3D Object Recognition using Time of Flight Camera with Embedded GPU on Mobile Robots

Benjamin Kelényi, Szilárd Molnár and Levente Tamás[a]
Department of Automation, Technical University of Cluj-Napoca, Memorandumului St. 28, 400114 Cluj-Napoca, Romania

Keywords: Time of Flight, 3D Point Clouds, Mobile Robots, Embedded Devices, Depth Image Processing.

Abstract: The main goal of this work is to analyze the most suitable methods for segmenting and classifying 3D point clouds using embedded GPU for mobile robots. We review the current main approaches including, the point-based, voxel-based and point-voxel-based methods. We evaluated the selected algorithms on different publicly available datasets. Simultaneously, we created a novel architecture based on point-voxel CNN architecture that combines depth imaging with IR. This architecture was designed particularly for pulse-based Time of Flight (ToF) cameras and the primary algorithm's target being embedded devices. We tested the proposed algorithm on custom indoor/outdoor and public datasets, using different camera vendors.

1 INTRODUCTION

With the appearance of the compact radiometric sensors the perception for the mobile robotic applications passed through a paradigm shift: from 2D image-based sensing the direct depth information was available. Besides the already existing 3D laser scanners, the consumer level depth cameras such as Kinect, Asus Xtion or RealSense allowed the integration of these depth sensors on mobile platforms providing rich geometric, shape and scale information. By the fusion of spectral information from the 2D cameras with the point-wise depth ensured an environment perception for a large spectrum of applications including mapping (Tamas and Goron, 2014), object detection (Tamas and Cozma, 2021), augmented reality (Blaga et al., 2021) or pose estimation (Frohlich et al., 2021)

Due to the discrete point characteristics of the 3D data acquired from the sensors, this is the preferred representation method in favour of mesh, parametric or voxel spatial representation. Recently, with the evolving of the deep learning techniques for 2D images, or natural language processing the attention of these methods shifted towards the 3D domain as well (Tamas and Jensen, 2014). Although the implementation of the discrete convolution over the 3D space is demanding due to its high dimensionality and having a limited number of public datasets compared to the 2D domain, the deep learning-based solutions for the 3D data are still in research focus today (Fooladgar and Kasaei, 2020). Furthermore, the development of embedded or real-time solutions for the point-cloud processing (Oleksiienko and Iosifidis, 2021) using these techniques is still in progress. With the evolution of the embedded GPU processing power, early phase solutions exist for real-time applications for point-clouds. The main focus of this paper is the evaluation of the existing methods for the feasibility of training and evaluation of custom point-cloud data-based object recognition on embedded GPU platforms.

In this work, we focused on the Time of Flight (ToF) depth cameras, more specifically the pulse-based variants returning beside the discrete depth points the infrared reflectively as well of the measured surface. We analyzed several methods suitable for embedded GPU processing and based on public and our custom dataset we summarized our findings related to the specific 3D object recognition task. From the publicly available datasets, we considered the NYU (Nathan Silberman and Fergus, 2012), S3DIS (Armeni et al., 2017), and Shapenet (Yi et al., 2016) variants. For the own dataset, we created a bunch of recordings from different depth camera vendors including Microsoft, Intel, Analog Devices, and Asus.

The methods selected for analysis are point-based variants, and the major criteria were that they should be usable also on recent embedded GPU platforms

[a] https://orcid.org/0000-0002-8583-8296

Kelényi, B., Molnár, S. and Tamás, L.
3D Object Recognition using Time of Flight Camera with Embedded GPU on Mobile Robots.
DOI: 10.5220/0010972200003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, pages 849-856
ISBN: 978-989-758-555-5; ISSN: 2184-4321

targeting mobile robot applications. As of the current state of the art (Shi et al., 2021; Le and Duan, 2018; Jiao and Yin, 2020), our selection was on the point-based methods, the fastest runtime variants being in this category. Based on these criteria, our selection resulted in the PointNet (Qi et al., 2017a), Point Transformer (Zhao et al., 2021), and mixed Point-voxel CNN (Liu et al., 2019) methods.

Besides the choice of these three algorithms, we extended the Point-voxel CNN method with pulse-based Time of Flight (ToF) camera-specific IR image data to have a better recognition performance on embedded GPU platforms.

