This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

ToFNest: Efficient normal estimation for time-of-flight depth cameras

Szilárd Molnár, Benjamin Kelényi and Levente Tamás
Faculty of Automation and Computer Science,
Technical University of Cluj-Napoca, Romania
Levente.Tamas@aut.utcluj.ro

Abstract

In this work, we propose an efficient normal estimation method for depth images acquired by Time-of-Flight (ToF) cameras based on feature pyramid networks (FPN). We perform the normal estimation starting from the 2D depth images, projecting the measured data into the 3D space and computing the loss function for the point cloud normal. Despite its simplicity, our method called ToFNest proves to be efficient in terms of robustness and runtime. In order to validate ToFNest we performed extensive evaluations using both public and custom outdoor datasets. Compared with the state of the art methods, our algorithm is faster by an order of magnitude without losing precision on public datasets. The demo code is available on https://github.com/molnarszilard/ToFNest

1. Introduction

The 3D perception for autonomous robots is becoming a standard today. Typical sensors include LiDAR and RGB-D cameras, which capture the geometric characteristics of the environment in the form of sparse point clouds. Although there is a wide range of the 2D convolutional neural network solutions for the spectral cameras, for the radial information represented by discrete points the processing solutions are rather limited. This is mainly due to the challenges of the 3D spatial convolution operator which does not scale well with the number of points from the point cloud or voxels from the depth data.

An essential geometric feature of a point cloud is the computed normal for each point in the 3D data. This lies at the basis of several methods used for object recognition [34], pose estimation [14] segmentation [10], mesh generation [18, 31] or ray-tracing [5] thus the runtime and robustness of this preliminary processing step is relevant.

The normal estimation challenges are mainly due to the varying sampling density of a point cloud as well as the noise corrupting the measurement or representation of the environment. A standard method for normal estimation includes the selection of a support size for the patch attached to the point, the estimation of the geometric properties of the patch (e.g. the parameters of the fitted plane) and the computation of the normal vector concerning the estimated patch parameters [20]. Although this method is runtime efficient, the choice of the point neighbour patch size is non-trivial: small scale patches will reflect the fine grade characteristics of the objects, being prone to measurement artifacts while larger support size has better robustness for noise, but fails to encode the small scale details. This suggests an adaptive method for the normal support size, for which many methods were proposed including conventional [16] and CNN based algorithms as well [4]. The major drawback of these algorithms is their runtime, i.e. they fail to run in real time setups.

In this work, we propose an efficient normal estimation method for ToF depth cameras, which is able to provide robust normal estimation similar to the adaptive algorithms in the state of the art, with almost real time runtime characteristics on embedded platforms. Inspired by the multi-scale pyramid approach of the feature pyramid networks (FPN) [25], we propose an encoding on different levels of the input ToF depth images. For computing the point cloud normal, we lift at the training phase the 2D depth image points in the 3D metric space and compare it with the reference normal. Based on the ground truth normal, the algorithm can successfully encode how the reference was generated from single scale, multi-scale or adaptive methods, as well.

The contribution of this paper is summarized as follows: 1) We propose a novel feature pyramid network-based architecture for the point cloud normal estimation, which can encode multi-scale normal estimation inheriting from the training data; 2) We solve the normal estimation problem at the training phase for the ToF depth images by lifting the 2D points in the 3D space, thus directly operating on point clouds normal; 3) We perform extensive testing of the algorithm based on publicly available datasets as well as custom data. Our baseline algorithm is generic enough to be augmented with 2D information either from IR or RGB images, thus yielding to a multi-modal normal estimation approach.

2. Related work

The normal estimation of the point clouds has a long history in the research community, with publications on this topic dating back several years [20]. It is still a hot topic with a considerable amount of publications in the last year focusing on learning based methods [24, 38, 26, 36, 3]. We review the normal estimation methods grouping them into traditional and learning-based techniques as well as multi-modal approaches.

2.1. Traditional normal estimation

The classical approach for normal estimation is based on Principal Component Analysis (PCA) [20], which considers a point neighbourhood with a certain size (support size) and by analyzing the covariance in this local patch defines the normal vectors as the eigenvector corresponding to the smallest eigenvalue. Starting from this idea, derived works analyzed the effect of support size, curvature, sampling density and noise effect on the normal estimation, [9, 16, 6] which often require additional computational steps, thus increasing the overall complexity [22].

