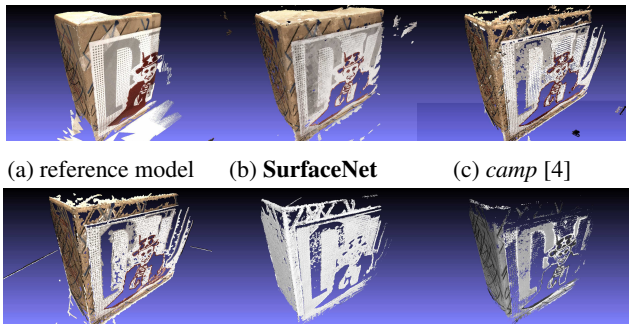
SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis

Mengqi Ji*[1], Juergen Gall[3], Haitian Zheng[2], Yebin Liu[2] and Lu Fang†[2]

[1]Hong Kong University of Science and Technology, Hong Kong, China
[2]Tsinghua University, Beijing, China
[3]University of Bonn, Bonn, Germany

Abstract

This paper proposes an end-to-end learning framework for multiview stereopsis. We term the network SurfaceNet. It takes a set of images and their corresponding camera parameters as input and directly infers the 3D model. The key advantage of the framework is that both photo-consistency as well geometric relations of the surface structure can be directly learned for the purpose of multiview stereopsis in an end-to-end fashion. SurfaceNet is a fully 3D convolutional network which is achieved by encoding the camera parameters together with the images in a 3D voxel representation. We evaluate SurfaceNet on the large-scale DTU benchmark.



(a) reference model (b) **SurfaceNet** (c) *camp* [4]

(d) *furu* [8] (e) *tola* [27] (f) *Gipuma* [9]

Figure 1: Reconstruction of the model 13 of the DTU dataset [1] in comparison to [4, 8, 27, 9]. Our end-to-end learning framework provides a relatively complete reconstruction.

1. Introduction

In multiview stereopsis (MVS), a dense model of a 3D object is reconstructed from a set of images with known camera parameters. This classic computer vision problem has been extensively studied and the standard pipeline involves a number of separate steps [4, 27]. In available multiview stereo pipelines, sparse features are first detected and then propagated to a dense point cloud for covering the whole surface [8, 11], or multiview depth maps are first computed followed with a depth map fusion step to obtain the 3D reconstruction of the object [15, 17]. A great variety of approaches have been proposed to improve different steps in the standard pipeline. For instance, the works [27, 4, 9] have focused on improving the depth map generation using MRF optimization, photo-consistency enforcement, or other depth map post-processing operations like denoising or interpolation. Other approaches have focused on more advanced depth fusion algorithms [13].

Recent advances in deep learning have been only partially integrated. For instance, [32] uses a CNN instead of hand-crafted features for finding correspondences among image pairs and [10] predicts normals for depth maps using a CNN, which improves the depth map fusion. The full potential of deep learning for multiview stereopsis, however, can only be explored if the entire pipeline is replaced by an end-to-end learning framework that takes the images with camera parameters as input and infers the surface of the 3D object. Apparently, such end-to-end scheme has the advantage that photo-consistency and geometric context for dense reconstruction can be directly learned from data without the need of manually engineering separate processing steps.

Instead of improving the individual steps in the pipeline, we therefore propose the first end-to-end learning framework for multiview stereopsis, named *SurfaceNet*. Specifically, SurfaceNet is a 3D convolutional neural network that can process two or more views and the loss function is directly computed based on the predicted surface from all available views. In order to obtain a fully convolutional network, a novel representation for each available viewpoint, named *colored voxel cube* (CVC), is proposed to im-

*mji@connect.ust.hk
†luvision.net@gmail.com

plicitly encode the camera parameters via a straightforward perspective projection operation outside the network. Since the network predicts surface probabilities, we obtain a reconstructed surface by an additional binarization step. In addition, an optional thinning operation can be applied to reduce the thickness of the surface. Besides of binarization and thinning, SurfaceNet does not require any additional post-processing or depth fusion to obtain an accurate and complete reconstruction.

2. Related Works

Works in the multiview stereopsis (MVS) field can be roughly categorised into volumetric methods and depth maps fusion algorithms. While earlier works like space carving [14, 22] mainly use a volumetric representation, current state-of-the-art MVS methods focus on depth map fusion algorithms [27, 4, 9], which have been shown to be more competitive in handling large datasets in practice. As our method is more related to the second category, our survey mainly covers depth map fusion algorithms. A more comprehensive overview of MVS approaches is given in the tutorial article [7].

The depth map fusion algorithms first recover depth maps [30] from view pairs by matching similarity patches [2, 18, 33] along epipolar line and then fuse the depth maps to obtain a 3D reconstruction of the object [27, 4, 9]. In order to improve the fusion accuracy, [4] mainly learns several sources of the depth map outliers. [27] is designed for ultra high-resolution image sets and uses a robust decriptor for efficient matching purposes. The *Gipuma* algorithm proposed in [9] is a massively parallel method for multiview matching built on the idea of patchmatch stereo [3]. Aggregating image similarity across multiple views, [9] can obtain more accurate depth maps. The depth fusion methods usually contain several manually engineered steps, such as point matching, depth map denoising, and view pair selection. Compared with the mentioned depth fusion methods, the proposed SurfaceNet infers the 3D surface with thin structure directly from multiview images without the need of manually engineering separate processing steps.

