Structure from Neuronal Networks (SfN²)

Julius Schöning*, Gunther Heidemann, Ulf Krumnack

Institute of Cognitive Science, Osnabrück University, Wachsbleiche 27, DE-49090 Osnabrück, Germany.

* Corresponding author. Tel.: +49 541 969 2706; email: juschoening@uos.de

Manuscript submitted December 9, 2017; accepted March 4, 2018.

doi: 10.17706/jcp.13.8.988-999

Abstract: Multiple View Geometry (MVG) with its underlying mathematical principle is mainly used for 3D reconstruction. The most common approaches based on MVG are the Structure from Motion (SfM) methods which create 3D point clouds from a collection of images or video frames. The emerging use of artificial neural networks (ANNs) in almost every domain leads to the question, if ANNs can learn the underlying mathematical geometric mappings of SfM pipelines? To answer this question, three different ANN architectures based on the three different key point matching strategies of SfM were benchmarked. Since we want to learn the mathematical, geometrical mapping of SfM approaches and not the categories or shapes of natural 3D objects, we trained and tested our ANNs on 2D projections of random 3D shapes built from small random cubes. For 3D shapes with a grid size of $\langle 3,3,3 \rangle$ voxels, all architectures show a high prediction accuracy of the reconstructed shape. When scaling up the grid size of the 3D cubes, we recognize a significant decrease in accuracy. These initial results show that all of the different ANN architectures we considered can learn to reconstruct unknown 3D shapes from images. In a more detailed analysis of our results, we investigate how the choice of architecture influences the prediction accuracy of the 3D shape on voxel and overall shape level and if non-occluded voxels can be predicted independently of scale. Finally, we discuss, if a voxel-based representation of the 3D shape can be scaled to a useful technical resolution due to its high impact on the size of the ANN as well as the required training data.

Key words: 3D reconstruction, artificial neural networks, multiple view geometry, structure from motion.

1. Introduction

Image-based 3D reconstruction and all multiview stereo approaches are usually realized with the mathematical principle of MVG. Almost all approaches use the principle of SfM to estimate the 3D geometry of an, e.g., object from different 2D projections showing the object from different perspectives. The resulting reconstructed 3D objects or scenarios are usually represented by a set of points—a point cloud—in 3D space. Thus, MVG solves nonlinear least squares problems for creating the point cloud of an object. Commercial 3D reconstruction software like *PhotoScan* [1], or *123D Catch* [2] as well as state of the art academic approaches e.g. *ARC 3D* [3], or *VisualSfm* [4] show good accuracy measurements in benchmarks [5]. Thus, the accuracy of their reconstructed 3D point cloud with respect to the ground truth is mostly sufficient for the specific use case. One major drawback of these approaches is that the reconstruction accuracy depends on the number of 2D projections and thus exponentially impacts the processing time for the MVG in the SfM pipeline.

Overcoming this drawback, we are going to develop, train and test ANNs for image-based 3D reconstruction. Hence, the first question to answer is, if ANNs can learn the underlying mathematical

geometric mappings of SfM pipelines. Note, that the *underlying mathematical principle* should be learned and not the usual shapes of 3D object classes. Thus we are introducing (cf. Section 0) a scalable dataset of 2D projections of arbitrary 3D shape. This dataset allows a systematic investigation of ANNs for MVG and gives insights, which parts of the ANNs are important as well as which hyperparameter of the ANNs performs best. However, to bridge the gap between the classical SfM pipeline and Structure from Neural Networks (SfN²) we designed in this paper, ANN architectures are inspired by the three most important strategies [6] of SfM pipelines—*global*, *hierarchical*, and *incremental* key point matching. Ensuring that the different architectures of the global, hierarchical, and incremental ANNs are comparable, all ANNs have the same number of parameters, i.e. weight. The design of these ANNs is detailly described in Section 5. The number of parameter scales by the output grid size as well as the number of filters and kernel size of each convolutional layer.

