupervised way by using a *domain classifier* able to distinguish features of the classifier generated from the source domain or from the target domain. The *domain classifier* is used to implement an adversarial loss, as done in GANs [41], to apply the domain adaptation. In [42], [43] a generator network is used to modify some labeled synthetic data to look similar to real data, which are then used to train a classifier. Unsupervised domain adaptation has been widely used also for semantic segmentation [44] and a very relevant work is the cycle-consistent domain adaptation scheme proposed in [45], that translates synthetic color images in real ones, by ensuring the semantic consistency in the domain translation. A similar approach was used for semantic segmentation on depth data in [46].

Regarding regression tasks, in [47] feature alignment from synthetic to real is used to adapt a network trained for multiple tasks as normal, edge and depth estimation from color images. In the field of depth estimation from stereo and mono camera systems, [48] introduces a technique for unsupervised domain adaptation leveraging on classical stereo disparity estimation and confidence measure.

Related to ToF depth refinement, the domain shift issue was analyzed in [2]. We proposed a unsupervised domain adaptation with adversarial learning in [6], that is the first application of domain adaptation in this field. In this paper, we extend the previously proposed network training procedure, presenting three different unsupervised domain adaptation frameworks with the task of improving the refinement performance on real data, without using real ground truth.

3 Depth Refinement With Deep Learning

Even if many analytic approaches for ToF data denoising have been proposed, deep learning approaches proved to be able to outperform classical techniques also on this type of data. In particular, the approach introduced in [2] and further refined in [6] has achieved state-of-the-art performance. We will use this approach as the starting point for this work: the target is to refine ToF depth data, i.e., we aim at removing both MPI and sensor noise. Here, the denoising deep network is fitted into a novel adversarial learning framework in order to apply unsupervised domain adaptation to real data. In this section, we introduce the denoising CNN while the adversarial learning framework for domain adaptation will be described in detail in Section 4.

Before describing the network input, let us recall that the phase offsets carried by the interfering rays causing MPI are frequency dependent and this *frequency diversity* can be used to understand if MPI is acting on the depth acquisition and can give cues for its correction [17], [49], [50]. For this reason, the input of the CNN are the ToF depth and amplitude images acquired at different modulation frequencies (20, 50 and 60 MHz). The raw data acquisitions are pre-processed before being fed to the CNN. First, the multi-frequency depth maps are phase unwrapped to extend the maximum unambiguous range up to 15 m. Then, the amplitude and the depth images are pre-processed in order to extract a representation that highlights relevant information that can be exploited by the deep network to estimate the MPI presence and strength. The raw ToF data are used to produce the CNN input I that is composed by five channels

$$I = \left(d^{60}; d^{20} - d^{60}; d^{50} - d^{60}; \frac{A^{20}}{A^{60}} - 1; \frac{A^{50}}{A^{60}} - 1 \right), \tag{1}$$

where we denoted with d^f and A^f the ToF depth and amplitude maps acquired at f MHz. The first channel is the ToF depth map d^{60} at 60 *MHz* that is the depth map we aim at refining. d^{60} is the most accurate among the collected depth maps since it was acquired at the highest modulation frequency and depth noise is inversely proportional to the modulation frequency [51]. The last four channels are a combination of the ToF depth and amplitude maps acquired at different modulation frequencies and are informative about the MPI distortion. As presented in [2], these features tend to zero in absence of MPI, instead they will assume a different value in case of MPI due to the frequency diversity of this phenomenon. Please, notice that if the network is trained for a given set of modulation frequencies, then it will not be able to achieve very good performances on a different set of frequencies. This is due to the fact that MPI distortion is happening differently for each frequency.

The preprocessed ToF data (Eq. (1)) are fed to the refinement CNN (R) introduced in [6]. R has a *Coarse-Fine* architecture enabling a multi-scale processing of the data. As shown by Fig. 1, the coarse part of the network takes I as input and estimates a low resolution version of the refined depth field.

The fine network processes both I and the output of the coarse network to estimate the full resolution refined depth map. The *Coarse-Fine* approach increases the receptive field of the CNN without increasing too much the network complexity or reducing the resolution of the input. A wide receptive field is important because the reflections causing MPI can happen in different areas of the scene.

The coarse network allows to have a wide receptive field, since it applies downsampling with pooling layers after the first and second convolutional layers. It is made of a stack of 4 convolutional layers with 3×3 kernels and 32 filters. Each convolutional layer is followed by a ReLU. The last layer is a single 3×3 convolutional filter. The output of the coarse network is a low resolution estimate of the scene depth (note that, differently from [2], the network directly estimates the depth map and not the MPI corruption). The output of the last layer is finally up-sampled using bilinear interpolation. The up-sampled output $d^{R,C} = R_C(I)$ is a coarse estimate of the refined depth map.

The fine network works at full resolution and allows to obtain an accurate representation of edges and details (the output of this network is denoted with $d^R = R(I)$ and is also the final output of the proposed method). The first 4 convolutional layers have 3×3 kernels, 64 filters, ReLU activation and no pooling. The output layer is a single 3×3 convolutional kernel. In order to exploit the global information, the up-sampled output of the coarse network ($d^{R,C}$) is given in input to the 4th layer of the fine network.

4 Unsupervised Domain Adaptation for Denoising Networks

The training of the depth refinement CNN needs a sufficiently large and diversified dataset. However, it is expensive and time consuming to collect such a labeled dataset composed of real ToF data with ground truth. For this reason, synthetic datasets have been used for the training [2], [3], [4], [5]. Even if synthetic data is generated trying to emulate the characteristics of real world scenes and the real sensor behavior, when the synthetic trained network is tested on real data a degradation of performance can be experienced due to the domain shift issue as shown in [2]. The domain shift issue arises due to the differences between the two sets of data, e.g., usually the synthetic data model just diffuse reflective objects [2], [3], [4], and does not account for the complex reflective properties of real materials. The main contribution of this paper is the analysis of different methods for unsupervised domain adaption of deep learning models for ToF depth refinement to real data. Doing so, we aim at improving the performance of synthetic trained networks on real data, without using real world ground truth information during the training.

In this paper, we introduce three different techniques in order to improve the performances of the depth refinement CNN on real data using a set of labeled synthetic data (i.e., provided with the related depth ground truth), and a set of unlabeled real data (containing just ToF acquisitions). In the considered methods, a small set of labeled real data will be used for validation purposes only.

The main difference among the three proposed domain adaptation methods is where domain adaptation is applied. We name them as *in-DA*, *feat-DA* and *out-DA* since they are respectively applied on the input of the CNN, at the feature level and on the output of the CNN. The remaining of this section will describe in detail the three domain adaptation methods highlighting the differences among them.

4.1 *In-DA*: Adaptation at the Input Level

The first method (*in-DA*) aims at adapting the ToF synthetic acquisitions in order to translate them into a representation with statistical properties more resembling the ones of real world data. To perform domain translation, we introduce two neural networks, namely $T_{s \Rightarrow r}$ and $T_{r \Rightarrow s}$. For the two networks we used the architecture introduced in CycleGAN [52]. Inspired from [45], [52], the key idea of this method is that the translation network $T_{s \Rightarrow r}$ has the task of transforming the synthetic ToF data, i.e., the input of the depth refinement network R, to a representation more resembling real data. Vice versa, $T_{r \Rightarrow s}$ has to apply the reverse transformation by transforming real data into synthetic-like representations. Notice that a key difference with respect to the CyCADA approach [45] is that while it has been developed for semantic segmentation (i.e., a pixel-wise classification problem), here we deal with depth refinement, that is a regression problem. For this reason, we introduced additional constraints in the training to deal with this different scenario.

$T_{s \Rightarrow r}$ and $T_{r \Rightarrow s}$ cannot be trained in a supervised way, because it is nearly impossible to have couples of aligned synthetic and real data representing the same scene with the same properties (same scene structure, object geometry, materials and so on). For this reason, these networks are unsupervisedly trained by means of adversarial learning [41]. In order to implement this idea, two additional discriminator networks are introduced, D_r and D_s. These domain discriminators have the task to respectively understand if the input data are coming from the real or from the synthetic domain. By defining $\mathbb{I}_s$ and $\mathbb{I}_r$ as the synthetic and real sets of input data, the target is that

$$D_s(I) = \begin{cases} 1, & \text{if } I \in \mathbb{I}_s \\ 0, & \text{otherwise} \end{cases}; \quad D_r(I) = \begin{cases} 1, & \text{if } I \in \mathbb{I}_r \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

Both D_r and D_s are composed by a stack of five 4×4 convolutional layers. The first four layers are composed respectively of 16, 32, 64 and 128 filters with stride 2 and each of them is followed by a batch norm layer and a leaky ReLU with 0.2 slope. The fifth, output layer is composed by just a single filter and does not have the activation function. The same network structure will be used for the discriminators employed in *feat-DA* and *out-DA*.

