Structure from Motion by Artificial Neural Networks

Julius Schöning, Thea Behrens, Patrick Faion, Peyman Kheiri, Gunther Heidemann, and Ulf Krumnack

Institute of Cognitive Science, Osnabrück University, Osnabrück, Germany
{juschoening, tbehrens, pfaion, pkheiri, gheidema, krumnack}@uos.de

Abstract. Retrieving the 3D shape of an object from a collection of images or a video is currently realized with multiple view geometry algorithms, most commonly Structure from Motion (SfM) methods. With the aim of introducing artificial neuronal networks (ANN) into the domain of image-based 3D reconstruction of *unknown* object categories, we developed a scalable voxel-based dataset in which one can choose different training and testing subsets. We show that image-based 3D shape reconstruction by ANNs is possible, and we evaluate the aspect of scalability by examining the correlation between the complexity of the reconstructed object and the required amount of training samples. Along with our dataset, we are introducing, in this paper, a first baseline achieved by an only five-layer ANN. For capturing life's complexity, the ANNs trained on our dataset can be used a as pre-trained starting point and adapted for further investigation. Finally, we conclude with a discussion of open issues and further work empowering 3D reconstruction on real world images or video sequences by a CAD-model based ANN training data set.

1 Introduction

Nowadays, image-based 3D reconstruction, also known as multiview stereo approaches, are usually realized as an analytic solution, based on multiple view geometry. In general, almost all approaches use the principle of *Structure* from *Motion* (SfM)—or rephrased *real wold object shapes* from *different 2D images captured from different locations*. The reconstructed 3D objects are usually represented by a set of s points in a 3D space calculated from $w \geq 2$ 2D images captured from different camera positions. This kind of representation leads to a nonlinear least squares problem with $2 \cdot w \cdot s$ constraints and $6 \cdot w + 3 \cdot s$ unknowns. Benchmarks of state of the art commercial 3D reconstruction software like *123D Catch* [2], or *PhotoScan* [1], as well as academic approaches e.g. *VisualSfM* [29], or *ARC 3D* [26], show that the accuracy of the reconstructed 3D point cloud with respect to the ground truth is mostly sufficient [23] for the specific use case, but exponentially impacts the processing time. For solving the nonlinear least squares problem in these SfM approaches, methods such as gradient descent, conjugate gradient, Gauss-Newton, Levenberg Marquardt, and singular value

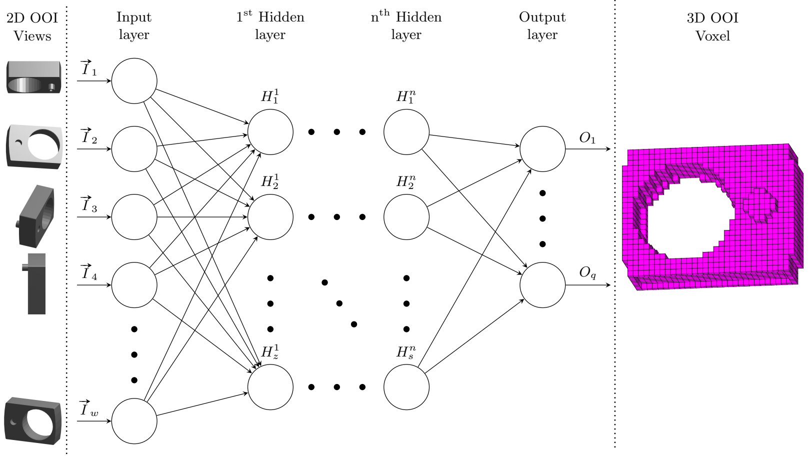
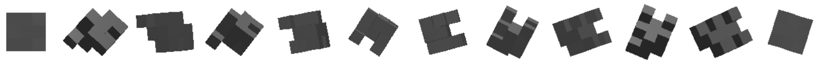
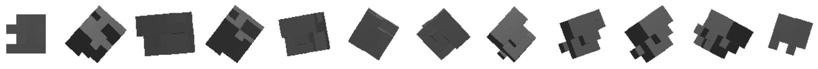
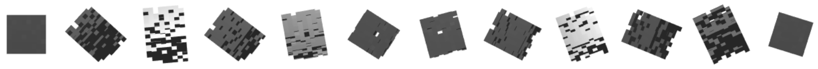
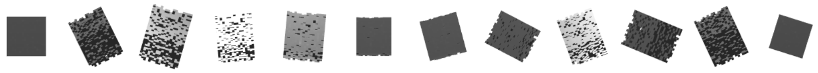


