ELSEVIER

Pattern Recognition Letters
journal homepage: www.elsevier.com

Supervised Learning of the Next-Best-View for 3D Object Reconstruction

Miguel Mendoza[a], J. Irving Vasquez-Gomez[a,b,**], Hind Taud[a], L. Enrique Sucar[c], Carolina Reta[d]

[a]*Instituto Politécnico Nacional (IPN), Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), Ciudad de México, México.*
[b]*Consejo Nacional de Ciencia y Tecnología (CONACYT), Ciudad de México, México.*
[c]*Instituto Nacional de Astrofísica Óptica y Electrónica, Puebla, México.*
[d]*Department of IT, Control, and Electronics, CONACYT-CIATEQ A. C., Av. Diesel Nacional #1 Ciudad Sahagún, Hidalgo 43990, México.*

ABSTRACT

Motivated by the advances in 3D sensing technology and the spreading of low-cost robotic platforms, 3D object reconstruction has become a common task in many areas. Nevertheless, the selection of the optimal sensor pose that maximizes the reconstructed surface is a problem that remains open. It is known in the literature as the next-best-view planning problem. In this paper, we propose a novel next-best-view planning scheme based on supervised deep learning. The scheme contains an algorithm for automatic generation of datasets and an original three-dimensional convolutional neural network (3D-CNN) used to learn the next-best-view. Unlike previous work where the problem is addressed as a search, the trained 3D-CNN directly predicts the sensor pose. We present an experimental comparison of the proposed architecture against two alternative networks; we also compare it with state-of-the-art next-best-view methods in the reconstruction of several unknown objects. Our method is faster and reaches high coverage.

1. Introduction

Autonomous three-dimensional (3D) object reconstruction or inspection is a computer vision task with applications to many areas, for example, robotics or cultural heritage conservation. It consists of generating a 3D model from a physical object by sensing its surface from several points of views. When the object is unknown, the task is performed iteratively in four steps: sensor positioning, sensing, registration and planning of the next sensor location [18]. In this work, we are interested in the challenging step of planning. According to the literature, the addressed challenge is called the next-best-view (NBV) problem and it has been defined as the task of computing the sensor position and orientation that maximizes the object's surface [4].

So far, most of the state-of-the-art techniques for NBV planning have been manually designed depending on the representations, needs, and constraints of reconstruction scene. Some methods, called synthesis methods, rely on analyzing the current information about the surface and directly synthesize the NBV [3], [17]. Synthesis methods are fast but their performance is usually decreased by objects with self-occlusions [18]. Another type of methods, called search-based, define a utility function and then perform a search over the feasible sensor positions in order to optimize the utility function. For instance, [24] and [2] represent the information with a probabilistic uniform grid, and they define as relevant features the frontier cells, as a result, they perform a search of the sensor position and orientation from where the maximum number of frontier cells is observed. Recent utility functions are based on the information gain contained in the model, [10]. In such cases, the target is to find the sensor pose that observes the cells (voxels) whose entropy is higher. Search-based methods are time-consuming given that the utility function has to be evaluated multiple times. Therefore, there is still a need for methods with the following characteristics: i) effective in spite of the object's shape and ii) efficient in terms of computation time.

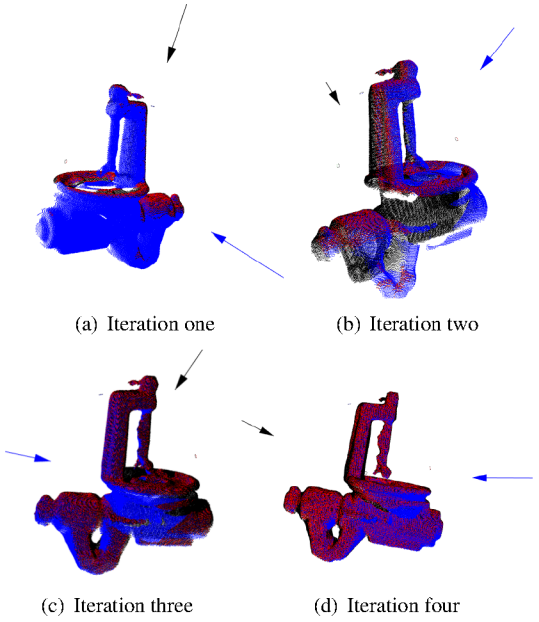
On the other hand, leveraged by technological advances in parallel processing hardware, the machine learning technique called deep learning (DL) [14] has dramatically improved the state-of-the-art in several areas, such as optical character recognition, two or three dimensional object recognition and classification [13]. One of the main advantages of DL is to automat-

**Corresponding author: Tel.: +5215543065029;
e-mail: jivasquezg@conacyt.mx (J. Irving Vasquez-Gomez)

ically discover the features that describe an entity or situation.

Our hypothesis is that the current NBV planning paradigm, usually seen as an optimization problem [6], can be modeled as a supervised learning problem. The trained function should take as input the partially reconstructed model and it should output the NBV. To the best of our knowledge, the use learning approaches, specifically DL, for NBV prediction is an unexplored path. Even though new approaches are trying to learn the utility function [7], the whole NBV prediction has not been addressed.