The results in Section 6 shows that for 3D shapes with a grid size of $\langle 3,3,3 \rangle$ voxels, all architectures show a high prediction accuracy of the reconstructed shape. When scaling up the grid size of the 3D cubes to $\langle 4,4,4 \rangle$ or $\langle 8,8,8 \rangle$ voxels, we recognize a significant decrease in accuracy. These first results evidence that all of the different ANN architectures we considered can learn to reconstruct unknown 3D shapes from images. In a more detailed analysis of our results, we investigate how the choice of architecture influences the prediction accuracy of the 3D shape on voxel and overall shape level and if non-occluded voxels can be predicted independently of scale. Finally, Section 7 discusses, if a voxel-based representation of the 3D shape can be scaled to a useful technical resolution due to its high impact on the size of the ANN as well as the required training data.

2. State of the Art

The reconstruction of known 3D objects with ANNs is still in its infancy, but recently a few approaches [7], [8] were introduced. Reconstruction of *unknown* 3D objects by learning the underlying mathematical principle of MVG mainly used for 3D reconstruction is, to our knowledge, not yet investigated a lot. Hence, our approach, just like any SfM approach works with unknown categories, we summarize in this section related topics relevant in this context like depth map generation as well as prediction, 3D object recognition, stereo image generation, and single-view 3D reconstruction.

For depth prediction, Eigen *et al.* [9] estimate the surface normal and perform semantic labeling by applying three increasing stacks of convolutional neural networks (CNNs) at the images. The smallest stack predicts a set of coarse features for the entire image, the intermediate stack produces mid-level resolution predictions and the largest stack leads to the predictions to higher resolutions. Thus, the depth estimation of this work is based on a multi-scale deep network [10]. Wang *et al.* [11] implement top-down and bottom-up CNNs for surface normal estimation. Taking the whole image as input, the top-down CNN captures the coarse structures which cannot be decoded by local evidence alone. On local patches extracted from the image, the bottom-up CNN captures local evidence at a higher resolution. In a fusion network, the outputs of these two networks are combined in a way that the fused network learns how to incorporate the predictions and returns the surface normal.

In 2016, a single-view 3D reconstruction method is proposed by Liu *et al.* [12]. This method uses a CNNs to estimate depth, normal, and symmetry correspondence for every pixel. Rectifying the depth information, they set up the symmetry correspondences as an optimization problem in their network. Based on Liu *et al.* work, the work [13] estimates depth on single images which are over-segmented into superpixels. Therefore a CNN is introduced to learn unary and binary potentials for the continuous conditional random field layer. Considering the estimation of depth as a regression problem, Li *et al.* [14] propose a method that extracts multi-scale image patches around the superpixel centers. A CNN learns the relationship between

these patches for depth estimation.

A convolutional regression forest, where each node in the forest is associated with a CNN, is presented by Roy *et al.* [15]. The CNN makes a depth estimation with a corresponding probability per pixel based on a sliding window around each pixel. By using only pairs of images with a small camera motion between themselves, Garg *et al.* [16] propose an unsupervised framework to train a deep ANN for single-view depth prediction.

Already back in 2004, Peng *et al.* [17] showed that object reconstruction by an ANN is more accurate than object generation by third order polynomials. The ANNs they used in their experiments have been entirely based on multilayer feedforward network designs.

3. Structure from Motion by ANNs

For, e.g., object recognition tasks [18]–[20], nowadays, images as input for ANNs have become quite common. For these recognition tasks, one usually defines the output layer of an ANN so that each output neuron represents a single object category. In computer graphics and computer vision, however, the shape of 3D objects is typically described by vertices, edges, and faces which differ depending on the complexity of the object. For the first computer games, a simplified representation of 3D objects for computers was introduced, which uses voxels (also known as volumetric pixels). Thus, one has to choose a suitable representation for 3D objects when designing an ANN to output 3D information. As consequence, either i) vertices, ii) edges, iii) faces, or iv) voxels of the object must be encoded by the output layer of the ANN.

Under consideration of ANN's performance in binary classification tasks, we chose a voxel-based representation of the 3D objects for the output layer. This kind of representation is also selected by related work [8], [7]. Also, voxel-based representation has the advantage that the resolution and therefore its accuracy, as well as its computational complexity, can be scaled. As opposed to the ANN's output and as it is common for any SfM approach, the input consists of images of the object captured from different viewpoints. These considerations led to the definition of the input as well as output architecture for ANN-based 3D reconstruction. The idea behind the input and output definition is to establish a scalable basis for higher resolution results. Thus, like in digital imaging, a general working architecture with a low resolution can be scaled to a higher resolution with constantly improving hardware. Therefore, we designed our dataset with low resolutions, but with an option to scale it up.

