Seminar Report: Depth Denoising

Claudius Kienle[1]
Autonomous Learning Robots Lab
Adenauerring 4
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: claudius.kienle@student.kit.edu

David Petri[2]
Autonomous Learning Robots Lab
Adenauerring 4
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: david.petri@student.kit.edu

***Abstract*—[12] Denoising of depth maps captured by RGB-D cameras is a relatively new and small field of research. The goal of the denoising is to reconstruct scene specific depth features, by canceling out the inaccuracies, which are created by the capturing depth sensor. These inaccuracies are mainly caused by the capturing technique used and the quality and size of the sensor. In this work we present five denoising approaches. These denoising approaches can be generally divided into two groups, namely non-machine-learning-based methods and machine-learning-based methods. For the former approaches we present *Bilateral Filter* by Tomasi et al. [1] and *Rolling Guidance Filter* by Zhang et al. [2]. For the latter approaches we present *Depth Denoising and Refinement Network* by Yan et al. [3], *Self-Supervised Deep Depth Denoising* by Sterzentsko et al. [4] and *Self-Supervised Depth Denoising using Lower- and Higher-quality RGB-D sensors* by Shabanov et al. [5]. In the end we provide a comparison between all five approaches, based on the results obtained by Sterzentsko et al. [4] and Shabanov et al. [5] and discuss their performance.**

I. Introduction[2]

Depth sensors represent a valuable tool for computer vision tasks, as they provide additional depth information in a fourth channel to the usual three RGB channels. In the field of robotics visual guided systems use this additional information to better perceive objects in their working space.

Different types of such RGB-D sensors exist, depending on the technologies used for depth measurement. The existing technologies can be coarsely divided into three groups.

The first relies on structured, coded light or infrared light that is projected onto the surfaces. As the projected pattern is known beforehand, a single sensor is sufficient to calculate the depth values through triangulation. Examples of such sensors are the Zivid product-line or the first version of the Microsoft Kinect.

The second method relies on a stereo view of the scene. Two or more cameras observe the scene, and as their relative distance to each other is known, the depth map is calculated through triangulation on characteristic features chosen among the images. Examples of such cameras are the Stereolabs ZED series and the Intel RealSense D-series.

[1]written by Claudius
[2]written by David

The third method uses the speed of light. The sensor emits a beam of light and measures the time the light takes until recapture. The distance then can be computed since the speed of light is known. An example of such a system is the Microsoft Kinect 2 and Azure Kinect.

Just like the technology of the depth sensors varies, so does the size and price of the sensors, but above all, the accuracy of their retrieved depth measurements. Lower-end systems, such as RealSense or Apple TrueDepth sensors produce rather noisy depth maps, compared to higher-end systems like Zivid or Kinect sensors. It is therefore of interest to find denoising processes that increase the quality of the depth maps captured with former sensors.

In the following, we present five approaches, by which means depth maps can be denoised. These approaches can be coarsely divided into two groups, namely filter or non machine learning based methods and machine learning based methods. For the first group we present *Bilateral Filter* [1] and *Rolling Guidance Filter* [2] in Section II. For the latter we investigate data-driven approaches, *Depth Denoising and Refinement Network* [3], *Self-Supervised Deep Depth Denoising* [4] and *Self-Supervised Depth Denoising using Lower- and Higher-quality RGB-D sensors* [5] in Section III. Conclusively, we compare the performance of all presented approaches based on the results of the last two approaches mentioned.

II. Non-ML-Based Methods

A. Bilateral Filter (BF)[1]

A rather old but very popular image denoising approach are bilateral filters (BF) introduced by Tomasi et al. [1] in 1998. Bilateral Filters are a non data-driven approach, relying on the combination of image domain and image range filtering. Here, the image domain refers to the spacial similarity between pixels, and the image range to the photometric similarity between pixels.

A domain filter applied to an image $\boldsymbol{f}(x)$ produces the output image $\boldsymbol{h}(x)$, given by Equation (1),

$$\boldsymbol{h}(x) = \frac{1}{k_d(x)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{f}(\xi) c(\xi, x) d\xi \tag{1}$$

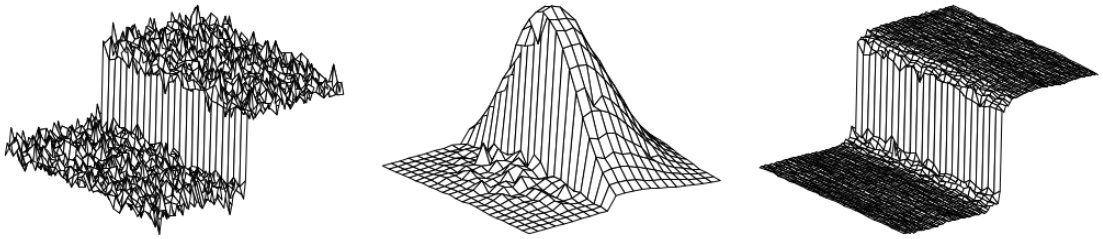

Fig. 1. Figure taken from Tomasi et al. [1]. It shows: in (a) a 100-gray-level step perturbed by Gaussian noise with $\mu = 10$ gray level. In (b) combined similarity weights $c(\xi, x)s(f(\xi), f(x))$ for a 23×23 neighborhood, centered on a pixel in vicinity of the step in (a), actively ignoring the pixels with dissimilar values. In (c) the step of (a) can be seen after bilateral filtering with $\mu_r = 50$ gray levels and $\mu_d = 5$ pixels. [1]

whereby the closeness function $c(\xi, x)$ captures the spacial similarity of neighboring pixel ξ to center pixel x. The closeness function $c(\xi, x)$ can e.g. be the euclidean distance between the two pixels ξ and x. To retrieve valid photometric values, $k_d(x)$ of Equation (2) normalizes $\boldsymbol{h}(x)$.

$$k_d(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) d\xi \tag{2}$$

The domain filter therefore only accounts for the spacial closeness, without accounting for photometric similarity. E.g. considering a depth image of a cube. A pixel near the cube's edge has different color and depth information than a spatially close pixel, which is part of the background. Not considering color information, but only relying on spacial similarity may therefore result in a blurred edge after filtering.

A range filter on the other hand, as defined in Equation (3),

$$\boldsymbol{h}(x) = \frac{1}{k_r(x)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{f}(\xi) s(\boldsymbol{f}(\xi), \boldsymbol{f}(x)) d\xi \tag{3}$$

accounts only for the photometric similarity between pixels for filtering. The similarity function $s(\boldsymbol{f}(\xi), \boldsymbol{f}(x))$ captures the photometric similarity of neighboring pixel ξ to center pixel x. Similar to domain filter, the normalizer $k_r(x)$ is given by Equation (4).

$$k_r(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\boldsymbol{f}(\xi), \boldsymbol{f}(x)) d\xi \tag{4}$$

Bilinear filters, as given in Equation (5), combine domain and range filter and thereby enforce both spatial and photometric similarity for filtering. The normalizer $k(x)$ is as before given in Equation (6).

$$\boldsymbol{h}(x) = \frac{1}{k(x)} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \boldsymbol{f}(\xi) c(\xi, x) s(\boldsymbol{f}(\xi), \boldsymbol{f}(x)) d\xi \tag{5}$$

$$k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) s(\boldsymbol{f}(\xi), \boldsymbol{f}(x)) d\xi \tag{6}$$

Bilateral filtering therefore replaces the pixel value of x with a weighted average of proximate and similar pixel values. The intuition behind Equation (5) is as follows, with c, s and k denoting closeness, similarity and normalization function respectively:

Considering a smooth image, pixels in close vicinity are similar to each other. Therefore, the normalized similarity function $\frac{s}{k}$ is close to one, as all s values are similar. As consequence, the bilateral filter does behave as a domain filter, averaging away small and weakly correlated differences between pixel values.