In this paper, we propose a supervised learning scheme to predict the NBV. We provide a methodology for generating datasets and an original 3D convolutional neural network, called NBV-Net. The scheme has been modeled as a classification problem where to a given partial model a possible sensor pose (class) is assigned. To validate the proposed scheme we have i) generated a dataset with more than 15,000 examples, ii) trained the proposed network iii) compared its accuracy against two related networks, iv) tested the trained network in the reconstruction of several unknown objects and v) compared its performance against several state-of-the-art methods. Our experiments have shown that NBV-Net improves the accuracy of related networks, and it has been capable of predicting the NBV during the reconstruction of several unknown objects (do not seen by the network during training) reaching a high reconstruction coverage. In addition, it is 1750 times faster with respect to the tested methods.



(a) Iteration one (b) Iteration two

(c) Iteration three (d) Iteration four

Fig. 1. Example of a 3D reconstruction. For each iteration, the current accumulated point cloud is displayed in black and the current sensor's pose is draw as a black arrow. Based on the current information, the NBV (blue arrow) is computed. The perception made at the NBV is drawn in blue. The overlap between the perception at the NBV and the accumulated point cloud is drawn in red. Finally, the new perception is registered to the accumulated point cloud. Note that in the next iteration the previous NBV is now the current sensor's pose. Figure best seen in color.

2. Next-best-view learning problem

The NBV concept originally rose up in the context of 3D object reconstruction [4]. In such a task, a sensor is placed at several poses around the object of interest in order to get its 3D shape. Due to the limited information about the object shape, the reconstruction is done iteratively by the steps of positioning, perception, registration and planning the NBV. Fig. 1 shows four iterations of the reconstruction of an example object. During the positioning, the sensor is placed at a given pose. The sensor's pose (also named view) defines its position and orientation, namely $v = (x, y, z, \alpha, \beta, \gamma)^T$, where α is a rotation about x axis, β is a rotation about y axis and γ is a rotation about z axis. At the perception step, the object's surface is measured and a point cloud (z) from the shape is obtained. After the perception, the acquired point cloud is registered to a single model [1]. As the reconstruction advances, the gathered information is stored into a partial model [9]. In this work, we use a uniform probabilistic occupancy grid, M, where each voxel has an associated probability that represents the likelihood that part or all the object's surface is inside the voxel's volume.

With respect to the NBV computation, so far, it has been treated as a search problem, where the target is to find the view that maximizes a metric. Unlike that approaches, in this work, we aim to directly predict the NBV based on the information provided by the partial model. Such prediction must be based on knowledge obtained from previous reconstructions. Henceforth, we define the NBV learning problem as the task of learning a function

$$f(M) : \mathbb{R}^n \rightarrow \mathbb{R}^3 \times SO(3) \quad (1)$$

so that the perception at $v = f(M)$ increases the information contained in the partial model about the object. The input of f is the voxel grid and it is written in eq. (1) as $\mathbb{R}^n$ where n is the amount of voxels in the grid. If we consider only the occupancy probability then the domain of f is only $[0, 1]^n$. The output is directly a sensor pose where a perception should be taken.

The proposed data-driven approach decomposes the NBV computation in two steps, the first step is the learning process of f, which has to be carried out off-line and using a plenty amount of examples from previous reconstructions; the second step is the prediction where f is used to estimate the NBV. The latter step is executed on-line during the reconstruction and it is hoped to be as quick as possible. In general, learning f implies to solve a regression problem and several challenges need to be addressed, therefore our first approach is to face the problem as a classification problem.

3. Dataset Generation

Deep learning approaches require vast amounts of examples to reach successful performances. In this NBV learning approach, we require a dataset with examples composed by an input (probabilistic grid) and a correct output (NBV). One of the challenges is to provide the "correct" or "ground truth" NBV. It is an important challenge because it implies to solve the NBV, which remains as an NP-Hard problem even though we could have the complete information about the object. Therefore, in this section, we propose a methodology to obtain a ground truth NBV using the complete information about the object and a dis-

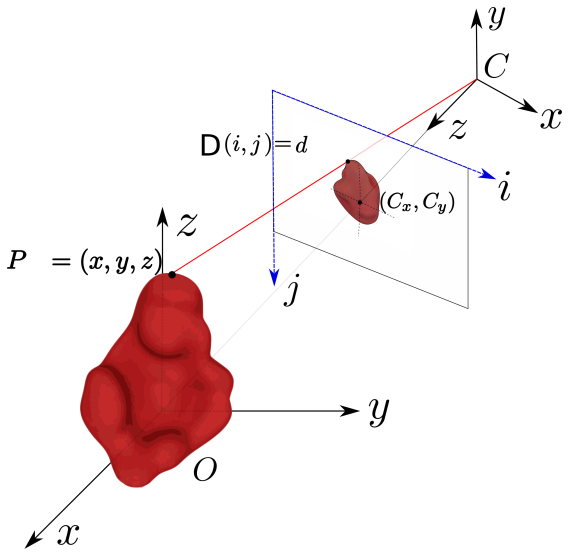


Fig. 2. Coordinate frames. An i measured point with respect to the camera's frame is denoted by $P_i^C = (x^c, y^c, z^c)$, the same point referenced to the global reference frame is denoted $P_i^O = (x, y, z)$.

crete search space. Making use of human expertise to label the dataset has been discarded since there are inherent constraints that even for a human expert are not easy to estimate, one of them is the registration constraint which is needed to build a unified model.

The proposed methodology works as follows. Given a demo object (an object from which we known the whole surface), 1) we establish a discrete NBV search space and then 2) we iteratively reconstruct it while we generate the NBVs. Below we describe the discrete search space, then we provide the definition and computation of a ground truth NBV and finally, we provide the iterative process for building the dataset.