Table 1. Extract from the Cube Dataset Showing Randomly Picked Objects in Different Setups

setup	12 different viewpoints of the cube object
$\langle 3,3,3 \rangle$	cube 090000 with its subcube pattern (1,0,1,0,1,1,1,0,0,0,0,0,1,1,0,0,1,0,1,1,1,0,1,1,0)
$\langle 4,4,4 \rangle$	cube 144400 with its subcube pattern (0,1,0,0,0,0,0,0,1,0,0,...,1,1,1,1,0,1,1,0,0,0,0,0)
$\langle 8,8,8 \rangle$	cube 010131 with its subcube pattern (0,1,1,0,0,1,0,1,1,1,0,...,0,0,0,1,1,0,1,1,0,0,0,0)

4. Dataset

As a first idea for providing the necessary amount of training and test samples to train deep ANNs, we considered a database generator based on a CAD definition file format like Wavefront obj. Since a dataset, e.g. downloaded from the internet, of a variety of 3D obj objects is composed of ordinary as well as complex objects at the same time, and because the object's complexity is quite difficult to quantize, we address oneself to designing a more uniform dataset. Thus, we suggested a cube-based dataset—for exemplary objects see Table 1. This dataset is generated in two steps. In the first step, a *cube generator*¹ computes cubes in a $\langle r, r, r \rangle$ space with a random but unique voxel constellation and stores them as 3D *obj* objects. The *images and voxel generator*, in the second step, creates w images showing these objects from different viewpoints. Further, the *images and voxel generator* compute and store the corresponding voxel cloud of the 3D *obj* objects in a defined resolution.

4.1. Cube Generator

Written in Matlab, this generator¹ randomly generates n 3D objects. By taking a unit cube in $\mathbb{R}^3$ and subdividing it into $\langle r, r, r \rangle$ subcubes (voxels), each such object is created. The user can define the parameter r for scaling the complexity of the objects. Taking the uniqueness of the cube distribution in the $\langle r, r, r \rangle$ grid into consideration, our generator is able to generate and export 2^{r^3} different 3D objects as 3D *obj* files. Using random binary values for every subcube, each 3D cube is generated by putting the subcubes into the $\langle r, r, r \rangle$ grid. Where two or more subcubes are connected with each other, duplicate vertices are combined and inner faces are deleted, before exporting the cubes as 3D *obj* objects.

4.2. Images and Voxel Generator

The images and voxel generator, available in Matlab¹ and Python generate w images and a voxel cloud from any 3D object stored in the *obj* format. For scalability, the user can define the number of images w taken from different viewpoints, their pixel resolutions $\langle x, x \rangle$, and the size v of the cubic voxel cloud $\langle v, v, v \rangle$.

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 [21]. From these viewpoints, gray scale images with a resolution of $\langle x, x \rangle$ 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 $\langle v, v, v \rangle$ grid, so that all voxels laying inside the object are defined as cubes.

4.3. Static Cube Datasets

As static benchmarks and in addition to the cube generators which generates random subcube setups, we have released three datasets—one with 3, 4, and 8 as v size, cf. Table 1. Ensuring the reproducibility of the proposed networks and their results, we released 100,000 different objects for the $\langle 3, 3, 3 \rangle$ setup, 300,000 different objects for the $\langle 4, 4, 4 \rangle$ setup, and 430,000 different objects for the $\langle 8, 8, 8 \rangle$ setup as static snapshots. The other user defined values are set to $w = 12$ different viewpoints and $\langle x, x \rangle = \langle 100, 100 \rangle$ pixel.

5. Global, Hierarchical, and Incremental ANNs for 3D Reconstruction

The designed ANNs should be able to acquire 3D voxel models out of 12 images, taken from different

¹At the project webpage <https://ikw.uos.de/%7Ecw/projects/sfnn>, all generator tools and the dataset could be downloaded. In addition, a supplementary document detailed describe the ANN architectures and results. Further, video visualization of the accuracy of each voxel over 1000 iterations can be watched.

viewpoints (cf. Table 1). The 3D voxel models are defined as a binary voxel grid where the values are either 0 when avoxel is not filled or 1 when avoxel is filled and visible.

