Everted U-Net for 3D scene reconstruction and segmentation

Vladislav Mymlikov[1], *Oleslav* Antamoshkin[1,2*], and *Maxim* Farafonov[1]

[1]Siberian Federal University, 79 Svobodny pr., Krasnoyarsk, 660041, Russian Federation
[2]Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Ave, Krasnoyarsk, 660037, Russia

Abstract. The field of data science related to the processing of three-dimensional objects is becoming more and more relevant. After the successes in image processing, the apotheosis of which was the development of generative neural net-works, the intensification of efforts in the direction of three-dimensional data processing looks logical. Although there are now numerous systems for the reconstruction of three-dimensional objects and other processing, almost all of the existing solutions are aimed at working with a single object. The advances in image processing with neural networks have largely been made possible by huge datasets. There are also large datasets available for model training in this area. Freely avail-able datasets such as ShapeNet and ModelNet contain many thousands of different models, allowing for a high diversity of data. However, most of them provide single individual objects, which allows them to be used in tasks involving the processing of a single three-dimensional object, but when working with scenes containing many objects, there is often a problem of finding an appropriate dataset. This work is aimed at solving the problem of reconstruction and segmentation of three-dimensional scenes, as well as generation of datasets for the task of processing scenes from the real world.

1 Introduction

Recently, outstanding results have been achieved in the field of image processing in the form of generative networks capable of drawing images on user request [1]. Be-fore that, many images processing tasks, such as classification and segmentation, had already been solved quite successfully [2]. A logical continuation of these successes should be similar advances in three-dimensional data processing.

The rapid development of digital technology brings human-computer interaction to a whole new level. The emergence and widespread use of virtual reality systems creates the need to develop software that allows the user to fully immerse themselves in the virtual world. At the same time the significant task is to provide feedback from digital agents, their ability to perceive the real world. As part of solving this problem there are many subtasks, which include reconstruction of real scenes in the virtual world, recognition of objects and people in 3D and many others. One of the most important subtasks is the task of reconstruction of

* Corresponding author: oleslav24@gmail.com

three-dimensional scenes from the real world for their subsequent processing. Since the mixed reality involves the joint presence in a certain volume of real and virtual objects, the computer needs to understand what the real objects surrounding it, it is necessary to obtain information about their size and shape, to avoid collisions, and other processing, such as classification. The present project is aimed to develop a system of recognition and classification of objects in mixed reality in order to resolve the problem mentioned above.

A prerequisite for the development of neural network solutions is the availability of large data sets suitable for model training. It is due to the abundance of pictographic data on the Internet that a breakthrough in the training of image processing networks has become possible [3]. Today there are already various developments in the field of three-dimensional data processing [4, 5], in particular existing systems for the reconstruction of three-dimensional objects, solutions for three-dimensional classification and segmentation [6, 7]. In addition, there are many data sets suitable for training models working with three-dimensional objects [8-10]. At the same time, most of the existing datasets for 3D tasks assume processing of single objects, rather than whole scenes. This paper also solves the problem of automatic generation of datasets for training neural networks working with three-dimensional data.

2 Related works

At the moment, there are quite a few solutions that allow performing three-dimensional reconstruction of objects both by one [4-6, 7-10, 14, 15, 17] and by several images [3, 11-13, 16]. However, one of the main problems in this field is still that most solutions are created for reconstruction of a single object [3-11, 13]. At the same time, systems working with many objects in a scene at once [12, 14-17] are less common. In other words, so far, there is no single universally accepted approach that solves the problem of three-dimensional scene reconstruction in full. There are also many solutions for the problem of three-dimensional segmentation separately [22-26], but putting the solution of this problem in a separate system will inevitably affect the performance, so it is more reasonable to consider combined solutions, as in [14, 16, 17].

The small number of projects working with three-dimensional scenes is largely due to the lack of suitable datasets that can offer complex three-dimensional scenes. Most often used datasets, such as ShapeNet [18], Pix3D [19], or ModelNet [20], contain only simple scenes with single objects in the frame. There is little choice in this area, so researchers have to create their own datasets combining pairs or triples of objects [14].

Thus, the creation of a system capable of reconstructing a scene with multiple objects is a promising area for scientific research. In this paper, we will attempt to develop our own generator of three-dimensional scenes and train the reconstructor of objects from a set of images with its help.

3 Proposed solution

As part of this work, it is necessary to develop a system for automatic generation of data sets for training neural networks that reconstruct three-dimensional scene from a set of input images in the form of a voxel grid, as well as to implement a neural network that performs this task.