Considering on the other hand a depth map of the distinct edge between an object and its background, as illustrated in Figure 1 (a). Let the bilateral filter be centered on a pixel x of the object close to the edge. The similarity function s will then consider values close to one for pixels on the object in x's vicinity. Pixels on the background in x's vicinity on the other hand will have a similarity value close to zero. These values then re-weight the weights of the closeness functions according to the photometric similarities of x's vicinity.

Gaussian filter: An example of a bilateral filter is the shift-invariant Gaussian filter. It defines the closeness and similarity functions as Gaussian functions of the euclidean distance between their arguments. The closeness function is therefore given by Equation (7), whereby the similarity function is analogously given by Equation (8).

$$c(\xi, x) = e^{-\frac{1}{2}(\frac{d(\xi, x)}{\sigma_d})^2} \tag{7}$$

$$d(\xi, x) = d(\xi - x) = ||\xi - x||_2$$

$$s(\xi, x) = e^{-\frac{1}{2}(\frac{\delta(\boldsymbol{f}(\xi), \boldsymbol{f}(x))}{\sigma_r})^2} \tag{8}$$

$$\delta(\phi, \boldsymbol{f}) = d(\phi - \boldsymbol{f}) = ||\phi - \boldsymbol{f}||_2$$

In Equation (7), σ_d accounts for the desired spread of the filtering. A large value for σ_d will have a higher spread and therefore more blur. Similarly, does the photometric spread σ_r in Equation (8) account for the tolerated dissimilarity in photometric values of neighboring pixels ξ. Pixels with photometric values closer to the center pixel x than σ_r get a high weight, and vice versa.

B. Rolling Guidance Filter[2]

A more recent, non-ml-based approach originates from Zhang et al. [2], who introduced rolling guidance filters in 2014. Filtering methods, such as the previously mentioned bilateral filtering, are designed to preserve high-contrast edges

and remove low-contrast or gradual changes. This is achieved by processing the information contained in the pixels' magnitude and distances. The scale of the regions filtered is, however, not taken into account. Consequently, small structures, such as texture, small object and noise may end up being treated as large scale information, such as boundaries, slow spatial transitions and flat regions. Latter are of more interest for the perception of the objects' arrangement in a scene. Zhang et al. [2] address this by proposing a scale-aware filter, that is able to remove different levels of details in any given input image. Their idea is based on the observation that after filtering an image with a Gaussian filter, nearly all texture patterns are removed, while large-scale intensity variations, like edges, are only blurred and can still be found.

The method proposed by Zhang et al. [2] are composed of two steps, firstly, the *small structure removal* step, which is followed by the iterative step of *edge recovery*.

Zhang et al. [2] first define the structure scale σ_s as the smallest Gaussian standard deviation, such that if a Gaussian filter $g_{\sigma_s^2}(x, y)$, as given by Equation (9),

$$g_{\sigma_s^2}(x,y) = \frac{1}{\sqrt{2\pi\sigma_s^2}} exp\left(-\frac{x^2+y^2}{2\sigma_s^2}\right) \tag{9}$$

is applied on an image, all to σ_s corresponding structures disappear. According to scale-space theory [6], the resulting image is at scale σ_s^2, meaning that all its structural components with a scale smaller than σ_s are completely removed.

Small Structure Removal: The first step aims to remove all small-scale structures, which are of insignificance to the information held by the image. RGF therefore uses the Gaussian filter G, given in Equation (10).

$$G(p) = \frac{1}{K_p} \sum_{q \in N(p)} exp\left(-\frac{||p-q||_2^2}{2\sigma_s^2}\right) I(q) \tag{10}$$

$$K_p = \sum_{q \in N(p)} exp\left(-\frac{||p-q||_2^2}{2\sigma_s^2}\right)$$

K_p normalizes G and $N(p)$ is the set of pixels in the vicinity of p. As described for Equation (9), this filter completely removes structural components with a scale smaller than σ_s.

Edge Recovery: The interactive edge recovery step presents the main contribution of Zhang et al. [2], in which the image J is iteratively updated to finally restore the blurred large-scale structures. Initializing J^1 with the output of the Gaussian filtering of step one, J^{t+1} equals the joint bilateral filtering from given input image I and the value of the previous iteration J^t as stated in Equation (11).

$$J^{t+1}(p) = \frac{1}{K_p} \sum_{q \in N(p)} exp\left(-\frac{||p-q||_2^2}{2\sigma_s^2} - \frac{||J^t(p) - J^t(q)||_2^2}{2\sigma_s^r}\right) I(q) \tag{11}$$

$$K_p = \sum_{q \in N(p)} exp\left(-\frac{||p-q||_2^2}{2\sigma_s^2} - \frac{||J^t(p) - J^t(q)||_2^2}{2\sigma_s^r}\right)$$

K_p normalizes the equation and σ_s and σ_r control the spatial scale and range scale respectively. Equation (11) can be intuitively understood, as that the filter smoothens the input I guided by the structure of J^t. For small and large structures the intuition behind Equation (11) can be understood as follows:

In Figure 2 the 1D signal of a small structure in (a) and a larger structure in (b) can be seen. For first example (a), the small structure is completely removed by the Gaussian filter of Equation (10) and $G = J^1$ is mostly flat. In consecutive iterations of Equation (11) the term $||J^t(p) - J^t(q)||$ is therefore almost zero for any (p, q) pairs. This makes the joint bilateral filter behave like a Gaussian filter. The small removed structure in J^{t+1} stays flat and none of the following iterations can restore the detail.

For large structures as (b), rolling guidance filtering behaves differently. The beforehand visible edge is only smoothed out by the Gaussian filter of Equation (10) and therefore still identifiable. Since each iteration takes a weighted average of I, the result J^{t+1} will be smoother than I but of similar sharpness as I. It is also guaranteed to be sharper than J^t. This holds, due to the involvement of the range weights $\frac{||J^t(p)-J^t(q)||_2^2}{2\sigma_s^r}$, as its numerator is now significantly different from zero. It can therefore be expressed, that for the image smoothness holds: $I \leq J^{t+1} < J^t$, and for the image sharpness holds: $I \geq J^{t+1} > J^t$, since significantly more range difference is involved in computing the weights. Zhang et al. [2] show in their paper, that their process of bilateral filtering actually converges to the expected outcome of high sharpness and smoothness.