Other approaches make use of the Voronoi cells of the 3D points [28, 13, 12]. Although these methods can encode the sharp feature, they are prone to errors. To handle this problem, the PCA based Voronoi approach proved to be a stable solution [1]. Multi-scale approaches also proved to be robust to density noise corrupted point clouds [27, 37] while Hough transforms also yield promising results [7].

Robust results were achieved based on the assumption that the point neighbour supports can be considered planar patches, especially for sharp edges in point clouds [2, 11].

2.2. Learning-based methods

The progress in the field of normal estimation has been platooning until the appearance of the learning-based approaches, which showed that a considerable enhancement can be achieved by using recent deep learning-based techniques [21, 24, 8, 35, 38, 17, 26, 4]. Even though the deep learning-based methods require large amounts of training data, or may be prone to adversarial attacks or have exhaustive runtime, their popularity is still growing. These methods can also be referred as adaptive methods.

The first approaches made use of 2D images either from the point cloud projection or the depth images from the ToF cameras. The main advantage of these solutions is in the straightforward use of 2D convolutional operators. The ordered point cloud or voxel-based approaches proved to be computationally expensive, the complexity of these variants growing fast with the number of points from the point cloud.

The second class of approaches focuses on unstructured point clouds and after the appearance of the point-set approach proposed in [32], this was reused for normal estimation in the works of [17, 21] by introducing a multi-scale set oriented approach for normal estimation. In [3] surface fitting approach based on normal estimation is considered. Subsequently, the multi-scale approach combined with Fischer vectors was proposed in [4] combined with multiple expert networks for normal estimation.

2.3. Multi-modal approaches

The multi-modal approaches make use of heterogeneous data including radiometric and spectral input, as well. A good overview of the recent multi-modal data processing including normal estimation can be found in [15]. Even though in our approach we focus on the depth images as a primary input source for the normal estimation, we augmented our baseline solution with the IR images from the ToF cameras as well, thus yielding to a multi-modal configuration.

The method presented in [4] is the closest to our approach, featuring the supervised 3D normal learning and the work by [25] as the architecture of the input network is based on the former but adopted for the normal estimation task, while the most competitive in terms of the runtime is the solution proposed in [24].

3. Our approach

The key insight in our approach was to combine the multi-scale capturing abilities of the feature pyramid networks (FPN) [25] with the normal estimation for the ToF cameras with known intrinsic parameters. Due to the specific depth measurement of these cameras, the depth information is stored in 2D depth images, which can be easily projected into the 3D space yielding an organized point cloud. Even though there is no one to one mapping between the different pyramid levels and the normal support sizes in the point cloud, these variable scales contribute to a multi-scale normal estimation in the reprojected 3D space. This intuition is supported by how the compact surfaces are measured with ToF cameras: they are likely to be represented as compact patches in the depth image. According to our knowledge, the FPN based ToF specific depth measurements were not yet treated in the main literature, the major advantage of the proposed method being the scalability and low runtime.

The pipeline of our approach is shown in Fig. 1 containing the following parts: the input 2D ToF depth image is stored on different channels for the FPN (either containing only the depth images multiple times or having other information such as infrared on other channels), the ground truth normal estimations is encoded in the RGB space for the training phase and converted to 3D normal vectors for the loss computation; point cloud with normal generated in the evaluation phase. In this way, we operated directly on the 2D depth images and perform the normal estimation in a 3D space.

