university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Multi-View 3D Object Recognition: Selecting the Best Sequences of Views

Final Research Project
(Computational Intelligence and Robotics)

Koen Buiten (s3861023)

August 23, 2022

Internal Supervisors: Dr. S.H. Mohades Kasaei (Artificial Intelligence, University of Groningen)
Prof. Dr. L.R.B. Schomaker (Artificial Intelligence, University of Groningen)

Artificial Intelligence
University of Groningen, The Netherlands

Abstract

Most state-of-the-art object recognition systems make use of a multi-view recognition approach. In most approaches, viewpoints are selected from predefined viewpoint setups, like orthographic, orbit, or hemisphere setups. The ability to have a more informed way of selecting viewpoints could potentially improve the performance of an object recognition system. For robotic systems, it has the additional benefit of travel reduction, because only the most informative viewpoints are visited. This viewpoint optimization problem is called the Next-Best-View (NBV) problem. Previous studies have shown that 2D shape measures such as view entropy and view area are suitable measures for determining the goodness of a view in the context of object recognition. We therefore propose a next-best-view selection model based on the predicted depth-entropy of views in the neighborhood of the current viewpoint. The input for the model is the point cloud of the object, taken from the current view. The output is a three-by-five depth-entropy map of neighboring views. We define depth-entropy as Shannon's entropy taken from the depth image of the corresponding view. Local extrema in the depth-entropy map determine the next-best-view. The backbone of our method is based on the PointNet model. In this thesis, we compare our model with various viewpoint selection methods. We have shown that our approach increases the classification performance and decreases the distance traveled between viewpoints compared to the baselines. Furthermore, experimental results showed that our model generalizes well when tested on unseen object classes.

Contents

List of Figures

List of Tables

1 Introduction

Present-day robotic systems use various sensors to perceive information about their environment. The information perceived by the sensors is used to do multiple tasks such as observing the environment and manipulating objects. Good information about the object is critical for accomplishing such a task, but not every view of a scene or object holds the same amount of information. Therefore, moving sensors in the environment can potentially increase the quantity and quality of the information. Optimizing this 'information gain' through different locations of the sensors is called the next-best-view (NBV) problem. The next-best-view problem is defined as finding the next-best sensor placement, which increases the total information gain for two consecutive sensor placements. The sensor placement can be considered as the view of the object from the robot's perspective. The NBV is constrained by the robot's working area and the cost of moving the sensors or objects relative to the sensors. For most robots, it is time-consuming to move sensors. Thus, in most definitions of the next-best-view problem, this is included.

The definition of the information gain differs between system applications and tasks. Direct information gain can be defined as the confidence in the success of the system's task, such as the confidence in the predicted class for a recognition task. Indirect information gain can be defined by maximizing a feature extracted from a view, such as depth-entropy, Kullback–Leibler divergence, mesh quality and many other measures. This definition assumes maximizing the feature's value will increase the task's performance. Additionally, different types of sensors are suitable for different measures.

In this thesis, we propose a solution to solve the next-best-view problem for object recognition. We define the information gain in terms of the entropy of a depth image. The depth-entropy of an image can be considered as a proxy for measuring the amount of information that can be observed from a viewpoint. Entropy measures are used in many other solutions to predict the best viewpoint or the next-best-viewpoint. Next to the indication from other research that depth-entropy can be used as the definition of information gain, we extensively test and validate this hypothesis by comparing it with several different methods, which we consider as baselines. Our solution consists of a model which estimates the depth-entropy of neighbouring views given a point cloud, and the depth-entropy value is used to choose the next-best-viewpoint. Additionally, we propose a model that directly predicts the classification confidence score of neighbouring views and the next-best-view given these scores.

The remainder of this thesis is organized as follows. Chapter 2 covers the theoretical background needed to understand the methods and techniques used in our proposed solution. In Chapter 3 related work about determining the goodness of a view and the next-best-view problem is explained. The methodology is explained in Chapter 4 and the setup of the experiments are covered in Chapter 5. Chapter 6 covers the results. In Chapter 7 we state our conclusions drawn from the results. Chapter 8 ends the report with a discussion about our results and we propose some ideas for future work.

1.1 Research Questions

1. Is it possible to develop a learning algorithm which can predict the goodness of neighbouring views?

    1.1 Which measure is best suited for determining the goodness of a view?

2. Does the extension of the developed NBV selection method improve the classification performance and decrease the travel distance compared to existing NBV selection methods?

3. How well does the NBV selection model perform when tested on unseen data?

2 Theoretical Background

This section provides the necessary background information to understand the topics related to solving the next-best-view problem and determining the goodness of a view.

2.1 Object Representation

2.1.1 Mesh

A mesh represents a 3D object in vertices, edges and faces. A mesh can have different forms in terms of the number of edges that form a face. In a triangular mesh, every face has three edges and three vertices. Most mesh files consist of the location of each vertex in x,y, and z coordinates, followed by the faces, which are represented as the index of the vertices.

2.1.2 Point cloud

Most point clouds are defined as a set of 3D points $\{P_i | i = 1, ..., n\}$ where every point P_i is a vector representing the x, y, and z location of the point and possibly other information about the point such as color or surface normal. Depth and lidar sensors can create point clouds for real-life objects by measuring distances from the sensor to the object. In a simulated environment, point clouds can be extracted from meshes or other volume-based data through point sampling.

2.1.3 Voxel grid

A voxel is the 3D form of a pixel, thus representing a cubic cell instead of a square cell. In most cases, a voxel grid has a size of $n \times n \times n$, making it a cube. But other rectangular configurations are possible as well. The most basic form of a voxel grid consists of an ordered grid where each cell holds a zero or a one. A one indicates the object's presence in that cell, and a zero represents empty space. It is possible to let other vector features be represented in the voxel, like the color of the mesh surface. Then the voxel has a value of (r,g,b); other vectors like surface normal or gray-scale value are other examples of voxel representations. A 3D occupancy grid is a special type of voxel grid where a voxel is set as occupied, empty, or unknown. The latter is mainly helpful for the partial representation of an object.

2.2 Classification

2.2.1 Neural networks

Neural networks are the main building blocks for solving most classification and regression problems using machine learning. Basic neural networks consist of weights and neurons. The weights are multiplied by the input values and go into neurons, which are activated if the input reaches a certain threshold, called the activation function. The task of machine learning is learning such weights that the difference between the correct output and the predicted out is the lowest. We call this difference

the loss, which can be calculated with different loss functions. Neural networks can be very 'deep' and consist of multiple layers, with numerous neurons in each layer. Several special neural network layers exist, like residual, sparse, and convolution layers. Convolution layers are broadly used in multi-dimensions data like images and voxel grids. The following section will go over the definition of a Convolutional Neural Network.

2.2.2 Convolutional neural networks

A Convolutional Neural Network (CNN) is helpful for multi-dimensional data because of the sparse connection between the inputs and the weights in a layer compared to a fully connected neural network (FCNN). A convolutional layer has a kernel, which in the case of an image is an n by m grid of weights; in many models, n = m. The kernel 'slides' over the images and accumulates the output of the n by m operations. One of the main benefits of CNNs is that we can easily use several channels, meaning we use multiple kernels representing different input features. Furthermore, CNN's are cheap in terms of memory and computation time compared to fully connected neural networks.

2.2.3 Loss functions

Cross-entropy The cross-entropy loss function predicts the loss between the predicted and expected class probability. In the case of object classification, the probability is represented as the probability that the respective object belongs to a particular class. In classification, the actual class probability is a vector with the length of the total number of classes, where the correct class is represented by one and the rest by zero. The cross-entropy can be considered the maximum likelihood loss in this case.

Mean squared error The mean squared error function takes the average of the sum of the squared difference between the predicted and ground truth data. Mathematically this is represented by the following formula:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(Y_i - \hat{Y}_i)^2 \tag{1}$$

where n represents the number of classes, Y_i represents the ground truth value, and $\hat{Y}_i$ represents the predicted value.

PointNet loss The loss function used in training the PointNet model consists of the combination of three loss values. The first loss function calculates the loss between the predicted and ground truth. The type of loss function depends on the output of the model. The original paper from Qi et al. [1] uses the cross-entropy loss for classification. The two other losses work as regularization and use the 3×3 matrix and 64×64 matrix from the input and feature transform, respectively (see figure 2). The loss is calculated by subtracting the product of the normalized matrix and its transposed matrix from the identity matrix. The mathematical definition is as follows:

$$L_{reg} = I_n - M \cdot M^T \tag{2}$$

where n is 3 or 64, defining the size of the matrix. The two losses are calculated and added to the cross entropy loss with a multiplication factor of 0.0001. This definition of the loss follows the work of a Pytorch implementation for the PointNet model. [1]

2.2.4 Optimizers

Adam The Adam [2] optimizer is a widely used optimizer for classification and regression problems. One of the key aspects of the Adam algorithm is the use of moments for updating the moving averages of the gradient. For more details, we refer to the paper of Kingma and Ba [2], because it does not lie in the scope of this report to explain the algorithm thoroughly.

2.2.5 Regularization

Batch Normalization Batch normalization is a technique used for regularization and has proven to reduce training time drastically [3] because the learning rate can be increased while keeping the same performance. It is a reasonably simple but effective technique; it normalizes the output of each instance in the batch over the whole batch.

Drop-out Layers Drop-out layers are another widely used technique for regularization. The algorithm works by randomly ignoring neurons (dropping them) during training. This has the effect that other neurons need to compensate for this non-existent behavior of the dropped neuron, and thus the network becomes more robust.

2.3 Object Recognition

We define object recognition as categorizing an object into a predefined object class. There are several methods for object recognition, mainly depending on the input data type. We separate the approaches into three categories based on the input data, view-based, point-based and volume-based methods. View-based methods try to find distinct features for the classes based on 2D images of the object. Volume-based and point-based data try to do this by using spatial information about the object, such as depth data in the form of a mesh, voxel grid or a point cloud. Voxel grids are considered a structured form of data compared to a point cloud because the spatial location of the voxel grid is discrete, whereas point clouds are in a continuous space.

2.3.1 View-based

View-based approaches use 2D images of an object as input for a recognition model. This approach is widely used in all fields of visual recognition. The most common approaches use Convolution Neural Networks (CNNs) as in the ResNet [4] and VGG models [5]. Multi-view recognition is a form of view-based object recognition where multiple views are used to predict the class. This approach is only useful when the sensor can take multiple images from different views of the object.

[1] https://www.kaggle.com/code/balraj98/pointnet-for-3d-object-classification-pytorch/notebook

RotationNet [6] uses such an approach and is one of the best-performing models for object recognition. The downside of RotationNet is that it instantly assumes a number of views available from around the object. Furthermore, the locations of these views are predefined.

2.3.2 Point-based

Most point-based object recognition approaches use hand-crafted features. A pre-defined function maps a point in R^3 to a feature in R^n, which can and ideally should contain all the local and global information about the point. Note that the function can also map from R^n to R^m in the case of additional feature vectors per point. The biggest challenge with point-based approaches is that a raw point cloud is an unordered set of points. Thus, creating a general model for a point cloud is much harder compared to an ordered set of points, like a voxel grid. Additionally, with real-time data, the point cloud's size is variable, which makes it harder to create a general model. But, compared to voxel grids, point-based methods do not lose spatial information. More information is present in less data, in theory this should improve a model. Nonetheless, point-based methods are not used as extensively as voxel-based methods because of the computational complexity.

2.3.3 Volume-based

Most volume-based approaches use volume-based data in the form of a voxel grid. A voxel grid can be considered as the 3D form of an image. Thus, it is not surprising a lot of voxel-based object recognition techniques use similar methods to view-based techniques. One of the most popular approaches uses 3D-CNNs. These special types of CNN have kernels in R^3 instead of R^2.

2.4 Grasp Synthesis

Grasp synthesis is the task of finding possible positions for a gripper to grasp an object successfully. Manipulating objects is one of the keys task of most robotic arms, the way of grasping an object is one of the critical aspects of successfully manipulating the object. It is a very challenging task because of the wide variety of object shapes and possible surface areas to grasp in different environments. Furthermore, a viable grasp option has several physical constraints to consider. The size of the gripper determines the potential area of the object that it can grasp, if it is at all possible. The available surface area of the object, meaning the areas with free space around the object determines the orientation from which the object can be grasped. This includes the space and orientation the gripper can move to and the orientation of the object. This available free space is why many grasp synthesis approaches constrain their solution to only produce grasps from the top since there is almost always free space. From these constraints, we can conclude that the orientation from which to get the best information can be critical and that the next-best-view problem is also applicable to grasp synthesis.

2.5 Pose Estimation

We define pose estimation as finding the orientation of an object or sensor. The pose of a sensor is relatively straightforward; it is the position in which all the actuators are in their zero position. For objects, the orientation is harder to define, because it differs for every object. There is no general rule which defines the front of an object.

2.6 Next Best Viewpoint Selection

Viewpoint selection is the task of selecting a viewpoint in an environment, that is best suited for a specific task. The premise of viewpoint selection is that the whole environment is known before doing the actions or that it is possible to move the sensor to another location relative to the environment. We define a system where the sensors can move as an active vision system. The next-best-view problem applies to both scenarios, with the difference that it solves a different task. In active vision, the goal is to reduce sensor movement while keeping high task performance. There is no sensor movement with static vision, so the goal is to reduce the number of data points (views) while keeping the task performance high. Sensor movement and data-point reduction are essential in most robotic systems to reduce the time for task completion.

Before choosing the best or next-best viewpoint, the goodness of a view needs to be determined. The goodness of a view is determined by the task and is not trivial. The optimal goodness of a view is the performance of the view for a specific task. The only problem with this definition is the cost of determining this performance. If we want to compare the goodness views in this way, the views have to go through the system and move to the view. The goal of reducing data points and camera movement is then lost. Therefore, determining the goodness of a view needs to be done using less costly performance measures. Optimally, these measures should be chosen or predicted without observing or fully observing the respective view to reduce the movement of the sensor. Concluding, we define viewpoint selection as selecting the best or next-best viewpoint for a specific task while reducing the movement of the sensor, the number of data points, and computational power to find the goodness of a view.