Fig. 1. Simplified scheme of an ANN system architecture for image-based 3D reconstruction. As input vectors ($\vec{I}_w$) for the ANN, a number of w 2D gray scale projections of a 3D object, taken from particular viewpoints, are used. The kind of ANN (feed-forward, convolutional, recurrent, etc.) and setup (numbers of layers $[n]$, numbers of neurons within each layer, kind of meshing, etc.) has to be varied to find the best possible network for this task. The output layer of the ANN consists of v^3 voxels, depending on the resolution of the voxel space. Our current training dataset consists of several $10,0000$ objects with different voxel resolutions. Each training item consists of $w = 12$ images from various viewpoints of each object that serve as input—seen on the left hand side of the figure—as well as v^3 binary voxel values as output—seen on the right hand side. Currently, we train and test our ANNs only on our virtual computer generated dataset. In the future, these pre-trained ANNs shall be applied for the training on "real-world" images and objects.

decomposition are commonly used. The complete processing pipeline of SfM approaches consists of a concatenation of algorithms, some of which have already been shown to be realizable by ANNs [20,27,4,10]. Hence, we introducing a scalable dataset, which provides 3D objects and 2D images of different complexity. Such a dataset, we argue, would allow for a systematic investigation of ANNs for multiview reconstruction, hopefully leading to a better understanding of which architectures are suitable for this task. Highlighting these opportunities, we include a first baseline showing that a single simple feedforward ANN can replace the whole 3D reconstruction pipeline. It already allows us to analyze some of its weaknesses, like a low reconstruction accuracy of occluded parts.

While recent deep neural network models have achieved promising results on related tasks, e.g. 3D object recognition, depth prediction, and single-view 3D reconstruction, which are introduced in detail in Section 2, the task of image-based 3D reconstruction of unknown object categories using ANNs is, to our knowledge, not yet systematically examined. This fact is partly due to the lack of datasets for this task, containing images and 3D ground truth of different 3D

Table 1. Overview of the cube dataset showing randomly picked objects in different setups as example.

setup	images of the object seen from 12 different viewpoints
$3 \times 3 \times 3$	 cube *012497* with its pattern $(1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0)$
$4 \times 4 \times 4$	 cube *030290* with its pattern $(0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, \ldots, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1)$
$8 \times 8 \times 8$	 cube *029598* with its pattern $(1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, \ldots, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1)$
$16 \times 16 \times 16$	 cube *000776* with its pattern $(1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1 \ldots, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1)$

objects. Filling this gap, we introduce in Section 3 the general architecture of ANN-based 3D reconstruction and discuss how the 3D objects can be represented in the output layer of an ANN. Following this initial discussion, we introduce our voxel-based dataset in Section 4 and release the less complex cases of $3 \times 3 \times 3$, $4 \times 4 \times 4$ and $8 \times 8 \times 8$ voxel spaces as a static dataset. For providing a scalable dataset, which allows different degrees of complexity, a generator for $n \times n \times n$ datasets is introduced as well. In Section 5, we describe the design of a simple feedforward ANN and show that it can reconstruct 3D objects in both the $3 \times 3 \times 3$ and $4 \times 4 \times 4$ setup with an adequate accuracy. Based on the results we achieved by training our ANN, we introduce the first baseline on this dataset in Section 6. Section 7 sets our approach in a broader context and discusses opportunities for transferring the results of ANN-based multiview 3D reconstruction from our computer generated dataset to real world images or video sequences with unknown 3D objects. We conclude this paper by pointing out open issues, proposing further working directions, and describing ongoing work such as our CAD model-based ANN training data set.

2 State of the Art

The reconstruction of 3D objects with ANNs is still in its infancy. To our knowledge, there is no approach that does 3D reconstruction of *unknown* 3D objects using only ANNs. For known 3D object categories, recently few approaches [7,30] were introduced. However, our approach worked with unknown categories just like any SfM approach. We summarize in this section work on related topics that seem relevant in this context, for example 3D object recognition, depth map generation as well as prediction, stereo image generation, and single-view 3D reconstruction.