3.1 Data generator

In the course of the work, the first step was to create a training dataset by developing our own scene generator. Three-dimensional models of objects for the training sample were typed from https://sketchfab.com and are freely available for download. As object classes we selected the most common categories of objects in our studio: tables, chairs, nightstands, laptops, bags/backpacks and a person in a mocap suit. A total of 190 models were collected, including: 40 tables, 40 chairs, 40 nightstands, 20 laptops, 40 bags/backpacks and 10 people in various poses.

The scene generator operates with three-dimensional models of objects, receiving as input a dictionary, which specifies the maximum number of objects in the scene for each class. In the current version, objects are limited to 2 tables, 3 chairs, 2 nightstands, 3 bags and 3 actors.

Each scene is a random set of objects that are randomly placed within a given volume of the virtual room. The objects selected for the scene are arranged one by one by applying random translation and rotation. It is important to avoid collisions between objects, because objects do not behave that way in real life. To do this, the generator checks that the current object does not intersect with all the previously placed objects by comparing their spatial bounding boxes. In case of intersection, the conflicting object reverts to its previous state and is randomly collocated again. If after N attempts the collision still occurs, the object returns to its original position and is hidden from view, not participating in the current scene.

An important element is the ability to combine objects in the scene, arranging small objects on larger ones: for example, putting laptops and bags on tables or nightstands. This is made possible by organizing a hierarchy of models, in which some objects are assigned as children to others. The arrangement of child objects is done in the same way as the arrangement of objects inside a room, only the area above the parent object is used as the volume. An example of the generated scene is shown in Figure 1.

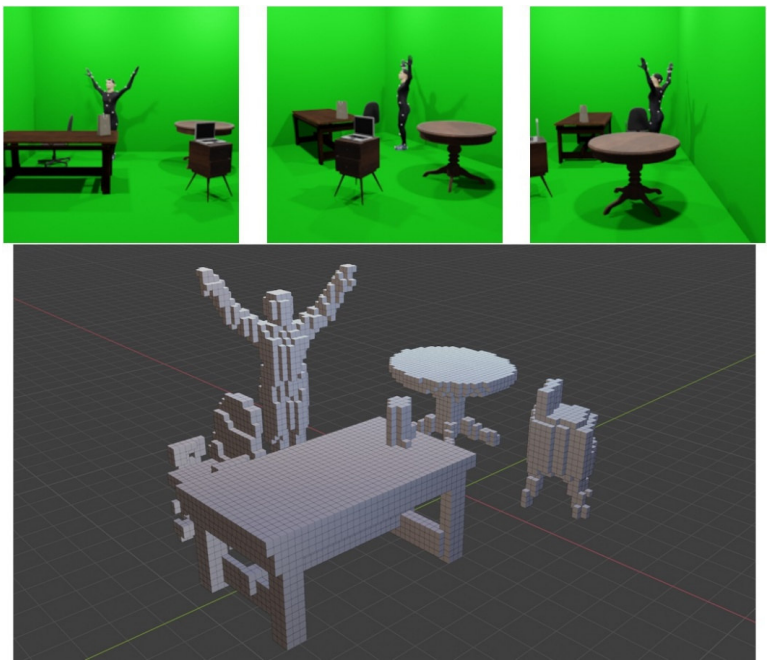

Fig. 1. The example of the generated scene. The three images above are renders of the generated scene from different angles, below is a voxel grid render in Blender.

3.2 Everted U-Net

Most similar works, dealing with reconstruction of single or multiple objects, are based on the encoder-decoder principle, where the encoder extracts features from the input images, and the decoder based on them performs a three-dimensional deconvolution producing a grid of voxels. The present solution is based on Pix2Vox++ [21] and the model from [25]. Pix2Vox++ allows to process several images of a scene from different angles, and the running time for three angles is close to real time. But this solution does not do any segmentation and does not classify objects. At the same time, the model from [25] performs reconstruction of multiple objects, their semantic segmentation and segmentation by entities, but does so only from one perspective. Our paper presents an attempt to combine the features of the above two solutions for reconstruction and segmentation of the set of objects in a scene. For this purpose, a model architecture combining the reconstruction and segmentation branch, which are placed according to the U-Net principle, is presented (see Figure 2).