2.7 ModelNet

There are two ModelNet datasets [7], ModelNet40 and Modelnet10. Modelnet40 is a 3D object dataset with CAD files with 40 different object classes, totaling 12431 CAD models. A smaller subset of ModelNet40 with ten object classes is called ModelNet10. It consists of the object classes which are considered as most popular. Furthermore, the dataset has a benchmark leader-board [2] recording the performance of more than 60 object recognition models on both datasets. The 3D models are available in the Object File Format (OFF) and are represented as triangle meshes.

[2] https://modelnet.cs.princeton.edu/

3 Related Work

3.1 Multi-View Object Recognition

Multi-view object recognition based methods are currently state-of-the-art in object recognition. The state-of-the-art in multi-view object recognition is RotationNet [6], it is also the leader on the ModelNet benchmark leaderboard. RotationNet is a multi-view approach that simultaneously predicts an object's pose and class. The method works so well that it has a 97.37% accuracy on the test set of ModelNet40 and a 98.46% accuracy on ModelNet10. The biggest downside of the ModelNet is that a lot of hypotheses need to be tested. If there are 40 possible views, and you want to use three views for prediction, you need 120 runs of the model to make the prediction.

Su et al. [8] proposed a solution for object recognition using a multi-view approach with a view-pooling layer. Several views go into a CNN model based on VGG-m [5] to create a feature vector. The feature vector of all the views goes into a maximum view pooling layer to create one feature. This feature goes into a second CNN and ends with a softmax layer which predicts the class probabilities. This is called the Multi-View CNN or MVCNN. The performance is not at the top of the leader-board, but the paper inspired a lot of other research to pursue this type of model [9, 10].

DeepPano is a model created by shi et al. [11] which uses a panoramic 2D view of an object as input to the model. The model is rotation invariant because of its row-wise max-pooling layer. The proposed model is not multi-view based because it uses one panoramic input image. But the panoramic image can be seen as a concatenation of multiple images into one, making it multi-view. The network consists of multiple CNN layers, a row-wise max-pooling layer, two fully connected layers, and a softmax at the end.

One of the key points we can get out of this section is that most state-of-the-art approaches use some CNN, where most of them use a pre-trained version of either VGG [5], AlexNet [12], ResNet [4] or models which are based on them. VGG and Alexnet were one of the first very deep CNNs with a good performance on the ImageNet dataset [13]. The ResNet model uses residual layers and is even deeper than VGG and Alexnet but less complex in some configurations. A residual connection in a neural network means a connection between the output of a layer and a layer further in the network, where at least one layer is skipped. He et al. [4] proposed five different forms of the ResNet model with respectively, 18, 34, 50, 101 and 156 layers. The ResNet model consists of one 7x7 convolution layer, multiple 3x3 convolution layers with 64, 128, 256, and 512 kernels, an average pooling, max pooling, fully connected, and a softmax layer. Every two layers between the pooling layers have a residual connection. The number of layers for each kernel size depends on the total number of layers. Figure 1 shows the architecture of ResNet18. The first two layers have a stride of two, as have the first layers for each time the kernel size goes up.

3.2 Point-Based Object Recognition

Point-based object recognition is a less used technique than image-based recognition because of the limited availability of datasets. But, the number of point cloud and mesh datasets is increasing. One of the main challenges of point-based object recognition is creating an object representation that can

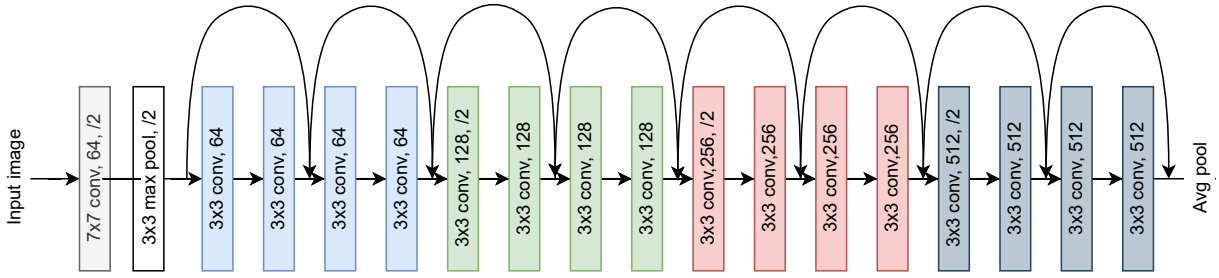

Figure 1: ResNet18 Architecture

be used for recognition. One of the most notable approaches to solving this problem is the PointNet model of Qi et al.[1]. The input for the model is a point cloud of $N \times 3$, where N is the number of points in the point cloud. Each point goes through the model consisting of fully connected, 1-D convolution, and matrix multiplication layers. The output of the central part of the model is an $N \times 1024$ vector which goes through a max-pooling layer to create a global feature (See Figure 2). For object recognition, the global feature goes into an FCNN with K outputs to predict K classes. In the original paper, the model also does part and semantic segmentation with a slightly different network but is not relevant for this project.

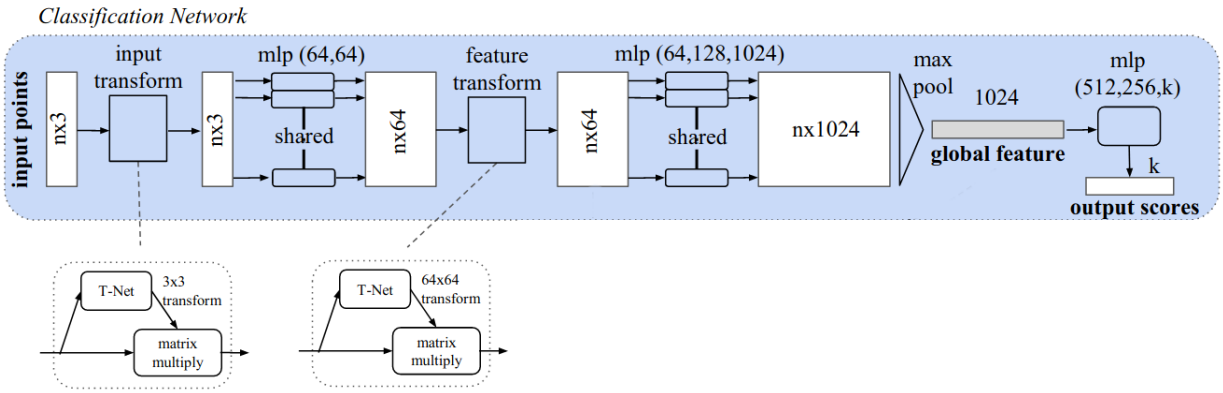

Figure 2: PointNet classification model [1].

After the first publication of the model, there have been several improvements. PointNet++ [14] tries to capture local structures in the point cloud by a hierarchical neural network and performs better than the original model. Ahmed et al. [15] created the Edge-Aware PointNet, using techniques to find the edges of objects in combination with predicting the class of the objects.

3.3 Volume-Based Object Recognition

The state-of-the-art in volume-based approaches uses voxel grids as input for their model. Wu et al. [7] proposed a volume-based method that uses a 3D convolutional deep belief network, transforming a 2.5D image into a voxel grid as input for the model. Furthermore, they proposed a system to select the next-best viewpoint, which can be used to improve the reconstruction and recognition of an object. Most notably, they created the benchmark dataset named ModelNet, which is used in many state-of-the-art 3D object recognition models.

The VoxNet model [16] uses an occupancy grid as input for the model. VoxNet uses a reasonable simple 3D CNN, with a max pooling layer, fully connected layer, and softmax layer. They are most notable for using Lidar sensor data and other depth data, on which they perform reasonably well.

The FusionNet model is one of the best performing models on the ModelNet datasets that use voxel grids as inputs. It uses a combination of two 3D-CNNs and one 2D-CNN and thus is not a complete volume-based technique.

Field-Probing Neural Networks (FPNN) [17] are a form of NNs which use a voxel grid as input. The network does not go over the entire grid but learns only to use the most informative points. It does so by using probing filters. They use 1024 filters which all have eight probing points. The initialization of the filters is done by spreading out the input locations of the points. The second layer in the network is a Gaussian layer which acts as the weight for the selected points. The most significant benefit of this solution is that the complexity stays the same, even if the voxel grid size increases, without necessarily losing information.

3.4 Shape Measures

The first step in next-best-view selection is determining the goodness of a view for the specific task. The most obvious choice for assessing the goodness of a view is the actual performance of a view for a particular task, such as classification performance. The downside of this method is that it is too computationally expensive in most cases, and more importantly, it does not generalize very well. Using such a measure for a specific task will probably only work for the particular task and dataset. Therefore it would be better to find a more general measure for the goodness of a view. Dutagaci et al. [18], Polonsky et al. [19] and Secord et al. [20] have done research in which they use human subjects to select the best viewpoints to specify the goodness of a chosen view and compare these choices with several computational measures which determine the goodness of a view. In the following section, different 2D shape measures proposed by the three papers are presented. The section is divided into five sub-sections based on the type of information of the object; area, silhouette, depth, curvature, and semantic information.

3.4.1 Area Measures

Entropy The entropy of a view should give the amount of information in that view, determined by the amount of uncertainty or differences in a random variable. The most common definition of

entropy for images is Shannon's entropy, and is mathematically defined as:

$$H(X) = -\sum_{i=1}^{n} p_i \, log(p_i) \tag{3}$$

where p_i is the probability of the pixel value occurring in the stream of pixel values, the image X. It differs between research as to how the entropy is calculated exactly. There are mainly differences in how the probability is calculated. In Dutagaci et al. [18] the probability is calculated as the ratio between a projected triangle's area and the view's total projected area. This assumes we have knowledge about the mesh of the 3D object. Most research use some ratio between local properties (triangles, pixels, faces) and the total projected area.

View Area The view area or projected area is the area of the object visible from a specific view. This can be represented as the projected area, such as the number of non-background pixels in an image, the number of points extracted from a point cloud, or the mesh area visible from a viewpoint.

Ratio of Visible Area The ratio of visible area [18], surface visibility [18] or visibility ratio [19] measures the ratio between the visible area of the viewpoint and the total area of the object. As with the view or projected area, the visible area is determined by the number of non-background pixels or the actual area of the object seen from the view.

3.4.2 Silhouette measures

Silhouette length The length of the silhouette can be calculated by edge detection of the view, the length of the edges combined is the silhouette length.

Curvature entropy Curvature entropy is the entropy over the curvature distribution of the visible surface of the object. In Dutagaci et al. [18] the measure is calculated by the ratio of the mean curvature and the curvature between vertices. They do not specify how they calculate the curvature. In Polonsky et al. [19] the estimated curvature at a vertex is defined by the standard angle-deficit approximation, given by:

$$C(v) = \frac{2\pi - \sum_i \theta_i}{3\sum_i A_i} \tag{4}$$

where θ_i is the sum of apex angles from triangles coincident with vertex v, and A_i is the sum of areas of the triangles coincident with v.

Silhouette entropy The silhouette entropy is similar to the curvature entropy but calculates the entropy with the outer edges of the visible area, the silhouette.

3.4.3 Depth measures

Depth distribution The depth distribution [20] is calculated by creating a histogram of depth values taken at each pixel of the view. The measure is defined by:

$$D(X) = 1 - \int H(z)^2 dz \tag{5}$$

where H(z) is the normalized histogram of the depth z of the view X per pixel. The measure becomes small when most points are at the same depth and is maximum when all points are at a different depth.

Maximum depth The maximum depth method in Secord et al. [20] tries to find the view with the maximum projected area and the maximum depth in a view.

3.4.4 Surface curvature measures

The **mean curvature** and **Gaussian curvature** [20] are both measures to calculate the amount of curvature in the mesh. For more details about these kinds of measure we refer to the paper by Secord et al. [20].

Mesh saliency The mesh saliency [18] [20] [18] is based on the local curvature of the surface of an object. It is constructed by the mean curvature at different levels of the mesh. The saliency score is the sum of the saliency values at each visible vertex from the respective view. The method was first proposed by Ha Lee et al. [21] and is used in other research to find visually interesting regions in a mesh object.

3.4.5 Semantic measures

Semantic measures are measures that prefer seeing some predefined part of the object. In Secord et al. [20] they tested three of these methods, above preference, eyes, and base. Above preference prefers views taken from above. The eyes method likes views from which eyes are visible, and the base method tries to avoid the base of the object, which is the bottom for most objects. These methods are only feasible if we have a lot of information about the object. The methods are very specific and can only be used for particular objects and use cases.

3.5 Next-Best-View selection

Best viewpoint selection and next-best-view selection are considered problems for many different tasks, the thing they have in common is that they use 3D object data. Examples of different tasks are thumbnail generation, 3D scene generation, surgery planning and view-based 3D object recognition. The different tasks also have different best viewpoints, making it very hard to create a general solution for the next-best-view problem. In both Secord et al [20] and [18] the entropy method is in the top of their proposed viewpoint selection methods and can be considered as a good general measure

to determine the goodness of a view.

The methods described in Section 3.4.1 do not test the viewpoint selection for classification or any other task. They only formalize general measures, which can determine the next-best or best viewpoints. Since this report proposes a solution for next-best-view selection for 3D object recognition, we also looked at other work concerning either next-best or best viewpoint selection. Leifman et al. [22] proposed a different hand-crafted feature approach for next best viewpoint selection using a model which first determines the regions of interest on the surface of an object, and then chooses the next best viewpoint. The regions of interest are selected by finding dissimilarities in vertices of the object, the dissimilarities are defined by a vertex descriptor based on various local descriptors.

Parrissoto et al. [23] proposed a model which predicts the depth-entropy of 60 views around a 3D object, which results in an entropy map of 12 by 5. The local-entropy peaks are chosen from this map as the best viewpoints and used in a classification model. The model does not test on next-best-viewpoint selection but tests best viewpoint selection for classification.

Ding et al. [24] developed a training algorithm consisting of a linear regression model trained to predict the surface complexity. They define the surface complexity by fusing the three view descriptors, variance, information entropy, and Vollath. The higher the predicted value, the better the viewpoint.

Shui et al. [25] proposed a system that uses the calculated viewpoint entropy from 42 views around the object, the 14 viewpoints with the highest entropy are chosen as the best viewpoints. This solution is not optimal because it assumes information about all the viewpoints is available.

