Learn-to-Score: Efficient 3D Scene Exploration by Predicting View Utility

Benjamin Hepp[1,2], Debadeepta Dey[2], Sudipta N. Sinha[2],
Ashish Kapoor[2], Neel Joshi[2], and Otmar Hilliges[1]

[1] ETH Zurich
[2] Microsoft Research

Abstract. Camera equipped drones are nowadays being used to explore large scenes and reconstruct detailed 3D maps. When free space in the scene is approximately known, an offline planner can generate optimal plans to efficiently explore the scene. However, for exploring unknown scenes, the planner must predict and maximize usefulness of where to go on the fly. Traditionally, this has been achieved using handcrafted utility functions. We propose to learn a better utility function that predicts the usefulness of future viewpoints. Our learned utility function is based on a 3D convolutional neural network. This network takes as input a novel volumetric scene representation that implicitly captures previously visited viewpoints and generalizes to new scenes. We evaluate our method on several large 3D models of urban scenes using simulated depth cameras. We show that our method outperforms existing utility measures in terms of reconstruction performance and is robust to sensor noise.

Keywords: 3D reconstruction, Exploration, Active vision, 3D CNN

1 Introduction

Quadrotors, drones, and other robotic cameras are becoming increasingly powerful, inexpensive and are being used for a range of tasks in computer vision and robotics applications such as autonomous navigation, mapping, 3D reconstruction, reconnaissance, and grasping and manipulation. For these applications, modeling the surrounding space and determining which areas are occupied is of key importance.

Recently, several approaches for robotic scanning of indoor [37] and outdoor [31,20] scenes have been proposed. Such approaches need to reason about whether voxels are free, occupied, or unknown space to ensure safety of the robot and to achieve good coverage w.r.t. their objective function (e.g. coverage of the 3D surfaces [31]). Model-based approaches require approximate information about free space and occupied space, which is typically acquired or input manually. This prevents such approaches from being fully autonomous or deployed in entirely unknown scenes [35]. Model-free approaches can be applied in unknown environments [19,27,24,8]. Irrespective of the type of approach used, all algorithms require a utility function that predicts how useful a new measurement (i.e. depth image) would be. Based on this utility function a planner reasons about the sequence of viewpoints to include in the motion plan.

This utility function is often a hand-crafted heuristic and hence it is difficult to incorporate prior information about the expected distributions of 3D geometry in certain scenes.

We propose to devise a better utility function using a data-driven approach. The desired target values for our utility function stem from an oracle with access to ground truth data. Our learned utility function implicitly captures knowledge about building and geometry distributions from approporiate training data and is capable of predicting the utility of new viewpoints given only the current occupancy map. To this end we train a 3D ConvNet on a novel multi-scale voxel representation of an underlying occupancy map, which encodes the current model of the environment. We then demonstrate that the learned utility function can be utilized to efficiently explore unknown environments.

The input to our network relies only on occupancy and hence abstracts away the capture method (i.e. stereo, IR depth camera, etc.). While our training data consists of perfect simulated depth images we demonstrate in our experiments that our learned model can be used with imperfect sensor data at test time, such as simulated noisy depth cameras or stereo data. The approach is not limited to environments with a fixed extent and generalizes to new scenes that are substantially different from ones in the training data. Our approach outperforms existing methods, that use heuristic-based utility functions [35,24,8] and is more than $10\times$ faster to compute than the methods from [24,8].

2 Related work

Exploration and mapping are well studied problems. We first discuss theoretical results and then describe approaches in the active vision domain and finally work in 3D vision.

Submodular sensor placement: In the case of a priori known environments and a given set of measurement locations, much work is dedicated to submodular objective functions for coverage [11,29]. Submodularity is a mathematical property enabling approximation guarantees on the solution using greedy methods. While work exists on dynamic environments where the utility of future measurements can change upon performing a measurement [16,22], these methods are usually difficult to scale to large state and observation spaces, which we considered in this paper as they are common in computer vision applications.

Next-best-view and exploration: In the next-best-view setting, the set of measurement locations is often fixed a priori as in the submodular coverage work described above. The work in this area is usually concerned with defining good heuristic utility functions and approximating the coverage task to make it computationally feasible [3,27,36,12,10]. A number of heuristics is explicitly compared in [24,8], and a subset of these is computed and used as a feature vector by Choudhury *et al.* [4] to imitate an approximately optimal strategy with ground-truth access.

Based on an a priori fixed set of camera poses and a binary input mask of already visited poses Devrim *et al.* [9] use reinforcement learning to regress a scalar parameter used in the selection algorithm for the next view. In contrast to our work the approach is concerned with a priori known, fixed environments and camera poses making it suitable for inspection planning.

In active vision, a large body of work is concerned with exploration through only partially known scenes. Frontier-based algorithms [38] are used for autonomous mapping and exploration of the environment using stereo [13], RGB-D, or monocular cameras [32]. Heng *et al.* [19] propose a method which alternates between exploration and optimizing coverage for 3D reconstruction.

All of the approaches discussed above either define or are based on heuristics to decide on the utility of the next measurement or require prior knowledge of environment and possible camera poses. Instead of hand-crafting a utility function our work is concerned with learning such a function that can outperform existing hand-crafted functions and is computationally cheaper to evaluate. Additionally, our approach does not need a priori knowledge of the map.