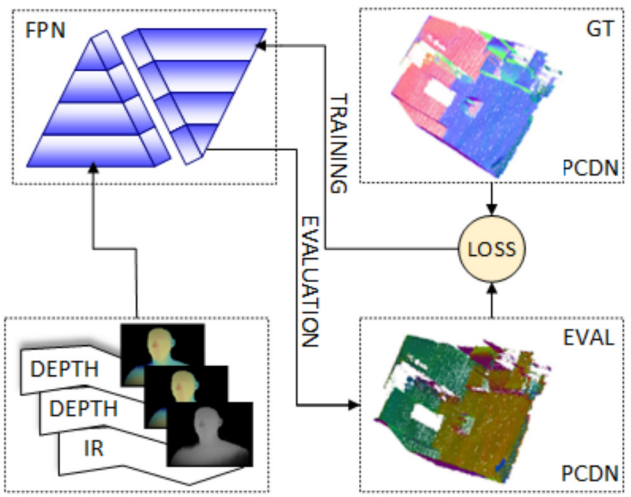


Figure 1: The training of the ToFNest algorithm: the information is encoded in depth and if necessary in IR information at the input; the custom FPN detailed in Fig 2; the FPN returns the prediction about the normals (Evaluation); the specific loss function compares it to the ground truth (GT) encoded in RGB space; The loss is then fed back to train the model (Training).

The encoding of the normal vectors into RGB space was straight forward. A vector has three coordinates: x, y, z. Their values range between $[-1, 1]$; after normalization, we can put these values in the range of $\{0, 255\}$. Here we might think that we are losing precision, since we are converting three 32 bit numbers into three 8 bit numbers. It might seem that 256 values are not enough to represent 360 degrees, but the channels are not independent, and they are working together. This means that the resolution is still considerably below 1 degree, and because most of the errors in the methods are at least 10 degrees, we can consider this type of loss negligible.

The details of the custom FPN are summarized as follows.

3.1. ToFNest architecture details

The input size of this network can be arbitrary (depending on the internal parameters of the ToF camera) and the output results in a proportionally sized normal feature map with the input image using a fully convolutional approach. The approach is generic in the sense that for the convolutional architecture custom variants can be adopted. In our solution, we adopted the ResNets[19] as the backbone with pre-trained weights from ImageNet for the initialization phase.

The construction of these pyramids involves a bottom-up and top-down path with lateral connections [25] as follows.

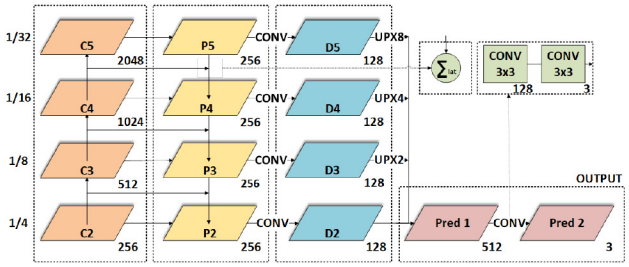


Figure 2: The connection diagram among the layers of the custom FPN used for normal estimation: bottom-up path (orange), top-down layers (yellow), fused feature map with lateral connections (blue), two consecutive conv operators(green) for the top layer and the final prediction-upsample part (red).

3.1.1 Bottom-up path

This path consists of the feed-forward computation of the convolutional network combining several feature maps from different levels. For the upsampling, we use a pixel-shuffle with a bi-linear interpolation to deal with inconsistent feature map size. The output of a layer is considered to be input for the next layer. In our specific case with the ResNet backbone, we use the feature activation output from the last stage residual block, which in our case is denoted with $\mathcal{C}_i$ from the i^{th} layer. In contrast with [25] we reuse the 0 layer as well to ensure a proper resolution at the output normal image, with the strides of $\{2, 4, 8\}$ between the layers. ReLU was used for these layers in order to achieve a stable output.

3.1.2 Top-down path

The top-down way mimics higher resolution features by upsampling the sparser but more descriptive feature maps from the upper layers denoted with $\mathcal{P}_i$. The lateral connections enforce these features including information from the bottom-up path as well using an upsampling addition operator. The connection among the different paths is shown in Fig. 2

The operator between the last two layers contains two consecutive $3x3$ convolutional layers for the final feature map processing followed by a Sigmoid activation function.

3.1.3 Lateral connections

The connection from the $\mathcal{P}_i$ layers are ensured through a $1x1$ convolution with a 1 stride and by element-wise addition. As the operations on the vertical level are performed on traditional feature design, the dimensions of the two branches match and are also inherited in the lateral layers from which the predicted $\mathcal{D}_i$ layers are generated. Obviously, more advanced architectures can be configured, but

we rather focused on an efficient solution also from runtime perspectives.