The extensive algorithm testing on public datasets as well as our real dataset was performed on different platforms ranging from high-end desktop GPUs through cloud-based variants to embedded GPUs. Also, we investigated for the mentioned algorithms the variance of performance concerning different types of recent commercial-grade depth cameras, as the characteristics returned by these devices heavily influence the recognition pipeline.

The contribution of this paper is summarized as follows: 1) overview and selection of embedded deep learning-based point cloud processing algorithms; 2) customization of the PV-CNN with ToF specific IR data for better accuracy; 3) extensive camera-specific algorithm evaluation for object recognition purposes and validation on custom indoor-outdoor datasets using different GPU architectures ranging from embedded devices to cloud solutions. The paper is organized as follows: in section 2 we shortly describe the state of the art for the point-cloud-based object recognition approaches with the focus on the embedded devices. Following this in 3, we present our customized PV-CNN (Liu et al., 2019) with combined depth and IR images. Next we summarize our finding with different types of ToF cameras for this algorithm in 4.1. Finally, in section 4.2 we present our results on the comparison of different embedded object recognition algorithms working with point-clouds.

2 RELATED WORK

In this section, we make an overview of the depth information processing in the era of deep learning with a special focus on real-time operation candidates for embedded platforms. Next, we shortly present the three methods which we considered for our test bunch on embedded platforms.

2.1 Traditional Methods

Traditional point cloud recognition methods were built on keypoint-feature pairs. Usually, these low-level semantic data about the objects of interest were used in statistical or kernel-based clustering methods in order to match in a template-observation pair the correspondences. For the consumer level ToF cameras, good overview of the available feature-based methods can be found in (Tamas and Jensen, 2014).

2.2 Multi-modal Approaches

The multi-modal approaches make use of heterogeneous data including radiometric and spectral input as well. This fusion can be done with a relative pose estimation between different camera modalities (Frohlich et al., 2021). A good overview of the recent multi-modal object recognition can be found in (Ophoff et al., 2019) while in (Jiao and Yin, 2020) a ToF camera focused two phase overview is presented.

2.3 Learning-based Representation for Embedded Processing

The main focus of this paper is based on the 3 methods (PointNet, Point Transformer, and Point-Voxel CNN) thus we describe each approach independently and summarize the algorithm's performance as follows.

2.3.1 PointNet (Qi et al., 2017a)

Typical convolutional architectures requires input data to be in a regular format, such as images and also temporal features. However, point clouds aren't in a regular form, we must to organize the point clouds into certain data structures.

The most popular solution is to convert the data to a 3D voxel grid (Xu et al., 2021). Another solution is to convert the point cloud to 2D projections which creates spatial consistency in the point set, allowing the usage of the convolution operators.

PointNet (Qi et al., 2017a) is the first method of a new form of neural network that employs unordered point clouds directly. The permutation invariance of points in the point cloud is also addressed. As shown in Fig. 1, PointNet can perform object classification, partial segmentation and scene semantic analysis.

The classification network maps each of the input points from three dimensions (x, y and z) to 64 dimensions using a shared MLP. Repeating this procedure the points from 64 dimensions are mapped

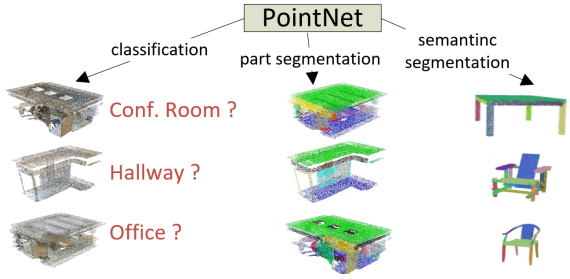

Figure 1: Typical applications of PointNet (Qi et al., 2017a).

to 1024 dimensions. After that, with the points obtained from the previous step max pooling is used to generate a global feature vector in $\mathbb{R}^{1024}$. Finally, a fully-connected network is used to map the global feature vector to predict the scores for all the candidate classes.