The proposed method in [8] describes a patch model that consists of a quasi-dense set of rectangular patches covering the surface. The approach starts from a sparse set of matched keypoints that are repeatedly expanded to nearby pixel correspondences before filtering the false matches using visibility constraints. However, the reconstructed rectangular patches cannot contain enough surface detail, which results in small holes around the curved model surface. In contrast, our data-driven method predicts the surface with fine geometric detail and has less holes around the curved surface.

Convolutional neural networks have been also used in the context of MVS. In [10], a CNN is trained to predict the normals of a given depth map based on image appearance. The estimated normals are then used to improve the depth map fusion. Compared to its previous work [9], the approach increases the completeness of the reconstruction at the cost of a slight decrease in accuracy. Deep learning has also been successfully applied to other 3D applications like volumetric shape retrieval or object classification [28, 25, 16, 19]. In order to simplify the retrieval and representation of 3D shapes by CNNs, [24] introduces a representation, termed geometry image, which is a 2D representation that approximates a 3D shape. Using the 2D representation, standard 2D CNN architectures can be applied. While the 3D reconstruction of an object from a single or multiple views using convolutional neural networks has been studied in [26, 5], the approaches focus on a very coarse reconstruction without geometric details when only one or very few images are available. In other words, these methods are not suitable for MVS. The proposed SurfaceNet is able reconstruct large 3D surface models with detailed surface geometry.

3. Overview

We propose an end-to-end learning framework that takes a set of images and their corresponding camera parameters as input and infers the 3D model. To this end, we voxelize the solution space and propose a convolutional neural network (CNN) that predicts for each voxel x a binary attribute $s_x \in \{0, 1\}$ depending on whether the voxel is on the surface or not. We call the network, which reconstructs a 2D surface from a 3D voxel space, *SurfaceNet*. It can be considered as an analogy to object boundary detection [29], which predicts a 1D boundary from 2D image input.

We introduce SurfaceNet in Section 4 first for the case of two views and generalize the concept to multiple views in Section 5. In Section 6 we discuss additional implementation details like the early rejection of empty volumes that speed up the computation.

4. SurfaceNet

Given two images I_i and I_j for two views v_i and v_j of a scene with known camera parameters and a voxelization of the scene denoted by a 3D tensor C, our goal is to reconstruct the 2D surface in C by estimating for each voxel $x \in C$ if it is on the surface or not, *i.e.* $s_x \in \{0, 1\}$.

To this end, we propose an end-to-end learning framework, which automatically learns both photo-consistency and geometric relations of the surface structure. Intuitively, for accurate reconstruction, the network requires the images I_i and I_j as well as the camera parameters. However, direct usage of I_i, I_j and their corresponding camera parameters as input would unnecessarily increase the complexity of the network since such network needs to learn the relation be-

tween the camera parameters and the projection of a voxel onto the image plane in addition to the reconstruction. Instead, we propose a 3D voxel representation that encodes the camera parameters implicitly such that our network can be fully convolutional.

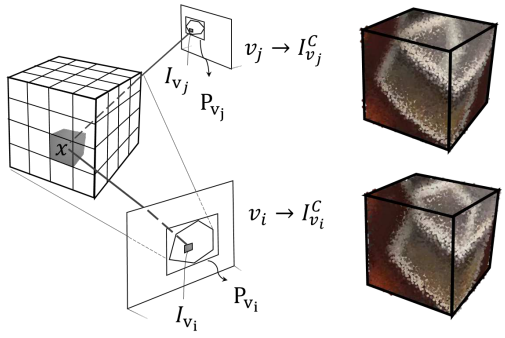


Figure 2: Illustration of two Colored Voxel Cubes (CVC).

We denote our representation as *colored voxel cube* (CVC), which is computed for each view and illustrated in Fig. 2. For a given view v, we convert the image I_v into a 3D colored cube I_v^C by projecting each voxel $x \in C$ onto the image I_v and storing the RGB values i_x for each voxel respectively. For the color values, we subtract the mean color [23]. Since this representation is computed for all voxels $x \in C$, the voxels that are on the same projection ray have the same color i_x. In other words, the camera parameters are encoded with CVC. As a result, we obtain for each view a projection-specific stripe pattern as illustrated in Fig. 2.

4.1. SurfaceNet Architecture

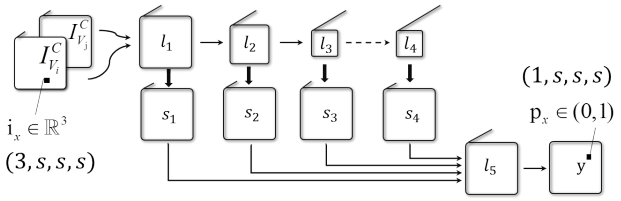


Figure 3: SurfaceNet takes two CVCs from different viewpoints as input. Each of the RGB-CVC is a tensor of size $(3, s, s, s)$. In the forward path, there are four groups of convolutional layers. The $l_{4.\cdot}$ layers are 2-dilated convolution layers. The side layers s_i extract multi-scale information, which are aggregated into the output layer y that predicts the on-surface probability for each voxel position. The output has the size $(1, s, s, s)$.