The predicted layer $Pred_1$ summarizes the contribution of the individual $\mathcal{D}_i$ layers and the $Pred_2$ upsamples the required output resolution (in our case being identical with the input depth image resolution).

3.2. Normal loss function

For the normal loss function, we investigated several variants including gradient, curvature loss, or the direct normal difference lost. The latter seemed to be the most efficient in terms of the normal estimation robustness. Starting from the normalized loss computation [4], we adopted the following loss function:

$$L_{normal} = \frac{1}{M} \sum_{1}^{M} \left(\frac{\|N_i \times N_{GT}\|}{\|N_i\| \cdot \|N_{GT}\|} \right) \tag{1}$$

where M is the number of points in the point cloud, N_i is the normal estimate and N_{GT} is the ground truth normal value.

In the evaluation phase of the proposed architecture, we used the absolute error angle between the ground truth and the predicted normal.

4. Evaluation

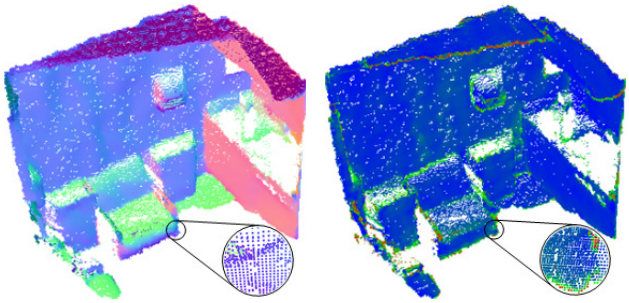
For the evaluation of the proposed method, we conducted a large scale test. We compared our method in terms of normal robustness and runtime against traditional and learning-based methods as well. We also investigated the effect of noise on the depth data, which is a common problem for depth measurements, especially for the outdoor ToF camera images.

4.1. Dataset used for evaluation

As public reference dataset, we considered the indoor NYU_V2 one [29] with normals based on [23] as ground truth, because comparing it to [24] resulted in smoother normal estimates. While for custom testing we created indoor and outdoor data with RGB-D data from ADI ToF camera with ground truth data from generated with a multiscale PCL [33]. We would mention that these two datasets were generated with a camera and not from a simulated model, which resulted in realistic dataset.

4.2. Performance evaluation and comparison

To encode efficiently the normal information for the training, we encoded the normal vectors into the RGB space which we, subsequently, converted at the output to point clouds with the normal vectors. These normal vectors then were compared to the normal vectors of the ground truth point clouds in terms of absolute normal orientation errors.



(a) $Input$ encoded in the RGB space (b) $Output$ of the estimation error distribution

Figure 3: The result obtained from our method encoded as a heat map for the absolute normal orientation error

All of these methods returned un-oriented normal vectors, i.e. we considered the orientations vectors the same for the flipped ones. At the output visualization of our estimate, we considered the heat map of the absolute error of the normal orientation, such as this is visible in Fig. 3. On this image, we can also see that the predicted normals seem to be slightly blurred at the edges. This might come from the fact that although the FPN works with many different layer sizes, it is not capable of per-pixel normal estimation, which means that in some cases it has a smoothing effect, especially at the edges.

For training purposes, we augmented the original NYU_V2 dataset containing 1449 depth images with simple horizontal and vertical flips as well as adding some Gaussian noise to the depth images. With this augmentation, we managed to cover the surface orientations, which are rarely present in the indoor scenes, thus yielding to more generic training. For the training dataset, we used more than $7.5K$ images, with an average point cloud size of $200K$ per depth image, while we used about $3.5K$ for testing and the original 1449 images as the evaluation dataset.

For the performance evaluation, we considered as primary metrics the mean absolute difference of the angle errors in deg between the estimated and the ground truth normal as well as the average histogram of the test cases computed in percentage as the dot product between the two vectors. Beside the normal estimation quality, we were also interested in the runtime of the algorithms, as this is a major criterion in the processing chain of several methods relying on normal estimation.