The four above introduced networks, $T_{s \Rightarrow r}$, $T_{r \Rightarrow s}$, D_r and D_s, are trained in an adversarial way. $T_{s \Rightarrow r}$ is trained to produce data following the statistic of real data when its input is synthetic data, instead D_r is trained to recognize data coming from $\mathbb{I}_r$ as belonging to the class 1 and data produced by $T_{s \Rightarrow r}$ as belonging to the class 0. Similarly, $T_{r \Rightarrow s}$ is trained to produce data following the statistic of the synthetic data when its input is a real data. Instead, D_s is trained to recognize data coming from $\mathbb{I}_s$ as belonging to the class 1 and the data produced by $T_{r \Rightarrow s}$ as belonging to the class 0 . More formally, the networks are jointly trained by

minimizing the following loss functions:

$$\min_{D_s} \mathcal{L}_{D_s} = \min_{D_s} \mathop{\mathbb{E}}_{I_s \in \mathbb{I}_s, I_s^f \in \mathbb{I}_s^f} \left[(D_s(I_s) - 1)^2 + D_s(I_s^f)^2 \right] / 2 \tag{3}$$

$$\min_{D_r} \mathcal{L}_{D_r} = \min_{D_r} \mathop{\mathbb{E}}_{I_r \in \mathbb{I}_r, I_r^f \in \mathbb{I}_r^f} \left[(D_r(I_r) - 1)^2 + D_r(I_r^f)^2 \right] / 2 \tag{4}$$

$$\min_{T_{r\Rightarrow s}} \mathcal{L}_{T_{r\Rightarrow s}} = \min_{T_{r\Rightarrow s}} \mathop{\mathbb{E}}_{I_r \in \mathbb{I}_r} (D_s(T_{r\Rightarrow s}(I_r)) - 1)^2 \tag{5}$$

$$\min_{T_{s\Rightarrow r}} \mathcal{L}_{T_{s\Rightarrow r}} = \min_{T_{s\Rightarrow r}} \mathop{\mathbb{E}}_{I_s \in \mathbb{I}_s} (D_r(T_{s\Rightarrow r}(I_s)) - 1)^2, \tag{6}$$

where $\mathbb{I}_s^f$ and $\mathbb{I}_r^f$ are respectively the set of *fake synthetic* and *fake real* data generated by $T_{r\Rightarrow s}$ and $T_{s\Rightarrow r}$. The sets $\mathbb{I}_s^f$ and $\mathbb{I}_r^f$ are built similarly to [52] using a buffer structure: in 50% of the cases a *fake* element generated with the translation network at the current training step is extracted, otherwise a *fake* sample generated in the last 512 training steps is randomly extracted. This way the discriminators are forced to focus on the statistics of *fake* data and not on the current structure of the translator networks.

Please note that as also suggested in CycleGAN paper [52], the implementation of the adversarial loss functions above follows the LS-GAN structure proposed in [53], where the standard negative log likelihoods are replaced by least squared losses in order to stabilize the learning process. The LS-GAN structure is also used in *feat-DA* and *out-DA*.

During the training process, $T_{s\Rightarrow r}$ will try to fool D_r by producing data with statistics resembling the ones of real data, in contrast D_r will become better at recognizing *fake* real data, from $T_{s\Rightarrow r}$. In this way, both $T_{s\Rightarrow r}$ and D_r will improve, as proven in [41]. If the training converges it will reach a saddle point in which $T_{s\Rightarrow r}$ is able to produce real-like data. A similar rationale applies to $T_{r\Rightarrow s}$ and D_s.

We added a cycle consistency constraint in the translation process that forces the sequential application of $T_{s\Rightarrow r}$ and $T_{r\Rightarrow s}$ to be closer to the identity operation. This is a desired and reasonable property of the domain translation process ensuring that no artifacts are introduced in the domain translation operation. The cycle consistency is implemented by minimizing the following loss:

$$\begin{aligned} &\min_{T_{s\Rightarrow r}, T_{r\Rightarrow s}} \mathcal{L}_{cycle} = \\ &\min_{T_{s\Rightarrow r}, T_{r\Rightarrow s}} \mathop{\mathbb{E}}_{I_r \in \mathbb{I}_r, I_s \in \mathbb{I}_s} \big[|T_{r\Rightarrow s}(T_{s\Rightarrow r}(I_s)) - I_s| + \dots \\ &\qquad \dots + |T_{s\Rightarrow r}(T_{r\Rightarrow s}(I_r)) - I_r| \big]. \end{aligned} \tag{7}$$

By jointly minimizing Eqs. (3), (4), (5), (6) and (7), it is possible to train $T_{r\Rightarrow s}$ and $T_{s\Rightarrow r}$ until they produce data which respectively follow the synthetic and real statistics as captured by the domain discriminators D_s and D_r. However, this does not guarantee that the original and the newly translated data are representative of the same geometric scene. For this reason, we added one supplementary loss term in order to enforce a *geometric consistency* in the translation process. This can be achieved by forcing $T_{s\Rightarrow r}$ to produce *fake* real data which have the same depth ground truth as their synthetic counterpart. $T_{s\Rightarrow r}$ can be trained in a way such that the output of the synthetic trained denoising network R^* is the depth ground truth corresponding to the synthetic data, when the *fake* real data generated by $T_{s\Rightarrow r}$ is given to it as input. Formally, we introduce an additional loss term to be minimized, enforcing this constraint:

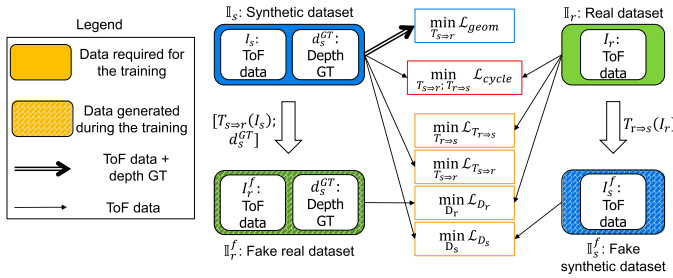

Fig. 2. Flux of data during the training steps of the translation networks $T_{s\Rightarrow r}$ and $T_{s\Rightarrow r}$ used inside the *in-DA* procedure.

$$\min_{T_{s\Rightarrow r}} \mathcal{L}_{geom} = \min_{T_{s\Rightarrow r}} \mathop{\mathbb{E}}_{(I_s, d_s^{GT}) \in \mathbb{I}_s} \left[|R^*(T_{s\Rightarrow r}(I_s)) - d_s^{GT}| \right]. \tag{8}$$

The system is trained by minimizing all the six losses introduced in this section, i.e., Eqs. (3), (4), (5), (6), (7) and (8). Fig. 2 shows how the different data sources are used during the training. The losses need to be opportunely weighted in order to ensure the convergence of the training, see Section 6.1 for the details of the parameters settings.

Regarding the synthetic depth refinement network R^*, used to implement $\mathcal{L}_{geom}$, it is pre-trained from scratch on labeled synthetic dataset by minimizing the loss function of Eq. (9) as also done in [6]:

$$\begin{aligned} &\min_R \mathcal{L}_{R,d} = \\ &\min_R E_{(I_s, d_s^{GT}) \in \mathbb{I}_s} [|R(I_s) - d_s^{GT}| + |R_C(I_s) - d_s^{GT}|]. \end{aligned} \tag{9}$$

The weights of R^* are not updated during the training of the translation networks.

When the four deep networks ($T_{s\Rightarrow r}$, $T_{r\Rightarrow s}$, D_r and D_s) reach the training convergence, in the second phase of the domain adaptation *in-DA*, $T_{s\Rightarrow r}$ can be used to translate all the labeled synthetic data contained in $\mathbb{I}_s$ thus obtaining a new dataset $\bar{\mathbb{I}}_r^f$ (with a little abuse of notation $\bar{\mathbb{I}}_r^f = T_{s\Rightarrow r}(\mathbb{I}_s)$). Please note that differently from the set of data $\mathbb{I}_r^f$ used in Eq. (4), that is dynamically updated at each training step, $\bar{\mathbb{I}}_r^f$ is a static dataset generated when $T_{s\Rightarrow r}$ is already trained and not updated anymore.

Since the translation is geometrically consistent (as forced by Eq. (8)), the ToF data contained in $\bar{\mathbb{I}}_r^f$ and $\mathbb{I}_s$ share the same depth ground truth. Thus, as shown in Fig. 3, the ToF depth refinement network CNN R^* can be fine tuned in a supervised way on the *fake* real data contained in $\bar{\mathbb{I}}_r^f$, using the loss function

$$\begin{aligned} &\min_R \mathcal{L}_{R,d_r^f} = \\ &\min_R E_{(I_r^f, d_s^{GT}) \in \bar{\mathbb{I}}_r^f} [|R(I_r^f) - d_s^{GT}| + |R_C(I_r^f) - d_s^{GT}|]. \end{aligned} \tag{10}$$

Since the statistical properties of $\bar{\mathbb{I}}_r^f$ resemble the ones of real data, this will enforce the unsupervised domain adaptation of the depth refinement CNN.

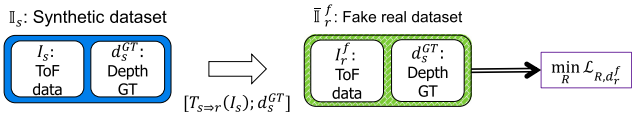

Fig. 3. When the two translation networks are trained, $T_{s\Rightarrow r}$ is used to translate the whole synthetic training set $\mathbb{I}_s$ in $\mathbb{I}_r^f$. The latter one is used to train R in order to adapt to real data without using real ground truth.