Some approaches do not use any predefined measure to predict the next-best-view. Mendoza et al. [26] propose a 3D-CNN from which they predict the next-best-view from a set of 14 possible viewpoints. The goal of the next-best-view selection model is to go to the viewpoint which can reconstruct the image as best as possible compared to the already constructed part of the object. McGreavy et al. [27] propose a solution for next-best-view planning for object recognition in cluttered environments. Their model consists of two stages; environment analysis and model analysis. The environment analysis determines the possible viewpoints the sensor can reach without visually occluding the to-be-evaluated object with another object. The model analysis determines the possible viewpoints from which the unexplored area of the object becomes visible. They do so by comparing the visible part of the object to the actual 3D model of the object.

Note that many models use some shape descriptor based on the 2D views of the object. Additionally, measures like information entropy and view area are recurring measures in many models and are considered suitable measures to determine the amount of information in an image.

4 Methods

This project aims to find a solution for the next-best-view problem by determining the goodness of a view for neighboring viewpoints. We propose a solution that tries to maximize the information gain by choosing the next-best-view according to the predicted depth-entropy of the neighboring views. To predict the depth-entropy, we propose a model based on PointNet [1] which predicts a local-depth-entropy map with the current view in the center.

Figure 3 shows the pipeline of the model including the classification and next-best-view selection. The pipeline starts with an initial random view chosen from the 40 possible viewpoints. The 2D RGB image of this view goes into the ResNet classification model, and the value of the output vector with the highest confidence score is compared to a threshold. If the threshold is reached, we consider the class which belongs to this confidence score as the predicted object class. If it is lower, the point cloud of the respective view serves as input for the PointNet model. A fully connected neural network predicts the local-depth-entropy map from the PointNet feature. The local-depth-entropy map's highest peak and lowest valley are extracted and used for the NBV selection. Depending on the object, the peak or valley is considered the next-best-view. The 2D image of this view serves as input for the ResNet classification model. The output of the ResNet model is added to the previously predicted features. The accumulated feature goes into a softmax layer and outputs the confidence scores. This continues until either the threshold for the confidence score is reached or if a total number of five viewpoints are explored. Next to the prediction of the local-depth-entropy map, we also trained a model which predicts the local classification performance map.

In the remainder of this chapter, we will explain in more detail which methods, techniques, and models are used to create and test the pipeline for NBV selection for classification, shown in Figure 3. In the first section, we explain which dataset is used and how we generated 2D images, depth images, and point clouds. Section 4.2 covers the proposed local map prediction model for both depth-entropy and classification performance. In section 4.3 we introduce the object-recognition model.

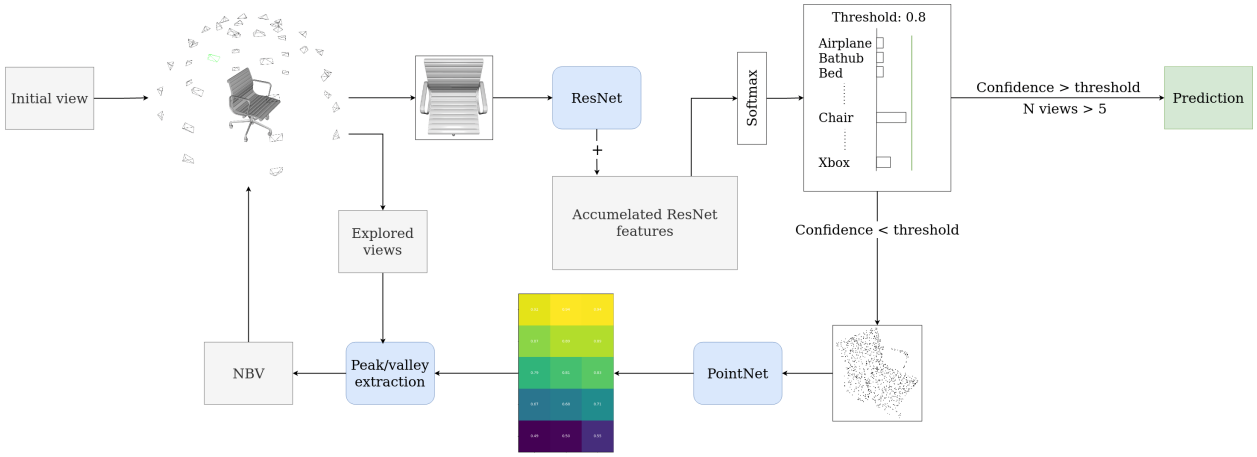

Figure 3: Pipeline for the next-best-view selection model for classification

4.1 Dataset

We choose to use the ModelNet dataset for training and testing our models. The large number of annotated 3D objects makes it easy to render 2D images, depth images, and generate point clouds. Furthermore, the ModelNet datasets are used extensively for object recognition; thus, we can easily compare our results. For ModelNet40 and 10, a few objects are left out due to broken files. The total number of object instances for ModelNet is 12313 and 4905 for ModelNet10. Figure 4 shows the distribution of objects per class in the ModelNet40 dataset. The number of objects is not very evenly distributed throughout the dataset. The classes in ModelNet10 are better distributed, but there is also still a significant difference in the number of objects per class (see Appendix 19)

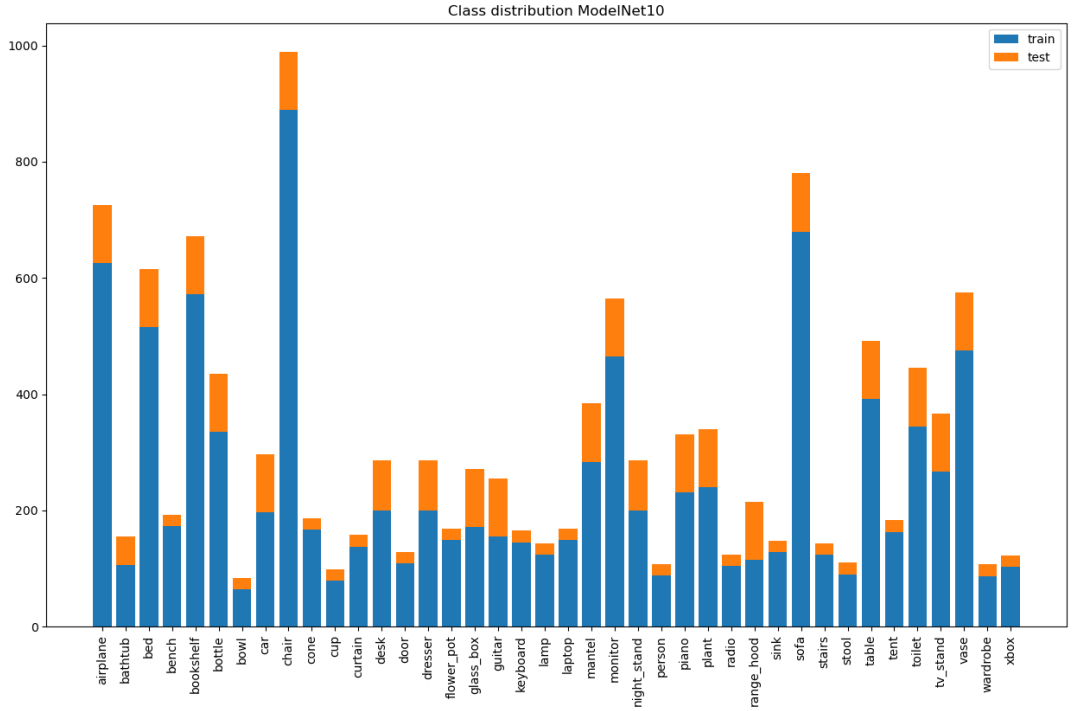

Figure 4: The number of objects for each class in the ModelNet40 dataset

4.1.1 Viewpoint distribution

We have chosen to use a fixed set of views around the objects to consider as viewpoints for the object recognition task. On both ModelNet10 and 40, we used two approaches regarding the number of viewpoints. The first approach renders 12 views around the object in a uni-directional manner at an angle of 45 degrees (Figure 5, left). Note that the 12 views approach is only used as preliminary testing for the classification model to reduce training time. The second approach generates views at 40 positions regularly distributed from the middle to the top of the object (right image in Figure 5). A

spherical coordinate system is used with five different values for the polar angle and eight different values for the azimuthal angle regularly distributed around the object. We refer to Appendix A.4 for more details about the coordinate frame. This creates five rings at different heights with eight views evenly spaced around the object. With this setup, we assume the object is always in an upright position. We choose 40 views, so we have enough views to create enough local maps with different views and still make sure the views are not too similar.

Figure 5: Viewpoint distribution pattern with 12 views (left) and 40 views (right)

4.1.2 Data processing

The 2D views, depth maps and partial point clouds are all created with the help of the open3d library [28]. The models are loaded in as meshes and the location of the vertices are normalized to [-1,1]. The 2D image, depth map and partial point cloud at each viewpoint position around the object are generated creating three datasets with a test and train set. The test-train split is 20-80 as is the default for the ModelNet dataset.

2D images The 2D images are rendered and saved as 224×224 pixels images, with rgb values between 0 and 255. We save each view by their polar and azimuthal angles. In the maps we also refer to the viewpoints by their index in the set of views, starting at 0,0 moving sideways and starting again at the row above (see Appendix A.5 for the schematic). The number of data-points per dataset are shown in Table 1.

Depth-entropy map The depth-entropy is calculated from the depth map, and the depth map is a 224×224 grayscale image. Pixels with a value of zero are pixels where no object is present. For

	ModelNet10	ModelNet40
12 views	58,850	147,756
40 views	196,200	492,440

Table 1: Number of 2D images per dataset

every view around the object, Shannon's entropy is calculated as defined in equation 3. The depth-entropy is zero if all the pixels are at the same depth and increases if the number of different depth values increases. If a depth image has a lot of pixels where an object is present (values above zero), the depth-entropy is higher than an image with a higher number of zero pixel values. Meaning that if more of the object is visible from a certain view, the depth-entropy is higher than if less of the object is visible. The entropy gives the average information or uncertainty of a random variable, in our case, the depth values of the pixels. The depth-entropy values are normalized for every object so the values range between zero and one (see Figure 6). The viewpoint with a value of one has the highest depth-entropy, and the viewpoint with a value of zero has the lowest depth-entropy.

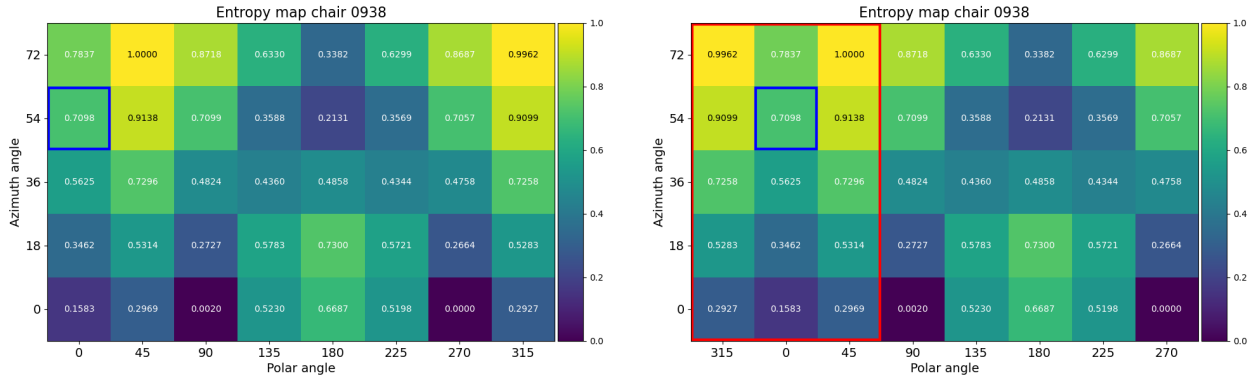

Figure 6: Extracting of the local-depth-entropy map from view (0,54)

To create the local-depth-entropy map for every view, the full map is shifted such that the current view is located in the second column of the map. The local-depth-entropy map is extracted from this representation by taking the three-by-five matrix from the full depth-entropy map with the second column as the center, see Figure 6.

Partial point clouds The partial point clouds are generated with the built-in function of Open3D library. It creates a point cloud with the zero-zero point in the middle of the object. The function uniformly samples 1024 points from the mesh, resulting in a vector of 3×1024. Additionally, the point clouds are normalized between zero and one to have the same scale for the input and output for the PointNet model.

4.2 Viewpoint Selection

4.2.1 Local-depth-entropy Map Prediction

The Local-depth-Entropy Map (LEM) prediction model aims to create a three-by-five depth-entropy map of the neighbouring views from one viewpoint. The three columns represent the angle around the object (azimuthal angle) and the rows represent the polar angles. (see Figure 7).

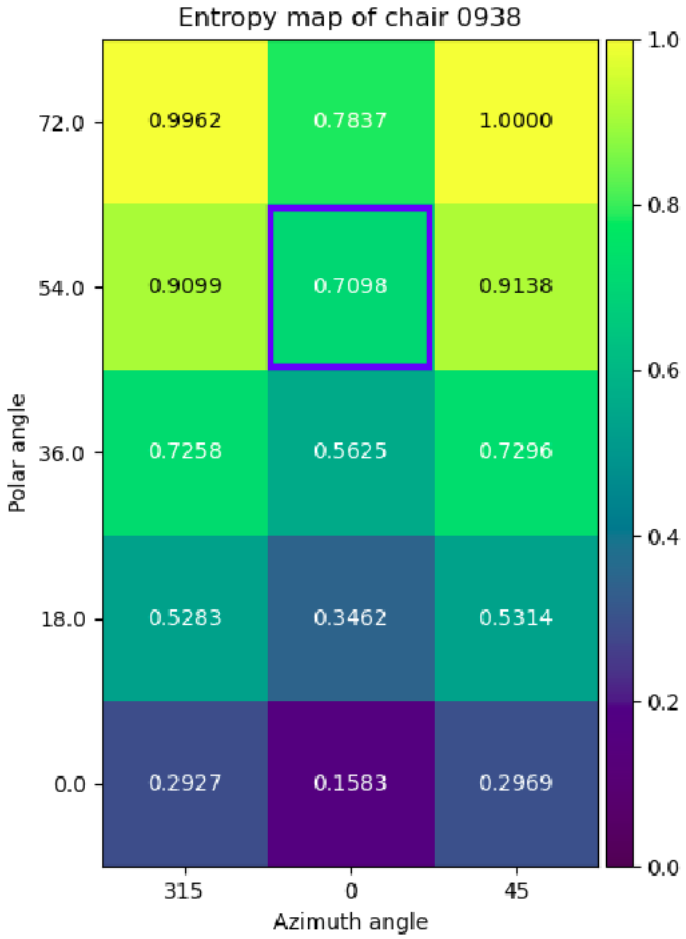

Figure 7: Normalized depth-entropy map of chair 938 from the (0, 54) viewpoint (highlighted in blue).