3D convolutional neural networks: A large body of work in computer vision is concerned with processing of 3D input and output data using convolutional neural networks. In some cases this data stems from RGB-D images such as in Song *et al.* [33] where the goal is to detect objects. In other contexts, volumetric input in the form of binary occupancy labels or signed distance functions are used for diverse tasks such as shape classification and semantic voxel labeling [6,30], surface completion [7], hand pose estimation [14], or feature learning [40]. These works are concerned with passive tasks on uniform input grids of fixed dimensions, containing the object or surface of interest. This prevents reasoning across large distances or requires one to reduce the level of detail of the input.

Different representations of occupancy grids have been proposed to mitigate the trade-off of large uniform input dimensions and level of detail [30]. However, in the context of our work the occupancy map is often not very sparse as it is generated by casting rays into a tri-state map and updating continuous values which results in very few homogeneous regions which would benefit from the formulation by Riegler *et al.* [30]. Also related to our work are approaches to multi-view reconstruction [5] where the output is predicted based on a sequence of input images. In contrast to our work Liu *et al.* [28] reconstruct small objects in a fixed size volume whereas we are concerned with large city scenes containing several buildings.

3 Problem Setting and Overview

Our work is concerned with the automatic exploration of an a priori unknown 3D world with the ultimate goal of reconstructing the surfaces of the scene in an efficient manner. In this setting, illustrated in Fig. 1, an algorithm has to make decisions about the next viewpoint based only on the current map information. In Fig. 1 the camera is surrounded by some space, already known to be free (white) and part of the surface has been observed (blue). The next viewpoint is restricted to the known free space, whereas moving into unknown space (light green) could lead to collisions. The main difficulty stems from the fact that the algorithm needs to predict how much unknown surface can be discovered from a new viewpoint. Much work has been dedicated to developing and studying various heuristics to compute a score that quantifies the expected value of possible viewpoints [24,8].

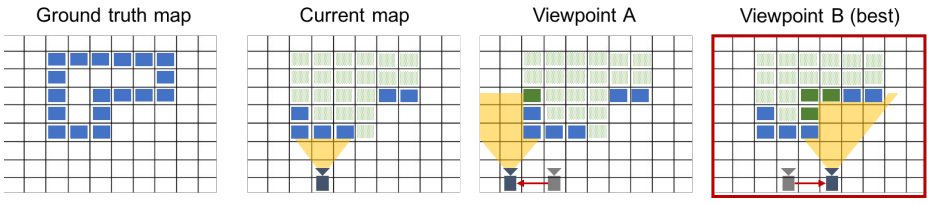

Fig. 1: The exploration task (here depicted in 2D for clarity) is to discover occupied surface voxels (shown here in blue). Voxels are initially unknown (shown here in light green) and get discovered by taking a measurement, e.g., shooting rays from the camera into the scene. Voxels that are not surface voxels will be discovered as free voxels (shown here in white). Each possible viewpoint has a corresponding utility value depending on how much it contributes to our knowledge of the surface (shown here in dark green). To decide which viewpoint we should go to next, an ideal utility score function would tell us the expected utility of viewpoints before performing them. This function can then be used in a planing algorithm to visit a sequence of viewpoints with the highest expected utility.

We propose a data-driven approach where we use supervised learning to find a utility function that imitates an oracle. The oracle has access to the ground truth map and can compute the true utility score. For this task we introduce a map representation consisting of multi-scale sub-volumes extracted around the camera's position. For all possible viewpoints this data is fed into a 3D ConvNet at training time together with the oracle's score as a target value. Intuitively, the model learns to predict the likelihood of seeing additional surface voxels for any given pose, given the current occupancy map. However, we do not explicitly model this likelihood but instead provide only the oracle's score to the learner. We experimentally show that our formulation generalizes well to new scenes with different object shape and distribution and can handle input resulting from noisy sensor measurements.

We follow related work [24,8,9] and evaluate our method on simulated but high-fidelity environments. This allows for evaluation of the utility function and reduces the influence of environmental factors and specific robotic platforms. Our environments contain realistic models of urban areas in terms of size and distribution of buildings. Furthermore it is important to note that our technique only takes occupancy information as input and does not directly interface with raw sensor data. In addition we test our approach on real data from outdoor and indoor scenes to demonstrate that our method is not limited to synthetic environments.

4 Predicting View Utility

We first formally define our task and the desired utility function and then introduce our method for learning and evaluating this function.

4.1 World model

We model the world as a uniform voxel grid V with resolution r. A map M is a tuple $M = (M^o, M^u)$ of functions $M^o : V \to [0, 1]$, $M^u : V \to [0, 1]$ that map each voxel $v \in V$ to an occupancy value $M^o(v)$ describing the fraction of the voxel's volume that is occupied and an associated uncertainty value $M^u(v)$, i.e. 1 for total uncertainty and 0 for no uncertainty. Maps change over time so we denote the map at time t as M_t.