3.1. Search Space Generation

In this step, for all demo objects a discrete set of views (search space) is build. Such views will be the only available poses to place the sensor. Therefore, the perceptions and the NBV are restricted to them. Formally, the discrete search space is denoted by V_d where $V_d \subset V = \mathbb{R}^3 \times SO(3)$. To standardize the process, we reuse the method proposed in [8], where a view sphere around the object and a set of range images are obtained for each view. To be precise, the object is placed in the center of a global reference frame O. Then, a set of possible sensor poses, V_d, is created by rotating a polar coordinate's point. The magnitude of the polar coordinate's vector is fixed and it provides the distance from the object's center to the sensor position. Next, the sensor poses are obtained by uniformly moving the angles. In our experiments, we use the same discretization provided by [8] (1312 positions). Each view is pointed to the center of the object. Figure 2 shows the reference frames and an example of a sensor pose.

Once that the views are established, for each demo object we generate the perception that correspond to each view in V_d. This step is included for performance reasons because later during training and validation it will be necessary to observe the object from the selected views. Hence, for each view a range image (D) is acquired. Next, the range images are transformed to point clouds, first with respect to the camera's reference frame and later with respect to the global reference frame. At the end of this step, for each view and for each object, we have a point cloud in terms of the global reference frame. From now on, we will call to each point cloud in the global reference frame a perception and we will write it as z. To remark that each perception is generated from a view, we will write a it as $z(v)$. The set of possible perceptions will be written as Z_d.

3.2. Ground Truth Next-Best-View

The NBV has been ideally defined as the view that increases the amount of information about the object of interest [4], but in practice, it is usually the view that optimizes a utility function. Such a utility function indirectly measures the increment of information. For example, the functions that count the number of unknown voxels [22] (voxels with probability 0.5) assume a positive correlation between the hidden object's surface and the unknown voxels. The same reasoning applies for the information gain approaches [10], where it is assumed that reducing the entropy of the partial model will provide more scanned surface. Those approaches are good to provide an approximation of the goodness of the view. However, they do not provide the real NBV because for obvious reasons they do not know the object's shape. Therefore, in this section, we establish a methodology for computing the NBV that will be taken as ground truth for a given probabilistic grid. The methodology incorporates the fact that the NBV must maximize the scanned surface but also incorporates the overlap that is needed in real world's reconstructions.

First, let us state some concepts. The set of views where the sensor has been placed is denoted by $S = \{s_0, \ldots, s_n\}, S \subseteq V_d$. The demo object is denoted by W_{obj} and it is a point cloud. The object's point cloud should be dense enough so that the distance between points will be smaller than the voxel resolution. Also, let us denote the object's accumulated point cloud as P_{acu}. Recalling, given that the reconstruction is an iterative integration of perceptions, P_{acu} is the result of integrating the sensor perceptions until the iteration i, namely $P_{acu} = \bigcup_{i=0,n} z(s_i)$. The percentage of coverage of a point cloud A with respect of the object ground truth W_{obj} is computed with the function $\texttt{Coverage}(A, W_{obj})$ [22].

Now, let us define the ground truth NBV (v^*) as the view that increases the accumulated point cloud, formally:

$$v^* = \arg\max_v \texttt{Coverage}(z(v) \cup P_{acu}, W_{obj}) \quad (2)$$

subject to the following constraints:

- v^* must be collision free, namely, it must lie in a free space.
- The perception $z(v^*)$ must have an overlap with P_{acu} higher than a threshold, $\texttt{overlap}(z(v^*), P_{acu}) > thresh_1$.
- The common region between surfaces must have at least three 3D features [15] ($thresh_2 \geq 3$) because besides of the amount of overlap, 3D features are needed to complete a 3D registration on the three axis.

In our experiments, we have set $thresh_1$ to 50% according to the experimental research presented in [22]. To compute the overlap, a radius, called gap, is defined to compare the two point clouds, P_{acu} and z. Due to the amount of data contained in these point clouds, a KdTree algorithm is needed to make an efficient search over all the neighbors of P_{acu} in z within the radius gap. NARF points [20] are used as 3D features.

3.3. Iterative Examples Generation

To generate the dataset we propose Algorithm 1. It reconstructs a demo object several times using different initial sensor poses and stores the computed NBVs. During the reconstruction, the current grid (M), accumulated point cloud (P_{acu}) and computed NBV (v^*) are stored as an example. It is worth to say that, even for the same object a different initial view will produce a different partial model and a different sequence of views.

The dataset generation requires as input a demo object (W_{obj}), a correspondent point gap ($gap \leftarrow 0.005$), a stop reconstruction criteria (S_{cov}), a maximum number of iterations (max_{iter}), an overlap percentage ($thresh_1$), the set of views ($V_d = \{v_1 \dots v_n\}$) and the set of perceptions ($Z = \{z_1, z_2 \dots z_n\}$). The object is assumed to be reconstructed when P_{acu} reaches a percentage of coverage equal to S_{cov} or the amount of iterations has reached max_{iter}, line 6 of Algorithm 1. Each reconstruction starts from a different initial view v_i (line 2) by taking each $v \in V_d$. Each point cloud z perceived at the pose $v*$ is read and added to the current P_{acu} (line 7); next, a filtering operation is performed to P_{acu} in order to maintain a uniform density (line 8). Then, in line 9, the probabilistic grid is updated according to the last perception z. From line 11 to 22, an exhaustive search is done over all possible views looking for the view whose perception, z, maximizes the increment, Δ, on the current P_{acu}. To avoid confusions with the current perception, z, we define z' as a perception in evaluation (line 12). z' must satisfy a minimum overlap (line 13) and must have at least $thresh_2$ 3D features computed as NARF points (line 14). Then, the NBV, $v*$, is the one that maximizes the increment and satisfies the constraints. Finally the algorithm saves a three-tuple which contains P_{acu}, M and $v*$ (line 24).