The model is based on PointNet [1] (Figure 2), the input is a 1024×3 point cloud and outputs a vector of 15 values, which represents the three-by-five depth-entropy map. The network only differs from the original PointNet model in the last layers. The last fully connected layer has an output of length 15 and uses the Relu activation function.

4.2.2 Local classification Performance Map

The Local Classification performance Map (LCM) is also a three-by-five map, but this map consists of the predicted classification confidence scores for every view. The first step in obtaining these maps is using all the images from the test and train dataset separately as input for the trained ResNet model. For every view, the confidence score and prediction are saved. Only the maps of the objects which are classified correctly for at least one view are included in the dataset. Furthermore, the confidence score is set to zero if the class is mispredicted. This decreases the training dataset to 202640 instances and the test dataset to 97120 instances for ModelNet40. But this is still enough to train the model properly. Furthermore, the ratio of train/test is now about $0.68/0.32$. We decided not to swap more test instances to train instances for consistency between all the datasets in terms of which objects belong to which set.

4.2.3 Next-Best-View selection

We use the peak_local_max function from the scikit-image package to find peaks in the local maps. The function uses a maximum filter of three by three. The filter convolves over the map and sets the value of each viewpoint as the highest value within the bounds of the filter. The new map is compared to the original map, and the values of the points that are equal in both maps, and are not direct neighbors of each other, are considered local peaks. If two peaks are too close, the highest peak is chosen. In our method, we choose the highest peak as the next-best-view if it is not already explored. If it is already explored, we select the next highest peak. If there is no peak, the view with the next highest depth-entropy value is chosen as NBV. For some objects, we choose to find the local

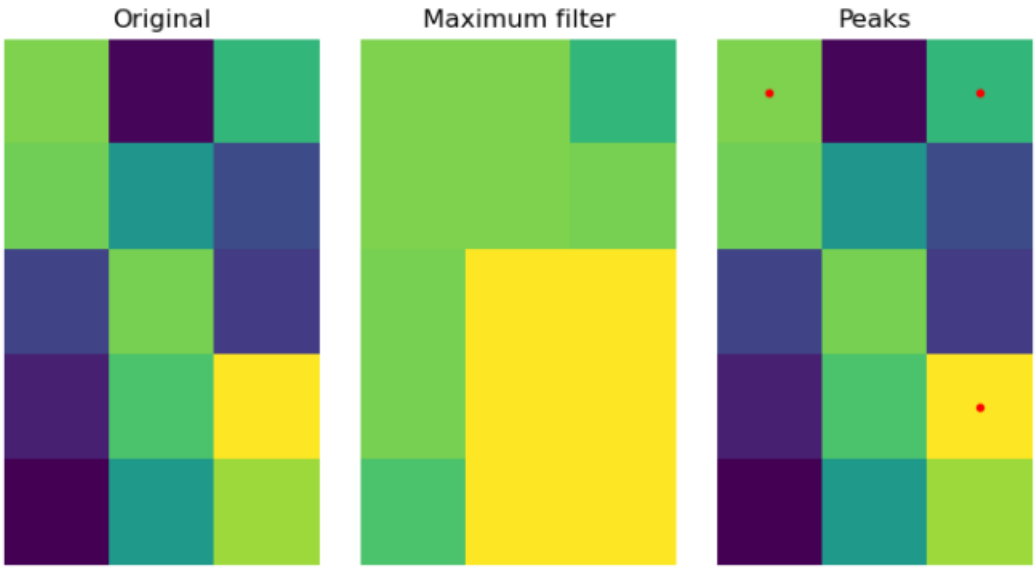

Figure 8: Peak selection method

valley in the map. This method works the same as peak selection, but only it tries to find the lowest local values in the map.

4.3 Object Classification

4.3.1 Resnet

We used the ResNet object classification model to test our viewpoint selection model. Resnet is state-of-the-art in view-based object recognition and image-based recognition in general. Furthermore, pre-trained models are available for multiple packages and different programming languages, and much open-source code is available. The model's architecture is the same as the original one, as explained in Section 3.1. An RGB image of 224×224 serves as input for the model, and the output is a $n \times 1$ vector of the probability that the object belongs to a class. Where n is 10 or 40, depending on the number of classes in the dataset.

4.4 Multi-View

In the next-best-view and classification pipeline (3) the ResNet model is used as a multi-view object recognition approach. If multiple views are explored, the feature vectors of size n are accumulated before they go into the softmax layer. The index of the value with the highest score after the softmax layer is considered the predicted class.

5 Experiments

All training is done on the RUG's High-Performance Cluster (HPC). We used the NVIDIA V100 GPU with a virtualized Intel Xeon Gold 6150 @ 2.70GHz CPU. The models are created, trained, and tested with the help of Pytorch 1.7.0 and Torchvision 0.8.1, in combination with several other packages. The models are tested locally on an NVIDIA GeForce GTX 1050 GPU with an Intel Core I7 @ 2.8GHz CPU. For more details about the code, we refer to the Github page of the project [3].

5.1 Classification

The experiments aim to train and test a ResNet model that can be used in the NBV pipeline. High performance is not of the greatest concern for this model since we are not trying to optimize the object recognition network but rather optimize the network's input through an NBV model. Thus we do only a minimal fine-tuning of the hyper-parameters for this network.

Training As a proof of concept and to tune some parameters, we trained a ResNet18 network on the ModelNet10 dataset with 12 views and a batch size of 32. To test if the batch size makes a difference during training, we trained the ResNet18 model on ModelNet10 with 40 views and a batch size of 4, 8, 16, 32, and 64. A batch size of 32 has a slightly better performance than the rest of the models; thus, this batch size iwas chosen for further training of the networks. With a batch size of 32, ModelNet10 with 40 views is trained with all the five possible ResNet architectures. We did this to see if a bigger ResNet model would improve the performance. To decrease training time we did the same for ModelNet40, but with 12 views. Both tests determined that ResNet34 performed best, and thus ResNet34 with a batch size of 32 was trained on ModelNet40 with 40 views. During training, batch normalization is done after every convolutional layer as a regularization. The Adam optimizer is used in combination with the cross entropy loss function. The learning rate was set to 0.001 and halves every fifth epoch.

5.2 Point Correspondence

The number of points that correspond between each view is calculated to determine the right size for the local maps. This so-called point correspondence is visualized for each view in a point correspondence map. The first step in calculating the correspondence map is determining which points from the whole point cloud of an object are visible from every view for every object. For every view, it is determined which points are the same in every other view. This leads to a five-by-eight map for every view. Figure 9 shows an example of the point correspondence map of the view at 18 degrees polar and a 90 degrees azimuthal angle. The maps are normalized such that the view with the lowest correspondence is zero and the view with the highest correspondence is one, which is the evaluated view.

[3] https://github.com/koenbuiten/PointNetNBV

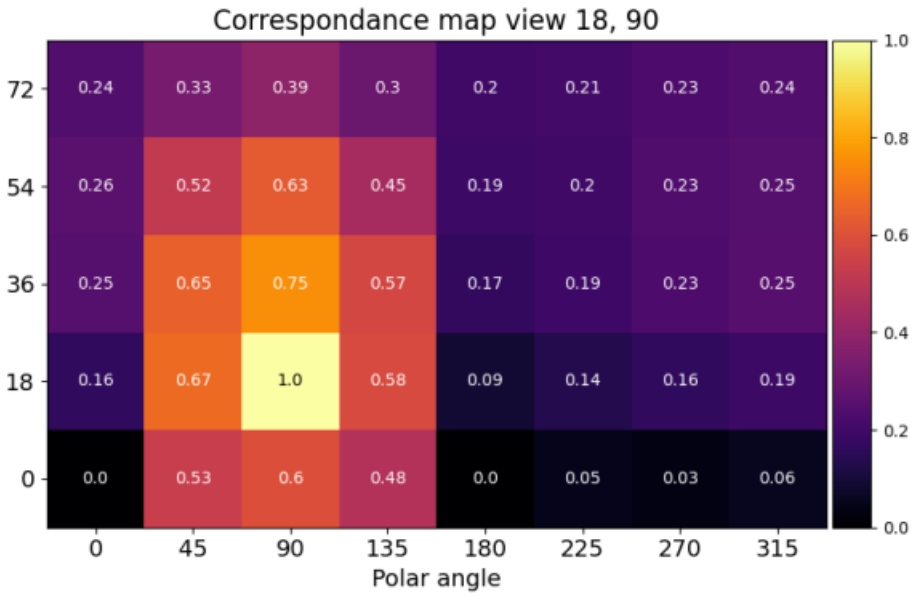

Figure 9: The point correspondence map of view 18, 90 (polar, azimuthal)

5.3 Next-Best-View Selection

These experiments aim to train an NBV selection network and compare it to existing baselines. Due to a limited amount of time, only hyper-parameter tuning for the learning rate and scheduler is done with some preliminary testing. The other parameters are defined by the parameters used in the original paper [1].

Training After some preliminary tests, a batch size of 16 is chosen for training the PointNet network. The learning rate is set at 0.001 with a decay factor of 0.1 every 5th epoch. The Adam optimizer and the loss function described in section 2.2.3 are used to train the model. In total, four different networks are trained, local-depth-entropy prediction and local performance prediction on the ModelNet10 and ModelNet40 with 40 views.

Testing As a preliminary test for viewpoint selection, we tested the classification model with four viewpoint selection methods based on the highest depth-entropy, lowest depth-entropy, random, and best possible viewpoints. This is done on the ResNet34 model trained on ModelNet40, so all the classes are included. The test is done by selecting the five best views according to the viewpoint-selection method and using those as input for the classification model. The goal of these tests is to have an indication about the performance of our NBV model before training and testing the actual NBV model. Furthermore, we want to know which classes react better to selecting the highest or lowest depth-entropy. The best possible viewpoints are selected by their performance on the classification model, which is defined by using all the views as input for the model and saving the prediction together with the confidence score.

To calculate the performance of our NBV model based on depth-entropy and confidence, we defined two metrics. We consider the loss between the predicted and the correct maps for the first metric. The second metric calculates the euclidean distance between the predicted highest peak and the correct highest peak in the map. We set three thresholds for the euclidean distance. We consider the prediction correct if the metric is lower or equal to the threshold. In Figure 10 a visualization of the metric is shown.

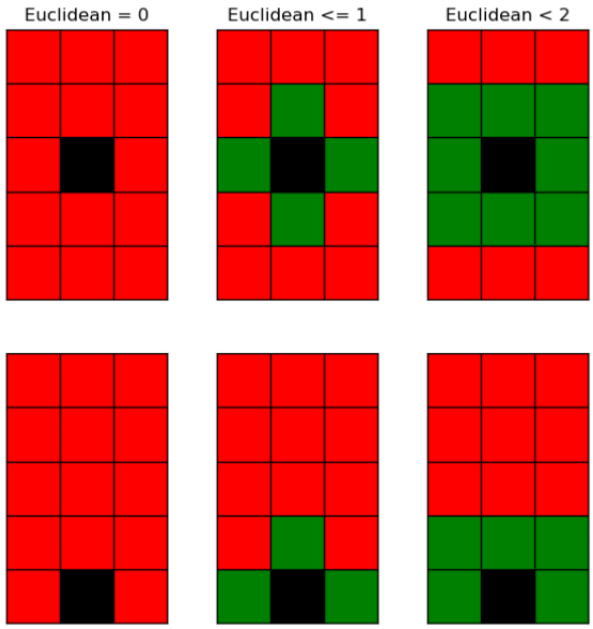

Figure 10: Euclidean metric visualization, black is the ground truth, green corresponds to predictions that are considered correct, and red to a wrong prediction.

We compare the performance of the proposed viewpoint selection approach with several baselines (Figure 11). The easiest baseline method for viewpoint selection is random viewpoint selection. Our methods work with local NBV prediction. Thus, the random baseline also selects a view in the three-by-five grid of possible local viewpoints. Furthermore, it only selects views that have not previously been visited. The second baseline method chooses the viewpoints furthest away from the current viewpoint. The last baseline consists of a uni-directional viewpoint selection method. The method starts with a random view from which the viewpoint on the next azimuth angle at the same polar angle is chosen. All the baseline methods work such that the viewpoint selection stops once a certain threshold for the confidence score is reached or if it has explored five views. During testing, the thresholds are set to 0.85, 0.9, 0.95, 0.97, and 0.99. Because the prediction starts with a random viewpoint, we run every method ten times. Additionally, we ensure that the initial random views are the same for every method and threshold but differ for every tested object and per run.