For depth prediction, Eigen et al. [8] use series of convolutional neural network (CNN) stacks applied at increasing resolution for surface normal estimation and semantic labeling. The first scale predicts a set of coarse features for the entire image, the second scale produces mid-level resolution predictions, and the third scale refines the predictions to higher resolution. This work is based on a multi-scale deep network for depth estimation [9]. A global coarse-scaled network consisting of convolutional and max-pooling layers estimates the depth at a global level. A fully convolutional fine-scaled network aligns these coarse predictions with local details. For surface normal estimation, Wang et al. [28] implement two CNNs. A top-down CNN takes the whole image as input and captures the coarse structures which cannot be decoded by local evidence alone. A bottom-up CNN acts on local patches extracted from the image and captures local evidence at a higher resolution. The output of these two networks is combined with a fusion network that learns how to incorporate their predictions.

In 2016, Liu et al. [18] propose a single-view 3D reconstruction method using a CNN to estimate per-pixel depth, normal, and symmetry correspondence. Rectifying the depth information, they set up the symmetry correspondences as an optimization problem in their network. Another work [17] introduces a CNN to learn unary and binary potentials for the continuous conditional random field layer that estimates depth on single images which are over-segmented into superpixels. Li et al. [16] use a method that extracts multi-scale image patches around the superpixel centers, and a CNN learns the relationship between these patches. Thus, the estimation of depth can be formulated as a regression problem.

Roy et al. [22] present a convolutional regression forest where each node in the forest is associated with a CNN which makes a depth estimation for a window around each pixel with a corresponding probability. Without requiring labeled data by using only pairs of images with a small camera motion between themselves, Garg et al. [11] propose an unsupervised framework to train a deep DNN for single-view depth prediction.

Already back in 2004, Peng et al. [20] showed that object reconstruction by an ANN is more accurate than object generation by 3^{rd} order polynomials. The ANNs they used in their experiments were completely based on multilayer feedforward network designs.

3 Multiview 3D Reconstruction by ANNs

Images as input for ANNs have become quite common for, e.g., object recognition tasks [14,12,24]. For these tasks, the output layer of an ANN is usually defined in a way that each output neuron represents a single object category.

However, in computer graphics and computer vision, the shape of 3D objects is usually described by vertices, edges, and faces which vary depending on the complexity of the object. A simplified representation of 3D objects for computers was introduced by the first computer games, which use volumetric pixels—so called voxels. Thus, when designing an ANN to output 3D information one has to choose a suitable representation for 3D objects, i.e., the output layer of the ANN must encode either i) vertices, ii) edges, iii) faces, or iv) voxels of the object.

In connection with the performance of ANNs in binary classification tasks, we chose a voxel-based representation of the 3D objects for the output layer. This form of representation has the additional advantage that the resolution and therefore its accuracy, as well as its computational complexity, can be scaled. The input consists of images captured from the object from different viewpoints, as is common for any SfM approach. These considerations lead to the scheme of a general architecture for ANN-based 3D reconstruction, illustrated in Fig. 1. Like in digital imaging, a general working architecture with a low resolution should establish the basis for higher resolution results. Hence, we designed our dataset with low resolutions, but with an option to scale up.

4 Dataset

As a first idea for creating a dataset which provides the necessary amount of training and test samples to train larger ANNs, we considered a database generator based on a geometry definition file format like Wavefront *obj*. Since a dataset of a variety of 3D *obj* objects will consist of simple and complex objects at the same time, and because the complexity of an object is quite difficult to quantize, we turned towards designing a more conservative dataset. Thus, we brought up a cube-based dataset, cf. exemplary objects of it in Table 1. The generation of this dataset is done in two steps. First, a *cube generator* computes random cubes in a $r \times r \times r$-space and stores them as 3D *obj* objects. Second, the *images and voxel generator* creates w images showing an object from different viewpoints and its corresponding voxel cloud in a defined resolution.

4.1 Cube Generator

This generator[1], written in Matlab, randomly generates n 3D objects. Each such object is created by taking a unit cube in $\mathbb{R}^3$ and subdividing it into $r \times r \times r$ subcubes. The parameter r can be defined by the user. By ensuring the uniqueness of the cube distribution in the $r \times r \times r$ grid, this generator is able to generate 2^{r^3} different 3D objects and export them as 3D *obj* object files. Each 3D object is generated by filling the $r \times r \times r$ grid with random binary values, where 1 is interpreted as a filled subcube, while 0 signifies an empty space. To generate the object description, each subcube in the $r \times r \times r$ grid is described

[1] cf. supplementary material on `https://ikw.uos.de/%7Ecv/publications/SCIA17`

by 8 vertices connected by 6 rectangular faces. Before exporting them as 3D *obj* objects, duplicate vertices are combined and inner faces, where two subcubes are connected with each other, are deleted.