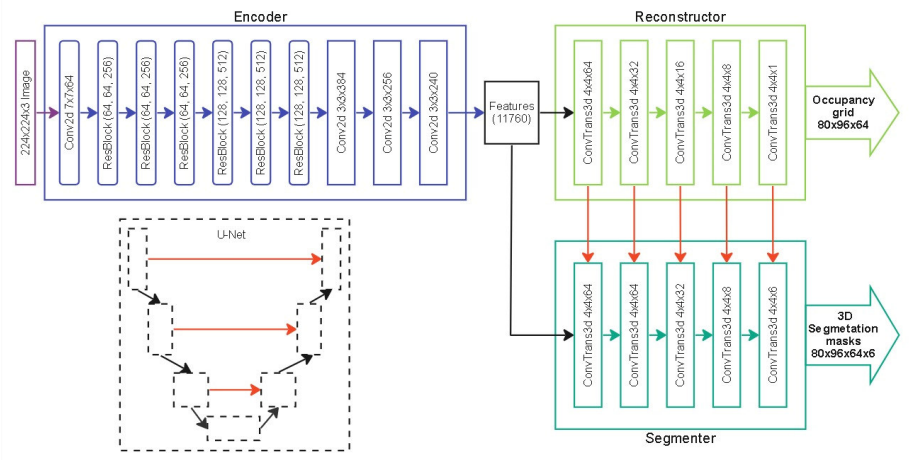

Fig. 2. A schematic representation of the architecture of our solution. The first half of the network is a conventional feature extractor, the role of which will be played by ResNet50. One branch (above) performs scene reconstruction in the form of voxels, similar to Pix2Vox++. The lower branch performs scene segmentation, and in the process, it uses data from the reconstructor sweep layers, by analogy with how U-Net does it.

The system receives three 224x224 images as input, which after passing through ResNet50 will be represented as a set of 98x5x6x4 shape features, which are then goes into the reconstructor branch, which is a set of transposed 3D convolution layers, and the segmentation branch, which has the same structure. In addition to output from the previous layer each segmentation layer also concatenates output from a parallel layer from the reconstruction branch. The above operations are performed in parallel for each of the three input images, after which fusion layers are applied to the resulting reconstructions and segmentation masks, similar to how it is done in Pix2Vox++.

4 Experiment

The project is developed in the Python programming language version 3.10, the PyTorch library is used for the implementation, in particular its CUDA version to work with graphic cards. In addition, Blender, a free open-source program for 3D modeling, is used to generate the dataset via our generator-script.

To train our neural network, we first generated a training dataset consisting of 25000 scenes, as well as an additional set of 1000 scenes with other object models to test the trained networks. The training set is divided into a training sample of 22500 scenes and a validation sample of 2500 scenes. Each scene is presented as a folder containing three RGB images at 224x224 resolution, and two serialized .npy files containing an 80x96x64 voxel grid for the occupancy and an 80x96x64x6 voxel grid for masks, where the last dimension corresponds to the number of classes. This resolution of the voxel grid was adopted because it corresponds to the aspect ratio of the green room in our studio for work with which this system is supposed to be used.

When training the network, the weights of the feature extractor are frozen, and only the reconstructor and segmenter layers are trained. The training process of the developed ntwork consists of two stages. First, the reconstructor network is trained; this is necessary because the segmenter is dependent on data from the reconstructor, and therefore it makes no sense to train the segmenter before training of the reconstructor is completed. Each network is trained in two stages: in the first stage the prediction is performed on only one random image from the set, in the second stage the training continues on triples of images. Binary cross-entropy is used as the loss function; the optimization function is Adam with an initial learning rate of 0.001. StepLR scheduler with step 10 and multiplier 0.5 was used to control the learning rate of the segmenter.

Initial training of the reconstructor and segmenter networks has been performed so far. Training was performed on an Nvidia RTX 3090 Ti with 24 GB of video memory over 100 epochs for each branch using early stopping after 10 consecutive epochs without improvements on the validation sample. The mini-batch size was 16 and 8 for the reconstructor and the segmenter respectively. The mIoU is used as the quality metric, and the threshold value is chosen according to the best result on the validation sample.

Next, a meta-optimization of the networks structure was performed by enumerating a number of training parameters and the sizes and types of network layers. Since one cycle of network training took about 20 hours, it would take months or even years of continuous work to enumerate all combinations of parameters or to use any meta-optimization algorithm. For this reason, the search for the best strategy is carried out according to the following principle: changes are made to the current strategy, if they give a positive result, these changes become the baseline, and the next experiments are conducted on their basis. As part of their training, each model was also selected by the lowest validation error, and comparison with other models was done based on the error on the test sample. The best strategy was selected by the minimum test error.

For the experiments, the data size was reduced to 5000 training examples (4500 training and 500 validation examples) and 250 test examples to slightly speed up the results. The early stopping parameter was also reduced to 5 consecutive failures per validation. In addition, the initial state of the network was frozen so that all experiments start from the same point.