4.2 *Feat-DA*: Adaptation at the Feature Level

The second considered unsupervised domain adaption approach is named *feat-DA*. It aims at statistically aligning the CNN internal features produced from synthetic or real samples. This way, we force the network to have the same behavior on the two domains, thus reducing the domain shift of the network on real and synthetic data.

The CNN feature alignment is achieved through an adversarial loss, implemented by means of the discriminator network D_{feat}. The input of this discriminator are the internal features at a selected layer: in this work, we used the output of the fourth layer of the *Fine* network (denoted with R_4). These features are good candidates because they combine the information coming from the *Coarse* branch with the output of the previous layers in the *Fine* branch (see Fig. 1). The discriminator D_{feat} is made of a stack of five convolutional layers and it has the same structure as the discriminator networks used in *in-DA* (check Section 4.1 for more details). D_{feat} is trained to output 1, if the input are features generated by the R network when its input are synthetic data, and 0 if the features have been generated from a real world input. This can be formulated as

$$D_{feat}(f) = \begin{cases} 1, & \text{if } f = R_4(I_s),\ I_s \in \mathbb{I}_s \\ 0, & \text{if } f = R_4(I_r),\ I_r \in \mathbb{I}_r \end{cases}. \tag{11}$$

D_{feat} is trained by minimizing the following loss:

$$\min_{D_{feat}} \mathcal{L}_{D,feat} = \min_{D_{feat}} \mathop{\mathbb{E}}_{I_s \in \mathbb{I}_s; I'_r \in \mathbb{I}_r} \left[(D_{feat}(R_4(I_s)) - 1)^2 + D_{feat}(R_4(I'_r))^2 \right]/2. \tag{12}$$

Simultaneously, the refinement network R is trained in an unsupervised way on the real dataset by trying to mimic the feature statistical distribution obtained on the synthetic dataset. This is implemented by means of the loss function

$$\min_R \mathcal{L}_{R,D_{feat}} = \min_R \mathop{\mathbb{E}}_{I''_r \in \mathbb{I}_r} \left[(D_{feat}(R_4(I''_r)) - 1)^2 \right]. \tag{13}$$

R is also trained in a supervised way on synthetic data to estimate an error free depth estimation by minimizing the loss $\mathcal{L}_{R,d}$ from Eq. (9). This loss function is introduced in [2] and it is used to train both the fine and the coarse branches of R. As shown in Fig. 4, at each training step the loss functions of Eqs. (12) and (9) are optimized on one batch of samples extracted from the synthetic dataset and one batch of samples extracted from the real dataset. A different real batch is used to minimize Eq. (13). More in detail, we simultaneously optimize

$$\min_R \mathcal{L}_{R,d} + \lambda_{feat} \cdot \mathcal{L}_{R,D_{feat}};\ \min_{D_{feat}} \mathcal{L}_{D,feat}, \tag{14}$$

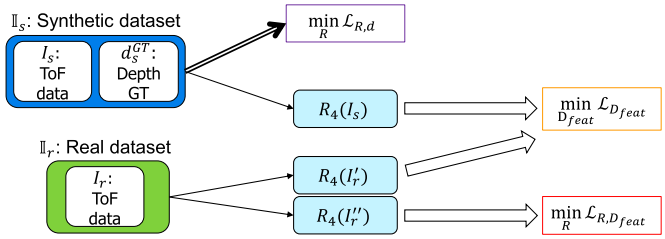

Fig. 4. Flux of data during a training step of *feat-DA* method.

where λ_{feat} is a weighting factor for the adversarial loss (see Section 6.1). As a result of this optimization, D_{feat} will improve in recognizing synthetic and real features and R will try to estimate error free depth maps, starting from raw ToF data and to create internal features which are domain indistinguishable. This procedure forces the network to reduce the domain shift issue and so to improve the refinement performance of R in an unsupervised way to real data.

4.3 *Output-DA*: Adaptation at the Output Level

The third approach for domain adaptation described in this paper works at the output level of R and has been introduced in the conference version of this work [6]. The supervised training of the denoising network R on a synthetic dataset is combined with an adversarial learning scheme, used for unsupervised domain adaptation. The adversarial loss is implemented by means of a discriminator network D_{out}, that is trained to distinguish between the ground truth depth statistics and the statistics of the output of R.

In the next of this section, the discriminator network is described and its use in the domain adaptation framework is explained in detail while the training parameters are detailed in Section 6.1.

4.3.1 Adversarial Discriminator Network

We designed the discriminator to distinguish between denoised depth maps (i.e., the output of R) and the ground truth data. The evaluation will be based on the joint analysis of the noisy ToF depth data and the related noise pattern, computed or from the input depth map and the ground truth or from denoised data. The differences between these situations will be used to drive the adversarial learning process that will force R to produce correctly denoised depth maps that resemble the properties of ground truth data, starting from both, synthetic and from real world data.

The input of the discriminator D_{out} are the noisy ToF depth map d^n and its error map E. For the current implementation, we used the raw ToF depth map at 60 MHz (d^{60}) as d^n but any other acquisition frequency could be used. Please notice that the error map E can be either the difference between the noisy depth map and the ground truth depth $E^{GT} = d^n - d^{GT}$ or the difference between the noisy depth map and the depth refinement network output $E^R = d^n - d^R$. The task of the discriminator is to distinguish between exactly these two cases. More in detail, the discriminator is trained to capture the joint statistics of the couple $I^{D_{out};GT} = (d^n; E^{GT})$, that can be rewritten as

$$I^{D_{out};GT} = (d^n; E^{GT}) = (d^n; d^n - d^{GT}) = (d^{GT} + E^{GT}; E^{GT}), \tag{15}$$

giving output 1, if the input follows this distribution and 0 otherwise. Indeed, we want the discriminator to discard all the data which do not follow the ground truth statistics and are generated by R. To clarify, the output of D_{out} should be 0, if the input is $I^{D_{out};R} = (d^n, d^n - d^R) = (d^n, E^R)$ and 1 if the input is $I^{D_{out};GT}$, i.e., it follows Eq. (15). Please notice that the straightforward approach of feeding the discriminator with d^{GT} as positive example, and d^R (the output of R) as negative example does not lead to very good performance, since it leaves too much freedom to the refinement network to produce data that has ground truth statistics but is far from the geometry of the input depth map. Thus, we employed the proposed two channel features. This choice forces D_{out} to focus on the raw ToF depth map and on how the estimated error is related to it, preventing the output of R to deviate from its input.

The discriminator D_{out} is made of a stack of five convolutional layers and it has the same structure as the discriminator networks used in *in-DA* and *feat-DA* (more details in Section 4.1). The discriminator can be trained by minimizing the following loss function:

$$\min_{D_{out}} \mathcal{L}_{D_{out}} = \min_{D_{out}} \mathop{\mathbb{E}}_{I_s \in \mathbb{I}_s} [D_{out}(I^{D_{out};GT})^2 + (D_{out}(I^{D_{out};R}) - 1)^2]. \tag{16}$$

Recall that for the training of the whole system we use a synthetic dataset, provided with the ground truth depth of the scenes (d_s^{GT}) and an unlabeled real dataset. As in the previous sections, we use the "s" and "r" subscripts to distinguish between synthetic and real data.

In the *true* case (i.e., discriminator output 1), $I^{D_{out};GT}$ requires the ground truth d^{GT} and consequently it can be constructed only on the synthetic dataset. On the other hand, the *fake* data $I^{D_{out};R}$ (discriminator output 0) does not require ground truth information and can be constructed for both real and synthetic datasets.

In order to obtain better performance, we chose to train D_{out} on synthetic data only. The real data will instead be used in the adversarial training procedure for R, described in the next section. Otherwise, D_{out} would always recognize real data as fake, since they were always used as negative examples. This allows to avoid training the discriminator to distinguish between real and synthetic data instead of learning the statistics of $(d^n; E)$ correctly.

On the other hand, the choice of using only synthetic data limits the capability of D_{out} to generalize to real data. One of the main reasons for this is that the amount of noise on real data depends on several factors and can be slightly different from synthetic simulations. In order to better generalize and train a network that is able to adapt to different levels of noise, we apply a novel data augmentation strategy on $I_s^{D_{out};GT}$. Using the ground truth information, we can separate data and noise on the training set and then produce different versions of the scene with slightly increased or decreased amounts of noise. The idea is to use the couple of data in Eq. (17) as *true* (positive) example for D_{out}

$$I_s^{D_{out};GT} = (d_s^{GT} + \hat{E}^{GT}; \hat{E}^{GT}), \tag{17}$$

with $\hat{E}^{GT}$ given by

$$\hat{E}^{GT} = k \cdot (d_s^{60} - d_s^{GT}) = k \cdot E_s^{GT}, \tag{18}$$

where k represents a uniform random variable in the range $[1-\epsilon; 1+\epsilon]$ that acts as a scaling factor for the noise on simulated data. The parameter ϵ has been set to 0.5 for optimal domain adaptation performance by validation. This data augmentation strategy leads to a wider and more general data distribution of which the synthetic statistics is a subset. It forces D_{out} to learn more generic pairs of *(noisy depth; error image)*, preventing it from focusing too much on synthetic ToF noise statistics. Doing so, D_{out} learns to judge how well the error map from R fits the noisy ToF depth.