Algorithm 1 Dataset Examples Generation. The algorithm outputs several next-best-views given a known object surface.

Require: W_{obj}, gap, S_{cov}, max_{iter}, $thresh_1$, $thresh_2$ V_d and Z_d.

1: **for** $i \leftarrow 1 : n$ **do**
2: $v* \leftarrow v_i$
3: $iter \leftarrow 0$
4: $P_{acu} \leftarrow \emptyset$
5: `Initialize`(M)
6: **while** `Coverage`(P_{acu}, W_{obj}) $< S_{cov}$ and $iter < max_{iter}$ **do**
7: $P_{acu} \leftarrow P_{acu} \cup z(v^*)$
8: $P_{acu} \leftarrow$ `DownsizeFiltering`(P_{acu})
9: $M \leftarrow$ `UpdateGrid`($M, z(v*)$)
10: $\Delta_{max} \leftarrow 0$
11: **for** $j \leftarrow 1 : n$ **do**
12: $z' \leftarrow$ `Perception`(v_j)
13: **if** `overlap`(z', P_{acu}, gap) $> thresh_1$ **then**
14: **if** |`NARF`($z' \cap P_{acu}$)| $> thresh_2$ **then**
15: $\Delta \leftarrow$ `Coverage`($z' \cup P_{acu}, W_{obj}$) − `Coverage`(P_{acu}, W_{obj})
16: **if** $\Delta > \Delta_{max}$ **then**
17: $v* \leftarrow v_j$
18: $\Delta_{max} \leftarrow \Delta$
19: **end if**
20: **end if**
21: **end if**
22: **end for**
23: $iter++$
24: `SaveToDataset`($P_{acu}, M, v*$)
25: **end while**
26: **end for**

4. NBV Class Prediction using Deep Learning

Deep learning is a new branch of machine learning. It has improved the classification results provided by the conventional techniques in different domains. The key aspect of DL resides in the self-extraction of features or variables learned by the same learning system. Thereby, there is no need for a considerable experience and a careful choice by a user to choose, design and extract a set of features or representations where the performance of the system depends mainly on them.
DL scheme can be represented by two parts between input data and output data. Composed by the hidden layers or a deep network structure, the first part carries out the features extraction. The second one involves one or two layers which predict the class label. Depending on the problem, the first part can be accomplished by a supervised machine learning approach as the convolutional neural networks (CNN), unsupervised one as Auto-Encoders (AEs) or both as the convolutional AEs.
To face the NBV learning problem, we propose a classification approach, where we consider a possible sensor pose as a class. In the next sections, we will present the discretization proposed to establish the classes and an original 3D convolutional neural network (3D-CNN) architecture to predict the correct class.

4.1. Classification

In general, a sensor pose is defined in a continuous space. However, to use a classification approach it is necessary to have a discrete set of possible classes. In consequence, we group the poses generating a reduced set. This imply that predictions of the 3D-CNN will be a pose centered in a region representing a group of poses. To create a discrete set of views a sphere tessellation or other techniques can be applied. From now on, we will call to each possible pose a class.

4.2. Architecture

A 3D convolutional neural network (3D-CNN) is a special case of a CNN. In this kind of architecture, the kernels are 3D volumes instead of 2D maps. In the literature, there are several 3D-CNN architectures for 3D representations, for example VoxNet [16] and ShapeNets [23] are applied to volumes recognition. To the best of our knowledge no 3D-CNN has been

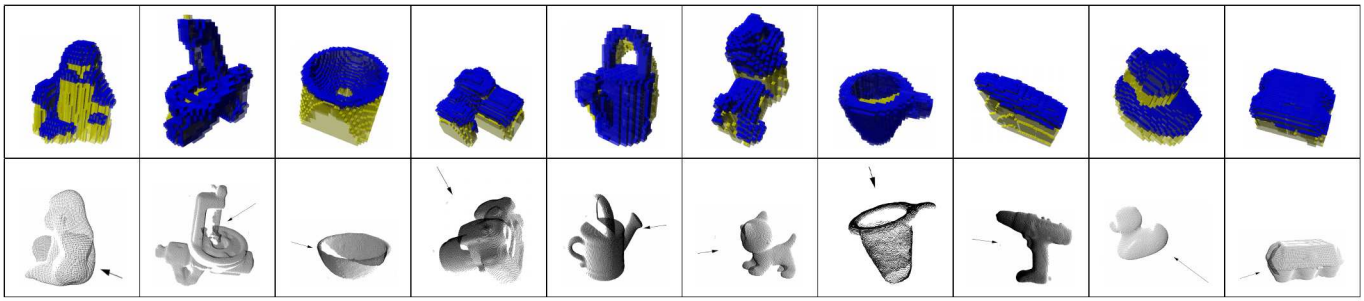


Fig. 3. Ten object examples contained in the dataset generated for experimentation. Inputs are P_{acu} and M and the corresponding output is the NBV. The black arrows show the NBV in the space. Blue voxels represent occupied space and yellow voxels unknown space.