We design three different ANNarchitectures¹ trying to emulate the SfM strategies of global, hierarchical, and incremental key point matching, cf. scheme of the strategy in Table 2. Ensuring the comparability, all these three architectures have the same number of parameter—weights. By 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 create the three different architectures out of an input layer containing 12 images as a 2D matrix, several convolutional layers which are used for creating the architectural differences and as last three layers before the output layer, a dropout (20%), pooling, and a fully connected layer is used. The last three layers are the same in every architecture. The dropout layer is only implemented for creating a more robust network.

Table 2. Results of the Global, Hierarchical, and Incremental ANN on the Synthetic Cube Dataset in the Different Setups $\langle 3,3,3 \rangle$, $\langle 4,4,4 \rangle$, and $\langle 8,8,8 \rangle$. The Accuracy of Correct Predicted Voxels is Significantly Dropping by Increasing the Size of the Voxel Cube. It can be Seen that All Strategies Perform almost Identically on this Task

(a) 1st iteration of training

(b) 361st iteration of training

Fig. 1. Voxel accuracy after different iteration of training (a) – (c) of the ANNs with a filter size of 32. The voxel accuracy is color coded from 0.5 random chance–blue to 1.0 every time right–dark red. The video1 of all 1000 iterations visualized in detail at which iteration a certain voxel is learned by the ANN.

The activation functions of all nodes are *rectifiers*. For testing the impact of the filter size on the result, the filter size is varied at a fixed filter kernel size of 5×5 . For covering a wide scale of filter sizes, filter sizes based on the power of two are selected (8, 16, 32, 64, and 128). Note, the ANNs are designed in that way that they still have the same number of weights for all values of filter and kernel size.

Attuning the available hardware resources, the data was preprocessed. The input images were downsampled from $\langle 100, 100 \rangle$ to $\langle 20, 20 \rangle$ pixels. We then stored all objects with their 12 images and their corresponding voxel matrix to an *HDF5* file to make the objects easier to load directly from hard disk to relax the working memory consumption during training, testing and validating. During the entire benchmark, the first 30% of cubes are used as training samples, the next 50% of cubes are used as testing samples, and the last 20% are used as validation samples. All ANNs settings were trained over 1000 epochs—iterations.

6. Results

The best performance we achieved during the benchmarking of our dataset is a voxel accuracy of 98.04% which leads to 58.60% completely correct reconstructed cubes. As seen in Table 3, this result is achieved by the global ANN learned and validated on the $\langle 3, 3, 3 \rangle$ dataset. Table 3 also summarizes the percentage of correct predicted visible voxels. In Table 2, the voxel accuracy over the epochs for any of the three architectures on the $\langle 3, 3, 3 \rangle$, $\langle 4, 4, 4 \rangle$, and $\langle 8, 8, 8 \rangle$ setup for both the testing and training samples are plotted. A comparison of the impact of the hyperparameter filter size to the cube accuracy, where all voxels of a cube are predicted correctly, is illustrated in Table 4. The footprint on the hard disk of all trained ANNs varied from 1MB to 160MB per network depending on the filter size and the used cube setup. The training for all architectures on all setups were done until no significant decrease of the loss over 200 iterations happened or after 1,000 iterations.

Table 3. Best Results on the Accuracy of All Voxels of the Different ANN Strategies by Varying the Filter Size.

Next, to the Value of the Best Accuracy of All Voxels, the Accuracy of All Visible Voxels, the Percentage of Complete, Correct Cubes and the Associated Epoch is Listed. The Best Result of Each Strategy is Highlighted