The architecture of the proposed SurfaceNet is shown in Fig. 3. It takes two colored voxel cubes from two different viewpoints as input and predicts for each voxel $x \in C$ the confidence $p_x \in (0, 1)$, which indicates if a voxel is on the surface. While the conversion of the confidences into a surface is discussed in Section 4.3, we first describe the network architecture.

The detailed network configuration is summarized in Table 1. The network input is a pair of colored voxel cubes, where each voxel stores three RGB color values. For cubes with s^3 voxels, the input is a tensor of size $6 \times s \times s \times s$. The basic building blocks of our model are 3D convolutional layers $l(\cdot)$, 3D pooling layers $p(\cdot)$ and 3D up-convolutional layers $s(\cdot)$, where $l_{n.k}$ represents the kth layer in the nth group of convolutional layers. Additionally, a rectified linear unit (ReLU) is appended to each convolutional layer l_i and the sigmoid function is applied to the layers s_i and y. In order to decrease the training time and increase the robustness of training, batch normalization [12] is utilized in front of each layer. The layers in l_4 are dilated convolutions [31] with dilation factor of 2. They are designed to exponentially increase the receptive field without loss of feature map resolution. The layer $l_{5.k}$ increases the performance by aggregating multi-scale contextual information from the side layers s_i to consider multi-scale geometric features. Since the network is fully convolutional, the size of the CVC cubes can be adaptive. The output is always the same size as the CVC cubes.

layer name	type	output size	kernel size
input	CVC	$(6, s, s, s)$	-
$l_{1.1}, l_{1.2}, l_{1.3}$	conv	$(32, s, s, s)$	$(3, 3, 3)$
s_1	upconv	$(16, s, s, s)$	$(1, 1, 1)$
p_1	pooling	$(32, \frac{s}{2}, \frac{s}{2}, \frac{s}{2})$	$(2, 2, 2)$
$l_{2.1}, l_{2.2}, l_{2.3}$	conv	$(80, \frac{s}{2}, \frac{s}{2}, \frac{s}{2})$	$(3, 3, 3)$
s_2	upconv	$(16, s, s, s)$	$(1, 1, 1)$
p_2	pooling	$(80, \frac{s}{4}, \frac{s}{4}, \frac{s}{4})$	$(2, 2, 2)$
$l_{3.1}, l_{3.2}, l_{3.3}$	conv	$(160, \frac{s}{4}, \frac{s}{4}, \frac{s}{4})$	$(3, 3, 3)$
s_3	upconv	$(16, s, s, s)$	$(1, 1, 1)$
$l_{4.1}, l_{4.2}, l_{4.3}$	dilconv	$(300, \frac{s}{4}, \frac{s}{4}, \frac{s}{4})$	$(3, 3, 3)$
s_4	upconv	$(16, s, s, s)$	$(1, 1, 1)$
$l_{5.1}, l_{5.2}$	conv	$(100, s, s, s)$	$(3, 3, 3)$
y	conv	$(1, s, s, s)$	$(1, 1, 1)$

Table 1: Architecture of SurfaceNet. A rectified linear activation function is used after each convolutional layer except y, and a sigmoid activation function is used after the up-convolutional layers and the output layer to normalize the output.

4.2. Training

As the SurfaceNet is a dense prediction network, *i.e.*, the network predicts the surface confidence for each voxel, we compare the prediction per voxel p_x with the ground-truth $\hat{s}_x$. For training, we use a subset of the scenes from the DTU dataset [1] which provides images, camera parameters, and reference reconstructions obtained by a structured light system. A single training sample consists of a cube $\hat{S}^C$

cropped from a 3D model and two CVC cubes $I^C_{v_i}$ and $I^C_{v_j}$ from two randomly selected views v_i and v_j. Since most of the voxels do not contain the surface, *i.e.* $\hat{s}_x = 0$, we weight the surface voxels by

$$\alpha = \frac{1}{|\mathcal{C}|} \sum_{C \in \mathcal{C}} \frac{\sum_{x \in C}(1 - \hat{s}_x)}{|C|}, \tag{1}$$

where $\mathcal{C}$ denotes the set of sampled training samples, and the non-surface voxels by $1 - \alpha$. We use a class-balanced cross-entropy function as loss for training, *i.e.* for a single training sample C we have:

$$L(I^C_{v_i}, I^C_{v_j}, \hat{S}^C) = \tag{2}$$
$$- \sum_{x \in C} \left\{ \alpha \hat{s}_x \log p_x + (1 - \alpha)(1 - \hat{s}_x) \log(1 - p_x) \right\}.$$

For updating the weights of the model, we use stochastic gradient descent with Nesterov momentum update.

Due to the relatively small number of 3D models in the dataset, we perform data augmentation in order to reduce overfitting and improve the generalization. Each cube C is randomly translated and rotated and the color is varied by changing illumination and introducing Gaussian noise.

4.3. Inference

For inference, we process the scene not at once due to limitations of the GPU memory but divide the volume into cubes. For each cube, we first compute for both camera views the colored voxel cubes $I^C_{v_i}$ and $I^C_{v_j}$, which encode the camera parameters, and infer the surface probability p_x for each voxel by the SurfaceNet. Since some of the cubes might not contain any surface voxels, we discuss in Section 6 how these empty cubes can be rejected in an early stage.