Figure 3 illustrates how different values of σ_s remove structures at different scales. Overall, large-scale structures are preserved, while small structures, such as noise, are removed depending on the σ_s chosen.

The results of Figures 2 and 3 show the inherent difference of rolling guidance filtering to other edge-preserving filters by preserving edges independent of their photometric magnitudes.

III. ML-Based Methods

A. Depth Denoising and Refinement Network (DDRNet)[1]

Yan et al. [3] proposed a data-driven end-to-end pipeline to learn a model, which denoises depth maps of consumer-level RGB-D cameras. A structural overview of the framework can be seen in Figure 4. In their paper, Yan et al. [3] differentiate between low- and high-frequency information, referring to smaller- and larger-scale structures of the depth maps respectively. The proposed framework tackles the denoising of these depth maps by splitting the task into two subtasks. The first task aims to denoise the low-frequency domain through self-supervised learning with a UNet-like architecture, using near-to-ground-truth depth maps. In Figure 4 it is referred to as *denoising net*. The second task aims to refine the high-frequency domain through unsupervised learning by a shading-based criterion built on inverse rendering. In Figure 4 it is referred to as *refinement net*.

Data Generation: In order to train either network, Yan et al. [3] used a non-rigid dynamic fusion pipeline proposed

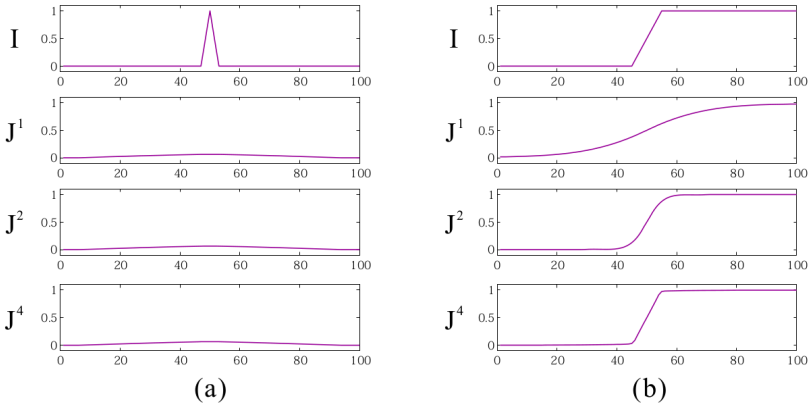

Fig. 2. Two examples and their results in rolling guidance of a 1D signal from the paper of Zhang et al. [2]. Example (a) shows a small structure, representing noise that should be removed. Example (b) in the other hand shows the edge of a larger structure that should be preserved, as it is of significance to the overall structural perception. [2]

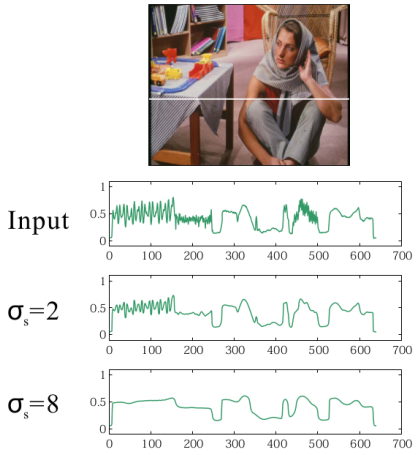

Fig. 3. Example from the paper of Zhang et al. [2], showing the 1D signal of the white line in the pictured image. The curves seen are input and two filtering results for different spacial similarity standard deviation σ_s. It can be seen, that overall significant large-scale edges and changes are preserved well. [2]

by Guo et al. [7], which reconstructs geometries of dynamic scenes from a single view RGB-D input. Therefore, the researchers captured sequences of synchronized RGB-D frames (D_{in}, C_{in}) at first. The non-rigid fusion pipeline [7] then produces a 3D mesh for every RGB-D tuple, which is then deformed using the estimated motion in each corresponding frame. Finally, rasterization at each corresponding view points generates the target depth maps D_{ref}. The target depth maps are therefore not actual ground-truth, but only close-to-ground-truth data. Additionally, they calculated foreground masks W^t by morphological filtering to indicate the region of interest.

The training dataset of Yan et al. [3] consequently contained 36,840 samples of human bodies, of which 11,540 originate from a structured light depth sensor and 25,300 result from a time-of-flight depth sensor. The authors moved the camera freely around the scene during capture and made no assumptions about the movement of the actor in the sequences.

Depth Map Denoising: The denoising net deployed by Yan et al. [3] was inspired by DispNet [8] and is a UNet-like architecture, as can be seen in Figure 5. The encoder aims to extract low-resolution high dimensional features from D_{in}. The skip connections are used to preserve geometric details of the input D_{in}. Instead of upsampling, transposed convolutions were used for the upward steps, adding complexity to the network.

The first part of the loss function used to train the denoising net is defined on the depth map itself, by taking the sum of the per-pixel L1 and L2 loss on the depth values, as seen in Equation (13), where $D_{dn} = \mathcal{D}(D_{in})$ and D_{ref} refers to the target depth maps. Yan et al. [3] state that "L2 and L1 losses may produce blurry results, however they accurately capture the low frequencies" [9], which is sufficient for the purpose of the denoising net.

To prevent high-frequency noise remaining in small local patches, a *normaldot* loss term is added.

$$\mathcal{L}_{dot}(D_{dn}, N_{ref}) = \sum_{i} \sum_{j \in \mathcal{N}(i)} [\langle P^i - P^j, N^i_{ref} \rangle]^2 \qquad (12)$$

As seen in Equation (12), it enforces for every point $P^i \in \mathbb{R}^3$ in the 3D coordinate system given by D^i_{dn} to have neighboring pixels P^j which are orthogonal to its reference normal N^i_{ref}. It therefore constrains the normal direction of the denoised depth map D_{dn} to be consistent with a reference normal direction N_{ref}. Yan et al. [3] retrieved the normal vectors N_{ref} from the reference depth map D_{ref} using a depth to normal layer. For each pixel it takes the surrounding 4 pixels to estimate the normal vector. The *normaldot* loss should additionally force the network to consider local dependence between neighboring pixels by learning locally joint distributions, to remove high-frequency noise.