	filter	$\langle 3, 3, 3 \rangle$				$\langle 4, 4, 4 \rangle$				$\langle 8, 8, 8 \rangle$			
		voxel	vis. voxel	cube	epoch	voxel	vis. voxel	cube	epoch	voxel	vis. voxel	cube	epoch
global	8	96.00%	96.11%	29.73%	489	92.97%	94.00%	1.72%	986	61.78%	67.09%	0.00%	931
	16	95.90%	96.03%	28.38%	650	92.93%	94.56%	1.32%	986	57.78%	60.64%	0.00%	989
	32	97.67%	97.84%	53.64%	360	94.42%	95.56%	4.35%	366	57.91%	60.74%	0.00%	998
	64	97.82%	98.03%	55.85%	135	94.37%	95.57%	3.95%	89	56.52%	60.65%	0.00%	914
	128	98.04%	98.23%	58.60%	31	94.54%	95.73%	4.23%	41	61.16%	68.03%	0.00%	995
hierarchical	8	97.59%	97.75%	52.86%	985	92.05%	93.03%	0.68%	993	66.74%	73.94%	0.00%	841
	16	97.86%	98.03%	56.36%	411	93.58%	94.60%	1.92%	955	65.24%	72.54%	0.00%	947
	32	98.03%	98.20%	58.77%	122	93.70%	95.03%	2.37%	894	55.60%	59.47%	0.00%	994
	64	97.61%	97.93%	52.34%	65	93.66%	95.06%	2.26%	181	56.24%	60.15%	0.00%	470
	128	97.47%	97.74%	50.69%	29	91.51%	92.96%	0.70%	141	50.01%	50.09%	0.00%	4
incremental	8	97.66%	97.81%	52.99%	584	92.37%	94.06%	1.00%	962	59.49%	65.22%	0.00%	647
	16	97.84%	98.00%	56.10%	598	90.85%	92.90%	0.35%	965	61.40%	67.98%	0.00%	966
	32	95.64%	95.99%	32.26%	323	88.43%	90.74%	0.12%	950	59.19%	64.32%	0.00%	967
	64	88.93%	89.14%	4.92%	299	80.43%	82.13%	0.00%	89	59.70%	65.16%	0.00%	104
	128	83.49%	81.68%	0.62%	33	92.78%	93.71%	0.94%	53	54.88%	58.08%	0.00%	8

Table 4. Impact of the Hyperparameter Filter Size and its Impact on the Cube Correctness on the $\langle 3, 3, 3 \rangle$ Setup. The Reached Percentage of Correctly Reconstructed Cubes Varies between the Set Number of Filters. Note that the Number of Filters Significantly Changes the Number of Oscillations and the Iteration when a Plateau Occurred

For investigating how well a voxel at a certain position was learned, and if there are blind spots, the accuracy for each voxel is calculated. Fig. 1 exemplarily shows the average accuracy of the ANN strategies after different iterations of training (a)–(c). The investigating of e.g. oscillations of certain voxels over epochs, videos visualizing all 1,000 epochs of all different setups were rendered. Plots of the confusion values and the average accuracy of visible voxels over the iterations as well as all the plots of all setups can be found in the supplementary material on our homepage1.

7. Discussion

The results indicate that ANNs can learn the underlying mathematical principle of MVG and can reconstruct voxel clouds of unknown 3D objects from multiple images captured from the same perspectives. However, are the resulting voxel clouds accurate enough and can they be scaled to a size where they are competitive to standard SfM approaches?

During the preparation of this dataset and while designing, training as well as testing these ANNs, our estimations on the expected accuracy varied from about 60% up to 98% object accuracy—both extreme expectations were reached. The standard processing pipeline of image-based 3D reconstruction can be seen as a concatenation of linear and nonlinear algorithms, thus in theory ANNs should be able to learn the underlying mathematical principle of image-based 3D reconstruction. Further, we expect that this could be learned in a few iterations only. Nevertheless, Table 4, indicates that at least 200 iterations of learning must be done to overcome the first plateau. By numbers, cf. Table 3 columns voxel and cube accuracy, the MVG problem is not solved as perfectly as one might expect. Further, by increasing the voxel count, the value of complete cube accuracy moves exponentially towards zero. A possible reason why classical SfM approaches [4], [1], [22], [23] seem to scale better, is that they reconstruct vertices and faces instead of reconstructing the object as voxel space. Thus they must not compute or predict values for hidden inner object voxels.