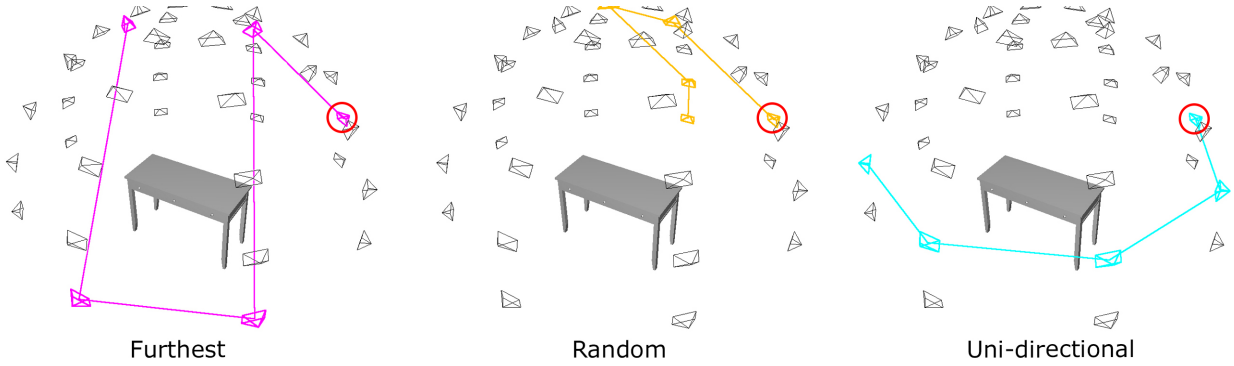

Figure 11: Examples of the three NBV selection baseline methods furthest, random and uni-directional. In all cases, the starting point is highlighted by the red circle

6 Results

6.1 Classification

The accuracy of the class prediction defines the performance of the classification model. Table 2 shows the instance accuracy of the two best models trained on ModelNet10 and ModelNet40 with 40 views. The entire table with all the trained models, including the models trained on 12 views, is visible in Appendix A.3.

Model	classes	viewpoints	batch size	accuracy
ResNet34	10	40	32	**87.10**%
ResNet34	40	12	32	**87.03**%

Table 2: Instance classification performance on the test set for the best model trained on ModelNet10 and 40 with 40 views.

Figure 12 shows the confusion matrix for the best model trained on ModelNet10 with 40 views. We noticed that the four classes, desk, dresser, night stand, and table, have low performance compared to the other classes. Furthermore, table and dresser are wrongly classified as each other. The same holds for night stand and dresser, where night stand is also predicted as a table in 10% of the cases. This behavior can be explained by looking at the type of object. A desk and a table are very similar, the definition of something being a table or desk is even for humans hard to tell. The same holds for the dresser and the night stand. They are generally both smaller tables with drawers and cabinets.

The same behavior occurs when testing the network trained on ModelNet40. Cup and vase, flower pot and plant are predicted as each other. The confusion matrix for the ResNet34 model trained on ModelNet40 is shown in Appendix A.6.

6.2 Next-Best-View Selection

As explained in Section 5, we did preliminary tests on the classification model regarding best-view selection. In this test, the five best views from the set of 40 views are chosen according to the low depth-entropy, high depth-entropy, random, or best viewpoint selection method. From the five views, a majority vote procedure is followed to select the predicted class. The procedure accumulates the ResNet features and lets the accumulated feature go through a softmax function to get the confidence scores. The index with the highest confidence score is considered the correct class. Table 3 shows the accuracy for the ModelNet10 classes tested with ResNet34 trained on ModelNet40. The table with all the classes is shown in Appendix A.15. These results show that some classes prefer views with low depth-entropy and others with high depth-entropy. The performance between low and high depth-entropy is best visible in the table class. The difference in performance between high and low depth-entropy is about 42%.

In Figure 13 the depth-entropy map of a table shown, two views are highlighted as examples for viewpoints with a high and low depth-entropy value. If we compare the depth-entropy of these views

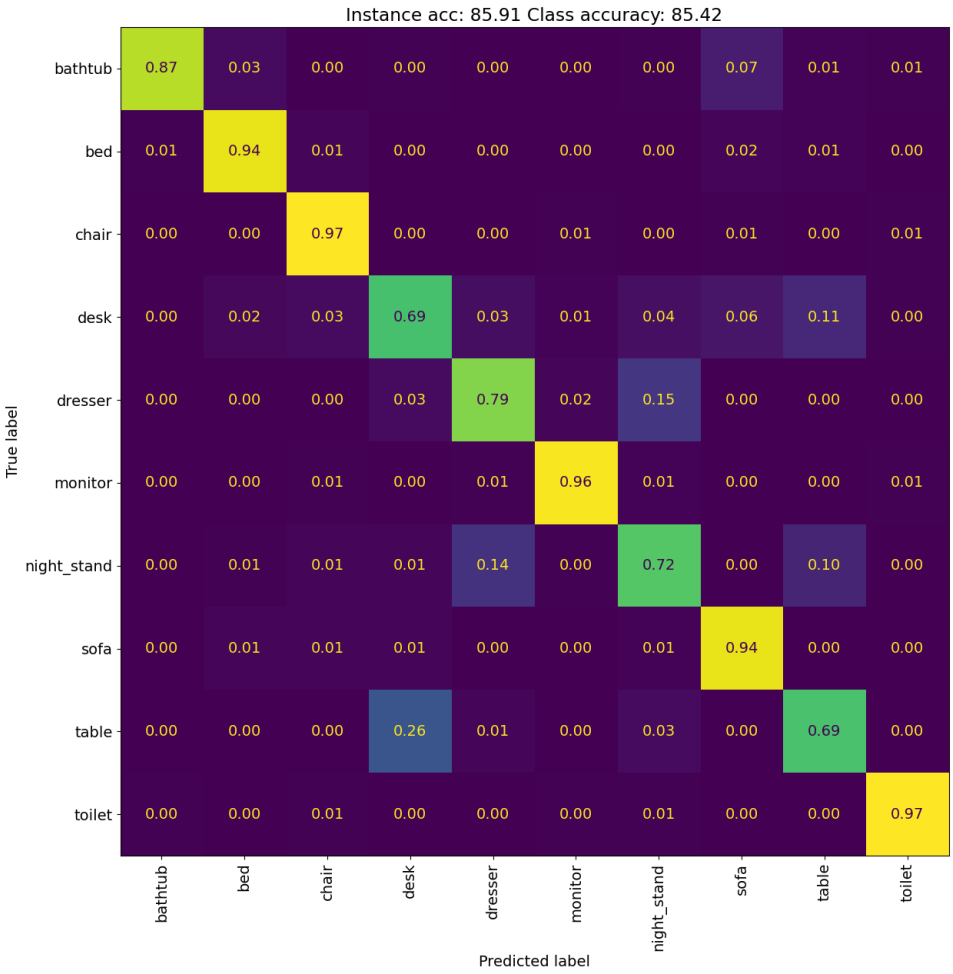

Figure 12: Confusion matrix for ResNet34 trained on ModelNet10 with 40 views

to their corresponding confidence score in Appendix A.7, we see that the values are exact opposites of each other. The view with a high depth-entropy has a low confidence score and the view with a low depth-entropy has a high confidence score. This can be explained by looking at the respective images of the views. The view at 0,72 shows only the tabletop, which is a rectangle. But because it is at a slight angle (18 degrees) compared to a full tabletop view, the depth values from the bottom of the image to the top are all different, and thus the depth-entropy is high. But intuitively, we can say that a rectangle does not give us much information about the object, it could also be a dresser, desk, night stand or anything with a rectangular flat surface. The view with a low depth-entropy shows the four legs of the table and part of the bottom of the tabletop. First of all, fewer pixels of the object are present in the image; thus, the chance of the depth-entropy being high is already lower. Secondly, the legs at the front are at the same depth as the back legs. Only the small part of the bottom side of the table top has some difference between depth values. For this reason, it does not surprise that

the classification model performs better on views with low depth-entropy than on views with high depth-entropy.

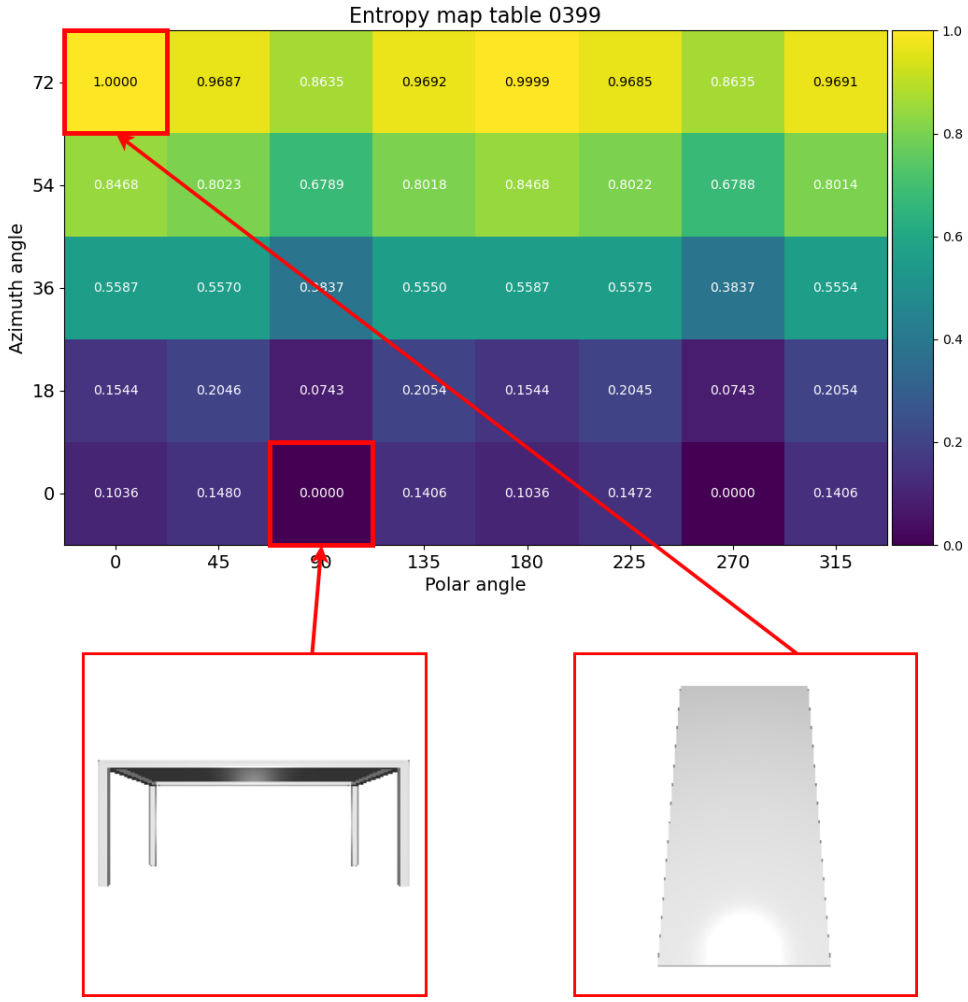

Figure 13: Depth-entropy map of table 0399 with the images of the views with the lowest and highest depth-entropy value in the map.

Table 3 also shows us the best possible performance by selecting the best viewpoints for each object. This performance is as expected, much higher than the other selection methods for about half of the classes. This indicates that a performance increase is possible for some of the classes, but that for the classes which already have a good performance on random selection, it will be challenging.

	low depth-entropy	high depth-entropy	Best	Random
bathtub	0.78	**0.9**	0.98	0.89
bed	0.98	**1.0**	1.0	1.0
chair	**0.97**	0.94	1.0	0.96
desk	**0.67**	0.65	0.99	0.76
dresser	0.69	0.69	0.95	0.68
monitor	**0.95**	0.94	1.0	0.96
night_stand	0.59	**0.67**	0.99	0.68
sofa	0.91	**0.97**	0.98	0.96
table	**0.84**	0.42	1.0	0.8
toilet	**0.99**	0.97	1.0	0.99

Table 3: Accuracy per class for low, high depth-entropy selection, best possible view, and random view selection for a fixed number of five views as input for the ResNet model. A majority vote chooses the prediction.

6.2.1 Point correspondence

From the point correspondence maps of all the views in Figure 14, we can see that a region of three by four views has a high correspondence for every view. For some views, the region is much wider, but at least a three-by-four map is visibly high in all the views. With this information, we decided to use a three-by-five local map of neighboring views from which we try to predict the depth-entropy and confidence score. We decided on using five rows instead of four to ensure the map is the same for every view in terms of rows and does not have to be shifted up and down.

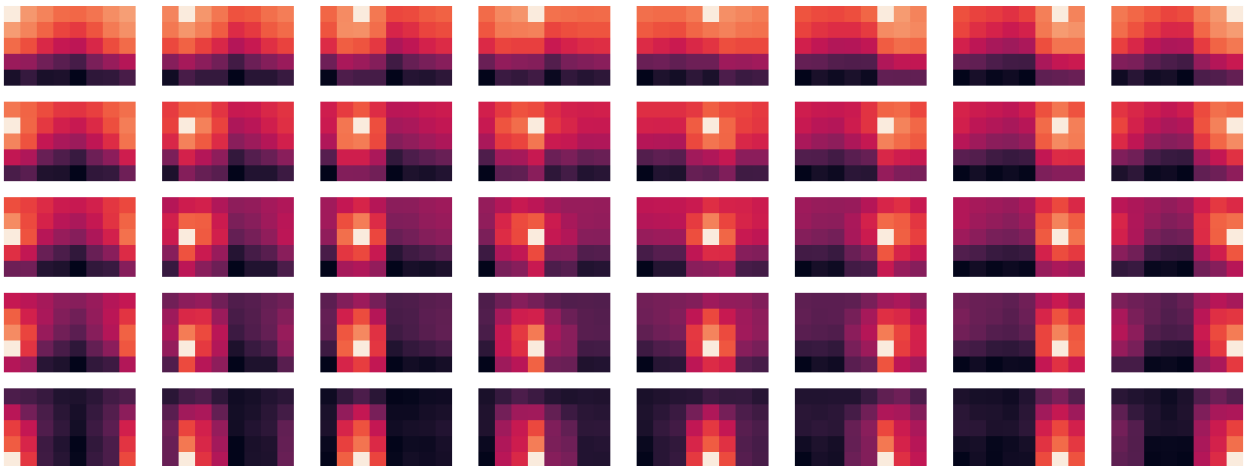

Figure 14: Average point correspondence maps for every view.

6.2.2 Map prediction

In this section, the results are shown for the depth-entropy and performance map prediction models. In Appendix A.2 the loss curves for testing and training the model on ModelNet10 are shown. The loss for testing and training converges nicely to a low loss value and settles around the 10th epoch. The overall loss on the test set is reasonably low for all the models, as shown in Table 4. The table also shows the peak selection accuracy (see Section 8) for each model trained on the different datasets, with the three different thresholds. The results for both the local-depth-entropy map models show that the predicted peaks are at least in the region of the correct peak in about 85% of the views. For the local confidence prediction model, this is not the case. Between views in the confidence maps, the difference in the scores is not as big as with the depth-entropy map, making it harder to predict the correct peak. A lot of values are either one; a good prediction and the maximum confidence score, or the values are 0 because the prediction is wrong. Values in between do occur, but just less frequently.