Based on these criteria, using the public dataset, we compared our method against the Nesti-Net [4], PCPNet with single and multiple scales [17], the single threaded normal estimation used in the Point Cloud Library (PCL) [33] as well as the Hough transform based normal estimation [7], and at last the Lessen et. al [24] method. In order for the PCL comparison to be fair, we considered the average of

Table 1: Summary of the comparison with other methods

Comparison between the normal estimation methods							
	Own	Nesti-Net [4]	PCPNet ss [17]	PCPNet ms [17]	PCL [33]	Hough [7]	Lenssen [24]
Avg. hist. [%]	0.922	0.913	0.911	**0.923**	0.914	0.829	0.897
Abs. angle [deg]	22.78	24.08	24.36	**22.52**	23.91	33.95	26.23
Avg. runtime [s]	**0.015**	1200	234	596	7.09	2.7	19.4

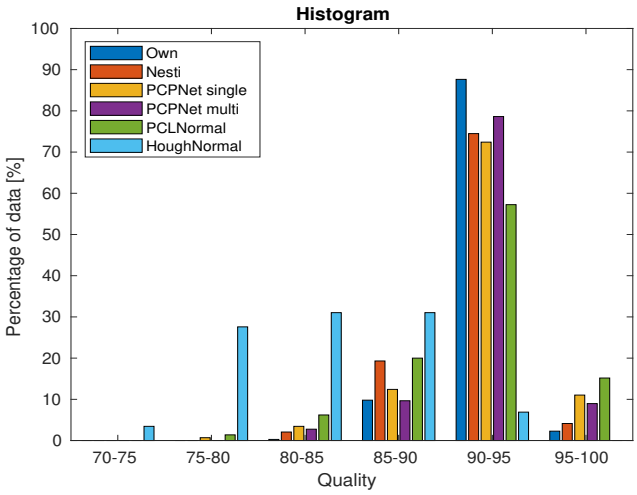
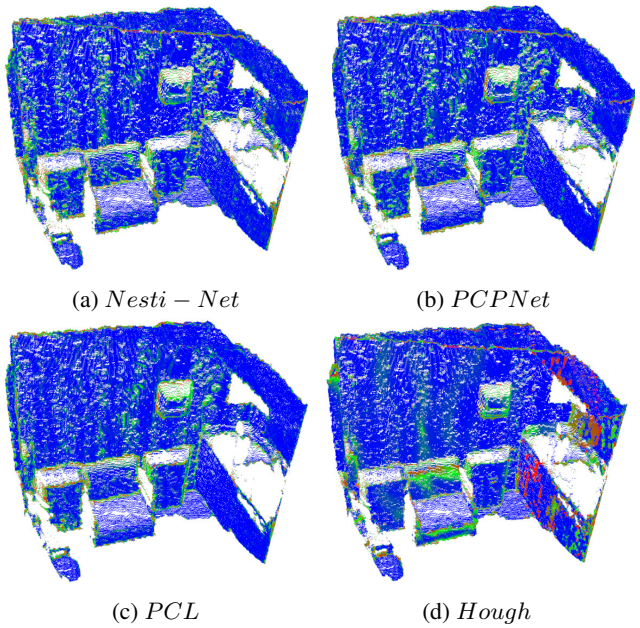


Figure 4: The histogram of the normal estimation for different methods



(a) $Nesti - Net$ (b) $PCPNet$

(c) PCL (d) $Hough$

Figure 5: Comparison of the output of the methods on the same data

the error from different support sizes used for normal estimation (from 2 to 10 centimeters, regarding the accuracy, the 4 centimeters support size was the closest to the average). The histogram dispersion of the quality indicator for the tested methods is presented in Fig. 4.

The results of the comparison are summarized in Table 1. The best normal estimation performance measured as mean angle error was achieved by the multi-scale PCPNet, our approach was with almost the same range of absolute error but featured a runtime that was orders of magnitudes smaller. The Nesti-Net was in the same range of absolute error, however this method had far the longest evaluation time required for an estimate. In terms of runtime, the PCL and Hough methods were close to our algorithm running on CPU. Although, for PCL there are other variants, the multi threaded PCL is computed in about 1 second.

We also tested the Integral Image method from PCL, and although it was about as fast as our method, it produced poor results only around 0.82. A visual comparison of a typical output for other methods tested in our experiments is visible in Fig. 5, for the same pointcloud visualized in Fig. 3.