After moving to a viewpoint $\mathbf{p}$ the camera acquires a new measurement in the form of a depth image and the map M is updated. We denote the updated map as $M|_{\mathbf{p}}$. The uncertainty is updated according to

$$M^u|_{\mathbf{p}}(v) = \exp(-\eta) M^u(v) \quad , \tag{1}$$

where $\eta \in \mathbb{R}_{>0}$ describes the amount of information added by a single measurement. This is a simple but effective measure providing a diminishing information gain of repeated measurements. Note that $M^u|_{\mathbf{p}}(v) \leq M^u(v)$ so uncertainty decreases with additional measurements. As is typical in occupancy mapping [34,23] we update the voxel occupancies $M^o(v)$ according to a beam-based inverse sensor model. Please see Sec. 4.4 for details on initialization of the map.

4.2 Oracle utility function

To select viewpoints, we need a utility function that assigns scores to all possible viewpoints at any time. We first introduce an oracle utility function with access to the ground truth (set of true surface voxels) during evaluation. It returns the desired true utility measure. We will then learn to imitate the oracle without access to ground truth.

We characterize a good viewpoint as one that discovers a large amount of surface voxels. Let *ObsSurf*(M) be the total number of observed surface voxels in map M weighted by their associated certainty value:

$$\textit{ObsSurf}(M) = \sum_{v \in \textit{Surf}} (1 - M^u(v)) \quad , \tag{2}$$

where *Surf* $\subseteq V$ is the set of ground truth surface voxels, i.e. all voxels that intersect the surface. Note that *ObsSurf*(M) increases monotonically with additional measurements because the certainty of voxels can only increase according to Eq. (1).

The decrease in uncertainty of surface voxels with a new measurement defines the oracle's utility score. We express this score as a function of the current map M and the camera pose $\mathbf{p}$:

$$\begin{aligned} s(M, \mathbf{p}) &= \textit{ObsSurf}(M|_{\mathbf{p}}) - \textit{ObsSurf}(M) \\ &= \sum_{v \in \textit{Surf}} (-M^u|_{\mathbf{p}}(v) + M^u(v)) = \sum_{v \in \textit{Surf}} (1 - \exp(-\eta))\, M^u(v) \geq 0 \quad . \end{aligned} \tag{3}$$

4.3 Learning the utility function

Computing the utility score introduced in Eq. 3 for any viewpoint requires access to the ground truth map. Our goal is to predict $s(M, \mathbf{p})$ without access to this data so we can formulate a regression problem that computes score values given occupancy maps as input.

Multi-scale map representation We propose to make predictions directly based on the occupancy map, rather than based on a temporal sequence of raw inputs. This occupancy map encodes our knowledge of already observed surfaces and free space and ultimately can be used to build up a map for both navigation and 3D reconstruction.

For use in a 3D ConvNet the map has to be represented with fixed dimensionalities. Here a trade-off between memory consumption, computational cost, reach and resolution arises. For example, extracting a small high resolution grid around the camera would constrain information to a small spatial extent whereas a grid with large spatial extent would either lead to rapid increase in memory consumption and computational cost or would lead to drastic reduction in resolution.

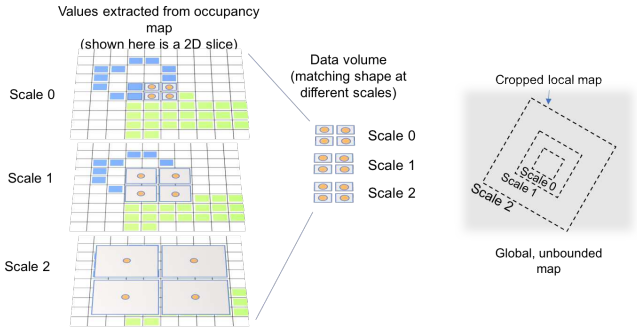

Fig. 2: Local multi-scale representation of an occupancy map. For clarity of presentation we shows the 2D case for a grid of size 2×2. The occupancy map is sampled with 3D grids at multiple scales centered around the camera position. Sample points on different scales are shown in orange and their extent in gray.

To mitigate this issue we introduce a multi-scale representation by sampling the occupancy map at multiple scales as depicted in Fig. 2. For each scale $l \in \{1, \dots, L\}$ we extract values on a 3D grid of size $D_x \times D_y \times D_z$ and resolution $2^l r$ (orange points in Fig. 2). On scale l the map values are given by averaging the 2^l closest voxels (gray rectangles in Fig. 2). This can be done efficiently by representing the map as an octree. The 3D grids are translated and rotated according to the measurement pose $\mathbf{p}$ and we use tri-linear interpolation of the map values to compute the values on the grid. This representation allows us to capture both coarse parts of the map that are far away from the camera but still keep finer detail in its direct surroundings. Furthermore, it provides an efficient data representation of fixed size, suitable for training of a 3D ConvNet. We denote the multi-scale representation by $x(M, \mathbf{p}) \in \mathbb{R}^{D_x \times D_y \times D_z \times 2L}$. Note that the factor 2 stems from extracting the occupancy and the uncertainty value on each scale.