In the segmentation network, each of the n input points must be allocated to one of the m segmentation classes. The points from 64 dimension are combined with the global feature vector. Concatenating the points from the 64 dimensional embedding space with the global feature vector, a vector for each points in $\mathbb{R}^{1088}$ will be obtained. MLPs are employed on the n points, as in a classification network, to reduce the dimensionality from 1088 to 128 and then to m. The results will be an array of $n \times m$ vector. This $n \times m$ vector represents the scores for each of then n points and each of the m semantic subcategories.

The main characteristic of PointNet is that the network is robust in terms of input disturbance. In addition, the network can learn to summarize the shape through a sparse set of key points, usually with a fixed point-width.

PointNet uses an operation that is symmetrical to its input, that is, the arrangement in the 3D point set is irrelevant. These operations include fully connected layers that share weights among all points and global maximum pooling.

The main disadvantage of using PointNet is that it does not directly utilize the local structure of the data, which makes learning more difficult. This shortcoming was overcome in the method developed in the work (Qi et al., 2017b).

2.3.2 Point Transformer (Zhao et al., 2021)

Many methods have been recently proposed to make better use of the local structure, typically by sampling a subset of representative points, using k-nearest neighbors to create a group around each point in the subset and then using a PointNet-like approach on each group. In the final step, these groups are mapped into local representations, which are then merged into global feature vectors through additional layers.

The work of (Zhao et al., 2021) offers a new method, utilizing the neural attention mechanism, which is already well-established in Natural Language Processing and is rapidly expanding to other fields of machine learning. The authors employ vector attention (Ioffe and Szegedy, 2015), a recent solution that is an extension of the traditional scalar attention approach.

The Point Transformer network used for the **segmentation** task follows the U-Net (Ronneberger et al., 2015) structure, consisting of 5 down-sampling point attention layers, 5 similar up-sampling layers and a final MLP layer, with a residual connection between the up and down layers.. The down-sampling part manages the **classification** task by using a global average pooling layer and an MLP layer.

2.3.3 Point-Voxel CNN (Liu et al., 2019)

The point-based networks are extensively used. However, as a result of the lack of explicit neighborhood information in point representations, most existing point-based techniques rely on time-consuming neighbor-finding algorithms.

Point-Voxel CNN proposed an approach that takes advantage of both point and voxel representation to achieve memory and computational efficiency at the same time.

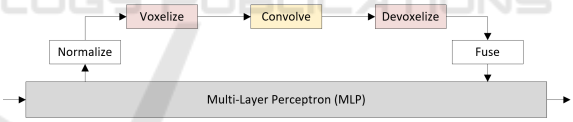

Figure 2: Point-Voxel CNN Architecture.

Point-Voxel CNN is based on a computationally efficient design, which is shown in Fig. 2. As one can see, the architecture is divided into two parts. The top part is voxel-based feature aggregation and the bottom part is Point-Based feature transformation. The upper branch (voxel-based) converts points into a low-resolution voxel grid, then combines nearby points using voxel-based convolutions, and finally devoxelizes them to points. The memory footprint is small since voxelization or devoxelization both need a single scan across all locations. It can afford a very high resolution since it does not combine the information of its neighbors.

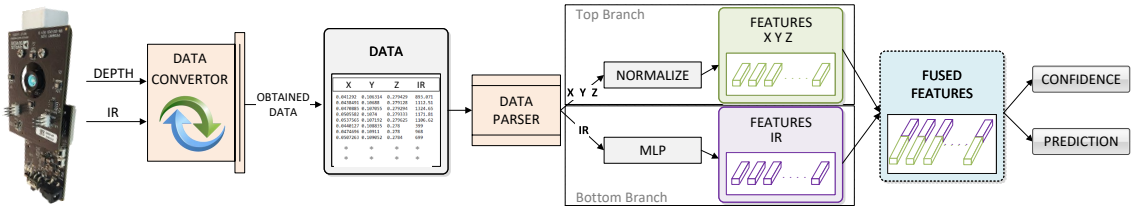

Figure 3: Proposed Depth-IR CNN Architecture based on PV-CNN using pulse-based ToF camera.