The final loss for the training of the denoising net is therefore given by Equation (13).

$$\mathcal{L}_{dn}(D_{dn}, D_{ref}) = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{dot}\mathcal{L}_{dot} \qquad (13)$$
$$\mathcal{L}_{rec}(D_{dn}, D_{ref}) = ||D_{dn} - D_{ref}||_1 + ||D_{dn} - D_{ref}||_2$$

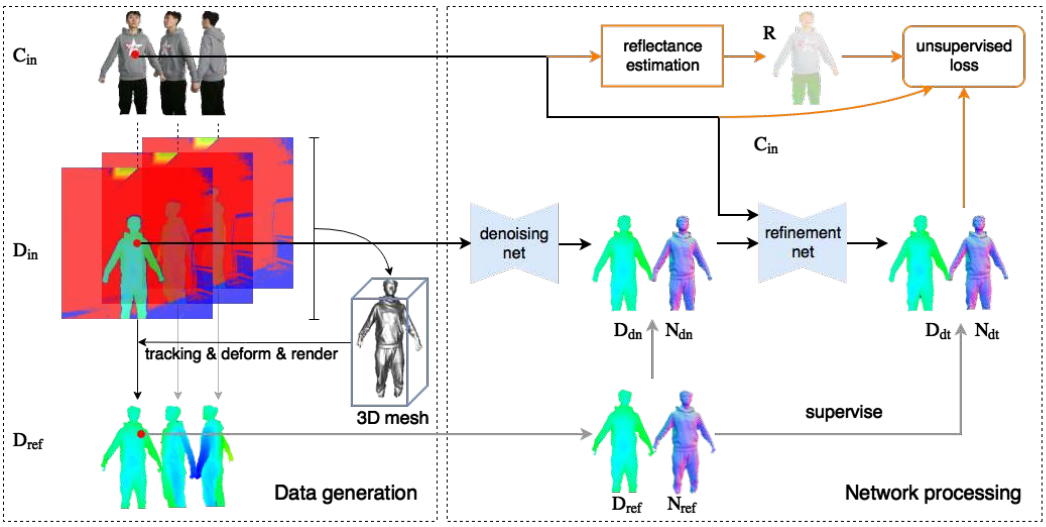

Fig. 4. Depth denoising and refinement pipeline proposed by Yan et al. [3]. It is composed of two parts, (1) data generation and (2) network processing. The black lines correspond to the forward pass at inference, the gray lines correspond to the supervise signal, and the orange lines relate to the unsupervised loss. [1]

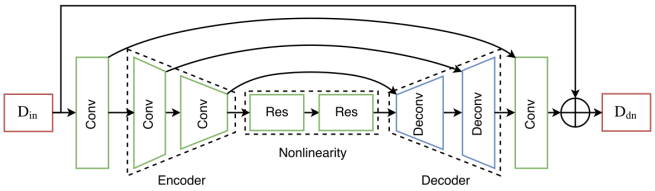

Fig. 5. Architecture of the UNet-like denoising net architecture of Yan et al. [3]. It consists of an encoder, a residual layer and a decoder. Additionally, one skip connection goes directly from the input to the output of the network. [1]

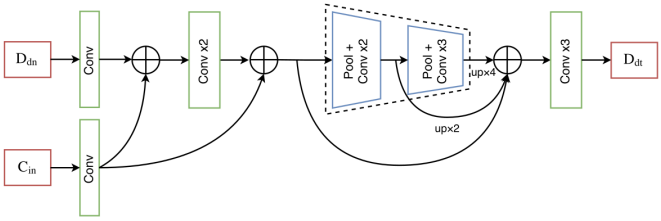

Fig. 6. Architecture of the refinement net used by Yan et al. [3]. The feature maps of D_{dn} are complemented/guided by corresponding feature maps from image input C_{in}. [1]

Depth Map Refinement: After denoising the depth map D_{in} using the denoising net, the output depth map D_{dn} may still lack details compared to the input image I_{in}. To restore high-frequency details in D_{dn}, Yan et al. [3] further deployed a relatively small CNN based on Hariharan et al.'s [10] hypercolumn architecture, seen in Figure 6. It intends to extract high-frequency features in the color image domain and use them as a guidance to (re-)construct local detailed structures on the denoised depth map D_{dn}.

The hypercolumn descriptor for a pixel results by concatenating the extracted feature maps of C_{in} twice with those of D_{in}. Max-pooling and convolutional down-sampling then fuses the hypercolumn descriptor in the spatial domains, ending with a multiscale fused feature map. This intends to transfer the fine structure from the color map domain to the depth map domain. To retrieve the original size, Yan et al. [3] employed up-sampling, for which they did not give any further detail, nor for the skip connections of the fusion layers. In the end, the authors used three post-fusion convolutional layers to learn a better channel coupling and added a final *tanh* activation to limit the output to the range of the input.

The loss function used for training the refinement net is depicted in Equation (14) and composed of the weighted sum of the three loss terms $\mathcal{L}_{sh}$, $\mathcal{L}_{fid}$ and $\mathcal{L}_{smo}$.

$$\mathcal{L}_{dt}(D_{dt}, D_{ref}, I) = \lambda_{sh}\mathcal{L}_{sh} + \lambda_{fid}\mathcal{L}_{fid} + \lambda_{smo}\mathcal{L}_{smo} \quad (14)$$

The first loss term $\mathcal{L}_{sh}$, shown in Equation (15), is the *per-pixel shading loss*, which is designed to penalize differences in intensity and gradient between the rendered image B and the corresponding image I.

$$\mathcal{L}_{sh}(N_{dt}, N_{ref}, I) = ||B(\boldsymbol{l}^*, N_{dt}, R) - I||_2 + \lambda_g||\nabla B(\boldsymbol{l}^*, N_{dt}, R) - \nabla I||_2 \quad (15)$$

N_{dt} is the normal map of the output D_{dt} of the refinement net and R is the albedo map estimated using Nestmeyer et al.'s "CNN + filter" method [11]. The light coefficients $\boldsymbol{l}^*$ can then be computed as the solution of the least squares problem stated in Equation (16).

$$\boldsymbol{l}^* = \arg\min_{\boldsymbol{l}} ||B(\boldsymbol{l}^*, N_{ref}, R) - I||_2^2 \quad (16)$$

The reflected irradiance B for Lambertian surfaces and low-frequency illumination can be expressed like Equation (17), where $H_b : \mathbb{R}^3 \mapsto \mathbb{R}$ is the basis function of spherical

harmonics (SH) and $\boldsymbol{l} = [l_1, ..., l_9]^T$ are the nine 2nd order SH coefficients representing the low-frequency scene illumination.

$$B(\boldsymbol{l}, N, R) = R \sum_{b=1}^{9} l_b H_b(N) \tag{17}$$

The second loss term $\mathcal{L}_{fid}$ aims to constrain the refined depth map to be close to the reference depth map using the L2 loss. The so-called fidelity loss is then given by Equation (18).

$$\mathcal{L}_{fid}(D_{dt}, D_{ref}) = ||D_{dt} - D_{ref}||_2 \tag{18}$$

The third and last component $\mathcal{L}_{smo}$ is given by the smoothness loss, depicted in Equation (19), which aims to regularize the refined depth map, by minimizing the anisotropic total variation of the depth

$$\mathcal{L}_{smo}(D_{dt}) = \sum_{i,j} |D_{dt}^{i+1,j} - D_{dt}^{i,j}| + |D_{dt}^{i,j+1} - D_{dt}^{i,j}| \tag{19}$$

Both networks, the denoising net and the refinement net, were then trained jointly with the following loss $\mathcal{L}_{total}$, given in Equation (20).

$$\mathcal{L}_{total} = \mathcal{L}_{dn} + \lambda \mathcal{L}_{dt} \tag{20}$$

B. Self-Supervised Deep Depth Denoising (DDD)²

Sterzentsenko et al. [4] in 2019 suggested another self-supervised approach based of a fully convolutional deep autoencoder, seen in Figure 7. The goal of the autoencoder is to exploit the photometric and depth information of different points of view on the same scene, relying on view synthesis as a supervisory signal. This approach therefore does not require clean depth measurements for training.