For our primary model, the local-depth-entropy map approach trained on ModelNet10, the performance per class is visible in Table 7. It shows that the model makes the prediction reasonably well for all the classes. This indicates that the model generalizes well. Additionally, in Appendix A.11 we show four examples of predicted local-depth-entropy maps together with their respective ground truth map. The examples show that even with a medium loss value, the map prediction is still good when tested on ModelNet10. Furthermore, the examples from ModelNet40 show us that the model generalizes pretty well because it can predict the map for never-seen-before object classes.

NBV selection	Classes train	Classes test	$d = 0$	$d <= 1$	$d < 2$	avg loss
LEM	40	40	0.1873	0.6019	0.8517	**0.0203**
LEM	10	10	**0.3988**	**0.8011**	**0.8677**	0.0222
LEM	10	30	0.0819	0.3642	0.6140	0.0973
LEM	10	40	0.0898	0.3790	0.6058	0.0890
LCM	10	10	0.0461	0.2332	0.4276	0.1120
LCM	10	30	0.0258	0.1549	0.3521	0.1345
LCM	10	40	0.0254	0.3402	0.6598	0.154

Table 4: Test results for local map prediction for the three models tested on three different datasets. The three models are tested on ModelNet10, ModelNet40, and the classes from ModelNet40 that are not in ModelNet10 having 30 classes. The results for the average loss and the peak selection performance for the three thresholds are also reported.

6.2.3 Next-best-view selection

As described in Section 5.3 we tested our two proposed methods and three baselines on ModelNet10 and ModelNet40. Additionally, we have set five different thresholds for the confidence score. The classification performance for all the tests done on ModelNet10 is shown in Figure 15. The ModelNet10 dataset consist of 4905 test object and the ModelNet40 dataset consist of 12311 test objects. The first thing we notice in the plot is that the performance generally goes up when the threshold

goes up. As expected, because more views will be explored when the threshold is higher. The next thing to note is that both the local-depth-entropy map prediction and local confidence map prediction outperform the three baselines when tested on ModelNet10. Additionally, the confidence method outperforms the depth-entropy method. The table in Appendix A.9 shows the accuracy per class for the five approaches tested on ModelNet40. One of the most important things to note from this table is that our approaches perform significantly better on most classes that are harder to predict overall. Which are table, desk, dresser, and night stand, as shown in the confusion matrix for ResNet34 tested on ModelNet10 (Figure 12). Night stand is the exception because it performs better with the further selection approach. The bathtub class performs significantly better on the random approach than our depth-entropy and confidence methods.

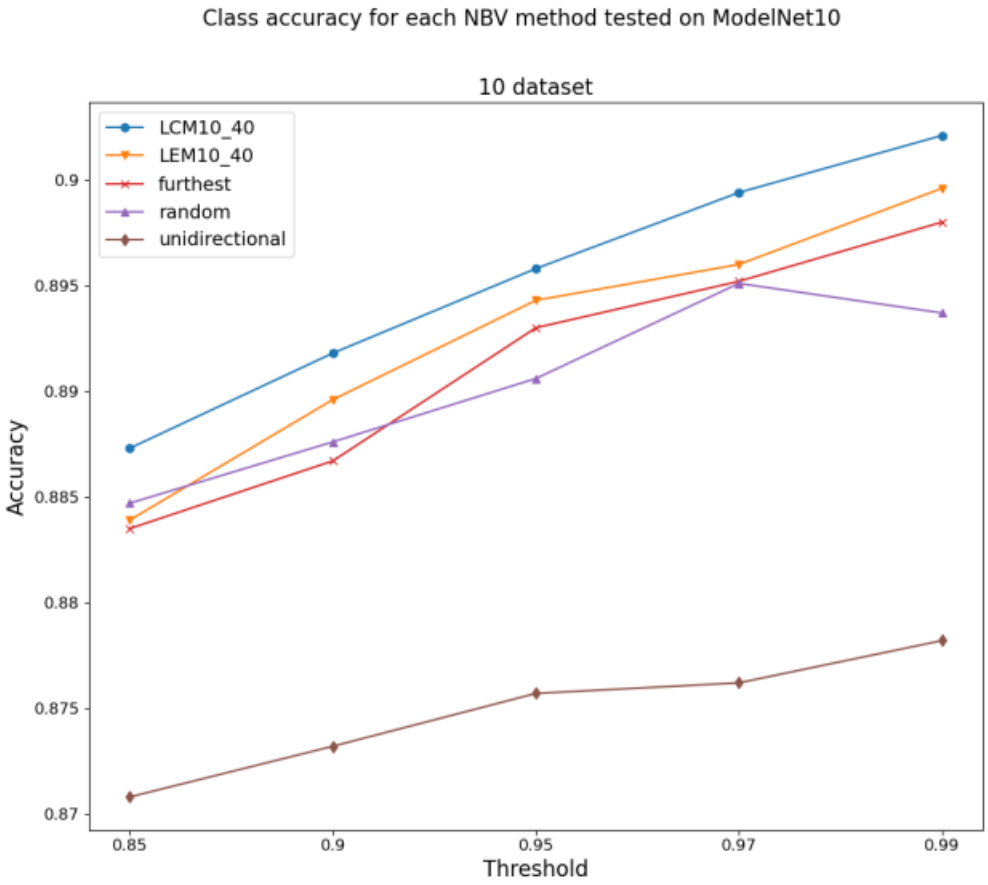

Figure 15: Class accuracy for each NBV method per threshold tested on ModelNet10. The test set consists of 4905 objects.

Table 5 shows the average classification performance and travel distance for all the approaches tested on ModelNet10. The results are shown for the respective best threshold, which is 0.99, except for the random approach. As shown in Figure 15, our method outperforms the baseline in term

Label	LCM10_40	LEM10_40	furthest	random	unidirectional
Threshold	0.99	0.99	0.99	0.97	0.99
Avg class acc	**0.9021**	0.8996	0.898	0.8951	0.8782
std	0.0865	0.0878	0.0935	0.0932	0.1084
Avg instance acc	**0.9067**	0.9056	0.9024	0.8988	0.8829
Avg class travel	**3.4095**	3.6659	7.4623	4.2816	1.8169
std	0.5421	**0.4357**	1.5967	0.8063	0.356
Avg instance travel	**3.7414**	3.7889	8.5512	4.8447	2.0848
std	2.3576	2.3662	5.427	3.2717	1.2145

Table 5: Classification performance and travel distance for five NBV approaches tested on ModelNet10.

of a classification performance. In the table we can also see the travel distance is lower for our methods compared to the baselines, where we disregard the unidirectional method, because it has by definition the lowest travel time and because its performance is so much worse compared to the other methods. The travel distance per class for the model tested on ModelNet10 is shown in Appendix A.10. On average and for all the classes, the travel distance is lower for our methods compared to the baselines.

Figures 16 and 17 show an example trajectory for both the depth-entropy and confidence approach. The confidence follows a uni-directional trajectory but the other way around than the baseline. The depth-entropy approach has a different trajectory but also follows a uni-directional trajectory at some point. Appendix A.12 show the trajectories resulting from the baseline approaches. All five examples predict the class correctly but with a variant confidence score.

In Figure 18 the results for all the methods tested on the ModelNet40 dataset are shown. Our methods outperform the random and unidirectional baselines, but the furthest baseline performs best and outperforms our methods.

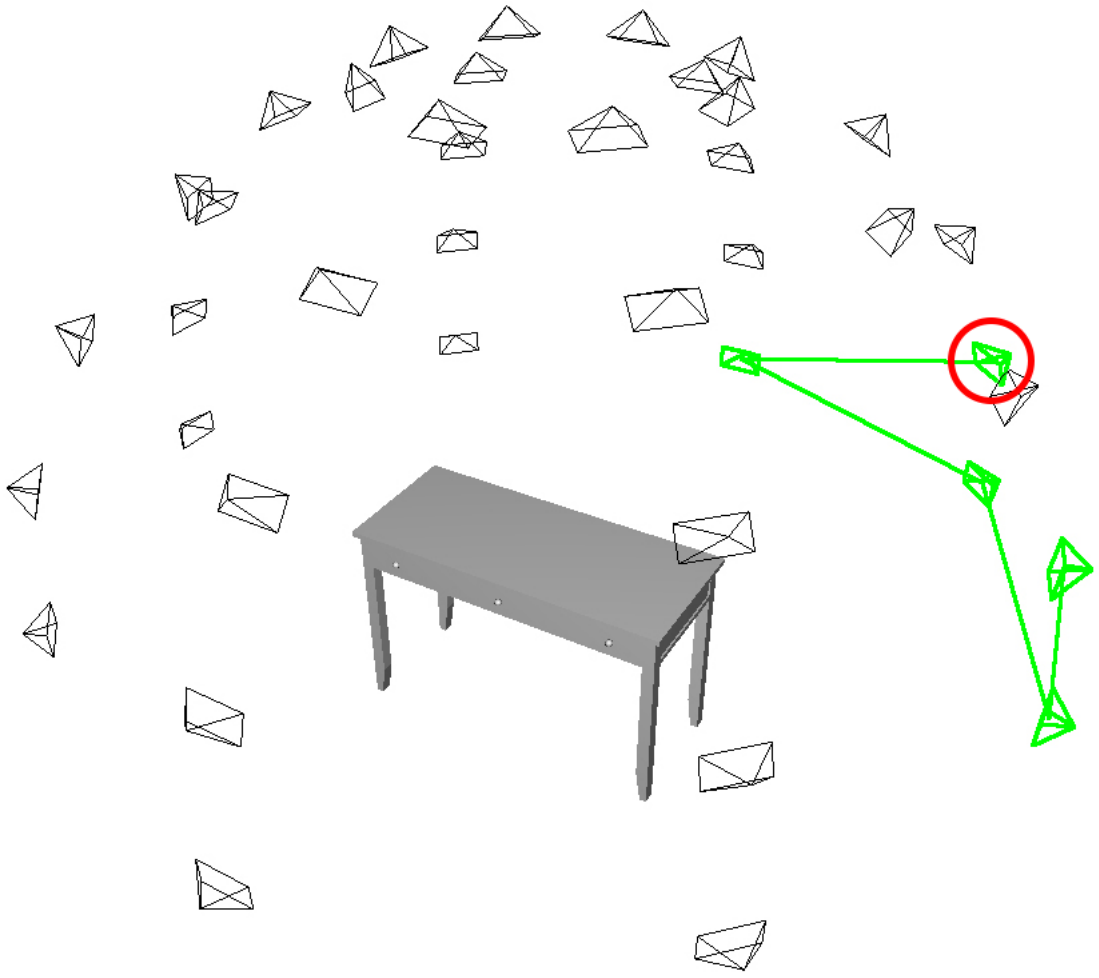

Figure 16: Viewpoint trajectory for using local-depth-entropy map NBV selection on table 0444. The trajectory goes through viewpoints 13, 12, 11, 10, and 9. The object is correctly predicted with a confidence score of 0.9954. The starting point (viewpoint 13), is indicated by the red circle.

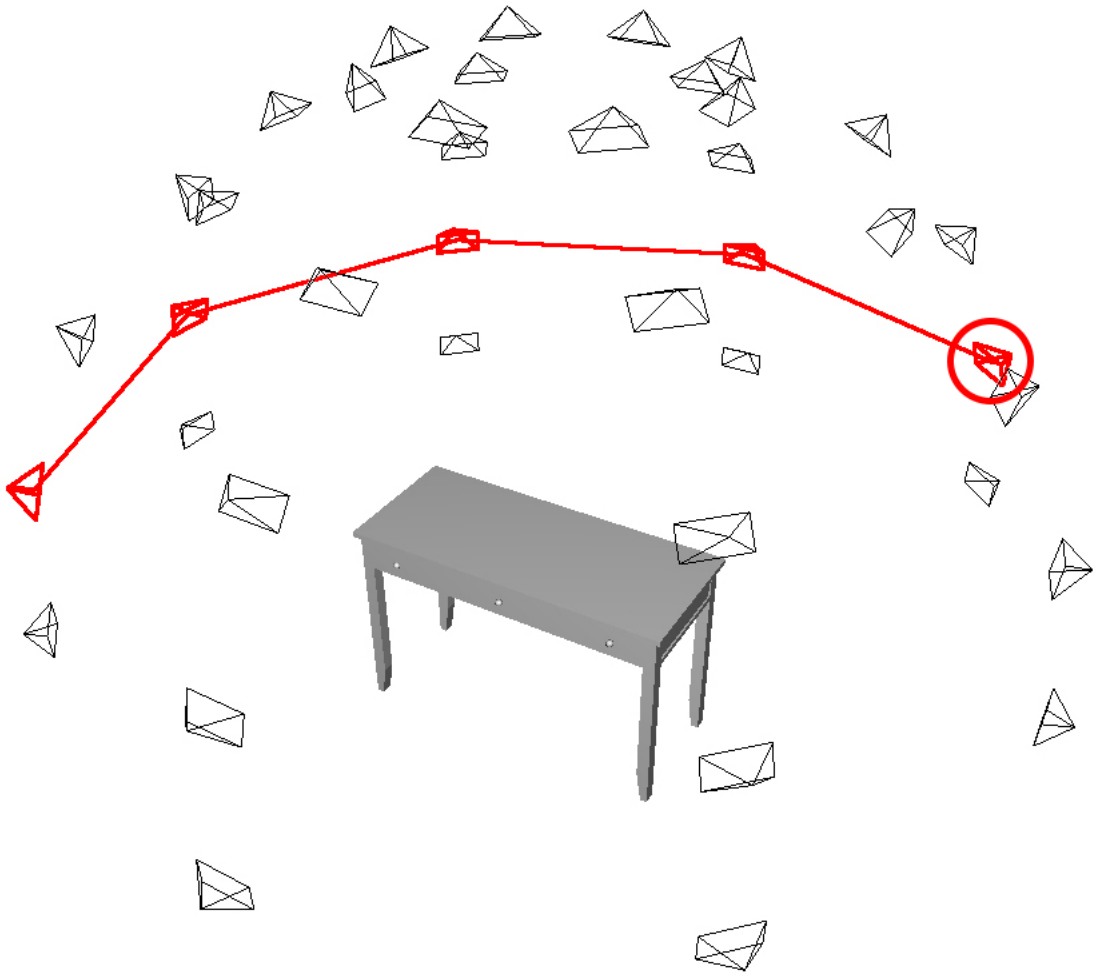

Figure 17: Viewpoint trajectory for using local confidence map NBV selection on table 0444. The trajectory goes through viewpoints 13, 4, 5, 6, and 14. The object is correctly predicted with a confidence score of 0.8464. The starting point (viewpoint 13), is indicated by the red circle.