ConvNet Architecture We now describe our proposed model architecture used to learn the desired utility function $f : \mathbb{R}^{D_x \times D_y \times D_z \times 2L} \rightarrow \mathbb{R}$. The general architecture is shown in Fig. 3 and consists of a number N_c of convolutional blocks followed by two

fully connected layers with ReLu activations. Each convolutional block contains a series of N_u units where a unit is made up of a 3D convolution, followed by Batch-Norm, followed by ReLu activation. Each 3D convolution increases the number of feature maps by N_f. After each block the spatial dimensions are downscaled by a factor of 2 using 3D max-pooling. The first fully connected layer has N_{h1} hidden units and the second one has N_{h2} hidden units. Note that we do not separate the input data at different scales so that the network can learn to combine data on different scales. More details on the exact architecture are provided in Sec. 5.1 and an evaluation of different variants is provided in Supplementary Material.

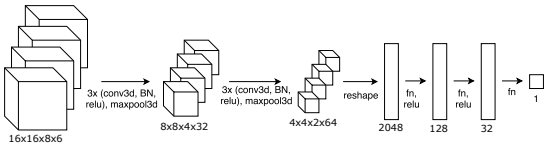

Fig. 3: Our architecture for an input size of $16 \times 16 \times 8$ with $L = 3$ scales resulting in $2L = 6$ channels. The model consists of blocks (made up of multiple units each performing 3D convolution, batch-norm and ReLu) followed by downscaling using 3D max-pooling. This pattern is performed until we arrive at a data volume with spatial dimension $4 \times 4 \times 2$. This is reshaped into a single vector followed by two fully connected layers with ReLu activation and a final linear layer predicting a scalar score value.

We use a weight-regularized $L2$ loss

$$\mathcal{L}(X, Y; \theta) = \sum_{i=1}^{N} \|f(X_i) - Y_i\|_2^2 + \lambda \|\theta\|_2^2 \quad , \tag{4}$$

where θ are the model parameters, λ is the regularization factor and (X_i, Y_i) for $i \in \{1, \ldots, N\}$ are the samples of input and target from our dataset.

4.4 3D Scene Exploration

To evaluate the effectiveness of our utility function, we implement a next-best-view (NBV) planning approach, to sequentially explore a 3D scene. Here we provide details of our world model and our methods for execution of episodes for the data generation phase and at test time.

We assume exploration of the world occurs in episodes. To initialize a new episode, the camera pose at time t_0 is chosen randomly in free space such that no collision occurs and the camera can move to each neighboring viewpoint without collision. A collision occurs if a bounding box of size $(1m, 1m, 1m)$ centered at the camera pose intersects with any occupied or unknown voxel. Initially, all voxels $v \in V$ are initialized to be unknown, i.e. $M^u_{t_0}(v) = 1, M^o_{t_0}(v) = v^{o,prior} \; \forall v \in V$, where $v^{o,prior}$ is a prior assumption on the occupancy and we use $v^{o,prior} = 0.5$ throughout this work. To enable

initial movement of the camera we clear (i.e. set to free space) a bounding box of $(6m)^3$ around the initial camera position.

At each time step t, we evaluate each potential viewpoint with our utility function, and move to the viewpoint $\mathbf{p}^*(t)$ that gives the best expected reward according to:

$$\mathbf{p}^*(t) = \mathit{argmax}_{\mathbf{p} \in P(t)} u(M_t, \mathbf{p}) \quad , \tag{5}$$

where $P(t)$ is the set of potential viewpoints and $u(\cdot)$ is the utility function in use.

At the start of each episode the set of potential viewpoints only contains the initial viewpoint. At each time step the set $P(t)$ is extended by those neighbors of the current viewpoint that do not lead to a collision. We ignore potential viewpoints if they have already been observed twice. Each viewpoint has 9 neighbors, 6 of them being positive and negative translations of $2.5m$ along each axis of the camera frame, 2 rotations of $25°$, clock-wise and counter-clockwise along the yaw axis, and a full turnaround of $180°$ along the yaw axis. We keep pitch and roll angles fixed throughout.

After moving to a new viewpoint, the camera takes a measurement in the form of a depth image and the map is updated (see Supplementary Material for details on the camera parameters and the map update). Note that we use ground truth depth when generating training data but later demonstrate that we can use noisy depth images and even stereo depth at test time.

Note that we assume that the utility function is submodular. While this is true for the oracle utility it is not necessarily the case for other utility functions (i.e. our learned model). Nevertheless, this assumption allows us to perform lazy evaluations of the utility function [26] (see Supplementary Material for details).

4.5 Dataset

To learn the utility function $f(x)$, approximating the oracle (see Eq. 3) we require labeled training data. Our data should capture large urban environments with a variety of structures typical for human-made environments. To this end we chose models from the *3D Street View* dataset [39]. These models feature realistic building distributions and geometries from different cities. Additionally, we chose a large scene from a photo-realistic game engine (`https://www.unrealengine.com`) containing small buildings in a suburban environment, including trees, smaller vegetation and power lines. All environments are shown in Fig. 4. Note that we only use data from *Washington2* to train our model. While *Washington1* and *Paris* are similar in terms of building height the building distribution and geometry is clearly different. A particular challenge is posed by the *SanFrancisco* scene which includes tall buildings never seen before in *Washington2*. Similarly, the buildings and vegetation in the *Neighborhood* scene are unlike anything seen in the training data.