The multi-view self-supervision works as follows: Given $\mathcal{V}$ spatially aligned sensors $v_s, v_t \in \{1, \ldots, \mathcal{V}\}$, whose relative viewpoint positions are known, any given source image's pixel coordinates $\boldsymbol{p}_s = (x, y) \in \mathbb{R}$ of sensor v_s can be transformed into the coordinate system of sensor v_t by Equation (21). In it, $\boldsymbol{T}_{s \to t}$ denotes the extrinsic transformation matrix between source s and target t sensor coordinate system, π and π^{-1} denote projection and deprojection functions of pixel coordinates to 3D coordinates and vice versa, the sensor's intrinsic matrix is denoted by $\boldsymbol{K}$, and D_s denotes the depth map of the source sensor.

$$\mathcal{T}_{s \to t}(\boldsymbol{p}_s) = \pi(\boldsymbol{T}_{s \to t} \pi^{-1}(D_s(\boldsymbol{p}_s), \boldsymbol{K}_s), \boldsymbol{K}_t) \tag{21}$$

Using Equation (21), color information from each view can be transferred to any other on a per-pixel basis. However, while traveling from one view to the other, the resulting color images will most probably be distorted due to noisy depth maps manifesting in incorrect re-projections in $\pi^{-1}(D_s(\boldsymbol{p}_s), \boldsymbol{K}_s)$. Sterzentsenko et al. [4] use this for a self-supervised depth denoising approach based on an inter-view color reconstruction under the photoconsistency assumption. While utilizing color and depth information during training, only a single depth map is required at inference time. Additionally, multiple depth maps of the same scene will offer varying noise structures and levels, thereby increasing the diversity of the data. It should be noted that due to the purely geometric nature of this approach, any number of unstructured sensor placements is supported, while in the experiment of Sterzentsenko et al. [4] deployed four RealSense D415.

The resulting transformed pixel coordinates $\mathcal{T}_{s \to t}(\boldsymbol{p}_s)$ may be at sub-pixel accuracy. Differentiable rendering [13] and forward splatting is then used to accumulate pixel-wise color information in the target image. Effectively, every source pixel splats its weighted color information to the four pixels, which are in the immediate neighborhood of $\mathcal{T}_{s \to t}(\boldsymbol{p}_s)$. A visualization of this is given in Figure 8. A splatting function is therefore trained to learn the weights for the contribution of each source pixel.

The architecture of the used deep autoencoder can be seen in Figure 7. The autoencoder receives as input four depth maps, one for each view of the same scene. After compression and decompression in the UNet-like network, forward splatting using the information from the other three non-target images and depth maps synthesizes a target image $\hat{\boldsymbol{I}}$. Sterzentsenko et al. [4] trained the network using a geometrically-derived photometric consistency loss $\mathcal{L}_{ph}$ and regularized by depth and normal priors $\mathcal{L}_{depth}$ and $\mathcal{L}_{surface}$ given by Equation (22).

$$\mathcal{L}_{total} = \underbrace{\lambda_1 \mathcal{L}_{ph}}_{\text{data}} + \underbrace{\lambda_2 \mathcal{L}_{depth} + \lambda_3 \mathcal{L}_{surface}}_{\text{priors}} \tag{22}$$
$$\lambda_1, \lambda_2, \lambda_3 \in (0, 1) \quad \lambda_1 + \lambda_2 + \lambda_3 = 1$$

Photometric consistency: The photometric consistency loss $\mathcal{L}_{ph}$ should minimize the pixel-wise error between $\boldsymbol{I}$ and $\hat{\boldsymbol{I}}$ and is linearly combined by two terms, namely the color-based loss $\mathcal{L}_{col}$ and the structural loss $\mathcal{L}_{str}$, like seen on Equation (23).

$$\mathcal{L}_{ph} = (1 - \alpha)\mathcal{L}_{col} + \alpha \mathcal{L}_{str} \quad \alpha \in (0, 1) \tag{23}$$

The color-based loss aims to penalize differences in the color intensity between $\ddot{\boldsymbol{I}}$ and $\hat{\boldsymbol{I}}$, where $\ddot{\boldsymbol{I}}$ denotes for the masked input image to account for missing color information in pixels in the splatted target image $\hat{\boldsymbol{I}}$. It is given by Equation (24), where $\rho(x) = \sqrt{x^2 + \gamma^2}, \gamma \approx 0$ is the Charbonnier penalty, used for robustness against outliers and $M(\boldsymbol{p})$ masking pixels, whose depth information is missing.

$$\mathcal{L}_{col} = \sum_{\boldsymbol{p}} \rho(M(\boldsymbol{p}) || \ddot{\boldsymbol{I}}(\boldsymbol{p}) - \hat{\boldsymbol{I}}(\boldsymbol{p})||_1) \tag{24}$$

The intuition behind the structural loss is, that it "forces prediction invariance to local illumination changes and structural information preservation" [4] and is given by Equation (25).

$$\mathcal{L}_{str} = 0.5 \sum_{p} \phi(M(\boldsymbol{p})(1 - \boldsymbol{SSIM}(\ddot{\boldsymbol{I}}(\boldsymbol{p}), \hat{\boldsymbol{I}}(\boldsymbol{p})))) \tag{25}$$

It uses the structural similarity index measure (SSIM), which is a perceptual metric to quantify image quality degradation [14]. $\phi(x)$ is the Tukey's penalty, which is used to reduce the magnitude of the outliers' gradients close to zero.

Depth regularization: The depth information of the prior is further used as regularization. Given the residual $r =$