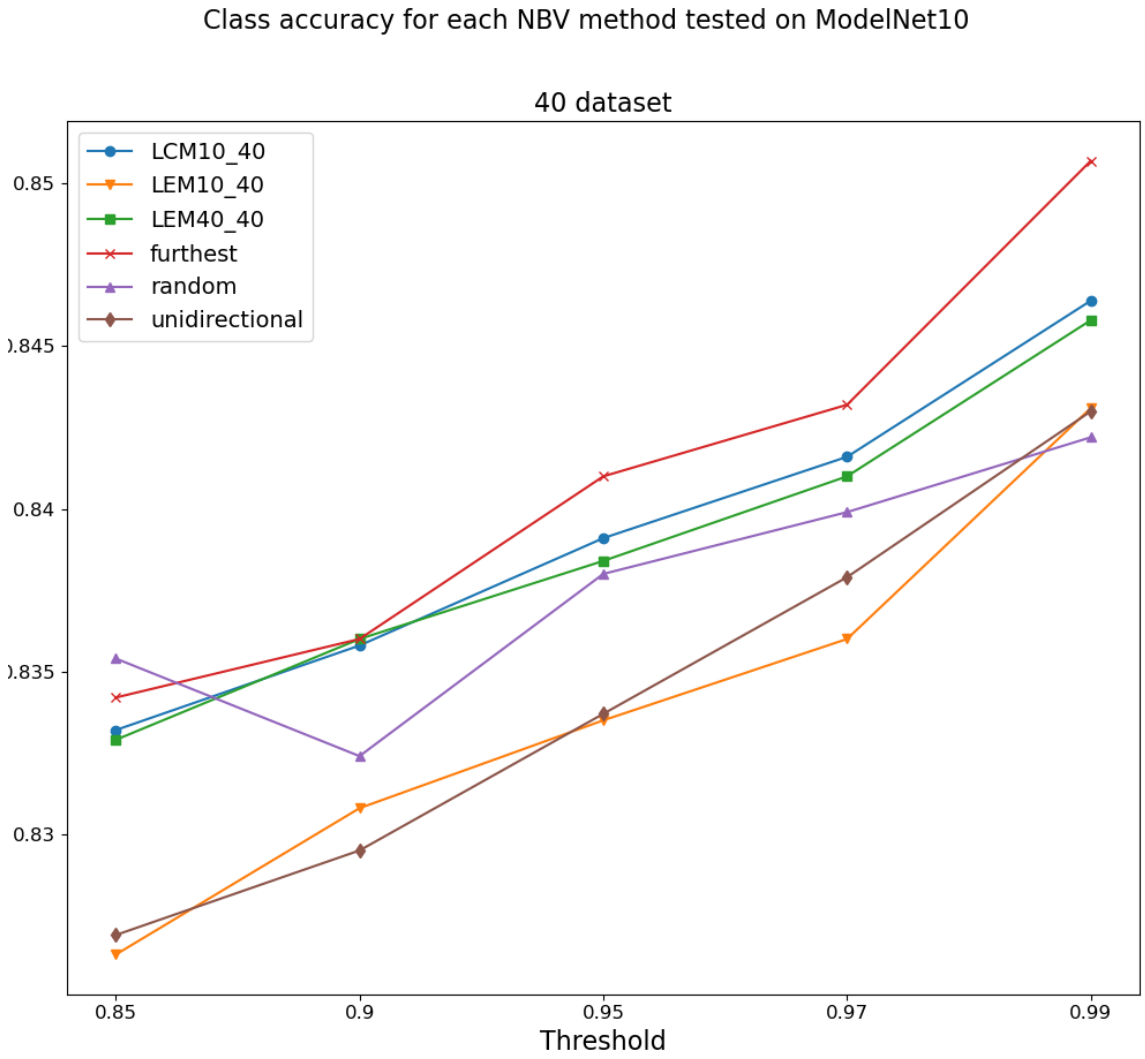

Figure 18: Class accuracy for each NBV method per threshold tested on ModelNet40. The test set consists of 12311 objects.

7 Conclusion

In this thesis, we tried to solve the next-best-view problem by training a system that can predict the goodness of neighboring views. The goodness of a view is determined by either the prediction of the depth-entropy or the prediction of the classification performance of the neighboring views. We developed a learning algorithm that can predict a map of these measures given a point cloud. We have proven that the prediction of the maps works reasonably well for the classes in ModelNet10. The model can also predict maps for the unseen classes in ModelNet40 with reasonable accuracy. This also indicates that a partial point cloud from a specific view holds enough information about its neighboring views to predict some of its properties, like depth-entropy.

Both our next-best-view methods outperform the baselines regarding classification accuracy and travel distance on the ModelNet10 dataset. We have seen that the classification model is very good at predicting some of the classes, with an accuracy between 0.97 and 1. Thus, the improvement on those classes is very limited or non-existent compared to the baselines. For the classes that are harder to predict, like table, dresser, and desk, our methods do significantly increase the classification performance compared to the baselines. Furthermore, the performance method outperforms the depth-entropy method by a small margin in classification accuracy and travel distance. Looking at the five worst performing classes (table, dresser, desk, night stand, and bathtub), we have to note that bathtub and night stand perform better on the baselines random and furthest, respectively. Thus for these models, the depth-entropy does not tell us anything about the information in a view regarding object recognition. One of the flaws of our current depth-entropy model is that flat surfaces viewed at an angle will produce a high depth-entropy, which in some cases does not represent a lot of information, as in the table class.

We tested or models trained on ModelNet10 on ModelNet40 and determined that our methods outperform the baselines in terms of travel distance when tested on this unseen data, but not in classification accuracy. We have to note that the ModelNet40 dataset is not very good since some objects are very easy to classify and some objects have only a few object instances. Thus it would be much more interesting to test models on a more evenly distributed dataset in terms of object instances and classification performance.

The literature research already gave us an indication that entropy is a good indicator of the goodness of a view. In this thesis, we also show that for some object classes the depth-entropy determines the goodness of a view in terms of object classification. It does differ between classes whether high or low depth-entropy is a good indicator of the goodness of a view.

8 Discussion and Future Work

One of our main goals was to develop a next-best-view selection model for object classification. We believe that the model we developed has the potential to be used in real-world applications when optimized. We do have to note that a more elaborate statistical approach in training and testing is needed to have a stronger belief in the current model. Such as multiple training attempts of the model with random weight initiation. Nonetheless, the current results do give enough belief to further investigate the model. The model can be optimized by training the learning algorithm on a different, potentially more challenging dataset concerning classification. The ModelNet10 dataset is in hindsight, not the best because about half the classes already have an excellent performance on classification using any view. Thus, there is almost nothing to improve.

We believe that the depth-entropy prediction model for neighboring views can be used on unseen data. Therefore, more tests on unseen and different data would be interesting. One of the most exciting things to see in future work would be the performance of the depth-entropy model in a real-life situation. Like using data directly from a depth camera to choose the next-best-viewpoint and physically move to that next viewpoint. Additionally, it would be interesting to see it being used in an active vision approach, meaning the distribution of viewpoints around the object is not predefined. But, the sensor is allowed to move to any point around the object. Another good test for generalization of the depth-entropy model would be to test it on a different task, like grasp synthesis or object reconstruction.

In our next-best-view prediction model, we used either high or low depth-entropy values for choosing the best view. It would be interesting to use a selection method with one definition for the next-best-view, so a higher value would always hold more information. A suggestion would be to take the direction of the gradient of the depth values into account so that we can detect flat surfaces such as a tabletop.

Lastly, since the performance of the confidence approach is better than the depth-entropy approach, it would be interesting to test if the confidence approach also works on a different dataset and on a different classification model without retraining. If it does work, it means the confidence approach learns a general measure for the goodness of a view that is not defined by a mathematical calculation such as depth-entropy.

References

[1] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv:1612.00593 [cs]*, April 2017. arXiv: 1612.00593.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. Technical Report arXiv:1412.6980, arXiv, January 2017. arXiv:1412.6980 [cs] type: article.

[3] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Technical Report arXiv:1502.03167, arXiv, March 2015. arXiv:1502.03167 [cs] type: article.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

[5] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015. arXiv: 1409.1556.

[6] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints. *arXiv:1603.06208 [cs]*, March 2018. arXiv: 1603.06208.

[7] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. *arXiv:1406.5670 [cs]*, April 2015. arXiv: 1406.5670.

[8] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *arXiv:1505.00880 [cs]*, September 2015. arXiv: 1505.00880.

[9] Charles R. Qi, Hao Su, Matthias Niessner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and Multi-View CNNs for Object Classification on 3D Data. *arXiv:1604.03265 [cs]*, April 2016. arXiv: 1604.03265.

[10] Jong-Chyi Su, Matheus Gadelha, Rui Wang, and Subhransu Maji. A Deeper Look at 3D Shape Classifiers. *arXiv:1809.02560 [cs]*, September 2018. arXiv: 1809.02560.

[11] Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. DeepPano: Deep Panoramic Representation for 3-D Shape Recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343, December 2015. Conference Name: IEEE Signal Processing Letters.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. ISSN: 1063-6919.

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. June 2017.

[15] Syeda Mariam Ahmed, Pan Liang, and Chee Meng Chew. EPN: Edge-Aware PointNet for Object Recognition from Multi-View 2.5D Point Clouds. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3445–3450, November 2019. ISSN: 2153-0866.

[16] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, Hamburg, Germany, September 2015. IEEE.

[17] Yangyan Li, Soeren Pirk, Hao Su, Charles R. Qi, and Leonidas J. Guibas. FPNN: Field Probing Neural Networks for 3D Data. *arXiv:1605.06240 [cs]*, October 2016. arXiv: 1605.06240.

[18] Helin Dutagaci, Chun Pan Cheung, and Afzal Godil. A benchmark for best view selection of 3D objects. In *Proceedings of the ACM workshop on 3D object retrieval - 3DOR '10*, page 45, Firenze, Italy, 2010. ACM Press.

[19] Oleg Polonsky, Giuseppe Patané, Silvia Biasotti, Craig Gotsman, and Michela Spagnuolo. What's in an image?: Towards the computation of the "best" view of an object. *The Visual Computer*, 21(8-10):840–847, September 2005.

[20] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM Transactions on Graphics*, 30(5):1–12, October 2011.

[21] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh Saliency. July 2005.

[22] George Leifman, Elizabeth Shtrom, and Ayellet Tal. Surface Regions of Interest for Viewpoint Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2544–2556, December 2016. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[23] Tommaso Parisotto and Hamidreza Kasaei. MORE: Simultaneous Multi-View 3D Object Recognition and Pose Estimation. *arXiv:2103.09863 [cs]*, March 2021. arXiv: 2103.09863.

[24] Bo Ding, Lei Tang, Zheng Gao, and Yongjun He. 3D Shape Classification Using a Single View. *IEEE Access*, 8:200812–200822, 2020. Conference Name: IEEE Access.

[25] Panpan Shui, Pengyu Wang, Fenggen Yu, Bingyang Hu, Yuan Gan, Kun Liu, and Yan Zhang. 3D Shape Segmentation Based on Viewpoint Entropy and Projective Fully Convolutional Networks Fusing Multi-view Features. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1056–1061, August 2018. ISSN: 1051-4651.

[26] Miguel Mendoza, J. Irving Vasquez-Gomez, Hind Taud, Luis Enrique Sucar, and Carolina Reta. Supervised Learning of the Next-Best-View for 3D Object Reconstruction. *Pattern Recognition Letters*, 133:224–231, May 2020. arXiv:1905.05833 [cs].

[27] Christopher McGreavy, Lars Kunze, and Nick Hawes. Next Best View Planning for Object Recognition in Mobile Robotics. page 9.

[28] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. Technical Report arXiv:1801.09847, arXiv, January 2018. arXiv:1801.09847 [cs] type: article.