3 OUR APPROACH

The voxel-based representation, according to our findings, is regular and has high memory locality. However, in order to avoid losing information, it requires a high resolution. Multiple points are compacted into the same voxel grid when the resolution is poor, and these points are no longer distinct. Because our main focus is on Time of Flight (ToF) cameras and the images received from these cameras are usually ordered points, we bypassed the voxelization and devoxelization aspects of our method based on PV-CNN. Simultaneously, we incorporate the IR field from the camera in order to enhance the model of the original PV-CNN.

The architecture provided in the PV-CNN (Liu et al., 2019) paper provides the foundation for our implementation. We were able to create a new architecture for Time of Flight cameras by altering the old architecture.

3.1 Architecture Details

Fig. 3 depicts our architectural design. The camera picture is obtained in the first step. The depth and infrared (IR) images are then recorded in a vector of arrays. Each point in the vector is represented by an array. The location coordinates (x, y, z) of each point are represented by the first three positions of the array, while the IR value is stored in the last position. The data from the vector are passed on to the two branches using a data parser. The top branch deals with depth image processing, whereas the bottom one extracts features, in this instance the IR picture. Finally, the two feature vectors derived from the two branches are merged together to calculate the confidence and prediction for the desired object.

3.2 Depth-IR Aggregation

Because our design is split into two branches for Depth and IR images, feature aggregation is done using a stack of 3D volumetric convolutions. After each 3D convolution, we apply batch normalization (Ioffe and Szegedy, 2015) and the nonlinear activation function (Maas et al., 2013), much like in traditional 3D models.

3.2.1 Multi-layer Perceptron

is applied to the input layer to extract attributes. The generated MLP operates on each point to extract its properties. There are 2048 points in a batch with 64-channel features (with batch size of 16). We explore combining data from each point's 125 neighbors and then transforming the resulting feature to obtain features of the same size.

3.2.2 Normalization

Different point clouds might have very different scales. As a result, before translating the point cloud to the volumetric domain, we normalize the coordinates $\{p\}$. To begin, we convert all points to a local coordinate system, with the gravity center as the origin. The points are then normalized by dividing all coordinates with $max||p||_2$ and then scaled to [0, 1].

3.2.3 Feature Fusion

the output of Depth and IR image processing branches have the same size and data types. Because the data is of the same kind, merging the two can be done adding them together.

4 EXPERIMENTS

4.1 Camera Comparison for Recognition

For the real-time demo, we had a choice of 4 cameras. Using RealSense D435i, Azure Kinect DK, Pico Zense DCAM710 and Asus Xtion PRO depth cameras we propose, a side-by-side geometric shapes recognition for these four ToF depth cameras. Using the ModelNet40 (Wu et al., 2015) dataset based pre-trained model, our goal was to compare the cameras

Table 1: Side-by-side comparison between confusion matrix for different cameras.

	Asus Xtion Pro				Azure Kinect DK				Pico Zense				RealSense D435i			
	cylinder	sphere	v. plane	h. plane	cylinder	sphere	v. plane	h. plane	cylinder	sphere	v. plane	h. plane	cylinder	sphere	v. plane	h. plane
cylinder	**0.94**	0.01	0.04	0.01	**0.86**	0.11	0.03	0.01	**0.88**	0.10	0.01	0.01	**0.90**	0.04	0.04	0.05
sphere	0.17	**0.76**	0.05	0.02	0.05	**0.94**	0.02	0.00	0.19	**0.79**	0.01	0.01	**0.41**	0.31	0.17	0.11
v. plane	0.01	0.00	**0.94**	0.04	0.00	0.00	**0.94**	0.06	0.05	0.00	**0.92**	0.03	0.17	0.01	**0.67**	0.15
h. plane	0.00	0.03	0.00	**0.97**	0.00	0.00	0.01	**0.99**	0.00	0.00	0.01	**0.99**	0.01	0.00	0.01	**0.97**

with the same recognition algorithms, on the same environment, same dataset and same lighting conditions, in this way ensuring that the comparisons were appropriate. The main idea is to create a more realistic comparison so that the camera can be used for real-time evaluation using embedded GPU's such as Jetson AGX.

4.1.1 Camera Comparison Method

In this section is presented our ground truth generation for the recognition pipeline based on geometric primitive extraction from point-clouds.