4.2 Images and Voxel Generator

For the generation of the w input images and a voxel cloud with the resolution v, a Matlab script is provided[1] which accepts 3D *obj* files as input data. For this generator, the user can define the number of images w taken from the 3D object, its pixel resolution $x \times x$, and the resolution of the voxel cloud $v \times v \times v$.

When generating w images showing the object from w different viewpoints, the generator will choose viewpoints that are uniformly distributed around the object to provide as different perspectives as possible. This is achieved by considering a sphere enclosing the object and evenly distributing the w viewpoints on the sphere using the Fibonacci lattice [13]. From these viewpoints, gray scale images with a resolution of $x \times x$ are rendered. For each scene, a light source is added next to the viewpoint. The images, as well as the voxel cloud of the objects, are created after scaling the object to the $v \times v \times v$ grid, so that all voxels laying inside the object are defined as cubes.

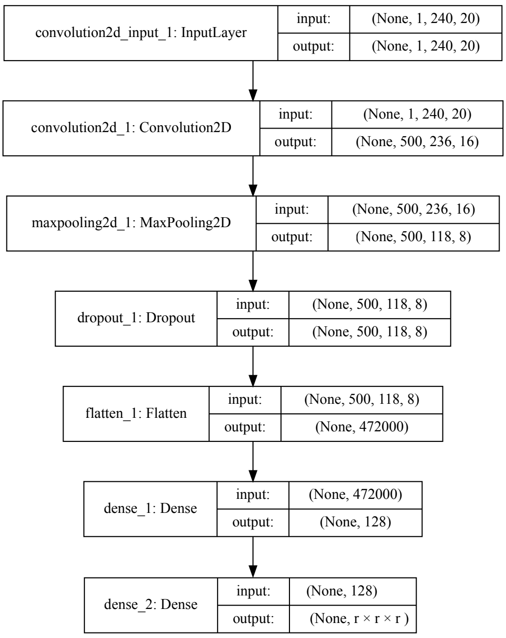


Fig. 2. The activation functions for all inner nodes are *rectifiers*, while the *tanh* is used for the output layer. The loss function used for training is the mean squared error.

4.3 Cube Datasets

In addition to the generators, we have released three datasets—a $3 \times 3 \times 3$, a $4 \times 4 \times 4$ and a $8 \times 8 \times 8$ setup, cf. Table 1—as static benchmarks. This ensures reproducibility for the proposed baseline. These static snapshots consist of $100,000$ different objects for the $3 \times 3 \times 3$ setup, $300,000$ different objects for the $4 \times 4 \times 4$ setup and $430,000$ different objects for the $8 \times 8 \times 8$ setup. The objects in both datasets are shown from $w = 12$ different viewpoints with a resolution of 100×100 pixel. This number of objects is, in our opinion, a sufficient amount for training an ANN. Further, we chose $w = 12$ different viewpoints since this is the minimum amount allowing us to solve this task by hand.

5 3D Reconstruction with ANNs

To show the usability of our dataset in a 3D reconstruction task, we trained an ANN to predict the 3D voxel representation of the object from its images. For each of the objects, there were 12 images, taken from different viewpoints (cf. Table 1), and the values in the binary voxel grid are either 0 when outside the object or 1 when inside the object. The aim is that the trained ANN acquires a model of this relationship and is able to output correct voxel grid labels, also for new unknown 3D objects in the same input format.

We prepared the data to fit the available hardware resources by downscaling the size of the input images, here from 100×100 to 20×20 pixels. We then concatenated all 12 images of one object into one matrix of the size 240×20, which then served as input to the ANN. To make the objects easier to load at once as training data for the different ANN setups, $100,000$ different objects were stored into one large matrix.

5.1 Baseline ANN

We created a first simple feedforward ANN using the Keras [6] neural networks library, with Theano [25] as backend. Due to the use of binary voxels as output, image-based 3D reconstruction becomes similar to a classification task (but allowing multiple classes per input). Hence, we adapt a simple example ANN [5,24], that was originally devised to recognize handwritten digits from the MNIST dataset [15]. The resulting network consists of five main layers i.e. the input layer, a convolutional layer, a pooling layer (max pooling), a fully connected dense layer, and the output layer, that is also fully connected.