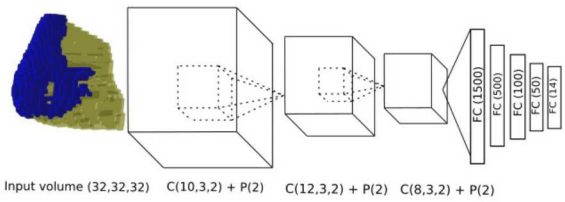


Fig. 4. NBV-Net architecture. A volume of size $32 \times 32 \times 32$ is fed to the network and 14 possible classes are given as output.

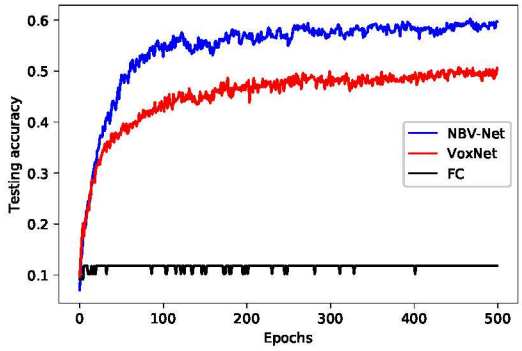


Fig. 5. Testing accuracy for VoxNet, NBV-Net and a FC Network.

applied to the NBV prediction. Consequently, we propose an architecture for NBV class prediction based on the problem's nature.

Henceforth, to simplify the notation, the next shorthand description is used:

- C(f,k,s): 3D convolution layer with f features, a kernel of size $k \times k \times k$ with stride $s \times s \times s$.
- P(s): A max pooling operation of stride $s \times s \times s$.
- FC(n): Fully connected layer with n parameters as output.

The proposed architecture, called NBV-Net, is connected as follows: C(10,3,2) – P(2) – C(12,3,2) – P(2) – C(8,3,2) – P(2) – FC(1500) – FC(500) – FC(100) – FC(50) – FC(C), where C indicates the number of classes. After each layer, except polling, the ReLu activation function is used. A softmax function with cross entropy is applied to provide one-hot enconding. Fig. 4 illustrates the network.

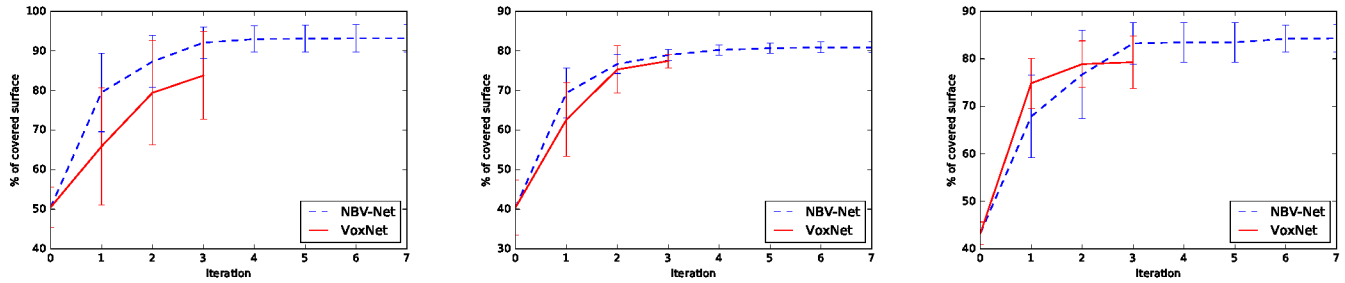
5. Experiments and Results

5.1. Dataset Analysis

The dataset generated for experimentation contains 15,364 examples that were generated using algorithm 1; available at `https://github.com/irvingvasquez/nbv-regression-dataset`. Each example is a tuple of the probabilistic grid and its corresponding NBV. See Fig. 3. The probabilistic grid has $32 \times 32 \times 32$ voxels and contains the whole object. The NBV designated for each grid is one of the set V_d. For this dataset, we create 20 sensor poses around the object, however only $C = 14$ were useful because the remaining 6 are below the floor and do not provide information about the object [8]. Processing time for generating the complete dataset was about 200 hours. An Intel® Core™ i7-2600 CPU to 3.40GHz with 8GR RAM was employed. The examples were created using 12 demo objects. Each object was reconstructed from 263 initial views; the views were selected to be uniformly distributed around the view sphere. We did not use the whole V_d set as initial views because we consider that very small differences between initial poses do not provide substantial information. The stop coverage criterion, S_{cov}, was set to 80%, this allows a reasonable trade-off between coverage and number of iterations. The maximum number of scans, max_{iter}, was set to 10. Distance for correspondent point gap was set to $0.005m$. During the reconstructions, the stop criterion (coverage or a maximum number of iterations) was reached at a different number of iterations due to the different object shapes and initial view. The objects were reconstructed from 3 to a maximum of 9 iterations and most of them are between 3 and 5 iterations. The dataset restricted to 14 classes is available at `https://www.kaggle.com/irvingvasquez/nbv-classification`

5.2. Training

NBV-Net was trained using the dataset described in section 5.1. In addition, two supplementary networks were implemented and trained in order to compare the accuracy of NBV-Net. One of them is a simply fully connected (FC) network of four layers; its architecture is FC(1500)-FC(750)-FC(100)-FC(50). We have included the fully connected network as a baseline. The second one is VoxNet [16], even though VoxNet



(a) Bottle (b) Iron (c) Telephone

Fig. 6. Reconstruction coverage for each object.

has been designed for object recognition we want to test its performance since its design is similar to the proposed NBV-Net.