In order to convert the probabilities into a surface, we perform two operations. The first operation is a simple thresholding operation that converts all voxels with $p_x > \tau$ into surface voxels and all other voxels are set to zero. In Section 6, we discuss how the threshold τ can be adaptively set when the 3D surface is recovered. The second operation is optional and it performs a thinning procedure of the surface since the surface might be several voxels thick after the binarization. To obtain a thin surface, we perform a pooling operation, which we call *ray pooling*. For each view, we vote for a surface voxel $s_x = 1$ if $x = \arg\max_{x' \in R} p_{x'}$, where R denotes the voxels that are projected onto the same pixel. If both operations are used, a voxel x is converted into a surface voxel if both views vote for it during ray pooling and $p_x > \tau$.

5. Multi-View Stereopsis

So far we have described training and inference with SurfaceNet if only two views are available. We now describe how it can be trained and used for multi-view stereopsis.

5.1. Inference

If multiple views $v_1, \ldots, v_V$ are available, we select a subset of view pairs (v_i, v_j) and compute for a cube C and each selected view v the CVC cube I^C_v. We will discuss at the end of the section how the view pairs are selected.

For each view pair (v_i, v_j), SurfaceNet predicts $p_x^{(v_i,v_j)}$, *i.e.* the confidence that a voxel x is on the surface. The predictions of all view pairs can be combined by taking the average of the predictions $p_x^{(v_i,v_j)}$ for each voxel. However, the view pairs should not be treated equally since the reconstruction accuracy varies among the view pairs. In general, the accuracy depends on the viewpoint difference of the two views and the presence of occlusions.

To further identify occlusions between two views v_i and v_j, we crop a 64×64 patch around the projected center voxel of C for each image I_{v_i} and I_{v_j}. To compare the similarity of the two patches, we train a triplet network [20] that learns a mapping $e(\cdot)$ from images to a compact 128D Euclidean space where distances directly correspond to a measure of image similarity. The dissimilarity of the two patches is then given by

$$d_C^{(v_i,v_j)} = \|e(C, I_{v_i}) - e(C, I_{v_j})\|_2, \tag{3}$$

where $e(C, I_{v_i})$ denotes the feature embedding provided by the triplet network for the patch from image I_{v_i}. This measurement can be combined by the relation of the two viewpoints v_i and v_j, which is measured by the angle between the projection rays of the center voxel of C, which is denoted by $\theta_C^{(v_i,v_j)}$. We use a 2-layer fully connected neural network $r(\cdot)$, that has 100 hidden neurons with sigmoid activation function and one linear output neuron followed by a softmax layer. The relative weights for each view pair are then given by

$$w_C^{(v_i,v_j)} = r\left(\theta_C^{(v_i,v_j)}, d_C^{(v_i,v_j)}, e(C, I_{v_i})^T, e(C, I_{v_j})^T\right) \tag{4}$$

and the weighted average of the predicted surface probabilities $p_x^{(v_i,v_j)}$ by

$$p_x = \frac{\sum_{(v_i,v_j) \in \boldsymbol{V}_C} w_C^{(v_i,v_j)} p_x^{(v_i,v_j)}}{\sum_{(v_i,v_j) \in \boldsymbol{V}_C} w_C^{(v_i,v_j)}} \tag{5}$$

where $\boldsymbol{V}_c$ denotes the set of selected view pairs. Since it is unnecessary to take all view pairs into account, we select only N_v view pairs, which have the highest weight $w_C^{(v_i,v_j)}$. In Section 7, we evaluate the impact of N_v.

The binarization and thinning are performed as in Section 4.3, *i.e.* a voxel x is converted into a surface voxel if at least $\gamma = 80\%$ of all views vote for it during ray pooling and $p_x > \tau$. The effects of γ and τ are further elaborated in Section 7.

5.2. Training

The SurfaceNet can be trained together with the averaging of multiple view pairs (5). We select for each cube C, N_v^{train} random view pairs and the loss is computed after the averaging of all view pairs. We use $N_v^{train} = 6$ as a trade-off since larger values increase the memory for each sampled cube C and thus require to reduce the batch size for training due to limited GPU memory. Note that for inference, N_v can be larger or smaller than N_v^{train}.

In order to train the triplet network for the dissimilarity measurement $d_C^{(v_i,v_j)}$ (3), we sample cubes C and three random views where the surface is not occluded. The corresponding patches obtained by projecting the center of the cube onto the first two views serve as a positive pair. The negative patch is obtained by randomly shifting the patch of the third view by at least a quarter of the patch size. While the first two views are different, the third view can be the same as one of the other two views. At the same time, we use data augmentation by varying illumination or adding noise, rotation, scale, and translation. After SurfaceNet and the triplet network are trained, we finally learn the shallow network $r(\cdot)$ (4).

6. Implementation Details

As described in Section 4.3, the scene volume needs to be divided into cubes due to limitations of the GPU memory. However, it is only necessary to process cubes that are very likely to contain surface voxels. As an approximate measure, we apply logistic regression to the distance vector $d_C^{(v_i,v_j)}$ (3) for each view pair, which predicts if the patches of both views are similar. If for less than N_{min} of the view pairs the predicted similarity probability is greater than or equal to 0.5, we reject the cube.

Instead of using a single threshold τ for binarization, one can also adapt the threshold for each cube C based on its neighboring cubes $\mathcal{N}(C)$. The approach is iterated and we initialize τ_C by 0.5 for each cube C. We optimize $\tau_C \in [0.5, 1)$ by minimizing the energy

$$E(\tau_C) = \sum_{C' \in \mathcal{N}(C)} \psi\left(S^C(\tau_C), S^{C'}(\tau_{C'})\right), \qquad (6)$$

where $S^C(\tau_C)$ denotes the estimated surface in cube C after binarization with threshold τ_C.