The activation functions of all inner nodes are *rectifiers* and for the output *tanh*. As the *tanh* function has outputs in the range of -1 to 1, the voxel grid labels are adapted accordingly. Because tuning the ANN is not the primary objective in this paper, the architecture of the layers are chosen without much consideration. Therefore, it is possible that other sizes would be more suitable for the problem—the values used can be found in Fig. 2. For the objective of providing a dataset for a sytematical development of multiview reconstruction

Table 2. Baseline of the image-based cube 3D reconstruction dataset. All results are generated by using the feedforward ANN seen in Fig. 2. The object accuracy is a measure of how many objects were reconstructed completely correct or more than 80% of their voxels.

	training cubes	epochs	voxel accuracy	object 100% accuracy	object 80% accuracy
$3 \times 3 \times 3$	1–10,000	4	78.01%	0.52%	40.80%
	1–10,000	8	82.03%	2.39%	63.05%
	1–30,000	2	84.66%	1.38%	60.91%
	1–30,000	4	**88.86%**	4.84%	81.72%
$4 \times 4 \times 4$	1–10,000	8	66.01%	0.00%	1.53%
	1–30,000	2	67.09%	0.00%	0.90%
	1–30,000	4	**70.80%**	0.00%	4.47%

by ANN, the generated results provide an initial step (cf. Section 7) as a starting point for further research.

For creating a more robust network, a dropout rate of 20% during training in the connections between the pooling layer and the dense layer is implemented in addition to the main layers cf. Fig. 2. For training we used, the *training sample size* and *epochs* hyperparameters mentioned in Table 2, a batch size of 200, the mean squared error as loss function and the *Adam* optimizer.

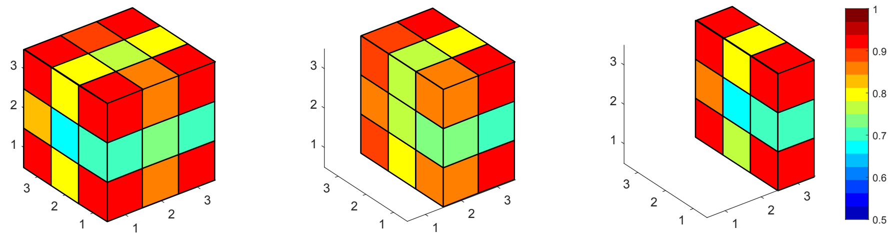
6 Baseline and Results

The best performance we achieved when training the ANN shown in Fig. 2 was 88.68% overall voxel accuracy on the $3 \times 3 \times 3$ setup, using objects 1 to 30,000 as the training set during 4 epochs of learning. In the $4 \times 4 \times 4$ setup, a maximum accuracy over all voxels of 70.80% could be reached, using the objects 1 to 30,000 as the training set over 4 epochs. In Table 2, a selected overview of trained ANNs is given as baseline, where we also provide the percentages of 3D objects which could be reconstructed 100% and 80% correct, respectively. The footprint on the hard disk of all trained ANNs was about $725MB$ per network. The overall run time varied from 2 to 6 hours using a non GPU accelerated version of Theano.

To investigatie how well a voxel at a certain position was learned, the accuracy for each voxel is calculated. Fig. 3 shows the average accuracy for (a) the $3 \times 3 \times 3$ setup and (b) the $4 \times 4 \times 4$ setup, where an accuracy of random chance—50%— is marked blue and an accuurracy of 100% is marked dark red.

7 Discussion

There is an ongoing discussion on what reconstruction accuracy can be reached with an ANN approach for image-based 3D reconstruction. During the preparation of this dataset, its generator, and while designing the first ANN architecture,