Networks training was performed by Adam stochastic optimization method [11]. The learning rate was 0.001 with a batch size of 200. Dropout, of 0.7, is added after the FC layers and the last convolutional layer. Convolutional layers were initialized with values from a truncated normal distribution with $\sigma = 0.1$ and $\mu = 0$. Our implementation was made in Tensorflow. The training was done using a GPU NVIDIA GeForce GTX1080 mounted on a desktop computer with 16GB of memory and a Xeon E5-2620 v4 2.10GHz processor. Traning for NBV-Net, FC and VoxNet was done using the same hyper-parameters. Training epochs was set to 500. The dataset was splitted in 80% for training and 20% for testing. The training was stopped in 500 epochs, based in the learning of the test set; from this epoch, the test accuracy was oscillating in a margin of ±0.02. Processing time for training was 1.2 hours for VoxNet and 1.5 hours for NBV-Net. Fig. 5 plots the testing accuracy reached by each network. It can be seen that testing accuracy for NBV-Net is better than VoxNet and FC network. Even though testing accuracy does not approximate to one, the use of the trained NBV-Net in the reconstruction of several unknown objects is satisfactory as the following experiment shows. Since the FC network has shown a poor validation accuracy it will not be tested in the following experiments.

5.3. 3D Reconstruction using Predicted NBV

In this experiment, we test the trained NBV-Net and VoxNet networks in a 3D reconstruction task. The objective is to validate that NBV-Net is capable of computing a sensor pose that increases the object's surface. The experiment's workflow is to place the sensor at a random initial pose, then to take a perception, next, the probabilistic grid is updated, then the grid is fed to each trained network and a forward pass is performed, at this moment each network provides a class (the NBV), next, the sensor is placed in the given class and a new perception is taken. The process is repeated until the coverage is not increasing or the network is providing an already visited sensor pose.

The reconstructions were conducted for three new objects: a telephone, an iron, and a bottle. It is important to remark that such objects were not seen by the networks during training. Each object was reconstructed 10 times by each network. The same random initial poses were used for both networks.

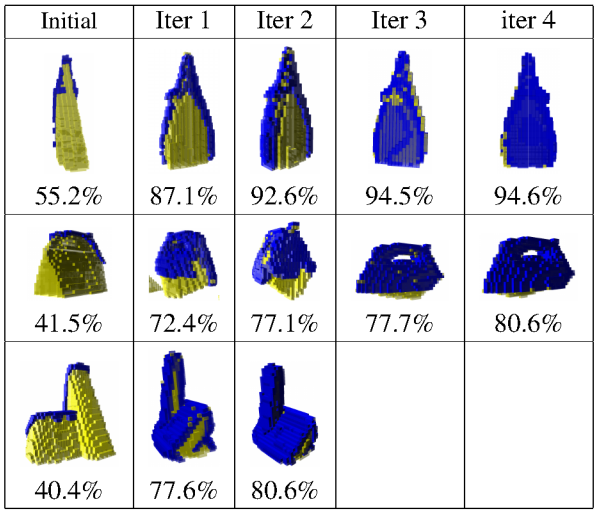


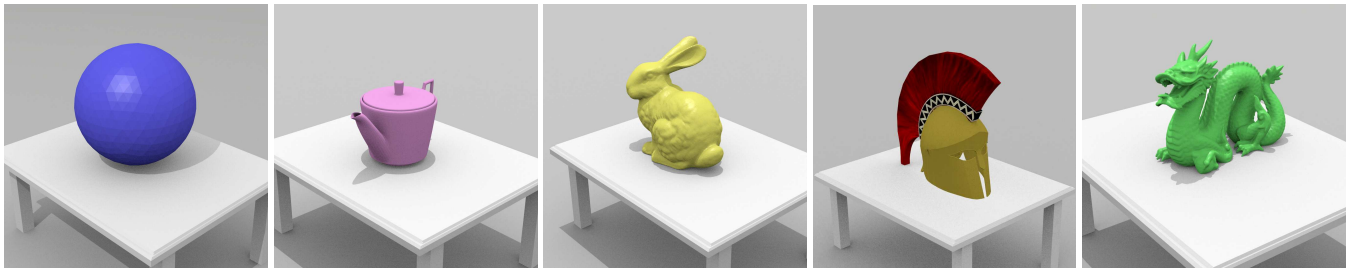
Fig. 7. Reconstruction progress for three objects using NBV-Net. From top to bottom: bottle, iron and telephone. Each frame shows the current state of the probabilistic grid. Blue voxels represent occupied space and yellow voxels unknown space.

Table 1. The average coverage (S), standard deviation (σ_s), and iterations (I) are shown as result of 10 reconstructions from random initial poses.

Object	NBV-Net			VoxNet		
	S_{cov}	σ_s	I	S_{cov}	σ_s	I
Bottle	93.18	3.44	4.1	83.88	11.05	2.2
Iron	80.77	1.41	4.5	77.64	1.8	2.6
Telephone	84.07	3.12	3.5	79.52	5.65	1.8

The results are summarized in figure 6. The mean and standard deviations of the covered surface are shown in Table 1. In `https://youtu.be/yw1L4HgDaPI` a video of three reconstructions using NBV-Net is shown.

As a result, NBV-Net reaches a higher coverage in comparison with the other networks. The overlap constraint (Section 3.2) is more evident in NBV-Net because the increments in percentage of covered surface are softer than VoxNet. In Fig. 7, the three objects were reconstructed, starting from three random poses and the iterations of the NBV predicted using NBV-Net are shown.



(a) Sphere (b) Teapot (c) Bunny (d) Helmet (e) Dragon

Fig. 8. Objects used for comparing NBV-Net versus current state-of-the-art approaches.