4.3.2 Unsupervised Domain Adaptation

The discriminator D_{out}, described in the previous section, is used to implement an adversarial loss to perform an unsupervised domain adaptation of the network to real world scenes. The denoising network R is trained both with synthetic data in a supervised way and with unlabeled real data in an unsupervised way. More in detail, the supervised training is performed with the patches extracted from a synthetic dataset $\mathbb{I}_s$ and allows to obtain good performance on synthetic scenes. On the other hand, the photometric differences between simulated and real world scenes reduces the performance of the network on real data [2]. For this reason, an unlabeled real dataset is used to perform domain adaptation on R by using the adversarial loss from the discriminator. The denoiser R is trained by minimizing a loss function composed of two parts:

$$\min_R \mathcal{L}_{out} = \min_R \mathcal{L}_{R,d} + \lambda_{out} \mathcal{L}_{R,D_{out}}, \tag{19}$$

where $\mathcal{L}_{R,d}$ was introduced in Eq. (9) and

$$\mathcal{L}_{R,D_{out}} = \mathop{\mathbb{E}}_{I_r \in \mathbb{I}_r} [\big(D(I_r^{D_{out};R}) - 1\big)^2]. \tag{20}$$

The first term is optimized in a supervised way on synthetic data only. It is modeled as the sum of the l_1 distances between the outputs of R (i.e., the output $d_s^R = R(I_s^R)$ of the fine network and the output $d_s^{R,C} = R_C(I_s^R)$ of the coarse one) and the ground truth depth.

The second part is trained in an unsupervised way on real data, from dataset $\mathbb{I}_r$, without using ground truth information. By minimizing the loss of Eq. (20), we aim at fooling the discriminator by modifying the output of R in order to generate depth maps similar to the ground truth ones. This allows to obtain samples of $I_r^{D_{out};R} = (d_r^n; d_r^n - d_r^R)$ (i.e., couples of noisy depth maps and related error images) similar to the couples constructed from ground truth data $I^{D_{out};GT}$. With the proposed training approach, we can train R to adapt to and denoise real world data without capturing depth ground truth for real scenes.

At each step of the training phase, a batch of real data and a batch of synthetic data are sampled from the two training datasets $\mathbb{I}_s$ and $\mathbb{I}_r$. At first, the synthetic data are used to train the discriminator as mentioned in Section 4.3.1. By following the idea introduced in [42], [52], we exploited a buffer to collect examples of fake data $I_s^{D_{out};R}$, produced by

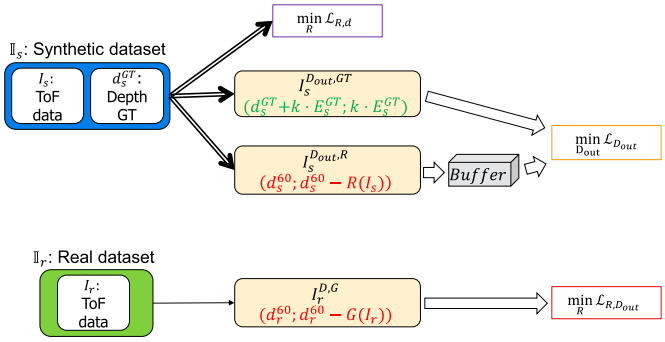

Fig. 5. Flux of data during a training step of *out-DA* method.

R when processing synthetic data in the previous training steps. Two different strategies can be selected with an equal probability. In the first, D_{out} is trained using data produced by R in the current training step. In the second, data stored in the buffer is extracted at random and used as fake examples for training, while the buffer is filled with the data produced by R. As we showed in [6], this approach allows to avoid that D_{out} overfits on the current status of R. Thus, it stabilizes the training process and lets D_{out} focus also on fake data related to previous training steps, since these always have to be classified as fake. In this way, D_{out} captures the statistics of $I^{D_{out};GT}$ better. Simultaneously, R is trained on unlabeled real data by minimizing the loss function of Eq. (20) and on the synthetic data by minimizing the loss in Eq. (9). In this way, it is trained to create depth maps resembling the ground truth ones also when using real data, even if no real world depth ground truth is used.

5 Datasets

This section presents the datasets used for the training, the validation and the testing of the proposed depth refinement methods. Five datasets have been used for this work: S_1 (used for training), S_2 (used for unsupervised domain adaptation), S_3 (used for validation purposes), S_4 and S_5 (both used for testing).

The dataset S_1 is a synthetic dataset generated by using the *Sony ToF Explorer* simulator developed at the Sony R&D Center Europe located in Stuttgart, Germany. This simulator is built starting from a ToF simulator initially developed at the Heidelberg University [54]. It can accurately simulate the different phenomena of real ToF camera acquisitions. This dataset was introduced in [2] and the employed camera parameters resemble the ones of a SoftKinetic DS541 ToF camera. The computer generated scenes used for the dataset have been downloaded from the *Blend Swap* website [55] and they have been adapted for the simulations inside the *Blender* 3D creation suite [56]. The scenes contain diffuse reflective elements like walls, furniture and objects of various shapes and textures placed in different indoor and outdoor environments. The depth of the scenes (from the ToF camera viewpoint) ranges from about 50 cm to 10 m. The dataset contains the 320×240 pxl depth and amplitude maps, acquired at 20, 50 and 60 MHz, simulating the ToF acquisitions on the 40 synthetic scenes together with the corresponding depth ground truth. This dataset is used as the dataset $\mathbb{I}_s$ for the supervised training of the deep networks in Section 6.

The datasets S_2, S_3, S_4 and S_5 instead contain real world acquisitions recorded with a SoftKinetic DS541 ToF camera using the modulation frequencies of 20, 50 and 60 MHz.

The dataset S_2 is an unlabeled real world dataset, composed by scenes acquired in an office with uncontrolled ambient light condition. The recorded scenes contain objects of common use in offices such as tables, chairs and lockers. The dataset contains 97 scenes with the calibrated depth and amplitude images from the ToF camera. The depth of the scenes ranges from 0.5 to about 10 m. No depth ground truth has been acquired for these scenes and for this reason S_2 has been used only for unsupervised domain adaptation playing the role of the $\mathbb{I}_r$ dataset.

The S_3 dataset contains short range scenes with puppets, small boxes, wooden corners and polystyrene cones and spheres. The depth of the scenes from the ToF camera viewpoint ranges from 0.5 to 2 m. This dataset, as S_4 and S_5, also includes depth ground truth acquired with the active stereo system used in [2] and registered on the pixel grid of the ToF camera. It contains 8 scenes and has been used as validation dataset for the proposed deep learning methods.

The S_4 dataset contains 8 real world scenes with ground truth whose subjects are wooden corners and objects of different materials, as plastic and ceramics, placed in a wooden box, where a lot of MPI is present. This dataset is used for testing purposes.

The S_5 dataset contains 8 scenes with ground truth containing boxes of various shapes and dimensions. This dataset is also named as *box dataset* in[6] due to the content of its scenes. It was collected for testing the performance of considered methods on a scenario in which ToF sensors are used in logistics and manufacturing for inspection, handling and dimensioning of box-shaped objects.

The five datasets S_1, S_2, S_3, S_4 and S_5 are available at https://lttm.dei.unipd.it/paper_data/MPI_DA_CNN_J.

6 Results

This section contains the qualitative and quantitative evaluation of the proposed domain adaptation techniques. In the first part, we present the implementation details for *in-DA*, *feat-DA* and *out-DA*. Then, their performances are evaluated and compared with competing state-of-the-art ToF refinement techniques. Next, we evaluate the combined usage of the proposed domain adaptation techniques. Finally, we present a detailed ablation study and we discuss the tuning of the hyper-parameters.

6.1 Implementation Details

Before evaluating the performance of *in-DA*, *feat-DA* and *out-DA*, we will focus on the description of their implementation details and on the used methods hyper-parameters. The networks used in *in-DA*, *feat-DA* and *out-DA* were trained by using the synthetic dataset S_1 (the whole dataset and not just the training split as in the conference version of this paper) for the supervised part of the training and the unlabeled real dataset S_2 for the unsupervised domain adaptation. The labeled real dataset S_3 has been used only for validation purposes to optimize the methods hyper-parameters for the best stability and performances. All the models were trained on a single Nvidia GTX1080TI

GPU using the Adam optimizer [57] in the TensorFlow framework.

In-DA. This method is the most complex from the implementation point of view due to the presence of six loss components and to the two-step training procedure. For its implementation and experimental evaluation, we scaled its losses using the weights: $\lambda_{T_{s\Rightarrow r}} = \lambda_{T_{r\Rightarrow s}} = \lambda_{D_r} = \lambda_{D_s} = 1$, $\lambda_{cycle} = 10$ and $\lambda_{geom} = 100$. The pedix of each weight indicates to which loss it is related. We used a fixed learning rate equal to $5 \cdot 10^{-6}$ using batches of 4 elements for $100k$ steps. Regarding the synthetic depth refinement network R^*, used to implement $\mathcal{L}_{geom}$, it has been pre-trained from scratch on the synthetic dataset S_1 for $150k$ steps using a fixed learning rate equal to $5 \cdot 10^{-6}$ by minimizing Eq. (9).