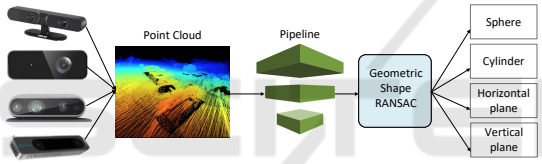

Figure 4: Our dataset processing pipeline.

The Fig. 4 shows how our pipeline works. In the first phase, the data acquisition was done by mounting the test cameras on a robot. The route of the robot was recorded, so we can assume that the comparison was made on the same dataset. After, storing the camera recording in ROS bag type file, the message conversion in point-cloud was performed offline using the PCL-ROS library.

4.1.2 Camera Comparison Experiments Details

In order to make our algorithm as generic as possible, we decided to create a configuration file. Everything in this configuration file may be modified dynamically, from the voxel filter settings to the tuning parameters like the diameter of the sphere or cylinder. For our RealSense D435i camera, we used as parameters for the voxel filter the value 0.02, the maximum circumference of the cylinder = 1, the minimum circumference of the cylinder = 0.3, the maximum number of iterations of the cylinder = 100000. With these settings, good results were obtained for the whole dataset from different cameras. The next step is to use the PointNet (Qi et al., 2017a) algorithm to recognize the following regular geometric shapes: cylinder, sphere, horizontal plane and vertical plane. The last phase of the process consists of training and validating the dataset.

To highlight the differences among the test cameras, we present the behavior of each camera on our dataset. The confusion matrix is designed for each camera to better visualization of the performance of our algorithm. Also, a comparison between results was made. In this way, we can get the conclusion, which camera is reliable for our algorithm.

4.1.3 Camera Comparison Results

Table 1 shows a side-by-side compassion between cameras by confusion matrix. Each row of the matrix represents the value in a predicted class while each column represents the value in an actual class. The correct object predictions are represented by the major diagonal of each confusion matrix. The average accuracy of the predictions is calculated by averaging the diagonals. Thus, the following values are obtained for the four cameras ordered by the head of the Table 1: 0.9025, 0.9325, 0.895, 0.7125. As a consequence, the following four cameras are ranked in order with the configs:

1. Azure Kinect DK - (640x576, range: 0.5 - 3.86m)
2. Asus Xtion PRO - (640x480, range: 0.8m - 3.5m)
3. Pico Zense 710 - (640x480, range: 0.2m - 8m)
4. RealSenseD435i - (1280×720, range: 0.3 - 3m)

4.2 Evaluation

To accomplish a large-scale examination, we compared the presented methods (PointNet, Point Transformer and Point-Voxel CNN) on a variety of datasets as well as devices. We trained and evaluated these methods to perform 3D object classification.

The model was trained and tested on a variety of platforms, ranging from high-performance computers with an RTX 1060 - 6GB VRAM and 16GB RAM, to low-performance embedded devices like Jetson Xavier NX.

Table 2: Summary of the comparison with other methods.