We generate samples by running episodes with $r = 0.4m$ until time $t_e = 200$ and selecting the best viewpoint $\mathbf{p}$ according to the oracle's score at each time step. For each step t we store tuples of input $x(M, \mathbf{p})$ and target value from the oracle $s(M, \mathbf{p})$ for each neighboring viewpoint.

Note that we record samples for each possible neighbor of the current viewpoint (instead of only the best selected viewpoint). This is necessary as our predictor will

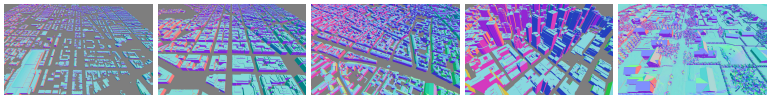

Fig. 4: Normal rendering of environments. From left to right: *Washington2*, *Washington1*, *Paris*, *SanFrancisco*, *Neighborhood*.

have to provide approximate scores for arbitrary viewpoints at test time. We record a total of approximately $1,000,000$ samples and perform a $80/20$ split into training and validation set. To encourage future comparison we will release our code for generating the data and evaluation.

5 Experiments

We describe our ConvNet architecture and then show different evaluations of our method.

5.1 ConvNet architectures and training

We evaluated different ConvNet variants by varying N_c, N_u and N_f. We also tried modifications such as using residual units [17,18]. We report these results in the supplementary material. Here we report results on the best performing model with input size $16 \times 16 \times 8$ ($N_c = 2$, $N_u = 4$, $N_f = 8$, $L = 3$, $N_{h1} = 128$, $N_{h2} = 32$), denoted as **Ours** in the rest of the section. Training of the model is done with ADAM using a mini-batch size of 128, regularization $\lambda = 10^{-4}$, $\alpha = 10^{-4}$ and the values suggested by Kingma *et al.* [25] for the other parameters. Dropout with a rate of 0.5 is used after fully-connected layers during training. Network parameters are initialized according to Glorot *et al.* [15] (corrected for ReLu activations). We use early stopping when over-fitting on test data is observed.

5.2 Evaluation

Our evaluation consists of three parts. First we evaluate our model on datasets generated as described in Sec. 4.5 and report spearman's rho to show the rank correlation of predicted scores and ground truth scores. Following this, we compare our models with previously proposed utility functions from [35,24,8]. We use the open-source implementation provided by [24,8] and report results on their best performing methods on our scenes, *ProximityCount* and *AverageEntropy*. We also compare with a frontier-based function measuring the number of unobserved voxels visible from a viewpoint as in [19,2]. For this comparison we use simulated noise-free depth images for all methods. Finally, we evaluate our models with depth images perturbed by noise and depth images produced by stereo matching in a photo-realistic rendering engine.

To demonstrate the generalization capability of our models we use four test scenes (column 2-5 in Fig. 4) that show different building distribution and geometry than the scene used to collect training data. We also perform the experiments on the training scenes where the exploration remains difficult due to random start poses and possible ambiguity in the incomplete occupancy maps.

To compute score and efficiency values, we run 50 episodes with $r = 0.4m$ until $t_e = 200$ for each method and compute the sample mean and standard deviation at each time step. To enable a fair comparison, we select a random start position for each episode in advance and use the same start positions for each method.

In order to report a single metric of performance for comparison we compute the area under the curve of observed surface versus time (see plots in Fig. 5):

$$\mathit{eff} = \sum_{t=0}^{t_e} \mathit{ObsSurf}(M_t) \quad . \tag{6}$$

We call this metric *Efficiency* as it gives a higher score to a method that discovers surface early on.

5.3 Model performance on different datasets

Here we evaluate the performance of our model on data collected from different scenes as described in Sec. 4.5. The model was trained on the training set of *Washington2* and we report Spearman's rho as well as the loss value from Eq. (4) in Tab. 1.

	Evaluation on different datasets					
	Washington2 train	*Washington2* test	*Washington1*	*Paris*	*SanFrancisco*	*Neighborhood*
Spearman's rho	0.88	0.87	0.83	0.69	0.73	0.48
Loss value	0.25	0.28	0.43	0.63	0.60	0.93

Table 1: Spearman's rho and loss values for our model on the different datasets. Despite the different building distribution and geometry of the test scenes (i.e. all scenes but *Washington2*) compared to training data Spearman's rho value shows a high rank correlation with the oracle score. This is even the case for the *Neighborhood* scene which features building shapes and trees unlike any in the training data.

The Spearman's rho shows a clear rank correlation even for the *Neighborhood* scene which features building distribution and geometry significantly different from *Washington2* which was used to generate training data. Interestingly, the model shows a high rank correlation for the *SanFrancisco* scene which features tall buildings and thus requires our model to generalize to different occupancy map distributions at high viewpoints.

5.4 Comparison with baselines

In Table 2 we compare the performance of our models against related hand-crafted utility functions [35,24,8]. Our method consistently outperforms the existing functions in terms of the efficiency measure, and as shown in Table 3, is faster to compute than other methods.