For the binary term ψ, we use

$$\psi(S^C, S^{C'}) = \sum_{x \in C \cap C'} (1 - s_x)s'_x + s_x(1 - s'_x) - \beta s_x s'_x. \qquad (7)$$

The first two terms penalize if S^C and $S^{C'}$ disagree in the overlapping region, which can be easily achieved by setting the threshold τ very high such that the overlapping region contains only very few surface voxels. The last term therefore aims to maximize the surface voxels that are shared among the cubes.

We used the *Lasagne* Library [6] to implement the network structure. The code and the trained model are publically available.[1]

7. Experiments

7.1. Dataset

The DTU multi-view stereo dataset [1] is a large scale MVS benchmark. It features a variety of objects and materials, and contains 80 different scenes seen from 49 or 64 camera positions under seven different lighting conditions. The provided reference models are acquired by accurate structured light scans. The large selection of shapes and materials is well-suited to train and test our method under realistic conditions and the scenes are complex enough even for the state-of-the-art methods, such as *furu* [8], *camp* [4], *tola* [27] and *Gipuma* [9]. For evaluation, we use three subsets of the objects from the DTU dataset for training, validation and evaluation. [2]

The evaluation is based on *accuracy* and *completeness*. Accuracy is measured as the distance from the inferred model to the reference model, and the completeness is calculated the other way around. Although both are actually error measures and a lower value is better, we use the terms *accuracy* and *completeness* as in [1]. Since the reference models of the DTU dataset are down-sampled to a resolution of $0.2mm$ for evaluation, we set the voxel resolution to $0.4mm$. In order to train SurfaceNet with a reasonable batch size, we use cubes with 32^3 voxels.

7.2. Impact of Parameters

We first evaluate the impact of the parameters for binarization. In order to provide a quantitative and qualitative analysis, we use model 9 of the dataset, which was randomly chosen from the test set. By default, we use cubes with 32^3 voxels.

As discussed in Section 5.1, the binarization depends on the threshold τ and the parameter γ for thinning. If $\gamma = 0\%$ thinning is not performed. Fig. 4 shows the impact of τ and γ. For each parameter setting, the front view and the intersection with a horizontal plane (red) is shown from top view. The top view shows the thickness and consistency of the reconstruction. We observe that τ is a trade-off

[1] https://github.com/mjiUST/SurfaceNet

[2] Training: 2, 6, 7, 8, 14, 16, 18, 19, 20, 22, 30, 31, 36, 39, 41, 42, 44, 45, 46, 47, 50, 51, 52, 53, 55, 57, 58, 60, 61, 63, 64, 65, 68, 69, 70, 71, 72, 74, 76, 83, 84, 85, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 107, 108, 109, 111, 112, 113, 115, 116, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128. Validation: 3, 5, 17, 21, 28, 35, 37, 38, 40, 43, 56, 59, 66, 67, 82, 86, 106, 117. Evaluation: 1, 4, 9, 10, 11, 12, 13, 15, 23, 24, 29, 32, 33, 34, 48, 49, 62, 75, 77, 110, 114, 118

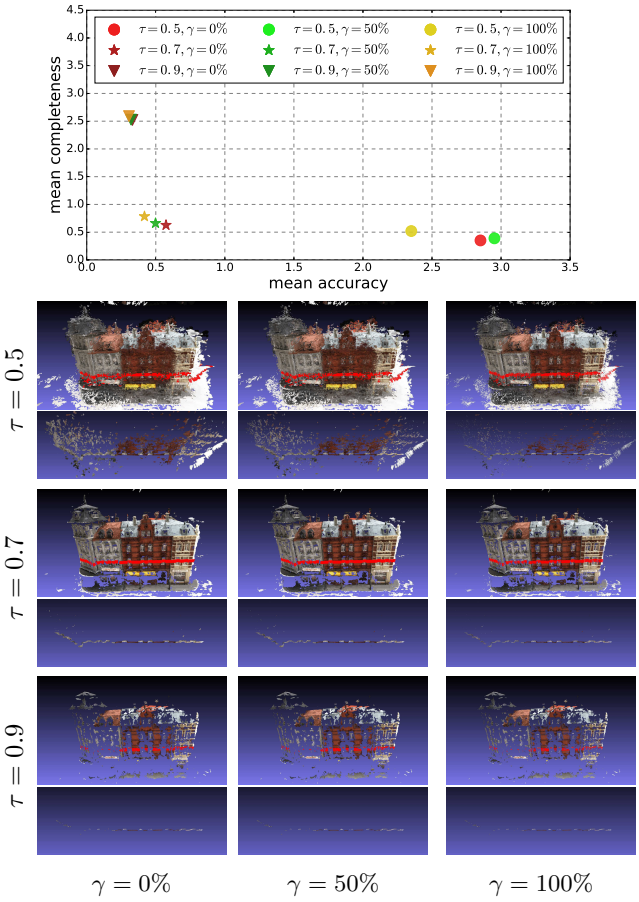


Figure 4: Quantitative and qualitative evaluation of τ and γ.