By neglecting all nonvisible voxels (cf. Table 3 column visible voxel accuracy), only a slight decrease with respect to the count of the visible voxels can be seen. Note: the drop in the visible voxel accuracy between the $\langle 4,4,4 \rangle$ and $\langle 8,8,8 \rangle$ setup might be caused by downscaling the input images so that two or more voxels might be represented by one pixel in the 2D projection only. Focusing on the visible voxels, Figure 1 shows, which positions are harder to predict in the voxel space. As expected, it can be seen that the corner voxels of the cubes will be trained in an early iteration (cf. Fig. 1(a)) and will be predicted with a high accuracy. Based on the fact that in the center of the visible hull can be blocked from view by neighboring voxels, the prediction accuracy drops.

On all results, we do not recognize significant differences between the global, hierarchical, and incremental ANNs regarding the voxel, visible voxel and cube accuracy, cf. Table 3. Especially seen in Table 4 the accuracy curves different in the oscillation and plateau over epoch's behavior. The filter size, as one hyperparameter, strongly influences these effects. Because there are no significant differences between the ANNs, the global architecture might be preferable for further investigation if SfN² for unknown objects from every perspective can be developed due to its easy scheme.

Even on our uncomplicated cube dataset, the used ANNs face difficulties. In the ongoing more thorough

analysis of ANNs trained on this dataset, we expect further insights. It may turn out, that 2D convolutions are not the appropriate network layer for this 3D reconstruction scenario and that other kind of layer e.g. 3D convolutions are more suitable for solving this task. Layer-wise relevance propagation [24] as visualization method may help to identify problematic regions in the input images. Rest on these visualizations, changes to the design of the ANN can be systematic performed. Furthermore, one may consider using the classification, in a non-binary manner for e.g. octree partitioning. Instead, train all the detail of a 3D object, an octree-based ANN first reconstruct the general shape of the object and the detailed areas. Such kind of network might work with convolutional layers only all trained on other octree levels, similar to the design of Eigen et al.[9].

8. Conclusion

Learning the underlying mathematical principles of MVG with ANNs, this paper introduces three different SfN² architectures inspired by the three most important strategies of classical SfM pipelines—the global, hierarchical, and incremental key point matching. Since the mathematical, geometrical mapping of SfM approaches and not the categories or shapes of natural 3D objects should be learned a scalable synthetic 3D cube dataset is introduced as well. The results indicate that ANNs can learn the underlying mathematical principle of MVG and can reconstruct voxel clouds of unknown 3D objects from multiple images captured from the same perspectives. However, the resulting voxel clouds are still, lack of accuracy especially by scaling up the 3D voxel space. We show that image-based 3D reconstruction of unknown objects by an ANNs is possible and can reach a voxel accuracy of 98%. Based on the results and the fact that we do not notice differences between the designed ANNs, we expect that further research can achieve a significant improvement in the performance. These enhancements can be done, e.g. through another design of the layers, as well as its connections, shared weights within the networks, more training examples, an increased number of the training samples or even through another non-voxel-based representation of the resulting 3D object.

The scalable synthetic 3D cube dataset, used here, should form the basis for developing, designing, training, testing, and benchmark various ANNs for the task of image-based 3D reconstruction. To get a deeper understanding, we plan to expand the existing video visualizations¹ over the epochs and visualized the features learned on each layer and in each node of the network. Besides our work, datasets which allow the evaluation of ANN based 3D reconstruction against classical SfM approaches are needed. Only such datasets will lead to meaningful benchmark results because the SfM and the SfN² approaches must use the same input images and must their results must be compared against the same 3D ground truth.

Establishing ANNs in the field of 3D reconstruction for e.g. overcoming the drawback of high processing times, further work must be done. Therefore the identification of open issues must be expedited, which leads to the discussion whether or not ANNs for 3D reconstruction are promising or misleading. For this purpose, our roadmap for further work consists of three packages which must not be processed sequentially: i) understand the general working principle of each designed ANN by visualizing of learned features; ii) introducing a real 3D object dataset with the corresponding images captured from different viewpoints and including the 3D ground truth data of the object; iii) proving 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.

The generation of an extensive, but in v , w , and x scalable dataset based on 3D CAD objects is one forthcoming work package. Thus, for allowing any volumetric *obj* 3D object to be converted into a dataset sample, we will expand our *images and voxel generator*. 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.