In the second phase, the dataset generated by translating the synthetic dataset S_1 into real data, $\bar{\mathbb{I}}_r^f$, is used to train R^* on Eq. (10) by optimizing it with Adam for $350k$ steps, using the same learning rate as for the other losses.

Among the presented domain adaptation methods, *in-DA* is the one requiring the longest training time. The overall training takes about 15 hours subdivided in the following way: about 84 minutes for the pretraining of R^*; about 640 minutes for the training of the $T_{s\Rightarrow r}$, $T_{r\Rightarrow s}$, D_r and D_s; about 175 minutes for the fine tuning of R^* for its adaptation to real data.

Feat-DA. In order to ensure the training stability of *feat-DA*, we set $\lambda_{feat} = 5 \cdot 10^{-4}$. We used a fixed learning rate equal to $5 \cdot 10^{-6}$ using batches of 4 elements for $140k$ steps. The training using *feat-DA* takes about 118 minutes.

Out-DA. For the training stability and the optimal performance of *out-DA*, we set $\lambda_{out} = 5 \cdot 10^{-4}$ in the loss of Eq. (19). We used a fixed learning rate of $5 \cdot 10^{-6}$, using batches of 4 elements for $160k$ steps. These training hyperparameters have been selected using the real validation set S_3. The training of the denoising network takes about 125 minutes, similarly to the *feat-DA* method.

6.2 *In-DA*, *Feat-DA* and *Out-DA* Evaluation

In the next of this section, the performance of the proposed methods have been evaluated on the test labeled real datasets S_4 and S_5. We compared their performances with other well-known techniques from literature. The methods used for the comparison are the SRA method [17], the DeepToF approach [3], the CNN-based refinement method CF-synth [2], that was proposed in our previous work and the unsupervised domain adaption method DeepCORAL [37]. More in detail, SRA uses a multi-frequency approach based on a linear optimization scheme to correct MPI. DeepToF uses an auto-encoder CNN, trained in a supervised way on synthetic data and in an unsupervised way on real data. CF-synth [2] is similar to the refinement CNN R used in this work, the main difference lays in the fully supervised training process on synthetic data only, without any domain adaptation procedure. Another key difference is that the CNN in [2] aims at estimating the MPI corruption, instead the proposed method outputs directly the denoised depths. DeepCORAL [37] reduces the domain gap between synthetic and real data by reducing the second order statistical discrepancy between internal network features computed from synthetic and real data. This is implemented by minimizing the Frobenius distance between the features covariance matrices generated when the network is fed with data coming from the different domains. DeepCORAL was originally developed for the domain adaptation of classification neural networks, but we adapted it for the considered regression task, ToF depth refinement. We implemented it on top of our depth refinement network R, and we used it on the features produced for each pixel in the fourth layer of the fine branch, $R_4(I)$, as we did for feat-DA.

6.2.1 Evaluation on the S_4 Dataset

As mentioned in Section 5, the dataset S_4 contains 8 scenes composed by small objects placed inside a wooden box. This scenario causes a high amount of light reflections, producing a severe MPI distortion. Fig. 6 shows a qualitative comparison of the proposed domain adaptation methods and compares them with CF-synth [2]. In particular, the first three rows of the figure depict scenes extracted from the S_4 dataset. All the depth refinement methods bring improvements to the standard ToF acquisitions, which are highly corrupted by MPI (the dark red color on the ToF error map highlights the MPI related depth over-estimation), and the considered refinement methods strongly reduce this kind of artifact. However, the synthetic trained CF-synth [2] is not able to properly remove the MPI distortion on the floor of the scenes. Furthermore, the regions close to boundaries show a quite large amount of remaining MPI corruption (e.g., on the sphere in row 2 or on the objects in row 3). This is due to the discrepancy between the synthetic and real domains respectively used for training and testing. The three domain adaptation approaches instead show a uniform behavior leading to a better correction with respect to the aforementioned method. They are able to further reduce the MPI distortion on the floor of the scenes (in all the 3 scenes) and on the boundaries of the objects, as it is possible to notice in the second and third row in Fig. 6. Finally, notice that also the MPI in proximity of the corner between the two walls in rows 1 and 2 is more efficiently reduced by the proposed strategies with respect to CF-synth.

While it is clear that the proposed approaches outperform the competing scheme, from a qualitative viewpoint it is not always easy to judge the differences among them. The three domain adaptation techniques have similar performances: *out-DA* seems to have slightly better correction capabilities on the floor, however, in average the results looks relatively similar.

More accurate conclusions can be drawn from the quantitative evaluation in Table 1. The table compares the proposed domain adaptation methods with other state-of-the-art depth refinement techniques specifically designed for ToF data. The methods used for comparison are SRA [17], DeepToF [3], CF-synth [2] and the the unsupervised domain adaption method DeepCORAL [37]. The Mean Absolute Error (MAE) to the ground truth depth is used as the metric to compare the method performances. As an additional measure for comparison, we used the relative remaining error between the refined depth map and the input depth map captured at the highest modulation frequency. We chose this measure since DeepToF uses just ToF data captured at 20 MHz, instead the other methods employ multi-frequency data whose highest used frequency is equal to 60 MHz in the considered implementations. The usage of relative error accounts for the fact

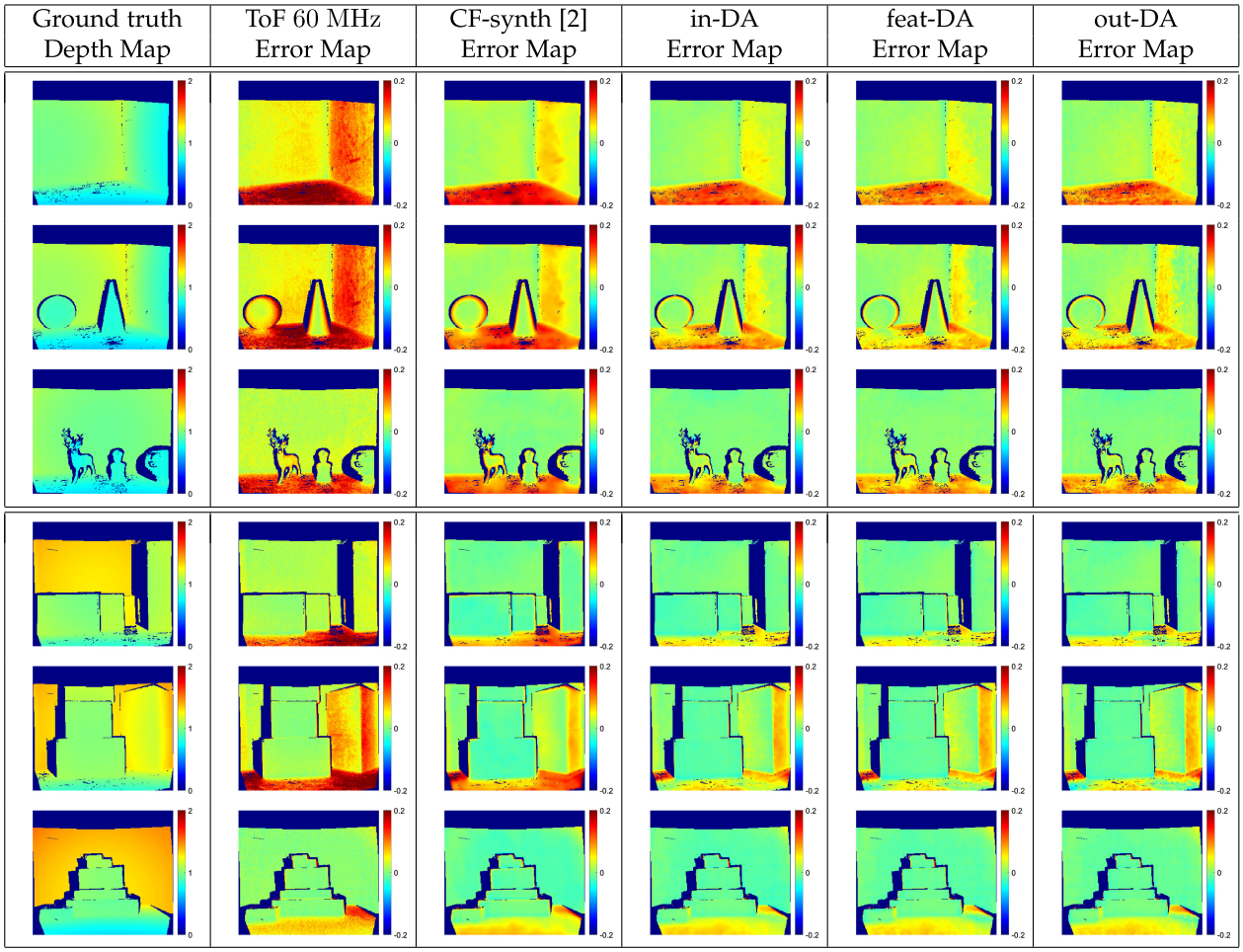

Fig. 6. Qualitative comparison: the images show the ToF depth error on some sample scenes. It compares the considered domain adaptation techniques (*in-DA*, *feat-DA* and *out-DA*) with the standard ToF acquisitions at 60 MHz and the output of the machine learning based depth refinement method CF-synth [2]. The error maps are computed as the estimated depth minus the ground truth depth. The first three rows depict scenes extracted from the dataset S_4, the last three are instead extracted from the dataset S_5. All the values are measured in meters.

that in ToF acquisitions a higher modulation frequency leads to a higher depth accuracy.