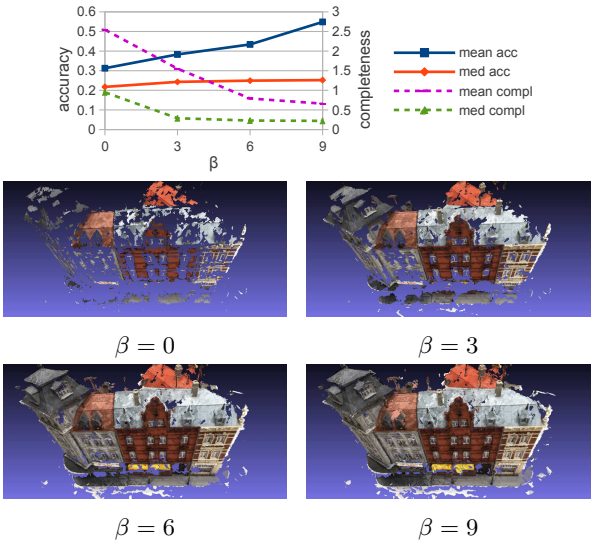
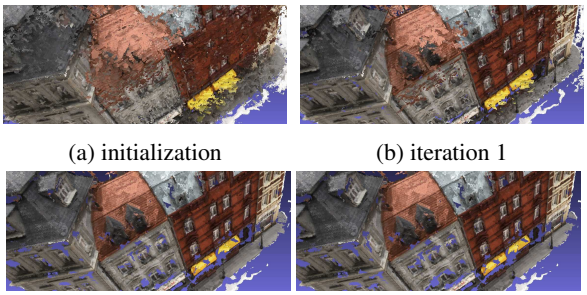


Figure 5: Quantitative and qualitative evaluation of β.



(a) initialization (b) iteration 1

(c) iteration 4 (d) iteration 8

Figure 6: The adaptive threshold is estimated by an iterative algorithm. The algorithm converges within a few iterations.

between accuracy and completeness. While a large value $\tau = 0.9$ discards large parts of the surface, $\tau = 0.5$ results in a noisy and inaccurate reconstruction. The thinning, i.e., $\gamma = 100\%$, improves the accuracy for any threshold τ and slightly impacts the completeness.

While the threshold $\tau = 0.7$ seems to provide a good trade-off between accuracy and completeness, we also evaluate the approach described in Section 6 where the constant threshold τ is replaced by an adaptive threshold τ_C that is estimated for each cube by minimizing (6) iteratively. The energy, however, also provides a trade-off parameter β (7). If β is large, we prefer completeness and when β is small we prefer accuracy. This is reflected in Fig. 5 where we show the impact of β. Fig. 6 shows how the reconstruction improves for $\beta = 6$ with the number of iterations from the initialization with $\tau_C = 0.5$. The method quickly converges after a few iterations. By default, we use $\beta = 6$ and 8 iterations for the adaptive binarization approach.

We compare the adaptive binarization with the constant thresholding in Table 2 where we report mean and median accuracy and completeness for model 9. The configuration

methods (mm)	mean acc	med acc	mean compl	med compl
$\tau = 0.7\ \gamma = 0\%$	0.574	0.284	**0.627**	**0.202**
$\tau = 0.7\ \gamma = 80\%$	0.448	**0.234**	0.706	0.242
adaptive threshold $\beta = 6\ \gamma = 80\%$	**0.434**	0.249	0.792	0.229
adaptive threshold $\beta = 6\ \gamma = 80\%$ w/o weighted average	0.448	0.251	0.798	0.228

Table 2: A quantitative comparison of two well performing parameter settings for τ and γ with the adaptive binarization procedure. The last row reports the result when the view pairs are not weighted. The evaluation is performed for the model 9.

$\tau = 0.7$ and $\gamma = 80\%$ provides a good trade-off between accuracy and completeness. Using adaptive thresholding with $\beta = 6$ as described in Section 6 achieves a better mean accuracy but the mean completeness is slightly worse. We

methods (mm)	mean acc	med acc	mean compl	med compl
camp [4]	0.834	0.335	**0.987**	**0.108**
furu [8]	0.504	0.215	1.288	0.246
tola [27]	0.318	0.190	1.533	0.268
Gipuma [9]	**0.268**	0.184	1.402	0.165
$s = 32, \tau = 0.7, \gamma = 0\%$	1.327	0.259	1.346	0.145
$s = 32, \tau = 0.7, \gamma = 80\%$	0.779	0.204	1.407	0.172
$s = 32$, adapt $\beta = 6, \gamma = 80\%$	0.546	0.209	1.944	0.167
$s = 64, \tau = 0.7, \gamma = 0\%$	0.625	0.219	1.293	0.141
$s = 64, \tau = 0.7, \gamma = 80\%$	0.454	0.191	1.354	0.164
$s = 64$, adapt $\beta = 6, \gamma = 80\%$	0.307	**0.183**	2.650	0.342

Table 3: Comparison with other methods. The results are reported for the test set consisting of 22 models.

also report in the last row the result when the probabilities p_x are not weighted in (5), *i.e.*, $w_C^{(v_i,v_j)} = 1$. This slightly deteriorates the accuracy.