Comparison between PointNet, Point-Voxel CNN and Point Transformer						
Method	Device	Dataset	mIoU[%]	mAcc[%]	Train/Img[ms]	Eval/Img[ms]
PointNet	GTX 1060	S3DIS	43.32	81.87	**272.9**	**207.4**
		NYU	68.92	82.39	437.7	**287.2**
		ShapeNet	78.75	-	**207.9**	**166.8**
		Own	89.29	93.76	**237.8**	462.4
	Jetson NX	S3DIS	41.24	80.00	**1674.2**	908.0
		NYU	67.1	84.07	**1955.4**	983.6
		ShapeNet	78.79	-	**2393.0**	745.2
		Own	87.95	93.76	**1294.2**	894.0
	Colab	S3DIS	44.19	79.98	334.1	**199.8**
		NYU	69.87	82.87	529.6	280.3
		ShapeNet	79.04	-	**331.8**	**155.3**
		Own	91.1	94.84	464.7	427.2
PVCNN	GTX 1060	S3DIS	54.99	86.24	595.6	345.7
		NYU	78.36	82.28	719.2	298.7
		ShapeNet	83.67	-	519.6	316.2
		Own	87.77	91.68	645.6	**314.7**
	Jetson NX	S3DIS	55.22	86.18	5920.0	**815.1**
		NYU	68.5	82.17	4403.3	**879.5**
		ShapeNet	**83.70**	-	6898.4	**694.3**
		Own	86.71	94.19	5493.8	**728.8**
	Colab	S3DIS	55.17	86.45	506.2	336.4
		NYU	78.94	82.31	611.3	**239.2**
		ShapeNet	83.67	-	441.1	231.7
		Own	86.94	94.19	427.6	**302.0**
Point Transformer	GTX 1060	S3DIS	**69.4**	74.8	328.8	889.5
		NYU	**78.69**	83.06	**342.2**	295.3
		ShapeNet	**84.3**	-	489.2	175.7
		Own	**91.64**	92.48	315.6	387.6
	Jetson NX	S3DIS	**69.5**	75.1	3990.2	899.7
		NYU	**77.1**	81.89	3890.7	886.4
		ShapeNet	82.8	-	5163.9	791.2
		Own	**92.37**	92.06	3707.0	812.4
	Colab	S3DIS	**69.7**	75.3	**318.9**	879.9
		NYU	**79.07**	82.92	**308.8**	280.0
		ShapeNet	**83.84**	-	435.2	186.0
		Own	**92.48**	93.66	**294.4**	358.8

4.2.1 Datasets Used for Evaluation

Four datasets were used in the analysis. Three of these datasets are open to the public, while one is custom. The public datasets are as follows: Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset (Armeni et al., 2017), ShapeNetPart dataset (Yi et al., 2016) and NYU Depth Dataset V2 dataset (Nathan Silberman and Fergus, 2012).

The S3DIS dataset contains 271 rooms in six regions from three different buildings for semantic scene processing. Each scan point is given a semantic name from one of thirteen categories (ceiling, floor, table, etc.). On Area 5, we assess each of the approaches provided. We utilize mean classwise intersection over union (mIoU) and mean of class-wise accuracy (mAcc) as assessment measures.

The ShapeNetPart dataset has 3D object component segmentation annotations. There are 16,880 models in all, divided into 16 form categories, including 14,006 3D models for training and 2,874 for testing. The number of pieces in each category ranges from 2 to 6, with a total of 50 distinct parts. We report category instance mIoU for assessment metrics.

The NYU dataset the subset comprises 1449 highly labeled color pictures and depth maps, with 795 images serving as the training dataset and 654 images serving as the testing dataset. As assessment metrics, we use mean class-wise intersection over union (mIoU) and mean of class-wise correctness (mAcc).

Our custom dataset consists of four classes as a structure and they are visible in Fig. 5. The first row of the picture depicts the object in RGB format, whereas the second row shown the identical objects but segmented. The four classes are cylinder, box,

bag, and robot frame. In each class, 1000 images depict the segmented item. As assessment measures, mIoU was employed.

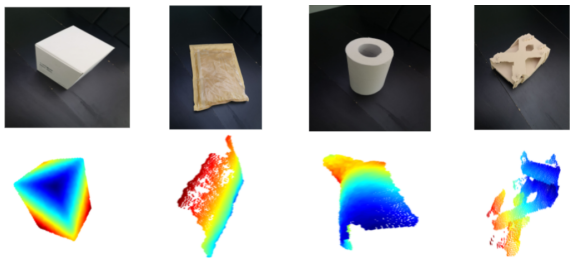

Figure 5: Our custom dataset.

4.3 Own Architecture Evaluation

The results of the evaluation of the proposed PV-CNN-based algorithm are shown in Table 3. The tests were carried out using the own and the shapenet public datasets. Because the IR field is missing from the shapenet dataset and estimating it without knowing the camera specifications is impossible, we utilized the **R**ed channel as IR, as it is the closest to IR. As a clear conclusion, we can see how the fusion between the IR and Depth image improves the design. As a visual validation, we can see how adding an IR image as a feature alongside the depth image improves the results in the video[1].