The left part of Table 1 shows the performance evaluation on the real dataset S_4. The SRA method is the least performing among the methods, bringing the MAE from 5.43 cm of the ToF acquisition at 60 MHz to 5.11 cm with a relative remaining error of 94.1%. The reason for the minimal correction capabilities of this method on the target dataset can be found in the implementation details of SRA. This method assumes that just a small number of rays are reflected in the scene. This is suitable for specular reflections, however in real scenarios as in the target dataset, there are also diffuse reflections. Diffuse reflections generate an infinite number of reflected rays in all directions and not just a single ray as in the case of ideal specular reflections. Due to this, the basic assumption of the SRA method is broken and the improvements are little under these conditions.

TABLE 1
Quantitative Evaluation on the Depth Refinement Performance of the Considered Domain Adaptation Techniques and of Other State of the Art Methods

Method	S_4 dataset		S_5 dataset	
	MAE [cm]	Relative error	MAE [cm]	Relative error
Input (60 Mhz)	5.43	-	3.62	-
Input (20 Mhz)	7.28	-	5.06	-
SRA [17]	5.11	94.1%	3.37	93.1%
DeepToF [3]	5.13	70.5%*	6.68	132%*
DeepToF [3]+calib.	5.46	75%*	3.36	66.4%*
CF-synth [2]	3.19	58.7%	2.22	60.5%
DeepCORAL [37]	2.39	44.0%	1.77	48.9%
in-DA	2.40	44.2%	1.74	48.1%
feat-DA	2.37	43.6%	**1.64**	**45.3%**
out-DA	**2.31**	**42.5%**	**1.64**	**45.3%**

The relative error is the ratio between the MAE of each method and the MAE on the input at 60 MHz, the highest employed frequency for all approaches, (*) except [3] that is compared with the MAE on the input at 20 MHz since it only uses this frequency.

The data driven method DeepToF [3] can achieve higher performance with respect to SRA. This method can reduce the initial MAE of the standard ToF acquisition at 20 MHz from 7.28 cm to 5.13 cm, with a relative remaining error of 70.5%. However, this approach is in turn outperformed by CF-synth [2] that achieves a relative error of 58.7%, corresponding to a MAE of 3.19 cm. As mentioned before, CF-synth shares a similar CNN with the depth refinement network used in this work, but it does not exploit any domain adaptation scheme. The DeepCORAL domain adaptation scheme implemented on top of the ToF depth refinement network R, outperforms the other methods from literature achieving a relative error of 44.0%, corresponding to a MAE of 2.39 cm.

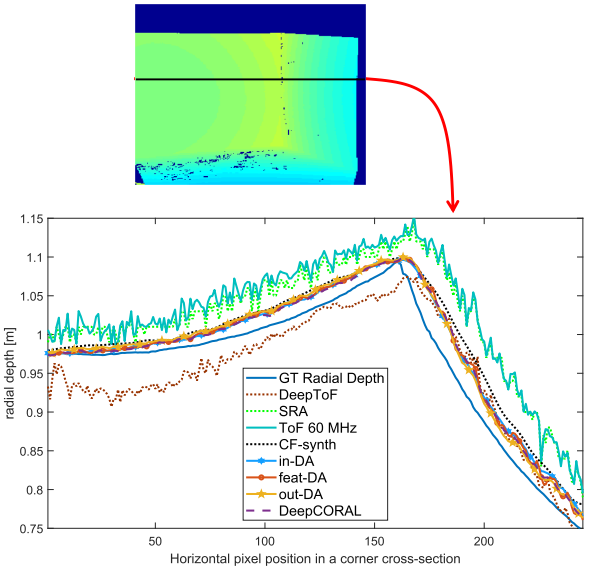

Fig. 7. Comparison on the reconstruction of a corner scene. The plot shows the output of the methods on the cross section highlighted with a red horizontal line on the corner depth map.

From Table 1, it is possible to notice that all the three domain adaption methods proposed in the paper, i.e., *in-DA*, *feat-DA* and *out-DA*, achieve very good performance. *In-DA* performs similarly as DeepCORAL, with a relative error of 44.2%. Comparing domain adaptation techniques working at the feature level of the network, *feat-DA* outperforms DeepCORAL with a relative error of 43.6%. However, *out-DA* is the best performing method on this dataset, being able to reduce the error to a relative remaining error of 42.5%, corresponding to 2.31 cm. It is important to notice the wide margin of improvement (more than 14% of error reduction) with respect to CF-synth [2] that shares a similar denoising architecture trained on synthetic data only. This proves that the adversarial models used in unsupervised training can be employed to improve the method performance on real data, without requiring the time-consuming collection of depth ground truth for real world scenes.

Fig. 7 shows an evaluation of the ToF refinement methods on the reconstruction of a corner scene from S_4. We can confirm that the domain adaptation methods outperform the competitors. In particular, focusing on the differences with CF-synth [2], we can note a slight improvement on the left side of the wall, and a higher margin on the right side, where the MPI distortion is stronger. In this kind of scenario, *out-DA* shows slightly better performance than the other domain adaptation methods. However, the domain adaptation schemes behave similarly good in this situation.

6.2.2 Evaluation on the S_5 Dataset

A similar evaluation has also been carried out on the dataset S_5. As described in Section 5, this dataset contains 8 real scenes composed by box-shaped objects. The fourth, fifth and sixth rows of Fig. 6 show a qualitative comparison among the error maps obtained with the three proposed domain adaptation methods and compare them with the synthetic supervised method CF-synth from [2]. The dataset S_5 appears to be less affected by MPI because the scenes are mainly table tops with no reflections coming from the sides of the scene as in the S_4 dataset. Also on this dataset, CF-synth [2] can reduce the overall depth overestimation due to MPI, but still a lot of depth distortion remains on the floor of the scenes. Differently, the proposed domain adaptation methods can highly reduce the impact of MPI. The three methods behave similarly with a very good reconstruction of the real depth of the scenes. Indeed, the error is almost completely corrected on the floor of the scenes in the forth and fifth rows of Fig. 6, especially by the *feat-DA* and *out-DA* methods, while *in-DA* leaves a bit more MPI on the floor (e.g., see row 4), but the difference is minimal.

The quantitative evaluation on the S_5 dataset is collected in the fourth and fifth columns of Table 1. Another time SRA is the least performing method, with a relative remaining error of 93.1%. DeepToF has issues on this dataset, indeed its basic implementation adds a systematic bias on its output, increasing the overall error instead of reducing it. To solve this issue, we calibrated the method by subtracting the offset that it adds on a flat wall, which is not affected by MPI. Doing so, DeepToF is able to properly reduce the error and obtain a relative error of 66.4%. CF-synth [2] outperforms again these two methods with a relative remaining error of 60.5%. The DeepCORAL domain adaptation method again outperforms the other methods from literature with relative remaining error of 48.9%, but it is less performing than all the proposed domain adaptation schemes.

The performance of *in-DA*, *feat-DA* and *out-DA* are consistent with those from the S_4 dataset. Here, they outperform the synthetic trained CF-synth [2] with at least a 12% of margin. In this case, *feat-DA* and *out-DA* are the best performing methods, both with a MAE of 1.64 cm. The performances of the *in-DA* method is close, having a MAE of 1.74 cm.

6.3 Combination of Domain Adaptation Methods

From the performance evaluation carried out on the S_4 and S_5 datasets, it comes out that the different unsupervised domain adaptation methods are all able to achieve very good performance in denosing real ToF data. A reasonable approach to obtain even better results, could be to apply together these domain adaptation methods.

For example, it is possible to perform domain adaptation using together *feat-DA* and *out-DA*. Another possibility is to first apply *in-DA* at the input level of the depth refinement CNN R and then train R using *feat-DA*, *out-DA* or both of them together using the synthetic data converted to *fake real data* (i.e., translated by $T_{s \Rightarrow r}$), as the set $\mathbb{I}_s$ (that will have a smaller domain shift w.r.t. real data if compared with the original synthetic data). This task is performed using the same translation networks used in *in-DA*, i.e., the translation networks have not been retrained with respect to the description in the previous section.