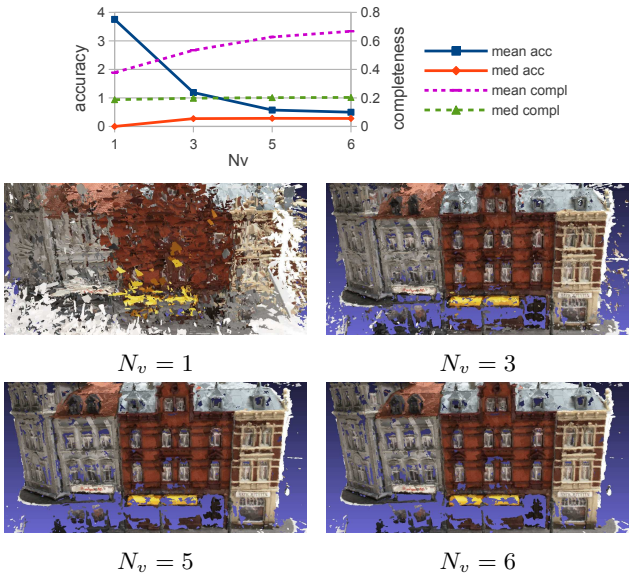


Figure 7: Quantitative and qualitative evaluation of N_v.

We finally evaluate the impact of fusing the probabilities p_x of the best N_v view pairs in (5). By default, we used $N_v = 5$ so far. The impact of N_v is shown in Fig. 7. While taking only the best view pair results in a very noisy inaccurate reconstruction, the accuracy is substantially improved for three view pairs. After five view pairs the accuracy slightly improves at the cost of a slight deterioration of the completeness. We therefore keep $N_v = 5$.

7.3. Comparison with others methods

We finally evaluate our approach on the test set consisting of 22 models and compare it with the methods *camp* [4], *furu* [8], *tola* [27], and *Gipuma* [9]. While we previously used cubes with 32^3 voxels, we also include the results for cubes with 64^3 voxels in Table 3. Using larger cubes, *i.e.* $s = 64$, improves accuracy and completeness using a constant threshold $\tau = 0.7$ with $\gamma = 0\%$ or $\gamma = 80\%$. When adaptive thresholding is used, the accuracy is also improved but the completeness deteriorates.

If we compare our approach using the setting $s = 64$, $\tau = 0.7$, and $\gamma = 80\%$ with the other methods, we observe that our approach achieves a better accuracy than *camp* [4] and *furu* [8] and a better completeness than *tola* [27] and *Gipuma* [9]. Overall, the reconstruction quality is comparable to the other methods. A qualitative comparison is shown in Fig. 8.

7.4. Runtime

The training process takes about 5 days using an Nvidia Titan X GPU. The inference is linear in the number of cubes and view pairs. For one view pair and a cube with 32^3 voxels, inference takes 50ms. If the cube size is increased to 64^3 voxels, the runtime increases to 400ms. However, the larger the cubes are the less cubes need to be processed. In case of $s = 32$, model 9 is divided into 375k cubes, but most of them are rejected as described in Section 6 and only 95k cubes are processed. In case of $s = 64$, model 9 is divided into 48k cubes and only 12k cubes are processed.

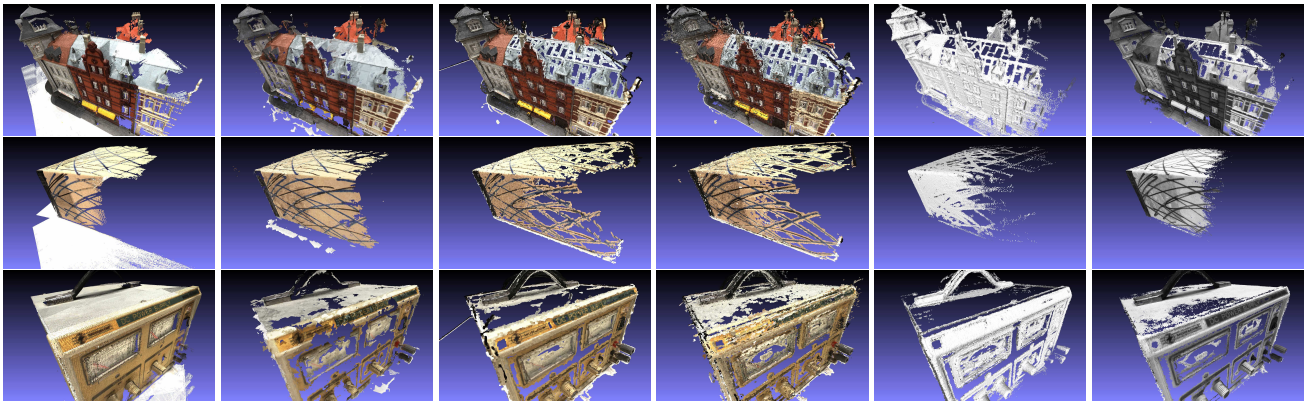
7.5. Generalization

methods (mm)	mean acc	med acc	mean compl	med compl
49 views				
$s = 32$, adapt $\beta = 6, \gamma = 80\%$	0.197	0.149	1.762	0.237
$s = 64, \tau = 0.7, \gamma = 80\%$	0.256	0.122	**0.756**	**0.193**
15 views				
$s = 32$, adapt $\beta = 6, \gamma = 80\%$	**0.191**	0.135	2.517	0.387
$s = 64, \tau = 0.7, \gamma = 80\%$	0.339	**0.117**	0.971	0.229

Table 4: Impact of the camera setting. The first two rows show the results for the 49 camera views, which are the same as in the training set. The last two rows show the results for 15 different camera views that are not part of the training set. The evaluation is performed for the three models 110, 114, and 118.