Table 3: Results obtained on own and shapenet dataset with/without IR using a Jetson NX.

	mIoU Depth	mIoU Depth + IR
Own dataset	94.82	**95.27**
Shapenet dataset	83.69	**83.87**

4.4 Performance Evaluation and Comparison

The results obtained for the presented three methods (PointNet, Point Transformer, PV-CNN) evaluated on different platforms (GTX 1060, Jetson NX, Colab) with different datasets (S3DIS, NYU, ShapeNet, Own) are shown in Table 2 and Table 3. We have been measuring the time for training and evaluating the image as well. During this time, the parsing of the configuration file and loading of the model are also calculated. For a better view of the results, best values across models for each dataset was outlined.

Our conclusions after analyzing the training and evaluation outcomes are as follows: Point Transformer outperforms all other platforms and datasets in terms of mIoU rankings. In terms of training time per image, PointNet is the most efficient, followed by Point Transformer. For image evaluation time, PointNet thrives on GTX 1060 and Colab platforms, whereas PV-CNN performs best on the Jetson NX device with 18 FPS.

[1]https://youtu.be/IrsOzx3VYXY

4.5 Training Details

The training was performed using the same batch size, number of points, epochs and the same platform to train and assess models for each method, even if we could have trained the model on a high-performance computer and subsequently assessed it on platforms with lower computing capability (Jetson Xavier NX). This technique was chosen because it allowed us to observe the training and the evaluation times for each platform at the same time.

5 SUMMARY

In this paper, we presented a comparison for the most popular methods for embedded point cloud processing at the moment, more precisely PointNet, Point-Voxel CNN and Point Transformer. We tested these approaches on a variety of datasets and platforms, ranging from high-performance computers to embedded devices. The tests were performed for object classification covering a wide range of datasets and algorithms for embedded devices. Alongside, we proposed a novel architecture for the Point-Voxel CNN for pulse-based Time of Flight (ToF) cameras by combining depth imaging with IR ones. The method was tested on custom dataset as well as public datasets.

ACKNOWLEDGMENTS

The authors are thankful for the support of Analog Devices GMBH Romania, for the equipment list and NVidia for the DGX grade server offered 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 PN-III-P3-3.6-H2020-2020-0060 and European Union's Horizon 2020 research and innovation programme under grant agreement No. 871295.

REFERENCES

Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). Joint 2D-3D-Semantic Data for Indoor Scene

Understanding. *Computing Research Repository*, abs/1702.01105.

Blaga, A., Militaru, C., Mezei, A.-D., and Tamas, L. (2021). Augmented reality integration into mes for connected workers. *Robotics and Computer-Integrated Manufacturing*, 68:102057.

Fooladgar, F. and Kasaei, S. (2020). A survey on indoor RGB-D semantic segmentation: from hand-crafted features to deep convolutional neural networks. *Multimedia Tools and Applications*, 79(7):4499–4524.

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

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Jiao, Y. and Yin, Z. (2020). A Two-Phase Cross-Modality Fusion Network for Robust 3D Object Detection. *Sensors*, 20(21).

Le, T. and Duan, Y. (2018). PointGrid: A Deep Network for 3D Shape Understanding. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 9204–9214.

Liu, Z., Tang, H., Lin, Y., and Han, S. (2019). Point-Voxel CNN for Efficient 3D Deep Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*.

Oleksiienko, I. and Iosifidis, A. (2021). Analysis of voxel-based 3D object detection methods efficiency for real-time embedded systems.

Ophoff, T., Van Beeck, K., and Goedemé, T. (2019). Exploring RGB+depth fusion for real-time object detection. *Sensors (Switzerland)*, 19(4).

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Computing Research Repository*, abs/1706.02413.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. (2021). PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection. pages 1–17.

Tamas, L. and Cozma, A. (2021). Embedded real-time people detection and tracking with time-of-flight camera. In *Proc. of SPIE Vol*, volume 11736, page 117360B.

Tamas, L. and Goron, L. C. (2014). 3D semantic interpretation for robot perception inside office environments. *Engineering Applications of Artificial Intelligence*, 32:76–87.

Tamas, L. and Jensen, B. (2014). Robustness analysis of 3d feature descriptors for object recognition using a time-of-flight camera. In *22nd Mediterranean Conference on Control and Automation*, pages 1020–1025. IEEE.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) oral presentation*.

Xu, Y., Tong, X., and Stilla, U. (2021). Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 126:103675.

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268.