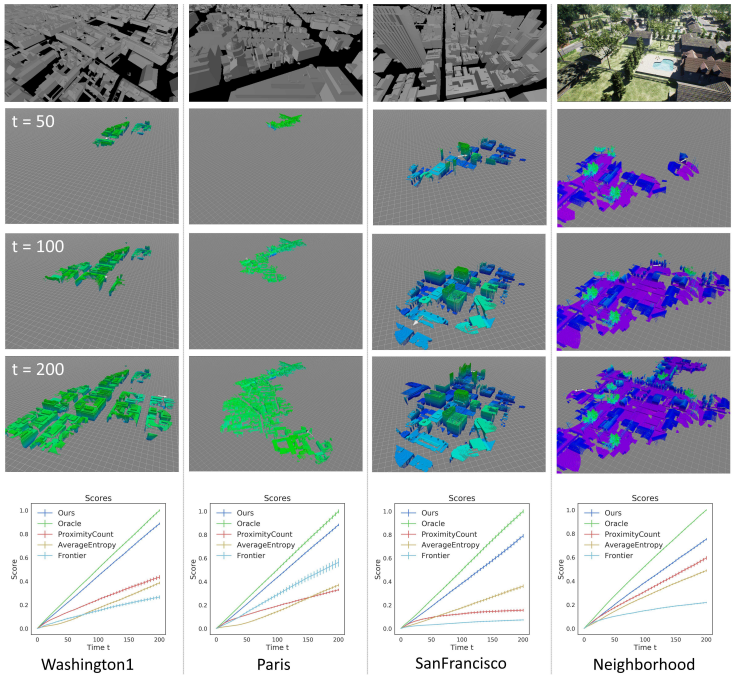

Fig. 5: Results on all test scenes. Top row: Visualization of the underlying mesh model. Row 2-4: Reconstructed 3D models at different time steps. Shown are only occupied voxels and the color coding indicates the voxel position along the z-axis. Bottom row: Plot of observed surface voxels vs. time for all methods, the oracle with access to ground truth and the baseline methods. Our method performs the best and approaches the oracle's performance. Best viewed in color and zoomed in. Larger versions in Supplementary Material.

We also show plots of observed surface voxels vs. time for our model, the oracle with access to ground truth and baseline methods in Fig. 5. Note that the scenes shown have not been used to generate any training data. The results show that our method performs better compared to the baseline methods at all time steps. Additionally the behavior of our method is consistent over all scenes while the performance of the baselines varies from scene to scene. The progression of reconstructed 3D models is shown by the renderings of the occupancy map at different times.

5.5 Noisy input sensor

While all our training is done on simulated data using ground truth depth images our intermediate state representation as an occupancy map makes our models robust to the noise characteristics of the input sensor. Here we evaluate the performance of our models at test time with depth images perturbed by noise of different magnitude. Addition-

	Evaluation on different scenes				
	Washington2	*Washington1*	*Paris*	*SanFrancisco*	*Neighborhood*
Frontier	0.40	0.29	0.57	0.09	0.27
AverageEntropy [24]	0.26	0.36	0.32	0.30	0.50
ProximityCount [24]	0.52	0.47	0.37	0.23	0.60
Ours	**0.91**	**0.88**	**0.87**	**0.77**	**0.74**
Oracle (GT access)	1.00	1.00	1.00	1.00	1.00

Table 2: Comparison of *Efficiency* metric. Our method achieves a higher value than the other utility functions on all scenes showing that our learned models can generalize to other scenes. Note that the model is trained only on data recorded from *Washington2*. *Efficiency* values are normalized with respect to the oracle for easier comparison.

	Computation time per step			
	Frontier	ProximityCount	AverageEntropy	Ours
Time in s	0.61	5.89	8.35	**0.57**

Table 3: Comparison of computation time per step. Our method is as fast as a simple raycast in the *Frontier* method and more than $10\times$ faster than *ProximityCount* and *AverageEntropy*.

ally we test our models with depth images computed from a virtual stereo camera. To this end we utilize a photorealistic game engine to synthesize RGB stereo pairs and compute depth maps with semi-global matching.

Episodes were run with noisy depth images and the viewpoint sequence was recorded. We replayed the same viewpoint sequences and used ground truth depth images to build up an occupancy map and measure the efficiency. Resulting *Efficiency* values are listed in Table 4. One can see that our method is robust to different noise levels. More importantly, even with depth images from the virtual stereo camera, resulting in realistic perturbations of the depth images (see Supplementary Material), our method does not degrade.

Evaluation using noisy depth images (normalized)						
Noise	none	low	medium	high	very high	stereo
eff	1.00	0.99	1.01	0.99	1.02	0.99

Table 4: Comparison of our method using noisy depth images. *Efficiency* values are normalized to the noise-free case. For the noise cases 40% of pixels in each depth image were dropped and each remaining pixel was perturbed by normal noise ($\sigma = 0.1m$ for low, $\sigma = 0.2m$ for medium, $\sigma = 0.5m$ for high, $\sigma = 1.0m$ for very high). In the case of stereo matching we used a photo realistic rendering engine to generate stereo images with a baseline of $0.5m$. A disparity and depth image was computed using semi global matching [21]. Note that all values have a standard deviation of $\approxeq 0.03$.