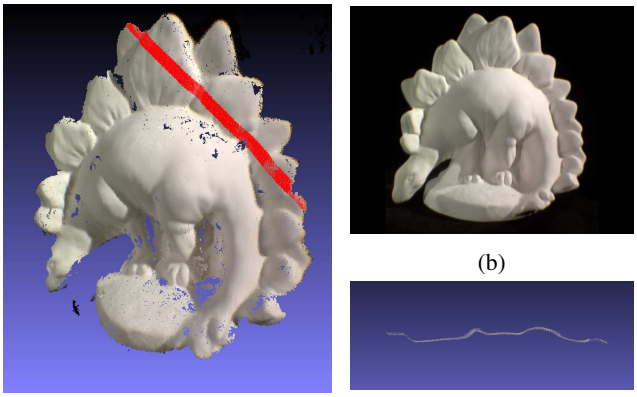
In order to demonstrate the generalization abilities of the model, we apply it to a camera setting that is not part of the training set and an object from another dataset.

The DTU dataset [1] provides for some models additional 15 views which have a larger distance to the object. For training, we only use the 49 views, which are available for all objects, but for testing we compare the results if the approach is applied to the same 49 views used for training or to the 15 views that have not been used for training. In our test set, the additional 15 views are available for the models 110, 114, and 118. The results in Table 4 show that the method performs very well even if the camera setting for training and testing differs. While the accuracy remains relatively stable, an increase of the completeness error is expected due to the reduced number of camera views.



a: reference model b: **SurfaceNet** c: *camp* [4] d: *furu* [8] e: *tola* [27] f: *Gipuma* [9]

Figure 8: Qualitative comparison to the reference model from the DTU dataset [1] and the reconstructions obtained by [4, 8, 27, 9]. The rows show the models 9, 10, 11 from the DTU dataset [1].



(a) (b) (c)

Figure 9: (a) Reconstruction using only 6 images of the dinoSparseRing model in the Middlebury dataset [21]. (b) One of the 6 images. (c) Top view of the reconstructed surface along the red line in (a).

We also applied our model to an object of the Middlebury MVS dataset [21]. We use 6 input images from the view ring of the model dinoSparseRing for reconstruction. The reconstruction is shown in Fig. 9.

8. Conclusion

In this work, we have presented the first end-to-end learning framework for multiview stereopsis. To efficiently encode the camera parameters, we have also introduced a novel representation for each available viewpoint. The so-called colored voxel cubes combine the image and camera information and can be processed by standard 3D convolutions. Our qualitative and quantitative evaluation on a large-scale MVS benchmark demonstrated that our network can accurately reconstruct the surface of 3D objects. While our method is currently comparable to the state-of-the-art, the accuracy can be further improved by more advanced post-processing methods. We think that the proposed network can also be used for a variety of other 3D applications.

Acknowledgments

The work has been supported in part by Natural Science Foundation of China (NSFC) under contract No. 61331015, 6152211, 61571259 and 61531014, in part by the National key foundation for exploring scientific instrument No.2013YQ140517, in part by Shenzhen Fundamental Research fund (JCYJ20170307153051701) and in part by the DFG project GA 1927/2-2 as part of the DFG Research Unit FOR 1505 Mapping on Demand (MoD).

References

[1] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph*, 28(3):24–1, 2009.

[3] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *British Machine Vision Conference*, volume 11, pages 1–11, 2011.

[4] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *European Conference on Computer Vision*, pages 766–779, 2008.

[5] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*, 2016.

[6] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, et al. Lasagne: First release, Aug. 2015.

[7] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[8] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[9] S. Galliani, K. Lasinger, and K. Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *IEEE International Conference on Computer Vision*, pages 873–881, 2015.

[10] S. Galliani and K. Schindler. Just look at the image: Viewpoint-specific surface normal prediction for improved multi-view reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5479–5487, 2016.

[11] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

[13] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3121–3128, 2011.

[14] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *IEEE International Conference on Computer Vision*, volume 1, pages 307–314, 1999.

[15] Y. Liu, X. Cao, Q. Dai, and W. Xu. Continuous depth estimation for multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2121–2128, 2009.

[16] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems*, pages 922–928, 2015.

[17] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[18] J. Pang, O. C. Au, Y. Yamashita, Y. Ling, Y. Guo, and J. Zeng. Self-similarity-based image colorization. In *IEEE International Conference on Image Processing*, pages 4687–4691, 2014.

[19] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3d data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[20] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[21] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006.

[22] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[24] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240, 2016.

[25] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[26] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337, 2016.

[27] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, pages 1–18, 2012.

[28] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[29] S. Xie and Z. Tu. Holistically-nested edge detection. In *IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[30] L. Xu, L. Hou, O. C. Au, W. Sun, X. Zhang, and Y. Guo. A novel ray-space based view generation algorithm via radon transform. *3D Research*, 4(2):1, 2013.

[31] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[32] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[33] A. Zheng, Y. Yuan, S. P. Jaiswal, and O. C. Au. Motion estimation via hierarchical block matching and graph cut. In *IEEE International Conference on Image Processing*, pages 4371–4375, 2015.