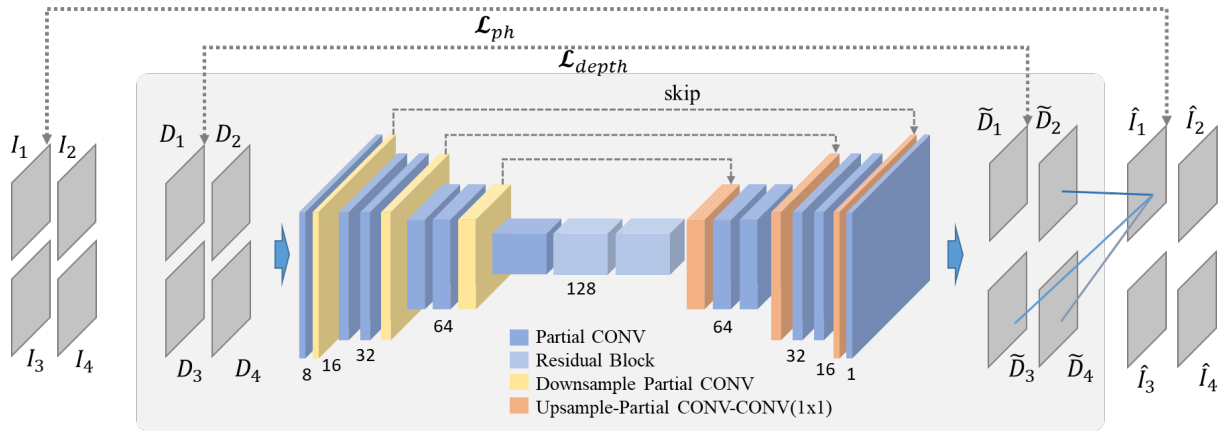

Fig. 7. Network architecture of fully convolutional autoencoder used in the approach of Sterzentsenko et al. [4]. Its core is based on the UNet architecture of Ronneberger et al. [12], but is with channel steps of [8, 16, 32, 64, 128] from top to bottom, much shallower than the UNet with [64, 128, 256, 512, 1028]. It receives the raw depth maps of all available sensors $(D_1 - D_4)$ and predicts the denoised depth maps $(\tilde{D}_1 - \tilde{D}_4)$. Using differentiable rendering and forward splatting a new target color image $\hat{\boldsymbol{I}}_1$ is created. $\hat{\boldsymbol{I}}_1$ is then used to compute the photometric loss L_{ph}, taking $\boldsymbol{I}_1$ as ground truth. This is iteratively done for all four input views and the total loss calculated as the sum of them. [2]

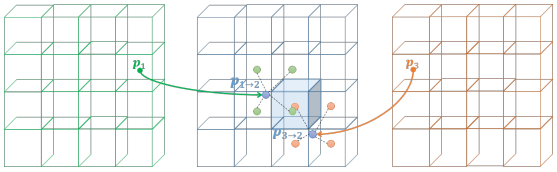

Fig. 8. Illustration of the multi-view forward splatting technique used by Sterzentsenko et al. [4] to render multiple transformed source images into a single target image. Source images 1 and 3 (green and orange respectively) splat their color information to the target image 2 (blue) After transformation using Equation (21), source pixel coordinates such as $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ are in sub-pixel accuracy in target coordinate system as $\boldsymbol{p}_{1\to2}$ and $\boldsymbol{p}_{3\to2}$. The color information of their respective source view contributes to the four neighboring pixels. Each contribution is weighted by the source pixel's confidence, which is incorporated in the learned splat function. As shown in the blueish highlighted pixel of image 3, multiple views contribute with their color information to an individual pixel. [2]

$M \odot (D - \tilde{D})$, the inverse Huber loss (BerHu), shown in Equation (26), is used, whereby c being the border value defined as 20 % of the maximum per batch residual $c = 0.2max(r)$.

$$\mathcal{L}_{depth} = \begin{cases} |r|, & |r| \leq c \\ \frac{r^2+c^2}{2c}, & |r| > c \end{cases} \tag{26}$$

BerHu was used, as it was found in [15] to be more appropriate as depth estimator.

Surface regularization: Lastly a surface regularization prior is used to enforce smoothness in the predicted depth maps. It is given by Equation (27), where $\boldsymbol{n}(\boldsymbol{p})$ denotes the normal vector of the 3D surface of the deprojected input image.

$$\mathcal{L}_{surface} = 1 - \sum_{\boldsymbol{p}} \sum_{\boldsymbol{p}' \in \Theta_{\boldsymbol{p}}} |\langle \boldsymbol{n}(\boldsymbol{p}), \boldsymbol{n}(\boldsymbol{p}') \rangle| \frac{M(\boldsymbol{p})}{G(\Theta_{\boldsymbol{p}})} \tag{27}$$

$\Theta_{\boldsymbol{p}}$ is the set of all 2D neighboring pixels around $\boldsymbol{p}$ and $G(\Theta_{\boldsymbol{p}})$ is an operator determining the number of valid (non-zero) depth values in $\Theta_{\boldsymbol{p}}$.

Experiments conducted by Sterzentsenko et al. [4] suggest, that their approach on one hand outperformed state-of-the-art approaches, among which are the already introduced Bilateral Filter (BF), Joint Bilateral Filter (JBF), Rolling Guidance (RGF), as well as the ml-based approaches DRR and DDRNet. On the other hand it is stated, that the presented self-supervised approach "maintains its performance when denoising depth maps captured [also] from other sensors" [4].

C. Self-Supervised Depth Denoising using Lower- and Higher-quality RGB-D sensors (SSDD)[1]

Shabanov et al. [5] in 2019 presented a technique to denoise depth images that relates the most to the approach from Sterzentsenko et al. [4]. They focused on denoising depth frames that depict human bodies. The denoising of the frames was performed with a two-level network structure that also respects temporal information within frames. For training, a custom dataset was recorded and processed. Besides the lower-quality (LQ) frames $(\boldsymbol{C}_{LQ}, \boldsymbol{D}_{LQ})$ captured with the TrueDepth camera of an iPhone X, the dataset also contains labels $(\boldsymbol{C}_{HQ}, \boldsymbol{D}_{HQ})$ that stem from the higher-quality (HQ) camera Microsoft Kinect V2. Compared to Sterzentsenko et al., they did not require both cameras to be synchronized beforehand for dataset generation. Instead, the cameras were mounted next to each other and aligned in space and time.

Temporal alignment was applied with the help of timestamp synchronization and frame filtering as post-processing. To align two pairs of frames with HQ timestamp t^i_{HQ} and LQ timestamp t^j_{LQ} in time, a constant time shift $\Delta_{HQ\to LQ} \in [-60ms, 60ms]$ is sampled with binary search. A mapping function $\Phi(i)$ then assigns each t^i_{HQ} to a $t^{\Phi(i)}_{LQ}$, such that the shifted timestamp difference

$$\Phi(i) = \underset{j \in [1 \dots N]}{\arg\min} |t^j_{LQ} - (t^i_{HQ} + \Delta_{HQ\to LQ})| \tag{28}$$

is minimal. Every time shift $\Delta_{H\to L}$ is evaluated with the loss from spacial alignment. Given a temporal mapping function $\Phi(i)$, temporally aligned tuples, as shown in Equation (29), can be conducted.

$$(\boldsymbol{C}_{LQ}^{\Phi(i)}, \boldsymbol{D}_{LQ}^{\Phi(i)}, \boldsymbol{C}_{HQ}^{i}, \boldsymbol{C}_{LQ}^{i}) \quad i \in [1 \dots N] \tag{29}$$

Spatial Alignment then transforms the HQ frames i onto the LQ's image plane $\Phi(i)$. The alignment is performed for every HQ pixel $p_{HQ} = (x, y, z)$ individually according to Equation (30), whereby the z value originates from the depth frame $\boldsymbol{D}_{HQ}$. Due to the known depth value z, the pixel can be unprojected to camera space with an unprojection function π_{HQ}^{-1}. The unprojected point is then transformed by T_{ex}, consisting of a rotation and a translation. Finally, the pixel is re-projected to LQ camera space resulting in a aligned pixel $\widetilde{p}_{HQ}$.

$$\widetilde{p}_{HQ} = \pi_{LQ}(T_{ex} * \pi_{HQ}^{-1}(p_{HQ})) \tag{30}$$

By selecting feature pixels $(p_{LQ}, p_{HQ}) \in U$ using the color images $\boldsymbol{C}_{HQ}$ and $\boldsymbol{C}_{LQ}$, a transformation T_{ex} can be learned that minimizes the L2 loss of the projected feature pixels $\widetilde{p}_{HQ}$ and the original feature pixels p_{LQ} as shown in Equation (31).

$$\mathcal{L}(U, T_{ex}) = \sum_{(p_{LQ}, p_{HQ}) \in U} ||\widetilde{p}_{HQ} - p_{LQ}||_2 \tag{31}$$

This alignment results in HQ and LQ frame pairs with a pixel-wise correspondence.