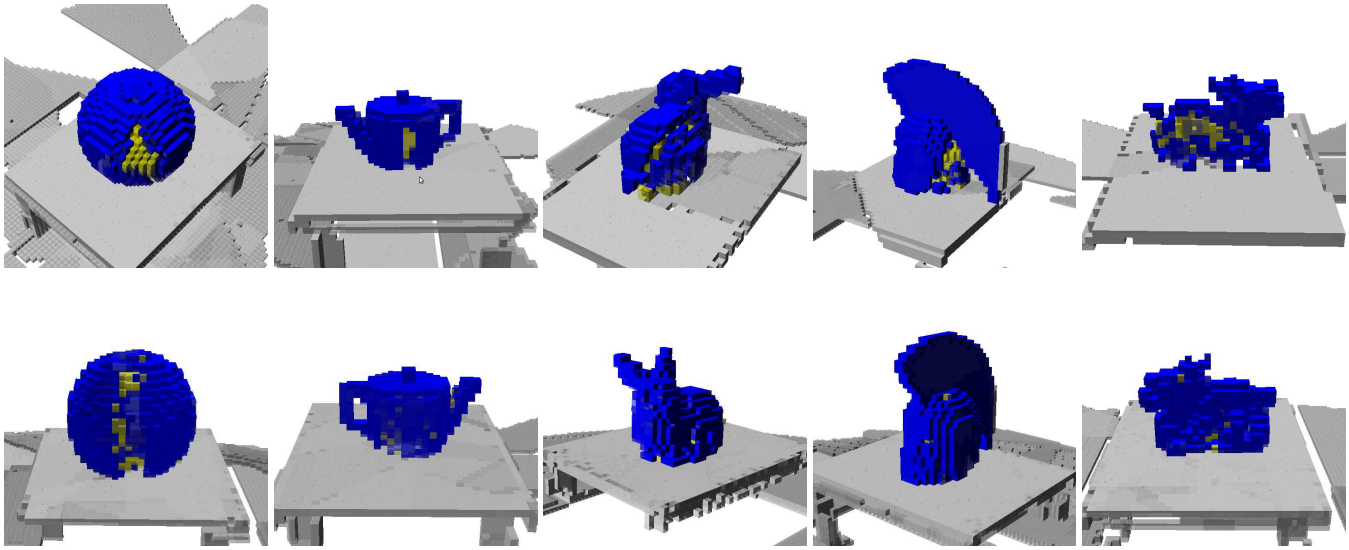


Fig. 9. Comparison of reconstructed grids. The pictures were selected in order to remark the missing parts for each model. Top: NBV-Net. Bottom: Information Gain. In both cases there are missing parts. The major difference is shown for the dragon model.

5.4. Comparison against non-machine-learning approaches

In this experiment, we compare the trained NBV-Net against several non-machine-learning approaches in the reconstruction of several unknown objects. The objective is to show the advantages and disadvantages of the proposed data-driven approach against the state-of-the-art methods in a more realistic scenario, where an unknown object placed over a table is reconstructed by a freeflyer Kinect sensor. The objects were not previously used neither for training nor validation nor testing. The compared methods are: Information gain (Inf. Gain) [12], unknown voxels counting (Unk. Vox.) [22] and rear side voxels (R.S.V) [5]. All methods require a set of candidate views in order to evaluate them according to their own metric. To make the comparison fair, we use 20 candidate views around the objects for all methods including NBV-Net. The tested objects are five (See Fig. 8): sphere [21], bunny [21], dragon [21], teapot [19] and helmet [19]. The experiment was carried out as follows: an initial scan from a fixed initial view is made, then each method computes the NBV and updates the model iteratively until 10 scans. The mean processing times are presented in Table 2 and the coverage reached is presented in Table 3. All methods were tested in a 3.60GHz Intel Core i7 machine with 8 GB of RAM; the GPU was disabled for NBV-Net and the forwards pass was made only on the CPU.

Table 2. Next-best-view computation times. Units are seconds.

	NBV-Net	Inf. Gain	Unk. Vox.	R.S.V
Time	**0.01 s**	29.9 s	17.5 s	18.0 s

Table 3. Comparison of NBV-Net versus non-machine-learning approaches. Coverage and number of scans for reaching such coverage are presented. After the given number of scans the surface did not increase.

	NBV-Net	Inf. Gain	Unk. Vox.	R.S.V
Sphere	96.7 / 9	96.2 / 7	**97.0 / 6**	97.0 / 8
Teapot	91.9 / 5	93.2 / 7	**93.7 / 6**	93.4 / 8
Bunny	90.0 / 3	98.1 / 9	97.4 / 9	**98.2 / 9**
Helmet	82.7 / 3	**86.5 / 8**	86.4 /10	85.7 / 9
Dragon	71.3 / 8	**90.4 / 10**	84.7 / 9	87.2 / 10

Based on the results, we observed that NBV-Net is very effective and efficient in early iterations, reaching a competitive coverage in a very short time, however it struggles in the last iterations. As we can see in Table 2, the processing time required by NBV-net is very short with respect to state-of-the-art methods; 1750 times faster than the second method (Unk. Vox.), this fact is explained because the current methods explicitly model the sensor and during the candidate evaluation, therefore they apply

Table 4. Comparison of NBV-Net versus non-machine-learning approaches after four scans. Coverage reached by each method is presented.