(a) voxel accuracy in $3 \times 3 \times 3$ setup

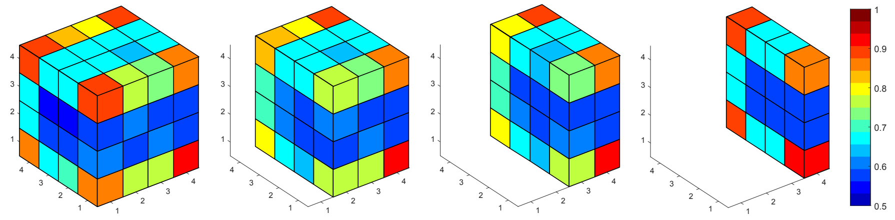

(b) voxel accuracy in $4 \times 4 \times 4$ setup

Fig. 3. Average accuracy of each voxel in two different setup (a) and (b); from the left to the right hand side the slice by slice walk through; accuracy is color coded from 0.5 random chance–blue to 1.0 every time correct–dark red

our estimations on the expected accuracy varied from about 60% up to 98%. As the standard processing pipeline used for image-based 3D reconstruction is a concatenation of algorithms, most of them applying only linear operations, one would expect that ANNs should be able to perform quite well on this kind of problem. However, the first observations, presented in Table 2, suggest that the problem is not as simple as one might expect. One possible reason is that classical SfM approaches [29,1,19,21] reconstruct vertices and faces instead of reconstructing the object as voxel space. This issue might increase the complexity, due to the *hidden* inner object voxels. These inner voxels may be occluded and thus not detectable on any image. Therefore, the additional interrelationship must be learned, that, e.g., inner object voxels are always to be filled.

Based on these considerations, we were interested if there are positions that are harder to predict within the voxel space. Thus, we expanded the results, with Fig. 3 showing the average accuracy for each voxel. It can be seen that the accuracy of the outer voxels of the cube in both setups is very high, while the accuracy drops towards the center of the voxel cube. This can be explained by the fact that these voxels are often blocked from view by other parts of the object.

Considering the issues that occur even on this simple cube dataset, one can expect further insights from a more thorough analysis of ANNs trained on this dataset. It may turn out that 2D convolution is not the appropriate architecture for this 3D scenario and that other architectures are more suitable. The recent technique of layer-wise relevance propagation [3] may help to identify problematic regions in the input images and suggest changes to the design of the ANN. Furthermore, one may consider using the classification, in a non binary manner. Instead, one can create *half* cubes by incorporating the correspondences to the surrounding blocks. In this way, more accurate objects can be reconstructed.

8 Conclusion

In this paper, we introduced a cube-based dataset to be used as a reference dataset for evaluating multiview 3D reconstruction algorithms. Also, we devised a first baseline ANN architecture to be trained on that dataset. Our initial results suggest that multiview 3D reconstruction of unknown objects by a simple five layer feedforward ANN is possible. Based on the results observed on our simple ANN, we expect that a significant improvement in the performance can be achieved. These enhancements can be done, e.g. through another design of the layers, as well as its connections, more training examples or epochs, another size of the training set or even through another non-voxel-based representation of the 3D object. As already stated in the introduction, the main goal of this paper is not to introduce an ANN with the best possible accuracy, but rather to introduce a dataset for a systematic investigation of ANNs for multiview reconstruction.

Consequently, the dataset presented here is the basis on top of which we will develop, design, train, test, and benchmark various ANNs. To get a deeper understanding, we plan to visualize the features learned on each layer and in each node of the network. We hope that the release of a standard scalable set of 3D training data will empower the community to head in the same or similar direction. In addition, such a set allows evaluating the best performing ANN against *classical* SfM approaches, which are nowadays used for exactly these tasks—the image- and video-based 3D reconstruction of objects. Such an evaluation will only lead to meaningful results if the SfM, as well as the ANN approaches, use the same input data for the 3D object and a comparable output format for the reconstruction process.

Achieving this ambitious goal, further work must be done, which also includes the identification of open issues and the discussion of ANN architectures whether they are promising or misleading. For this purpose, our roadmap of further work consists of five packages which not necessarily have to be processed sequentially:

– Proving a widely diversified set of ANNs which outperform the current baseline on our dataset by a significant value of at least 5%
– Visualization of learned features by each layer and node to understand the general working principle of each designed ANN.
– Developing, generating, and introducing a virtual 3D CAD dataset with a voxel resolution of at least $100 \times 100 \times 100$.
– Developing, generating, and introducing a real 3D object dataset with the corresponding images captured from different viewpoints and including the 3D ground truth data of the object.

– Analyzing if voxel-based multiview 3D reconstruction by ANN can be scaled up, e.g. by the use of pre-trained computer-generated training data, to reach a suitable voxel resolution with available hardware resources.