To denoise the LQ depth frames, a two-level training approach based on out-of-fold was utilized. A UNet is used for the first level and a LSTMUNet for the second to improve the UNet's results by respecting temporal information. Both networks were trained on the aligned dataset with L1 loss depicted in Equation (32).

$$\mathcal{L}(\boldsymbol{D}_{pred}, \widetilde{\boldsymbol{D}}_{HQ}) = ||(\boldsymbol{D}_{pred} - \widetilde{\boldsymbol{D}}_{HQ}) * \boldsymbol{M}_r * \boldsymbol{M}_{NaN}||_1 \tag{32}$$

To focus only on denoising the human bodies, the loss function was computed only on those pixels by computing a mask $\boldsymbol{M}_r$ beforehand. Due to the transformation that was applied to align the frames spatially, the HQ depth frames contain NaN values, which must also be ignored in the loss.

IV. Comparison[2]

In this section, results obtained by the five depth denoising approaches introduced are presented. The results presented are based on the results obtained by Sterzentsko et al. [4] and Shabanov et al. [5]. Former compared their approach of self-supervised deep depth denoising with bilateral filters, rolling guidance filters and the DDRNet. Latter, as it is more recent, compared its self-supervised approach with bilateral filters, rolling guidance filters and DDRNet and the former.

Sterzentsko et al. [4] used two types of scenes in their training set, as can be seen in Figure 9. The scenes were captured in sequences using four Intel RealSense D415 which were mounted on an H-stand, as can be seen in Figure 10. The first type consisted of a combination of "human actions simultaneously performed by multiple people" [4], and the

Fig. 9. Two training samples (image and depth pair) of training set used by Sterzentsko et al. [4]. Captured content consisted of balloons and multi-person activities. [1]

second type of "a special set of moving textured balloons of different colors" [4]. The authors by this intended to "create a dataset of sufficient depth variability" [4]. For quantitative evaluation purposes on the other hand, "due to the lack of ground truth, depth maps from Kinect V2 [were] [...] used as 'close to ground truth' data" [4]. Therefore, a Microsoft Kinect V2 and an Intel RealSense were mounted on a rigid-structure and "a 70-pair RGB-D set of instant samples with varying content" [4] were captured. An example of an evaluation image can be seen in Figure 11, depicting a human actor.

Shabanov et al. [5] on the other hand "recorded 51 simultaneous RGB-D sequences with different human actors, each about 30 seconds long" [5]. As seen in Figure 10, an Intel RealSense D415 and a Microsoft Kinect V2 were mounted next to each other to record the sequences. Actors were first asked to make a turnaround in front of the camera and after to move freely for the remaining time. In order to maximize the diversity of the recorded data, the recordings were done in six different environments. As the dataset of Shabanov et al. [5] already entailed near-to-ground-truth data in form of the higher-quality images captured by the Kinect V2, the additional step of Sterzentsko et al. [4] of capturing evaluation data is not needed.

To be mentioned is, that: First, it is not clear, whether Sterzentsko et al. [4] trained DDRNet on their data or used Yan et al.'s [3] trained model. It is only stated, that the "refinement net of the [DDRNet] network is omitted in order to have a fair comparison in denoising" [4]. Second, Shabanov et al. [5] did not retrain the used DDRNet and DDD net on their dataset, as it is challenging due to their different nature. For the DDRNet it was not possible, because "the RD depth is usually too noisy to perform 3D reconstruction reliably", which is at its core. For DDD it was not possible, as Shabanov et al.'s [5] data set does not represent a simultaneous multi-view of the scene.

Comparing the characteristics of the two datasets, the following similarities can be seen: First, the sensors used for capturing the input and target images of the model are the same. Second, the scenes captured are very similar, as both datasets consist of sequences of human actors. Only does the training set of Sterzentsko et al. [4] additionally contain sequences of balloons. Based on these similarities, the following comparisons can be made: First, the two non-ml approaches can be analyzed on performance and consistency

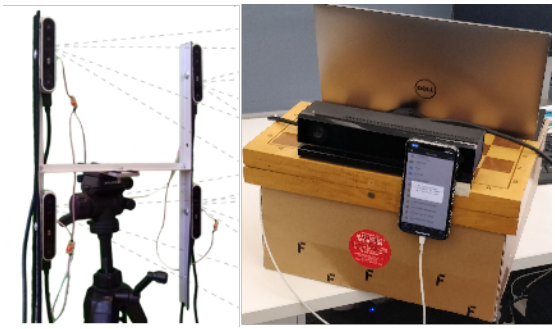

Fig. 10. Camera setup used by Sterzentsko et al. [4] (left) and Shabanov et al. [5] (right). [1]

at the depth denoising task. Second, DDRNet and DDD can be analyzed on generalizability. Therefore, the quantitative and qualitative results of Sterzentsko et al. [4] and Shabanov et al. [5] are opposed in Table I and Figure 11 respectively.

Instead of using the pixel-wise MSE as Shabanov et al. [5], Sterzentsko et al. [4] used the root MSE (RMSE) to evaluate the quality of approach. The error term is retrieved by first calculating the Least Square Planes for each point in the ground truth using all pixel points in a 5 mm radius. This constructs a plane, which minimizes the LSE for all points within the given radius. After "the distance of each calculated plane of the ground-truth point cloud against the closest point from the denoised point clouds contribute [then] a term to the final RMSE" [4]. An explanation, in which form this is a reasonable metric for comparison is not given. But it appears to rather interpolate the possible noisy surfaces in a rather high area of 5 mm, resulting in a biased error term.

First, analyzing the results of the filter based algorithms, one can observe an inconsistency. For the DDD dataset, BF outperforms RGB by the same margin as RGF outperforms BF for the SSDD dataset. This may be caused by the different way of error calculation used by Sterzentsko et al. [4]. Additionally, no information is given about the error of the raw dataset with respect to the near ground truth data. This makes it difficult to put any of Sterzentsko et al.'s [4] quantitative results into relation with their actual denoising performance with respect to the raw noisy depth maps. Assuming a similar error between raw and target depth maps as of Shabanov et al. [5] may be possible, as their sensors used are the same. Further, assuming that the error reduced by the filter based algorithms is within the range of 15 mm, as for the case of Shabanov et al. [5], the initial error may be around 90 and 100 mm. The improvement in the denoised images reached by Shabanov et al.'s [5] approach is therefore rather small compared the filter based approaches, whereas Shabanov et al. [5] halved the error with their approach.