	NBV-Net	Inf. Gain	Unk. Vox.	R.S.V
Sphere	**93.4**	92.2	90.9	87.2
Teapot	87.1	**93.0**	91.9	84.4
Bunny	90.0	**96.3**	94.0	94.3
Helmet	82.7	85.1	**86.0**	85.0
Dragon	71.3	**85.0**	82.8	58.0

an expensive ray tracing. With respect to the coverage, in Table 3, we can see that NBV-Net reaches a competitive coverage for the sphere, teapot, bunny and helmet but stalls in the dragon, we believe that the stall is caused by two reasons: i) the dataset does not contain a similar shape and ii) NBV-Net omits the sensor modeling of previous approaches. On the other hand, NBV-net reaches a high coverage with only a few scans. To show this fact, Tab. 4 presents the coverage until four scans, where we can observe that NBV-net wins for the sphere object and is very close to the Information Gain approach for the remaining objects, in addition, NBV-Net overcomes in three times the R.S.V. method which reaches the lowest coverage. As we mention earlier, NBV-net struggles in the last iterations where small missing surfaces need to be observed (See Fig. 9). We believe that such disadvantage is caused by the dataset where there is a small number of examples where the object is almost reconstructed. Recalling, the reconstructions made for the dataset were stopped when 80% of the surface is reached.

In conclusion, NBV-Net reaches a high coverage in early iterations using a very short processing time. Therefore, we recommend the use of NBV-Net during the initial scans and then to apply another method for the last iterations.

6. Conclusions

A scheme for computing the NBV for 3D object reconstruction based on supervised deep learning has been proposed. As part of the scheme, we have presented an algorithm for automatic generation of datasets and an original 3D convolutional neural network called NBV-Net. We have trained the network and we have compared its accuracy against an alternative net. We also have tested NBV-Net in the reconstruction of several unknown objects which were not seen during training. As a result, NBV-Net was capable of predicting the NBV covering the majority of the surface. In addition, the processing time for computing the NBV is significantly shorter than state-of-the-art methods given that it does not require the expensive ray tracing. In consequence, we have shown positive evidence that the NBV problem can be addressed as a classification problem when supervised deep learning is applied. Our future research interest is to face the NBV learning problem as a regression problem avoiding the restriction of a limited set of classes.

References

[1] Besl, P., McKay, N., 1992. A method for registration of 3-d shapes. IEEE Trans Pattern Anal Mach Intell 14, 239–256.

[2] Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R., 2016. Receding horizon" next-best-view" planner for 3d exploration, in: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE. pp. 1462–1468.

[3] Chen, S., Li, Y., 2005. Vision sensor planning for 3-d model acquisition. IEEE Transactions on Systems, Man, and Cybernetics 35, 894–904.

[4] Connolly, C., 1985. The determination of next best views, in: Robotics and automation. Proceedings. 1985 IEEE international conference on, IEEE. pp. 432–435.

[5] Delmerico, J., Isler, S., Sabzevari, R., Scaramuzza, D., 2018. A comparison of volumetric information gain metrics for active 3d object reconstruction. Autonomous Robots 42, 197–208.

[6] Foissotte, T., Stasse, O., Escande, A., Wieber, P.B., Kheddar, A., 2009. A two-steps next-best-view algorithm for autonomous 3d object modeling by a humanoid robot, in: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE. pp. 1159–1164.

[7] Hepp, B., Dey, D., Sinha, S.N., Kapoor, A., Joshi, N., Hilliges, O., 2018. Learn-to-score: Efficient 3d scene exploration by predicting view utility, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 437–452.

[8] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N., 2012. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes, in: Asian conference on computer vision, Springer. pp. 548–562.

[9] Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W., 2013. Octomap: An efficient probabilistic 3d mapping framework based on octrees. Autonomous robots 34, 189–206.

[10] Isler, S., Sabzevari, R., Delmerico, J., Scaramuzza, D., 2016. An information gain formulation for active volumetric 3d reconstruction, in: IEEE international conference on robotics and automation, IEEE, Stockholm, Sweden, 16-21 May 2016. pp. 3477–3484.

[11] Kingma, D., Ba., J., 2015. Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR).

[12] Kriegel, S., Rink, C., Bodenmüller, T., Narr, A., Suppa, M., Hirzinger, G., 2012. Next-best-scan planning for autonomous 3d modeling, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE. pp. 2850–2856.

[13] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105.

[14] LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. nature 521, 436.

[15] Low, K.L., Lastra, A., 2007. Predetermination of icp registration errors and its application to view planning, in: 3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on, IEEE. pp. 73–80.

[16] Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE. pp. 922–928.

[17] Maver, J., Bajcsy, R., 1993. Occlusions as a guide for planning the next view. IEEE Trans Pattern Anal Mach Intell 15, 417–433.

[18] Scott, W., Roth, G., Rivest, J., 2003. View planning for automated three-dimensional object reconstruction and inspection. ACM Comput Surv 35, 64–96.

[19] Sketchup, . 3d warehouse. Online. Https://3dwarehouse.sketchup.com, Accessed 01 December 2019.

[20] Steder, B., Rusu, R.B., Konolige, K., Burgard, W., 2010. Narf: 3d range image features for object recognition, in: Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).

[21] Turk, G., Levoy, M., . The stanford 3d scanning repository. online. Http://graphics. stanford. edu/data/3Dscanrep, Accessed 01 December 2019.

[22] Vasquez-Gomez, J.I., Sucar, L.E., Murrieta-Cid, R., 2017. View/state planning for three-dimensional object reconstruction under uncertainty. Autonomous Robots 41, 89–109.

[23] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920.

[24] Yamauchi, B., 1997. A frontier-based approach for autonomous exploration, in: Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on, IEEE. pp. 146–151.