The performance of the various combinations of domain adaptation techniques are presented in Table 2 together with the optimal balancing parameters computed on the validation set S_3. On the dataset S_4, it is possible to notice that by combining *in-DA* with *feat-DA* or *out-DA* or both of them together, it is possible to obtain some improvements with respect to the single methods applied alone. Differently, on the S_5 dataset, the combinations of the methods are not able to outperform *feat-DA* and *out-DA* applied alone. A possible

TABLE 2
Quantitative Evaluation of Domain Adaptation Method Combination

Methods			Weights		S_4 dataset	S_5 dataset
in-DA	*feat-DA*	*out-DA*	λ_{feat}	λ_{out}	MAE [cm]	MAE [cm]
✓			-	-	2.40	1.74
	✓		$5 \cdot 10^{-4}$	-	2.37	**1.64**
		✓	-	$5 \cdot 10^{-4}$	2.31	**1.64**
✓	✓		$2 \cdot 10^{-5}$	-	**2.24**	1.75
✓		✓	-	$2 \cdot 10^{-4}$	2.26	1.75
	✓	✓	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	2.37	1.70
✓	✓	✓	$5 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	2.27	1.71

explanation of this limited performance gain is related to the fact that *in-DA* can add some elements to the *fake real data*, which can be easily recognized by the discriminators used by the adversarial models in *feat-DA* and *out-DA*. This could make the adversarial domain adaptations less performing. For this reason, the combination of the proposed domain adaption methods for ToF depth refinement needs to be further explored in order to assess if it is worth the introduced extra complexity.

6.4 Ablation Study and Hyper-Parameter Tuning

To conclude this section, we present some analysis to motivate the design choices we took in implementing *in-DA*, *feat-DA* and *out-DA*. As already mentioned, the hyper-parameter tuning and the design choices have been made on the real validation set S_3 (see Section 5).

In-DA. The backbone structure and training of the translation networks $T_{s\Rightarrow r}$ and $T_{r\Rightarrow s}$ are inspired by CycleGAN [52]. However, we made them able to convert labeled synthetic data into the real world data and we used these data to train the depth refinement CNN R in a supervised way. This allows to reduce the domain shift with real data without using real ground truth. To do so, we had to impose a geometrical consistency between the original synthetic data and the newly created *fake real data* in order to make them share the same depth ground truth. We implemented this by adding the geometrical consistency loss $\mathcal{L}_{geom}$ weighted by λ_{geom} to the training. Fig. 8 depicts the effects of adding $\mathcal{L}_{geom}$ on the ToF depth translated by $T_{s\Rightarrow r}$. Here, the effect of $\mathcal{L}_{geom}$ is showed on depth data only to ease the visualization, but recall that the translation is applied also on the other source representations acquired by the ToF sensor and used as input to the network R. The third column of Fig. 8 shows the error map of the depth map translated by $T_{s\Rightarrow r}$ when $\mathcal{L}_{geom}$ is disabled during the training. The fourth column shows the effects on using $\lambda_{geom} = 100$. The value of λ_{geom} was tuned by validation on S_3. Since the nature of synthetic ToF data is already similar to real data (indeed the Sony ToF Explorer is a quite accurate simulator), the task of the translation network $T_{r\Rightarrow s}$ is to slightly modify the synthetic ToF data by adjusting the noise statistic and the MPI distortion to make them closer to the real data. Please note that when $\lambda_{geom} = 0$, the ToF depth is highly distorted and by consequence the synthetic ground truth depth may be less informative about the geometry of the translated data. However, as it is possible to see in the fourth column, thanks to $\mathcal{L}_{geom}$, the geometry of the synthetic scene is preserved and just the intensity of the MPI distortion and the level of noise are modified. These translation effects are

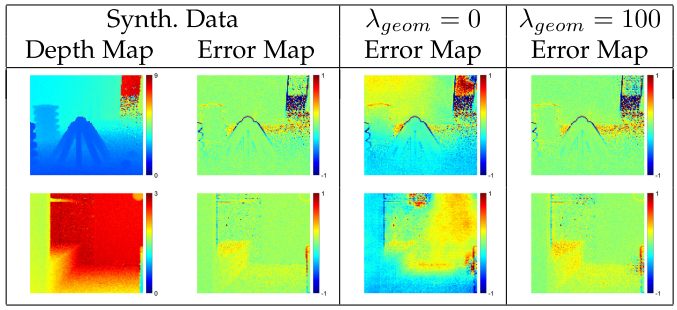

Fig. 8. Qualitative evaluation of the ToF depth translation operation from synthetic to *fake real data* using the network $T_{s\Rightarrow r}$. The scenes are extracted from the synthetic dataset S_1. The values on the color bars are measured in meters.

directly linked to the final performance of the domain adaptation technique. Indeed, after using the synthetic data translated by $T_{s\Rightarrow r}$ to train R, in the case $\lambda_{geom} = 100$ as we did in the proposed *in-DA*, we obtain a MAE of 1.88 cm on the real validation dataset S_3. However, when $\mathcal{L}_{geom}$ is disabled ($\lambda_{geom} = 0$) we obtain a MAE of 8.70 cm on S_3. This shows that the domain adaptation completely fails when the geometrical consistency loss is not used.

Another key component of *in-DA* is the use of cycle consistency in the translation process. This is controlled by the weight λ_{cycle}. Also in this case we notice a degradation of the performances when it is disabled, obtaining in this case a MAE of 1.93 cm on S_3 confirming the importance of using also the cycle consistency in the *in-DA* method.

Feat-DA has a much simpler structure and the only component affecting the domain adaptation performance is the value of the weight λ_{feat}. By validation on S_3, we set $\lambda_{feat} = 2 \cdot 10^{-4}$, achieving a MAE of 1.86 cm. The setting of this parameter has a strong impact on the overall performance. Indeed, by setting its value to $\lambda_{feat} = 5 \cdot 10^{-4}$, the MAE in the validation set S_3 increases to 1.98 cm, and by halving it $\lambda_{feat} = 10^{-4}$, the MAE reaches 1.90 cm.

Out-DA has three main parameters: λ_{out}, that manages the strength of the adversarial loss while training R; ϵ, that manages the amount of the domain data augmentation (see Section 4.3.1 for details); and if enabling or not the buffer of new and old examples of refined depth maps (from R) to train the discriminator network D_{out}. About the optimal value of λ_{out}, the best performances are achieved for $\lambda_{out} = 5 \cdot 10^{-4}$, with a MAE of 1.81 cm. Differently, by setting its value to $\lambda_{out} = 1 \cdot 10^{-3}$, the MAE in the validation set S_3 is 1.96 cm, and by reducing it to $\lambda_{out} = 2 \cdot 10^{-4}$, the MAE is 1.98 cm. Results show that the validation error increases when we disable the domain data augmentation ($\epsilon = 0$, leading to a MAE of 1.99 cm on S_3) or if we disable the usage of the buffer for training D_{out} (the MAE reaches 2.00 cm), motivating our method design choices.

7 Conclusion

In this paper, we tackled the challenging problem of training a deep network for ToF depth data denosing without exploiting real world depth ground truth information. We proposed three novel unsupervised domain adaptation strategies, working respectively on the input data, on the internal feature representation and at the network output

level. The proposed strategies proved to be able to adapt a coarse-fine denoising network, trained on synthetic data, to real world acquisitions, allowing to strongly reduce the overestimation due to the multi-path interference. The comparison with current state-of-the-art approaches showed how the proposed method is able to outperform them and to achieve superior denoising performances.

Further research will be devoted to investigate the combined usage of the proposed strategies. Novel domain adaptation schemes will also be investigated, in particular based on the analysis of the feature space. Finally, the proposed techniques can be applied not only to ToF data, but can be used for any image or depth data denoising task and we will investigate their application in different fields.

ACKNOWLEDGMENTS

We would like to thank the members of the Sony R&D Center Europe Stuttgart Laboratory 1 for the precious comments and insights, in particular Oliver Erdler, Markus Kamm, Tomoo Mitsunaga, and Shigeyuki Baba from Sony Semiconductor Solutions for supporting this work. A special thank to the members of the LTTM lab at the University of Padova for the fruitful discussions.

REFERENCES

[1] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, *Time-of-Flight and Structured Light Depth Cameras*. Cham, Switzerland: Springer, 2016.

[2] G. Agresti and P. Zanuttigh, "Deep learning for multi-path error removal in TOF sensors," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 410–426.

[3] J. Marco *et al.*, "DeepToF: Off-the-shelf real-time correction of multipath interference in time-of-flight imaging," *ACM Trans. Graphics*, vol. 36, no. 6, 2017, Art. no. 219.

[4] S. Su, F. Heide, G. Wetzstein, and W. Heidrich, "Deep end-to-end time-of-flight imaging," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6383–6392.

[5] Q. Guo, I. Frosio, O. Gallo, T. Zickler, and J. Kautz, "Tackling 3D ToF artifacts through learning and the flat dataset," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 381–396.

[6] G. Agresti, H. Schaefer, P. Sartor, and P. Zanuttigh, "Unsupervised domain adaptation for ToF data denoising with adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5584–5593.

[7] D. Lefloch *et al.*, "Technical foundation and calibration methods for time-of-flight cameras," in *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Berlin, Germany: Springer, 2013, pp. 3–24.

[8] M. Lindner, I. Schiller, A. Kolb, and R. Koch, "Time-of-flight sensor calibration for accurate range sensing," *Comput. Vis. Image Understanding*, vol. 114, no. 12, pp. 1318–1328, 2010.