Further, analyzing the results of the ml based approaches, one can observe a significant outlier for DDRNet within the results of Sterzentsko et al. [4]. It is almost three times higher than any other error within their observations, and possibly higher than the raw error. This may be due to the fact of Sterzentsko et al. only utilizing DDRNet's denoising net without the refinement net, which is a significant part of Yan et al.'s [3] pipeline. Yan et al. [3] did not suggest in their paper for their pipeline to be used in such form. For Shabanov et al. [5] DDRNet performs worse than the filter based approaches, but does not worsen the error as DDD does.

Taking a look at the qualitative results of Figure 11, the described malfunctioning of DDRNet for Sterzentsko et al. can be seen clearly. Comparing the depth maps of DDD to BF and the raw depth, a slightly higher smoothing of the noise can be seen, but no significant reconstruction of image specific high-frequency details. Opposing stands the SSDD model's results of Shabanov et al. [5], in which a significant smoothing with preservation of high-frequency details can be seen. E.g. are the relatively small contours along the hoody's zipper clearly to be seen. The other two ml-based approaches do not show any of such behavior and are mostly indistinguishable to the filter based approaches for the quantitative results presented by Shabanov et al. [5].

Finally looking at the complexity of each approach, the non-ml approaches are naturally the least complex and most straight forward. Among the ml-based approaches, Shabanov et al.'s [5] SSDD approach seems to be the most robust and straight forward approach, as it uses solely a standard L1 loss term to train its slightly modified UNet architectures. Yan et al.'s [3] DDRNet and Sterzentsko et al.'s [4] used more complicated approaches and complex loss terms without better results.

V. Conclusion[1]

In this work, five methods were presented, applicable for the denoising of depth maps. The two non-ml-based approaches *Bilinear Filters* and *Rolling Guidance Filters* are straight forward and easy ways to successfully denoise depth maps on a small scale. Due to their nature, these approaches lack the ability to reconstruct features with the help of learned information. On the other hand, ml-based approaches, such as *DDRNet*, *DDD*, and *SSDD* are potentially capable of using learned information for reconstruction purposes. Former two present rather complex and more complicated approaches, seeming to be difficult to generalize or be re-trained on other datasets. Latter presents a more straight forward and simple approach, which is easier to be retrained on datasets with available ground truth or close to ground truth data. Nevertheless, all of these approaches greatly depend on the domain of data on which they were trained and evaluated on.

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 839–846.

[2] Q. Zhang, X. Shen, L. Xu, and J. Jia, "Rolling guidance filter," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 815–830.

[3] S. Yan, C. Wu, L. Wang, F. Xu, L. An, K. Guo, and Y. Liu, "Ddrnet: Depth map denoising and refinement for consumer depth cameras using cascaded cnns," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 151–167.

	Raw	BF	RGF	DDRNet	DDD	SSDD (UNet)	SSDD (LSTM)	Paper
RMSE (mm)	-	73.25	81.35	239.06	**58.95**	-	-	**DDD** - Sterzentsko et al. [4]
MSE (mm)	57.22	56.39	49.49	57.08	84.07	**31.61**	**21.02**	**SSDD** - Shabanov et al. [5]

TABLE I

COMPARISON OF QUANTITATIVE RESULTS OBTAINED BY STERZENTSKO ET AL. [4] (FIRST ROW) AND SHABANOV ET AL. [5] (SECOND ROW) ON THE FIVE DEPTH DENOISING APPROACHES PRESENTED IN THIS WORK. [2]

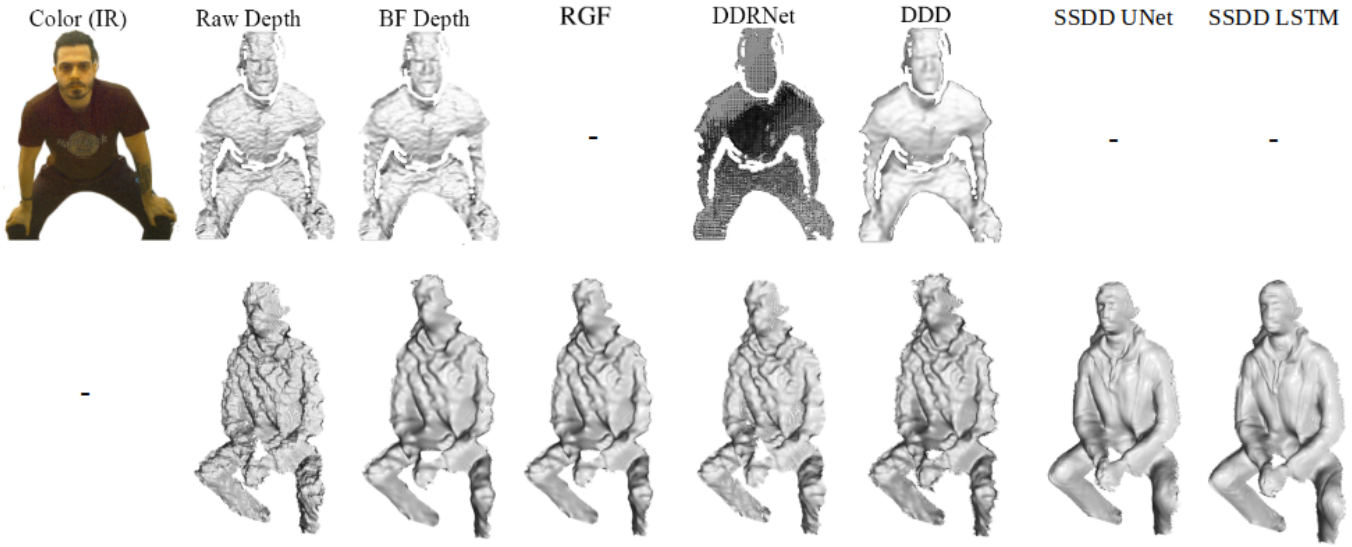

Fig. 11. Comparison of qualitative results obtained by Sterzentsko et al. [4] (first row) and Shabanov et al. [5] (second row) on the five depth denoising approaches presented in this work. [2]

[4] V. Sterzentsenko, L. Saroglou, A. Chatzitofis, S. Thermos, N. Zioulis, A. Doumanoglou, D. Zarpalas, and P. Daras, "Self-supervised deep depth denoising," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1242–1251.

[5] A. Shabanov, I. Krotov, N. Chinaev, V. Poletaev, S. Kozlukov, I. Pasechnik, B. Yakupov, A. Sanakoyeu, V. Lebedev, and D. Ulyanov, "Self-supervised depth denoising using lower-and higher-quality rgb-d sensors," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 743–752.

[6] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[7] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu, "Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, p. 1, 2017.

[8] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[10] B. Hariharan, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization. arxiv: 1411.5752," 2014.

[11] T. Nestmeyer and P. V. Gehler, "Reflectance adaptive filtering improves intrinsic image estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6789–6798.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[13] S. Tulsiani, R. Tucker, and N. Snavely, "Layer-structured 3d scene inference via view synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 302–317.

[14] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[15] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.