5.6 Additional results on real data

To show that our method is general and also works with real scenes we conducted additional experiments on high-fidelity 3D reconstructions of buildings and on the 2D-3D-S indoor dataset [1] that was acquired with a Matterport[3] camera. Result are shown in Tab. 5, Fig. 6 and Fig. 7. For the outdoor case we trained our model on the church (Fig. 6a) and evaluated on the historic building (Fig. 6.c). Despite the differences of both buildings in terms of geometry and scale (the historic building is about $2x$ smaller in each dimension) our model is able to generalize. For the indoor case we trained on *Area1* and evaluated on *Area5b* of the 2D-3D-S indoor dataset [1]. Both experiments demonstrate that our method also works with real detailed scenes.

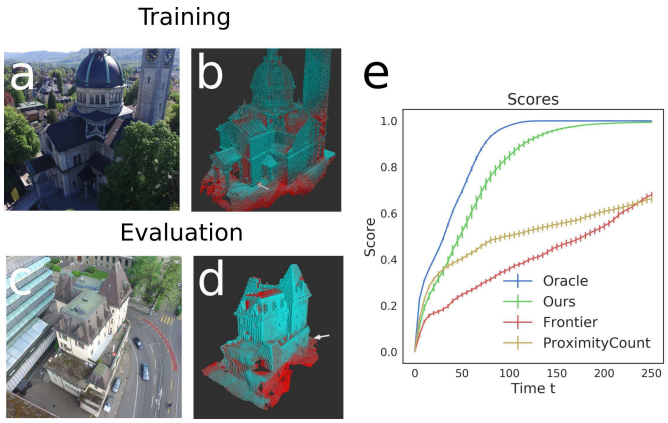

Fig. 6: Shown are example explorations on *real* outdoor data – (a) Picture of church scene. (b) Occupancy map of the church scene (training data) (200 steps). (c) Picture of historic building scene. (d) Occupancy map of the historic building scene (evaluation) (100 steps). (e) Performance plot for the historic building scene. Color coding of observed voxels: High uncertainty (red) and low uncertainty (cyan).

6 Discussion and Conclusions

We presented an approach for efficient exploration of unknown 3D environments by predicting the utility of new views using a 3D ConvNet. We input a novel multi-scale voxel representation of an underlying occupancy map, which represents the current model of the environment. Pairs of input and target utility score are obtained from an oracle that has access to ground truth information. Importantly, our model is able to generalize to scenes other than the training data and the underlying occupancy map enables robustness to noisy sensor input such as depth images from a stereo camera. Experiments indicate that our approach improves upon previous methods in terms of reconstruction efficiency.

[3] `https://matterport.com/`

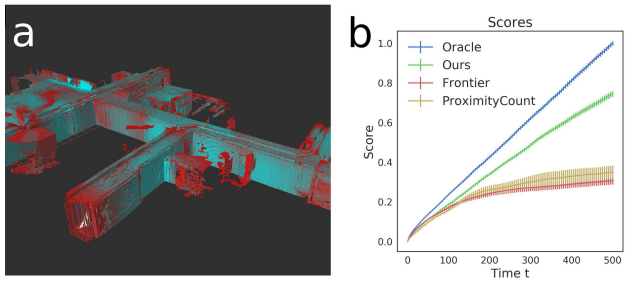

Fig. 7: Shown are example explorations on *real* indoor data – (a) Occupancy map of S3Dis Area5b (400 steps). (b) Performance plot for S3Dis Area5b (training on Area1). Color coding of observed voxels: High uncertainty (red) and low uncertainty (cyan).

Evaluation on additional real data				
	Frontier	ProximityCount [24]	**Ours**	Oracle (GT access)
Outdoor	0.46	0.58	**0.90**	1.00
Indoor	0.44	0.52	**0.78**	1.00

Table 5: Comparison of *Efficiency* metric on the additional real data. Our method achieves a higher value than the other utility functions on both ourdoor and indoor scenes. Note that in both cases the model was trained on data recorded from a single scene that was different from the evaluation scene. *Efficiency* values are normalized with respect to the oracle for easier comparison.

Limitations of our method include dependence on the surface voxel distribution in the training data. In future work, it would be interesting to see how the method performs on vastly different geometries such as rock formations and other natural landscapes. Similarly, our model is bound to the map resolution and mapping parameters used in the training data.

Another limitation is the underlying assumption on a static scene. A dynamic object such as a human walking in front of the camera would lead to occupied voxels that do not correspond to a static object. While these voxels can change their state from occupied to free after additional observations if the human walked away the intermediate occupancy map can lead to utility predictions that are not desired. A possible solution to address this problem is to identify and segment dynamic objects in the depth maps before integrating them into the occupancy map.

Our work suggests several directions for future work. We used our learned utility function to implement a greedy next-best-view algorithm; however, our utility function could be used to develop more sophisticated policies that look multiple steps ahead. In addition, our approach could be extended to be used in a generative way to predict future states of the 3D occupancy map or to predict 2D depth maps for unobserved views. This could be used for model completion or hole-filling which has numerous applications in computer vision and robotics.