One forthcoming work package is the generation of an extensive dataset based on 3D CAD objects. Although those objects will introduce an additional level of complexity and hence may be less suited for initial experiments, they definitely deserve attention in the future due to their practical relevance. Thus, we will expand our *images and voxel generator* to allow for any volumetric *obj* 3D object to be converted into a set of w images and a voxel cloud with a certain resolution. Furthermore, we plan to design, train, and test a diversified set of ANNs on our dataset to support our claim that ANN-based 3D reconstruction from images or video sequences is feasible. We encourage everyone to outperform our baseline and to improve image-based 3D reconstruction using ANNs.

References

1. Agisoft: Agisoft PhotoScan. (Jan 2017), http://www.agisoft.ru/
2. Autodesk, Inc.: Autodesk 123D Catch | 3D model from photos. (Jan 2017), http://www.123dapp.com/catch
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE 10(7), 1–46 (2015)
4. Brabandere, B.D., Jia, X., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in Neural Information Processing Systems (NIPS). Curran Associates, Inc. (2016)
5. Brownlee, J.: Machine learning mastery. http://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural (Jan 2017)
6. Chollet, F.: Keras. https://github.com/fchollet/keras (Jan 2017)
7. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In: Computer Vision – ECCV 2016. pp. 628–644. Springer International Publishing (2016)
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. IEEE International Conference on Computer Vision (ICCV) pp. 2650–2658 (2015)
9. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Advances in Neural Information Processing Systems (NIPS), pp. 2366–2374. Curran Associates, Inc. (2014)
10. Elizondo, D., Zhou, S.M., Chrysostomou, C.: Surface reconstruction techniques using neural networks to recover noisy 3D scenes. In: Artificial Neural Networks (ICANN), pp. 857–866. Springer Science + Business Media (2008)
11. Garg, R., B.G., V.K., Carneiro, G., Reid, I.: Unsupervised CNN for single view depth estimation: Geometry to the rescue. In: Computer Vision – ECCV 2016. pp. 740–756. Springer Nature (2016)
12. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
13. González, À.: Measurement of areas on a sphere using fibonacci and latitudelongitude lattices. Mathematical Geosciences 42(1), 49–64 (2009)

14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 1097–1105. Curran Associates, Inc. (2012)
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
16. Li, B., Shen, C., Dai, Y., van den Hengel, A., He, M.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. pp. 1119–1127. Institute of Electrical and Electronics Engineers (IEEE) (2015)
17. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5162–5170 (2015)
18. Liu, G., Yang, C., Li, Z., Ceylan, D., Huang, Q.: Symmetry-aware depth estimation using deep neural networks. arXiv preprint arXiv:1604.06079 (2016)
19. Pan, Q., Reitmayr, G., Drummond, T.: ProFORMA: Probabilistic feature-based on-line rapid model acquisition. British Machine Vision Conference (BMVC) pp. 112.1–112.11 (2009)
20. Peng, L.W., Shamsuddin, S.M.: 3D object reconstruction and representation using neural networks. In: Proceedings of the International Conference on Computer Graphics and iIteractive Techniques in Austalasia and Southe East Asia (GRAPHITE). pp. 139–147. Association for Computing Machinery (ACM) (2004)
21. Pollefeys, M., Nistér, D., Frahm, J.M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.J., Merrell, P., et al.: Detailed real-time urban 3D reconstruction from video. International Journal of Computer Vision 78(2-3), 143–167 (2008)
22. Roy, A., Todorovic, S.: Monocular depth estimation using neural regression forest. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
23. Schöning, J., Heidemann, G.: Evaluation of multi-view 3d reconstruction software. In: Computer Analysis of Images and Patterns (CAIP), pp. 450–461. Springer International Publishing (2015)
24. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. Seventh International Conference on Document Analysis and Recognition (ICDAR) 3, 958–962 (2003)
25. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688 (2016)
26. Vergauwen, M., Van Gool, L.: Web-based 3D reconstruction service. Machine Vision and Applications (MVA) 17(6), 411–426 (2006)
27. Waller, L., Tian, L.: Computational imaging: Machine learning for 3D microscopy. Nature 523(7561), 416–417 (2015)
28. Wang, X., Fouhey, D.F., Gupta, A.: Designing deep networks for surface normal estimation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 539–547 (Jun 2015)
29. Wu, C.: VisualSFM: a visual structure from motion system (Jan 2011), http://ccwu.me/vsfm/
30. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d ShapeNets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Institute of Electrical and Electronics Engineers (IEEE) (2015)