4.3. Performance evaluation on noisy data

To evaluate the robustness of ToFNest, we tested the normal estimation in presence with different levels of Gaussian noise corrupting the depth data. In order to do so, we generated the noisy data from the depth images with additional Gaussian noise, the covariance ranging from $1cm - 10cm$ and compared it against the normal ground truth from the clean data.

The comparison in terms of average histogram for the different methods is shown in Fig. 6. The best performance against the noise robustness was achieved by ToFNest followed by the PCPNet, while the most affected method was the one based on Hough transforms.

4.4. Runtime performance evaluation on different platforms

As for many applications, the runtime is relevant; we tested the performance of our method on different platforms ranging from embedded devices to cloud servers. For the embedded devices we considered the Jetson family from

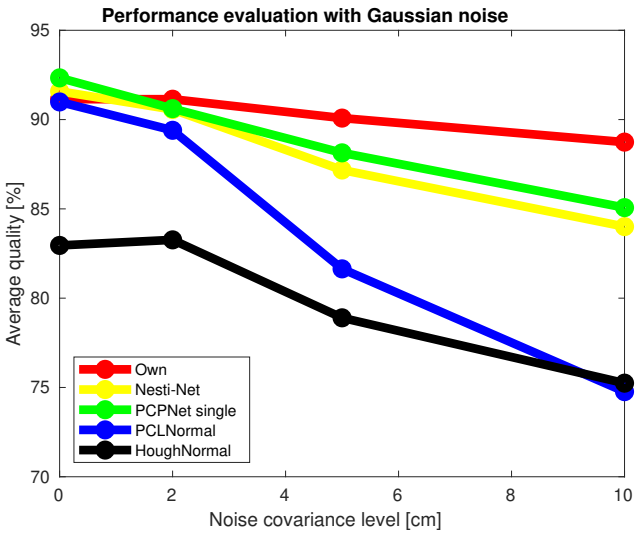


Figure 6: The effect of noisy data for different normal estimation methods

Nvidia including the NX and AGX variants, while for the cloud solution we used the Google Colab. As a baseline, we considered a commercial grade RTX 3080 GPU enabled PC.

Table 2: Summary of the runtime comparison for different devices with our method

Runtime comparison on different platorms					
Device	RTX 3080	Jetson NX	Jetson AGX	GTX 1060	Colab
Time [s]	0.04	0.31	0.234	0.047	0.11

The results of the comparison are summarized in Tab. 2. As it can be seen in this summary, our method runs with 4 fps even on embedded devices, thus yielding an efficient runtime solution also for mobile robot applications.

4.5. Performance evaluation on custom data

For our custom training and testing we considered two datasets. We created the datasets using a Pico Zense DCAM710 which is built on the ADDI9036 CCD TOF Signal Processor. Using this camera we could record the infrared and RGB images along with the depth images, as well. We recorded several types of environments including office, hallway and laboratory workspace. We have also augmented these images acquiring about $10K$ images that were split into training and testing datasets, about 66-33 rate.

For the outdoor dataset, we acquired with the same camera but with tuned noise reduction parameters images in normal daylight conditions. As ground truth for these datasets, we used manually scale tuned PCL method, nevertheless this can be replaced with an arbitrary method. We

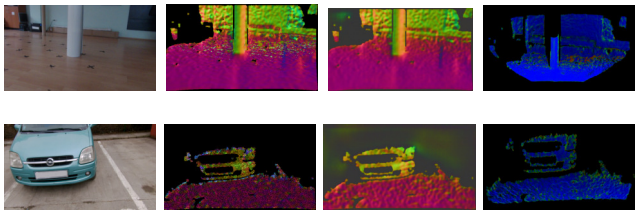


Figure 7: Indoor / Outdoor results on custom data

considered PCL because it runs faster, and tuning the support size it can give us fair results. This tuning is feasible in this case, but if we wanted to use this method it might not be valid.