Acknowledgment

The training and validation of the different ANNs were performed on a SGI UV2000 granted by the German Research Foundation (DFG) <http://gepris.dfg.de/gepris/projekt/239246210> as well as on a Titan XMax well donated by NVIDIA Corporation.

References

- [1] Agisoft. Agisoft *PhotoScan*. Retrieved from <http://www.agisoft.ru/>
- [2] Autodesk, Inc. *Autodesk 123D Catch | 3D model from photos*. Retrieved from <http://www.123dapp.com/catch>
- [3] Vergauwen, M., & Van Gool, L. (2006). Web-based 3D reconstruction service. *Machine Vision and Applications (MVA)*, 17(6), 411–426.
- [4] Wu, C. *VisualSFM: A Visual Structure from Motion System*. Retrieved from <http://ccwu.me/vsfm/>
- [5] Schöning, J., & Heidemann, G. (2015). Evaluation of multi-view 3D reconstruction software. *Computer Analysis of Images and Patterns (CAIP)*, 450–461.
- [6] Schonberger, J. L., & Frahm, J. M. (2016). Structure-from-motion revisited. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [7] Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. (2016). 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. *Computer Vision — ECCV*, 628–644.
- [8] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [9] Eigen, D., & Fergus, R. (2015). Predicting depth, surface normal and semantic labels with a common multi-scale convolutional architecture. *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2650–2658.
- [10] Eigen, D., Puhersch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in Neural Information Processing Systems (NIPS)*, 2366–2374.
- [11] Wang, X., Fouhey, D. F., & Gupta, A. (2015). Designing deep networks for surface normal estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 539–547.
- [12] Liu, G., Yang, C., Li, Z., Ceylan, D., & Huang, Q. (2016). *Symmetryaware Depth Estimation Using Deep Neural Networks*.
- [13] Liu, F., Shen, C., & Lin, G. (2015). Deep convolutional neural fields for depth estimation from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5162–5170.
- [14] Li, B., Shen, C., Dai, Y., van den Hengel, A., & He, M. (2015). Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFS. *Institute of Electrical and Electronics Engineers (IEEE)*.
- [15] Roy, A., & Todorovic, S. (2016). Monocular depth estimation using neural regression forest. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [16] Garg, R., BG, V. K., Carneiro, G., & Reid, I. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. *Computer Vision – ECCV*, 740–756.
- [17] Peng, L. W., & Shamsuddin, S. M. (2004). 3D object reconstruction and representation using neural networks. *Proceedings of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE)*, 139–147.

- [18] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 1097-1105.
- [19] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [20] Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, 3, 958-962.
- [21] González, À. (2009). Measurement of areas on a sphere using fibonacci and latitude/longitude lattices. *Mathematical Geosciences*, 42(1), 49-64.
- [22] Pan, Q., Reitmayr, G., & Drummond, T. (2009). ProFORMA: Probabilistic feature-based on-line rapid model acquisition. *British Machine Vision Conference (BMVC)*.
- [23] Pollefeys, M., Nistér, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P. (2008). Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision*, 78(2-3), 143-167.
- [24] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 1-46.

Julius Schöning received his master of science in intelligent embedded microsystems from the University of Freiburg in 2013. Since 2009 and in parallel, he worked as software requirement engineer for HMI systems for agriculture machines like tractors and combine harvesters. In 2014, Julius joined the Institute of Cognitive Science at Osnabrück University to work on his Ph.D Julius' research interests include 3D reconstruction, visualization, user interfaces, and computer vision. The main focus of his research is dedicated to developing new interactive 3D modeling methods to generate CAD-like models from video footage.

Gunther Heidemann received the diploma degree in physics from the University of Karlsruhe, and the doctorate degree in computer science from the University of Bielefeld, Germany. He held positions as assistant professor of computer engineering at the Florida State University and the Florida A&M University in Tallahassee, and as professor of computer science at the University of Stuttgart, Germany. Currently, he is a professor of cognitive science at Osnabrück University, Germany.

Ulf Krumnack studied computational linguistics and artificial intelligence as well as mathematics and received his doctorate degree in cognitive science in 2015 from the Osnabrück University, Germany. He currently works as a research assistant in the workgroup on biologically inspired computer vision at the Institute of Cognitive Science in Osnabrück, focusing on the analysis of features learned in deep neural networks.