During the experiments we tried the following parameters and values: learning rate 0.0001, 0.001, 0.01; Dropout 0.0, 0.1, 0.2, 0.3, 0.4, 0.5; Batch size 2, 4, 8, 16. Table 1 shows the best found parameter values for both models after completing the meta-optimization process.

Table 1. The best values of training parameters for final networks and their mean IoU metric.

Network	Learning rate	Dropout	Batch-size	Threshold	mIoU
Reconstructor	0.01	0.10	16	0.32	0.392
Segmenter	0.01	0.25	8	0.37	0.384

Thus, during the selection of the learning strategy it was found out that adding Dropout layers makes the most significant contribution to the reduction of the test error. Changing other parameters has less impact on the final accuracy. Examples of predicted scenes are shown in Figure 3 and Figure 4.

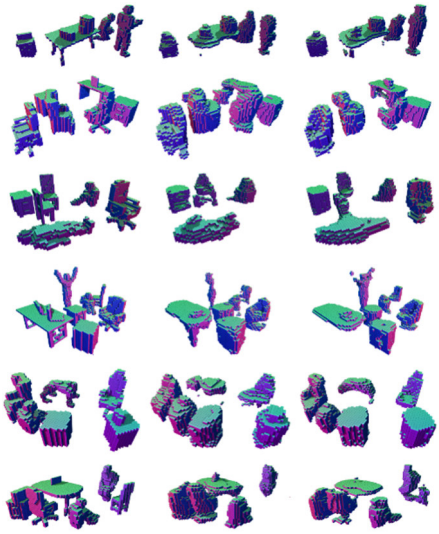

Fig. 3. Ground truth voxel grid (left), predicted voxel grid by the first version (center) and the final version (right) of the reconstructor. It successfully recognizes the silhouettes of objects in the scene, but is not yet able to reconstruct fine details. The latest version achieved some improvements in reconstruction quality and predicted objects look much more clean.

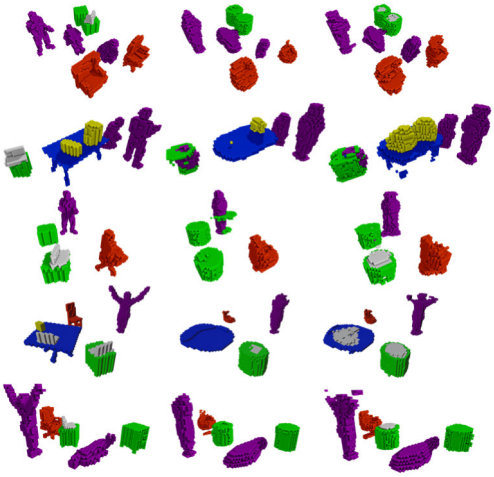

Fig. 4. Ground truth voxel masks (left), predicted class masks by the first version (right) and the final version of segmentation network. Each class has a different color (people are purple, chairs are red, nightstands are green, desks are blue, laptops are white, and bags are yellow). The segmenter is good at determining class affiliation, but the accuracy of the shapes so far leaves much to be desired. The final version also demonstrates much more accurate results in shape and class mark predictions.

The following Table 2 gives a summary of mIoU per class for the segmenter.

Table 2. Mean IoU per class. The final version outperforms the first one at almost all categories of objects. Laptop and Bag classes still show the lowest results.

Class	First version	Final version
Chair	0.264	**0.302**
Table	**0.366**	0.358
Nightstand	0.354	**0.420**
Laptop	0.114	**0.149**
Bag	0.040	**0.154**
Human	0.377	**0.450**
Overall	0.335	**0.384**

It is important to note that the current accuracy scores were obtained at objects that were not in the training dataset. In other words, the network tried to predict objects of the same classes, but having an unfamiliar appearance.

We have a number of assumptions about how to improve performance, but they need experimental verification, which will be done in the near future. There are also ideas to experiment with the positioning of virtual cameras to obtain more informative perspectives. The use of three perspectives improves the visibility of objects on the scene, but does not completely exclude situations when some objects overlap or are out of the lens of one of the cameras, so the network in any case has to "guess" the shape of such objects.

5 Conclusion

As part of this project, a new solution for automatic generation of three-dimensional datasets in voxel grid format was developed. A solution was also proposed to perform reconstruction and segmentation of three-dimensional scenes from a set of input images.

It is worth noting that the solution being developed is primarily intended for use in a studio environment, so the area of its operation is limited by the size of the room. Also, it can already be assumed that it is unlikely to achieve higher performance on real images, since synthetic data is used in training the model. The suggestion is that further improvement is limited by the amount of information that the network can extract from the current data set.