Typical results from the indoor and outdoor scenes as well as the normal estimate and difference heat maps are visible in Fig. 7. The results of the indoor and outdoor evaluations are summarized in Tab. 3, with better results in the indoor data as the outdoor dataset was more affected by measurement artifacts. Also, we mentioned previously that our model tends to smooth the prediction, but as we have seen, it is not that bad. This would be the base for future point cloud smoothing methods.

Table 3: Summary of the custom Indoor/Outdoor evaluation

Custom dataset performance		
	Indoor	Outdoor
Avg. hist. [%]	0.959	0.952
Abs. angle [deg]	16.46	17.82

4.6. Cross validation

For further testing purposes, we used the model which was trained with the NYU_V2 dataset and applied it to our dataset. Here the average quality was 0.901, while with the native model the average quality was 0.959. This can happen because the camera that we used was different; however if we compare this result to the NYU_V2 evaluation (avg. quality is 0.915), we can consider it acceptable.

In addition to this, we created in a simulated Blender model about one of our laboratory (where we made the indoor real dataset), and with the help of Isaac Sim [30] which is a simulation library in Omniverse created by Nvidia, we created a synthetic dataset in this scene. Testing this dataset with the model trained on the NYU_V2 dataset, we got 0.96 % avg. quality. But if we test the NYU_V2 dataset with the model trained on the synthetic data, we got only 0.8. The performance drop can be due to the differences of the simulated render in the Isaac Sim, which returned near-perfect depth images.

4.7. Training details

To be fair with the other learning-based training methods used in our comparison, we usually trained our models for 10 epochs, featuring a learning rate starting from 1e-3 with the SGD optimizer. In every epoch, the model evaluated all the data from the dataset, and after each batch a loss was calculated. Our method was implemented using PyThorch, the code being publicly available on github. The baseline comparisons were done using an Nvidia RTX 3080 GPU, and an Intel i9-10900K with 64GB RAM commercial PC.

4.8. Generalization to multi-modal data

The method itself is generic to accept additional information besides the depth image: 3 layers are included as input, for which 1 depth is used for pointcloud normal estimation, while the two additional layers enable to encode camera specific information such as IR intensity or color data, as well. According to our findings, these influence on limited manners the normal estimation, but allow the extension of the method.

5. Summary

In this work, we presented a runtime efficient and robust learning-based normal estimation method for the depth images acquired by ToF cameras. The method is based on the FPN architecture tailored to the depth image specific details with loss functions focusing on the absolute normal orientation difference. We evaluated our method against the traditional and learning based variants on large scale public indoor and custom outdoor data, our method being with similar normal estimation performance but with orders of magnitudes faster, running efficiently even on embedded devices. The method is generic enough for any central projective camera type featuring known intrinsic parameters returning radial information.

As future work, we intend to extend our normal estimation pipeline with point filtering variants such as statistical outlier removals. These ideas are built upon the precomputed normals for a point cloud, achieving in this way a runtime efficient preprocessing pipeline for ToF depth cameras.

Acknowledgments

The authors are thankful for the support of Analog Devices, for the equipment list (cameras, embedded devices) and for the embedded GPUs offered by Nvidia as support to this work. This work was financially supported by the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-III-P2-2.1-PTE-2019-0367 and also by Hungarian Research Fund grant number OTKA K 120367.

References

[1] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 39–48, Goslar, DEU, 2007. Eurographics Association. 2

[2] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. Sparse reconstruction of sharp point set surfaces. *ACM Trans. Graph.*, 29(5), Nov. 2010. 2

[3] Yizhak Ben-Shabat and Stephen Gould. DeepFit: 3D Surface Fitting via Neural Network Weighted Least Squares. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12346 LNCS:20–34, 2020. 2

[4] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2019. 1, 2, 4, 5

[5] Andreea Blaga, Cristian Militaru, Ady-Daniel Mezei, and Levente Tamas. Augmented reality integration into mes for connected workers. *Robotics and Computer-Integrated Manufacturing*, 68:102057, 2021. 1

[6] R. Bormann, J. Hampp, M. Hägele, and M. Vincze. Fast and accurate normal estimation by efficient 3d edge detection. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3930–3937, 2015. 2

[7] Alexandre Boulch and Renaud Marlet. Fast and robust normal estimation for point clouds with sharp features. *Computer Graphics Forum*, 31:1765–1774, 08 2012. 2, 4, 5