References

1. Armeni, I., Sax, S., Zamir, A.R., Savarese, S.: Joint 2d-3d-semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105 (2017)
2. Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R.: Receding horizon" next-best-view" planner for 3d exploration. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on. pp. 1462–1468. IEEE (2016)
3. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: A survey of recent developments. The International Journal of Robotics Research **30**(11), 1343–1377 (2011)
4. Choudhury, S., Kapoor, A., Ranade, G., Scherer, S., Dey, D.: Adaptive information gathering via imitation learning. Robotics Science and Systems (2017)
5. 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: European Conference on Computer Vision. pp. 628–644. Springer (2016)
6. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. http://arxiv.org/abs/1702.04405 (2017)
7. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. http://arxiv.org/abs/1612.00101 (2016)
8. Delmerico, J., Isler, S., Sabzevari, R., Scaramuzza, D.: A comparison of volumetric information gain metrics for active 3d object reconstruction. Autonomous Robots pp. 1–12 (2017)
9. Devrim Kaba, M., Gokhan Uzunbas, M., Nam Lim, S.: A reinforcement learning approach to the view planning problem. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6933–6941 (2017)
10. Dunn, E., Frahm, J.M.: Next best view planning for active model improvement. In: BMVC. pp. 1–11 (2009)
11. Feige, U.: A threshold of ln n for approximating set cover. JACM (1998)
12. Forster, C., Pizzoli, M., Scaramuzza, D.: Appearance-based active, monocular, dense reconstruction for micro aerial vehicles. In: Robotics: Science and Systems (RSS) (2014)
13. Fraundorfer, F., Heng, L., Honegger, D., Lee, G.H., Meier, L., Tanskanen, P., Pollefeys, M.: Vision-based autonomous mapping and exploration using a quadrotor mav. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 4557–4564. IEEE (2012)
14. Ge, L., Liang, H., Yuan, J., Thalmann, D.: 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1991–2000 (2017)
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)
16. Golovin, D., Krause, A.: Adaptive submodularity: Theory and applications in active learning and stochastic optimization. JAIR (2011)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
18. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision. pp. 630–645. Springer (2016)
19. Heng, L., Gotovos, A., Krause, A., Pollefeys, M.: Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In: ICRA (2015)
20. Hepp, B., Nießner, M., Hilliges, O.: Plan3d: Viewpoint and trajectory optimization for aerial multi-view stereo reconstruction. arXiv preprint arXiv:1705.09314 (2017)
21. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Transactions on pattern analysis and machine intelligence **30**(2), 328–341 (2008)

22. Hollinger, G.A., Englot, B., Hover, F.S., Mitra, U., Sukhatme, G.S.: Active planning for underwater inspection and the benefit of adaptivity. IJRR (2012)
23. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots (2013). https://doi.org/10.1007/s10514-012-9321-0, http://octomap.github.com, software available at http://octomap.github.com
24. Isler, S., Sabzevari, R., Delmerico, J., Scaramuzza, D.: An Information Gain Formulation for Active Volumetric 3D Reconstruction. In: ICRA (2016)
25. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
26. Krause, A., Golovin, D.: Submodular function maximization. Tractability: Practical Approaches to Hard Problems (2012)
27. Kriegel, S., Rink, C., Bodenmüller, T., Suppa, M.: Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects. Journal of Real-Time Image Processing **10**(4), 611–631 (2015)
28. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5162–5170 (2015)
29. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functionsi. Mathematical Programming (1978)
30. Riegler, G., Ulusoy, A.O., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
31. Roberts, M., Dey, D., Truong, A., Sinha, S., Shah, S., Kapoor, A., Hanrahan, P., Joshi, N.: Submodular trajectory optimization for aerial 3d scanning. In: International Conference on Computer Vision (ICCV) 2017 (2017)
32. Shen, S., Michael, N., Kumar, V.: Autonomous multi-floor indoor navigation with a computationally constrained mav. In: Robotics and automation (ICRA), 2011 IEEE international conference on. pp. 20–25. IEEE (2011)
33. Song, S., Xiao, J.: Deep sliding shapes for amodal 3d object detection in rgb-d images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 808–816 (2016)
34. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. MIT press (2005)
35. Vasquez-Gomez, J.I., Sucar, L.E., Murrieta-Cid, R., Lopez-Damian, E.: Volumetric next-best-view planning for 3d object reconstruction with positioning error. International Journal of Advanced Robotic Systems **11**(10), 159 (2014)
36. Wenhardt, S., Deutsch, B., Angelopoulou, E., Niemann, H.: Active visual object reconstruction using d-, e-, and t-optimal next best views. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–7. IEEE (2007)
37. Xu, K., Zheng, L., Yan, Z., Yan, G., Zhang, E., Nießner, M., Deussen, O., Cohen-Or, D., Huang, H.: Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. ACM Transactions on Graphics 2017 (TOG) (2017)
38. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on. pp. 146–151. IEEE (1997)
39. Zamir, A.R., Wekel, T., Agrawal, P., Wei, C., Malik, J., Savarese, S.: Generic 3d representation via pose estimation and matching. In: European Conference on Computer Vision. pp. 535–553. Springer (2016)
40. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In: CVPR (2017)