At the moment there are two important problems that need to be solved:

1. Renderer inefficiently uses hardware resources, which is expressed in long operation and high load on hardware with insufficiently high quality of the result. It is necessary to find a way to optimize the rendering process to obtain the best quality with full use of available computational resources.
2. The rendering of a three-dimensional scene should be as realistic as possible. We need a tool for automatic evaluation of rendering realism like in [26], which will allow us to determine the suitability of the generated data for training neural networks.

After obtaining the maximum possible values of reconstruction and segmentation accuracy, we will start optimizing the network inference time and adapting it for use in game

engines. In particular, we plan to port the algorithms to the C++ programming language. Then we are going to start working on integration of the network into streaming services' workflows.

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant № 075-15-2022-1121).

References

1. A. Stöckl, Evaluating a Synthetic Image Dataset Generated with Stable Diffusion. arXiv preprint arXiv:2211.01777 (2022)
2. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, L. Fei-Fei arXiv preprint arXiv:1409.0575 (2015)
3. C.B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. arXiv preprint arXiv:1604.00449 (2016)
4. H. Fan, H. Su, L. J. Guibas, A point set generation network for 3d object reconstruction from a single image. arXiv preprint arXiv:1612.00603 (2017)
5. M. Tatarchenko, A. Dosovitskiy, T. Brox, Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. arXiv preprint arXiv:1703.09438 (2017)
6. H. Kato, Y. Ushiku, T. Harada, Neural 3d mesh renderer. arXiv preprint arXiv:1711.07 566 (2018)
7. T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, M. Aubry, A papier-mâché approach to learning 3d surface generation. arXiv preprint arXiv:1802.05384 (2018)
8. B. Yang, S. Rosa, A. Markham, N. Trigoni, H. Wen, Dense 3D object reconstruction from a single depth view. arXiv preprint arXiv:1802.00411 (2018)
9. N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, Y. G. Jiang, Pixel2mesh: Generating 3d mesh models from single rgb images. arXiv preprint arXiv:1804.01654 (2018)
10. L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger, Occupancy networks: Learning 3d reconstruction in function space. arXiv preprint arXiv:1812.03828 (2019)
11. H. Xie, H. Yao, X. Sun, S. Zhou, S. Zhang, Pix2vox: Context-aware 3d reconstruction from single and multi-view images. arXiv preprint arXiv:1901.11153 (2019)
12. Z. Murez, T. Van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, A. Rabinovich, Atlas: End-to-end 3d scene reconstruction from posed images. arXiv preprint arXiv:2003.10432 (2020)
13. H. Xie, H. Yao, S. Zhang, S. Zhou, W. Sun arXiv preprint arXiv:2006.12250 (2020)
14. S. Popov, P. Bauszat, V. Ferrari, Corenet: Coherent 3d scene reconstruction from a single rgb image. arXiv preprint arXiv:2004.12989. (2020).
15. G. Gkioxari, J. Malik, J. Johnson, Mesh r-cnn. arXiv preprint arXiv:1906.02739 (2019)
16. M.J. Tyszkiewicz, K.K. Maninis, S. Popov, V. Ferrari, RayTran: 3D pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers. arXiv preprint arXiv:2203.13296 (2022)
17. M. Dahnert, J. Hou, M. Nießner, A. Dai, Panoptic 3d scene reconstruction from a single rgb image. arXiv preprint arXiv:2111.02444 (2021)
18. A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, F. Yu arXiv preprint arXiv:1512.03012 (2015)

19. X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, W. T. Freeman, Pix3d: Dataset and methods for single-image 3d shape modeling. arXiv preprint arXiv:1804.04610 (2018)
20. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes. arXiv preprint arXiv:1406.5670 (2015)
21. C.R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593 (2017)
22. L. Jiang, H. Zhao, S. Shi, S. Liu, C. W. Fu, J. Jia, Pointgroup: Dual-set point grouping for 3d instance segmentation. arXiv preprint arXiv:2004.01658 (2020)
23. W. Zhao, Y. Yan, C. Yang, J. Ye, X. Yang, K. Huang, Divide and Conquer: 3D Point Cloud Instance Segmentation With Point-Wise Binarization. arXiv preprint arXiv:2207.11209 (2022)
24. C. Liu, Y. Furukawa arXiv preprint arXiv:1902.04478 (2019)
25. H. Chen, Q. Dou, L. Yu, P. A. Heng arXiv preprint arXiv:1608.05895 (2016)
26. P. Peresunko, D. Mamatin, O. Antamoshkin, E. Peresunko, A. Nikitin, Models of Experts for Shaders Estimation of Rendering Complex 3D Scenes in Real Time. 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA) (2021)