[8] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35, 2016. 2

[9] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 2

[10] Erzhuo Che and M. J. Olsen. Multi-scan segmentation of terrestrial laser scanning data based on normal variation analysis. *Isprs Journal of Photogrammetry and Remote Sensing*, 143:233–248, 2018. 1

[11] Honghua Chen, Jin Huang, Oussama Remil, Haoran Xie, Jing Qin, Yanwen Guo, Mingqiang Wei, and Jun Wang. Structure-guided shape-preserving mesh texture smoothing via joint low-rank matrix recovery. *Computer-Aided Design*, 115:122–134, 2019. 2

[12] Tamal K. Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. *Computational Geometry*, 35(1):124–141, 2006. Special Issue on the 20th ACM Symposium on Computational Geometry. 2

[13] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: a comparison study for a voronoi based method. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 39–46, 2005. 2

[14] R. Frohlich, L. Tamas, and Z. Kato. Absolute pose estimation of central cameras using planar regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):377–391, 2021. 1

[15] Mingliang Gao, Jun Jiang, Guofeng Zou, Vijay John, and Zheng Liu. RGB-D-Based Object Recognition Using Multimodal Convolutional Neural Networks: A Survey. *IEEE Access*, 7:43110–43136, 2019. 2

[16] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3):23–es, July 2007. 1, 2

[17] P. Guerrero, Yanir Kleiman, M. Ovsjanikov, and N. Mitra. Pcpnet learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37, 2018. 2, 4, 5

[18] Taisuke Hashimoto and M. Saito. Normal estimation for accurate 3d mesh reconstruction with point cloud model incorporating spatial structure. In *CVPR Workshops*, 2019. 1

[19] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[20] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, page 71–78, New York, NY, USA, 1992. Association for Computing Machinery. 1, 2

[21] Janghun Hyeon, Weonsuk Lee, Joo Hyung Kim, and Nakju Doh. Normnet: Point-wise normal estimation network for three-dimensional point cloud data. *International Journal of Advanced Robotic Systems*, 16(4):1729881419857532, 2019. 2

[22] K. Jordan and P. Mordohai. A quantitative evaluation of surface normal estimation in point clouds. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4220–4226, 2014. 2

[23] L'ubor Ladický, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):468–484, 2014. 4

[24] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Deep iterative surface normal estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. 2, 4, 5

[25] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 1, 2, 3

[26] Dening Lu, Xuequan Lu, Yangxing Sun, and J. Wang. Deep feature-preserving normal estimation for point cloud filtering. *Comput. Aided Des.*, 125:102860, 2020. 2

[27] C. Mura, G. Wyss, and R. Pajarola. Robust normal estimation in unstructured 3d point clouds by selective normal space exploration. *The Visual Computer*, 34:961–971, 2018. 2

[28] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, 2011. 2

[29] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 4

[30] NVIDIA. Nvidia isaac sim | nvidia developer, 2019. Available at: https://developer.nvidia.com/isaac-sim. 6

[31] Songyou Peng, C. Jiang, Yiyi Liao, Michael Niemeyer, M. Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *ArXiv*, abs/2106.03452, 2021. 1

[32] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[33] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 05 2011. 4, 5

[34] Levente Tamas and Bjoern Jensen. All-season 3d object recognition challenges. In *ICRA Workshop on Visual Place Recognition in Changing Environments*, 2014. 1

[35] X. Wang, David F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–547, 2015. 2

[36] Zirui Wang and Victor Adrian Prisacariu. Neighbourhood-insensitive point cloud normal estimation network. In *BMVC*, 2020. 2

[37] J. Zhang, J. Cao, X. Liu, He Chen, B. Li, and Ligang Liu. Multi-normal estimation via pair consistency voting. *IEEE Transactions on Visualization and Computer Graphics*, 25:1693–1706, 2019. 2

[38] Haoran Zhou, Honghua Chen, Yidan Feng, Qiong Wang, Jing Qin, Haoran Xie, Fu Lee Wang, M. Wei, and J. Wang. Geometry and learning co-supported normal estimation for unstructured point cloud. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13235–13244, 2020. 2