[9] J. Jung, J.-Y. Lee, Y. Jeong, and I. S. Kweon, "Time-of-flight sensor calibration for a color and depth camera pair," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 7, pp. 1501–1513, Jul. 2015.

[10] H. Schäfer, F. Lenzen, and C. S. Garbe, "Model based scattering correction in time-of-flight cameras," *Opt. Exp.*, vol. 22, no. 24, pp. 29835–29846, 2014.

[11] F. Lenzen, H. Schäfer, and C. Garbe, "Denoising time-of-flight data with adaptive total variation," in *Proc. Int. Symp. Vis. Comput.*, 2011, pp. 337–346.

[12] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-Flight Cameras: Principles, Methods and Applications*. London, UK: Springer, 2012.

[13] R. Whyte, L. Streeter, M. J. Cree, and A. A. Dorrington, "Review of methods for resolving multi-path interference in time-of-flight range cameras," in *Proc. IEEE Sensors*, 2014, pp. 629–632.

[14] S. Fuchs, "Multipath interference compensation in time-of-flight camera images," in *Proc. Int. Conf. Pattern Recognit.*, 2010, pp. 3583–3586.

[15] S. Fuchs, M. Suppa, and O. Hellwich, "Compensation for multipath in ToF camera measurements supported by photometric calibration and environment integration," in *Proc. Int. Conf. Comput. Vis. Syst.*, 2013, pp. 31–41.

[16] D. Jiménez, D. Pizarro, M. Mazo, and S. Palazuelos, "Modeling and correction of multipath interference in time of flight cameras," *Image Vis. Comput.*, vol. 32, no. 1, pp. 1–13, 2014.

[17] D. Freedman, Y. Smolin, E. Krupka, I. Leichter, and M. Schmidt, "SRA: Fast removal of general multipath for ToF sensors," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 234–249.

[18] A. Bhandari, M. Feigin, S. Izadi, C. Rhemann, M. Schmidt, and R. Raskar, "Resolving multipath interference in Kinect: An inverse problem approach," in *Proc. IEEE Sensors*, 2014, pp. 614–617.

[19] R. Whyte, L. Streeter, M. J. Cree, and A. A. Dorrington, "Resolving multiple propagation paths in time of flight range cameras using direct and global separation methods," *Opt. Eng.*, vol. 54, no. 11, 2015, Art. no. 113109.

[20] G. Agresti and P. Zanuttigh, "Combination of spatially-modulated ToF and structured light for MPI-free depth estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 355–371.

[21] S. Achar, J. R. Bartels, W. L. Whittaker, K. N. Kutulakos, and S. G. Narasimhan, "Epipolar time-of-flight imaging," *ACM Trans. Graphics*, vol. 36, no. 4, pp. 1–8, 2017.

[22] M. Gupta, A. Velten, S. K. Nayar, and E. Breitbach, "What are optimal coding functions for time-of-flight imaging?," *ACM Trans. Graphics*, vol. 37, no. 2, pp. 1–18, 2018.

[23] F. Gutierrez-Barragan , S. A. Reza, A. Velten, and M. Gupta, "Practical coding function design for time-of-flight imaging," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1566–1574.

[24] T. Kushida, K. Tanaka, T. Aoto, T. Funatomi, and Y. Mukaigawa, "Phase disambiguation using spatio-temporally modulated illumination in depth sensing," *IPSJ Trans. Comput. Vis. Appl.*, vol. 12, pp. 1–13, 2020.

[25] K. Son, M.-Y. Liu, and Y. Taguchi, "Learning to remove multipath distortions in time-of-flight range images for a robotic arm setup," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3390–3397.

[26] Y. Chen, J. Ren, X. Cheng, K. Qian, L. Wang, and J. Gu, "Very power efficient neural time-of-flight," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2020, pp. 2257–2266.

[27] I. Eichhardt, D. Chetverikov, and Z. Janko, "Image-guided ToF depth upsampling: A survey," *Mach. Vis. Appl.*, vol. 28, no. 3–4, pp. 267–282, 2017.

[28] D. Qiu, J. Pang, W. Sun, and C. Yang, "Deep end-to-end alignment and refinement for time-of-flight RGB-D module," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9993–10002.

[29] G. Agresti, L. Minto, G. Marin, and P. Zanuttigh, "Deep learning for confidence information in stereo and ToF data fusion," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2017, pp. 697–705.

[30] G. Agresti, L. Minto, G. Marin, and P. Zanuttigh, "Stereo and ToF data fusion by learning from synthetic data," *Inf. Fusion*, vol. 49, pp. 161–173, 2019.

[31] C. Pu, R. Song, N. Li, and R. B. Fisher, "SDF-GAN: Semi-supervised depth fusion with multi-scale adversarial networks," 2018, *arXiv:1803.06657*.

[32] M. Poggi, G. Agresti, F. Tosi, P. Zanuttigh, and S. Mattoccia, "Confidence estimation for ToF and stereo sensors and its application to depth data fusion," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1411–1421, Feb. 2020.

[33] E. Buratto, A. Simonetto, G. Agresti, H. Schäfer, and P. Zanuttigh, "Deep learning for transient image reconstruction from ToF data," *Sensors*, vol. 21, no. 6, 2021 Art. no. 1962.

[34] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," *Proc. Mach. Learn. Res.*, vol. 37, pp. 97–105, 2015.

[35] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 2208–2217.

[36] P. Morerio, J. Cavazza, and V. Murino, "Minimal-entropy correlation alignment for unsupervised deep domain adaptation," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.

[37] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 443–450.

[38] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "AutoDial: Automatic domain alignment layers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5077–5085.

[39] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci, "Unsupervised domain adaptation using feature-whitening and consensus loss," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9471–9480.
[40] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1180–1189.
[41] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[42] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2242–2251.
[43] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 95–104.
[44] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: A review," *MDPI Technol.*, vol. 8, 2020, Art. no. 35.
[45] J. Hoffman *et al.*, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1989–1998.
[46] J. S. Katrolia, L. Kramer, J. Rambach, B. Mirbach, and D. Stricker, "An adversarial training based framework for depth domain adaptation," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2021, pp. 353–361.
[47] Z. Ren and Y. J. Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 762–771.
[48] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano, "Unsupervised domain adaptation for depth prediction from images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2396–2409, Oct. 2020.
[49] A. Bhandari *et al.*, "Resolving multipath interference in time-of-flight imaging via modulation frequency diversity and sparse regularization," *Opt. Lett.*, vol. 39, no. 6, pp. 1705–1708, 2014.
[50] M. Gupta, S. K. Nayar, M. B. Hullin, and J. Martin, "Phasor imaging: A generalization of correlation-based time-of-flight imaging," *ACM Trans. Graphics*, vol. 34, no. 5, 2015, Art. no. 156.
[51] R. Lange, P. Seitz, A. Biber, and S. C. Lauxtermann, "Demodulation pixels in CCD and CMOS technologies for time-of-flight ranging," in *Sensors Camera Syst. Sci. Ind. Digit. Photogr. Appl.*, vol. 3965, pp. 177–189, 2000.
[52] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251.
[53] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2813–2821.
[54] S. Meister, R. Nair, and D. Kondermann, "Simulation of time-of-flight sensors using global illumination," in *Proc. Vis. Model. Vis.*, 2013, pp. 33–40.
[55] Blender Swap, "Blend swap website," Accessed: Apr. 8, 2020. [Online]. Available: https://blendswap.com/
[56] The Blender Foundation, "Blender website," Accessed: Apr. 8, 2020. [Online]. Available: https://www.blender.org/
[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

Gianluca Agresti received the MSc degree in telecommunication engineering from the University of Padova in 2016 and the PhD degree from the Department of Information Engineering, University of Padova, in 2020. He is currently a senior engineer with the Sony R&D Center Europe Stuttgart Laboratory 1. His research interests include deep learning for ToF sensor data processing and multiple sensor fusion for 3D acquisition.

Henrik Schäfer studied physics from Heidelberg University and received the doctor of science degree in image processing from the Heidelberg Collaboratory in 2014, with his thesis on Image Enhancement and Parameter Estimation for ToF Cameras. He is currently a senior engineer with the Advanced Sensors and Modelling Group, Sony Europe R&D Center Stuttgart Laboratory 1.

Piergiorgio Sartor received the MSc degree in electronic engineering from the University of Padova. He is currently a principal engineer with the Computational Sensing Group, Sony Europe R&D Center Stuttgart Laboratory 1.

Yalcin Incesu studied electrical engineering at the Technical University of Hamburg, Harburg and received the diplom-Ingenieur degree in 2001. He is currently managing Advanced Sensors and Modelling Group, R&D Center Europe Stuttgart Laboratory 1. His research interests include on novel image sensors and their emulations and tight integration, with respective signal processing, and deep learning applications.

Pietro Zanuttigh (Member, IEEE) received the MSc degree in computer engineering and the PhD degree from the University of Padova, in 2003 and 2007, respectively. He is currently an associate professor with the Department of Information Engineering. His research interests include image and 3D data processing and analysis, ToF sensors data processing, domain adaptation, and incremental learning in semantic segmentation and hand gesture recognition.