A Appendix

A.1 Modelnet10 class distribution

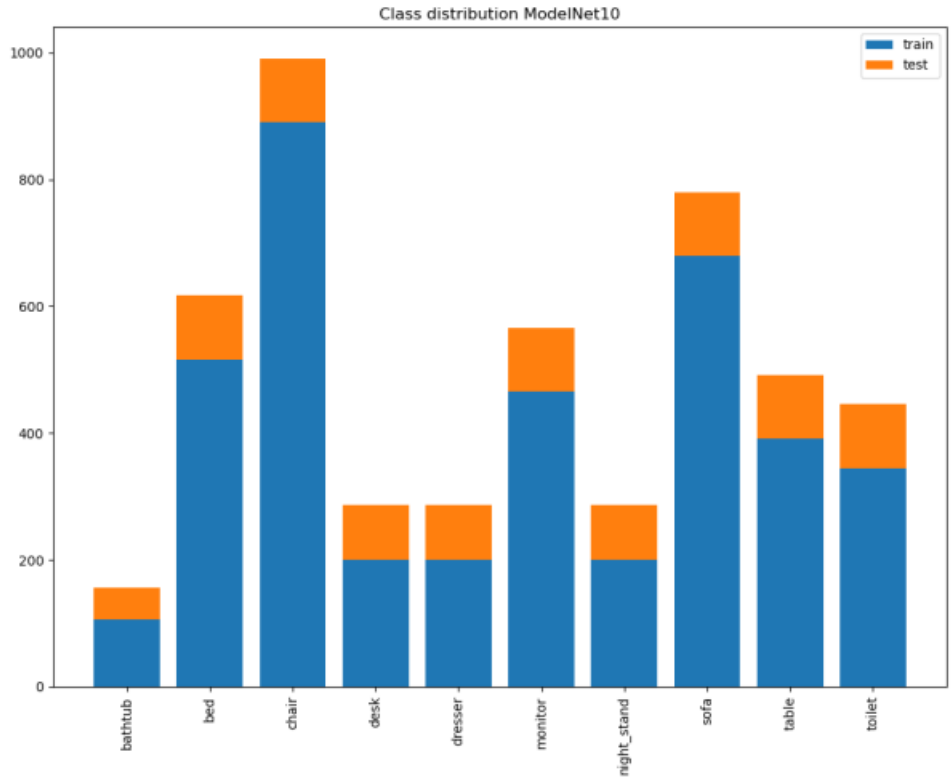

Figure 19: The number of objects for each class in the ModelNet10 dataset

A.2 Loss curves for the depth-entropy approach on ModelNet10

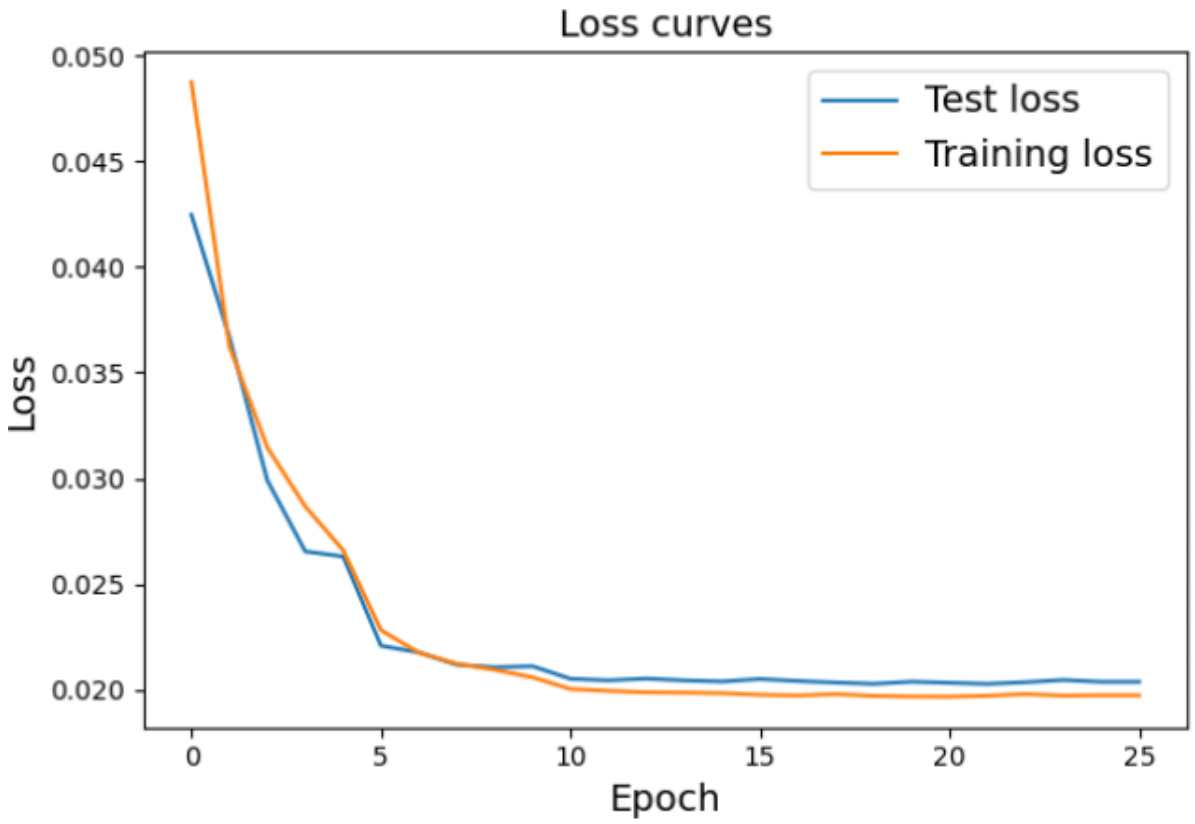

Figure 20: Loss curves for LEM PointNet trained on ModelNet10

A.3 ResNet training results

model	classes	viewpoints	batch size	accuracy
ResNet18	10	12	32	**89.01**%
ResNet18	10	40	4	85.88%
ResNet34	10	40	4	**85.99**%
ResNet50	10	40	4	84.44%
ResNet101	10	40	4	85.23%
ResNet152	10	40	4	86.20%
ResNet34	10	40	4	85.88%
ResNet34	10	40	8	86.42%
ResNet34	10	40	16	86.62%
ResNet34	10	40	32	**87.10**%
ResNet34	10	40	64	87.04%
ResNet18	40	12	32	86.29%
ResNet34	40	12	32	**87.03**%
ResNet50	40	12	32	86.87%
ResNet101	40	12	32	86.61%
ResNet152	40	12	32	86.73%
ResNet34	40	40	32	83.45%

Table 6: Instance classification performance for several ResNet models

A.4 Spherical coordinate system viewpoint distribution

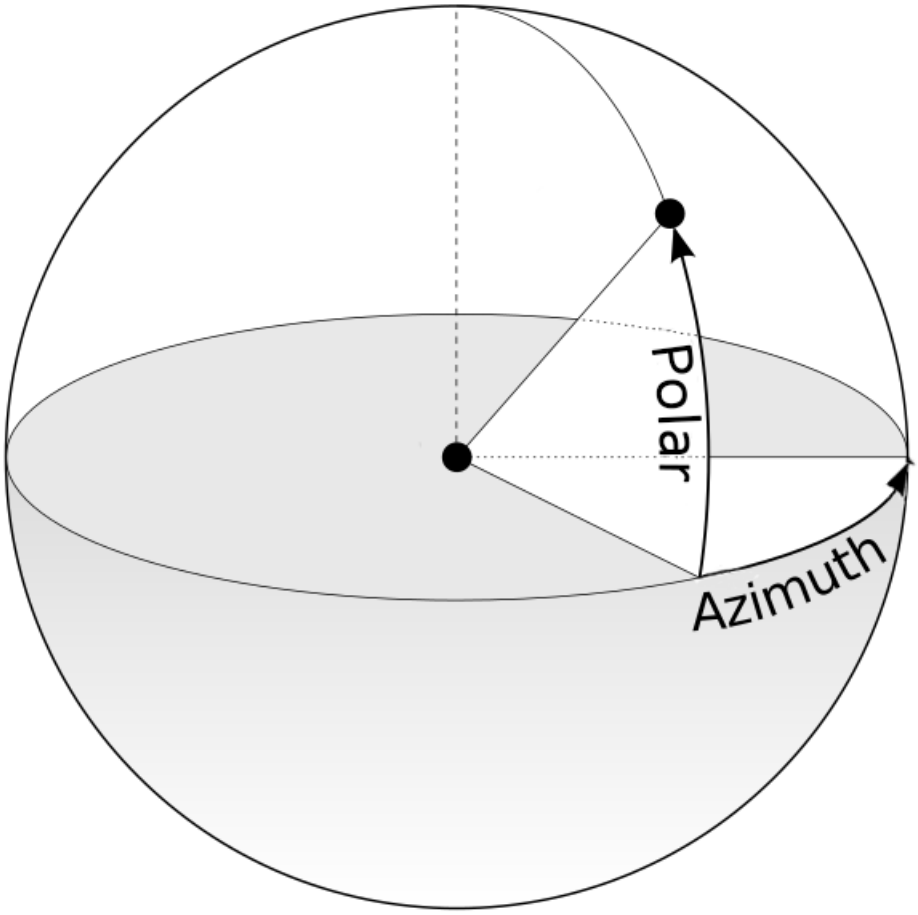

Figure 21: The spherical coordinate frame used to create the viewpoint distribution

A.5 Map indices

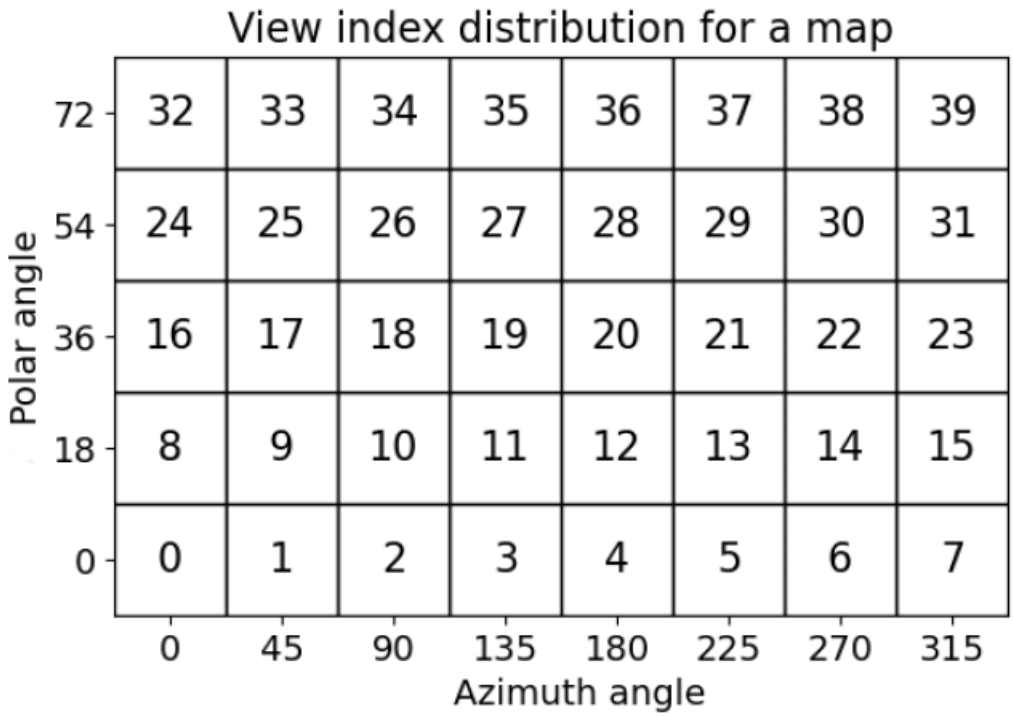

Figure 22: The index corresponding to every view in a map.

A.6 Confusion matrix for ModelNet40

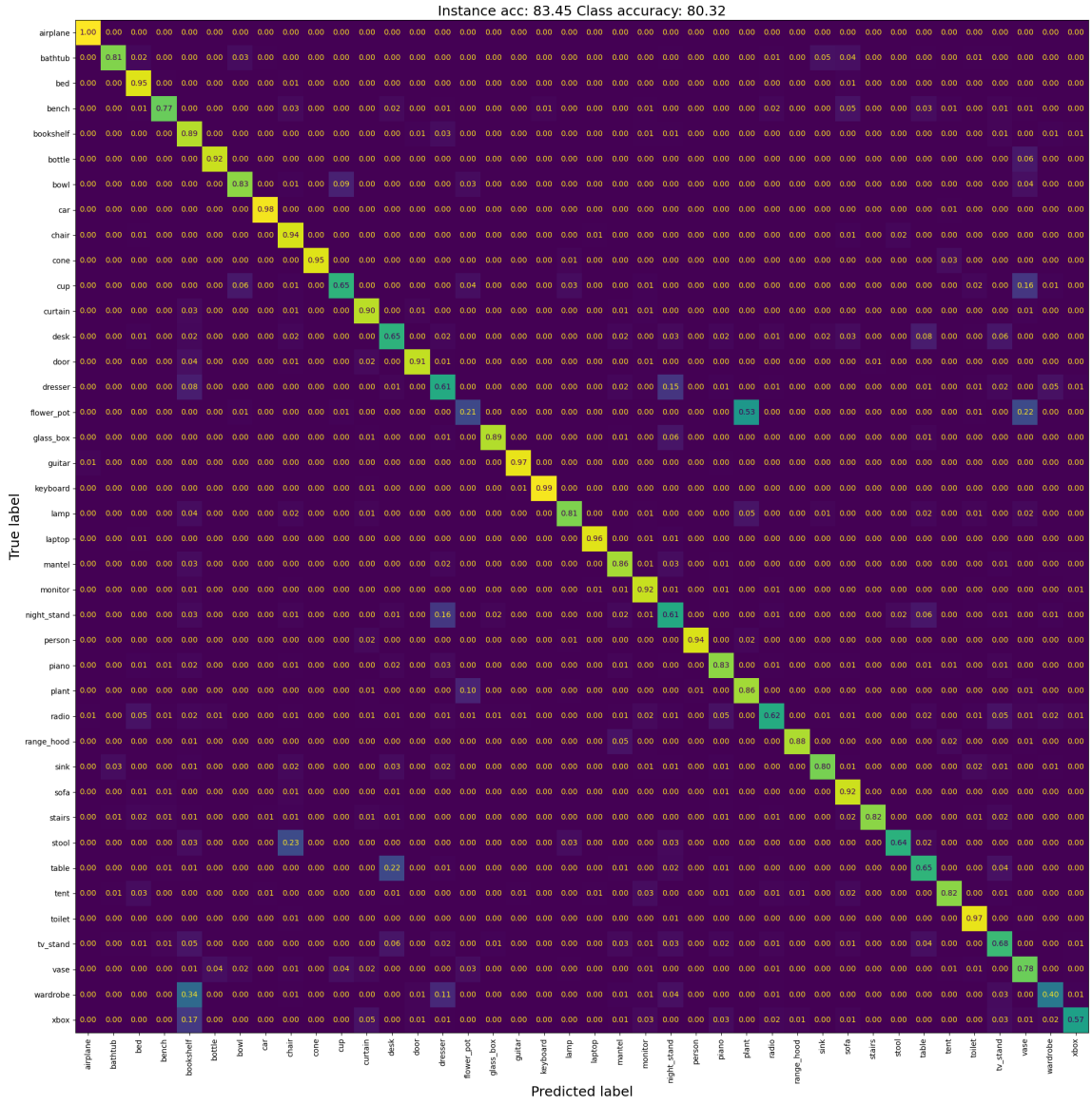

Figure 23: Confusion matrix for ResNet34 trained on ModelNet40 with 40 views

A.7 Confidence map table 0399

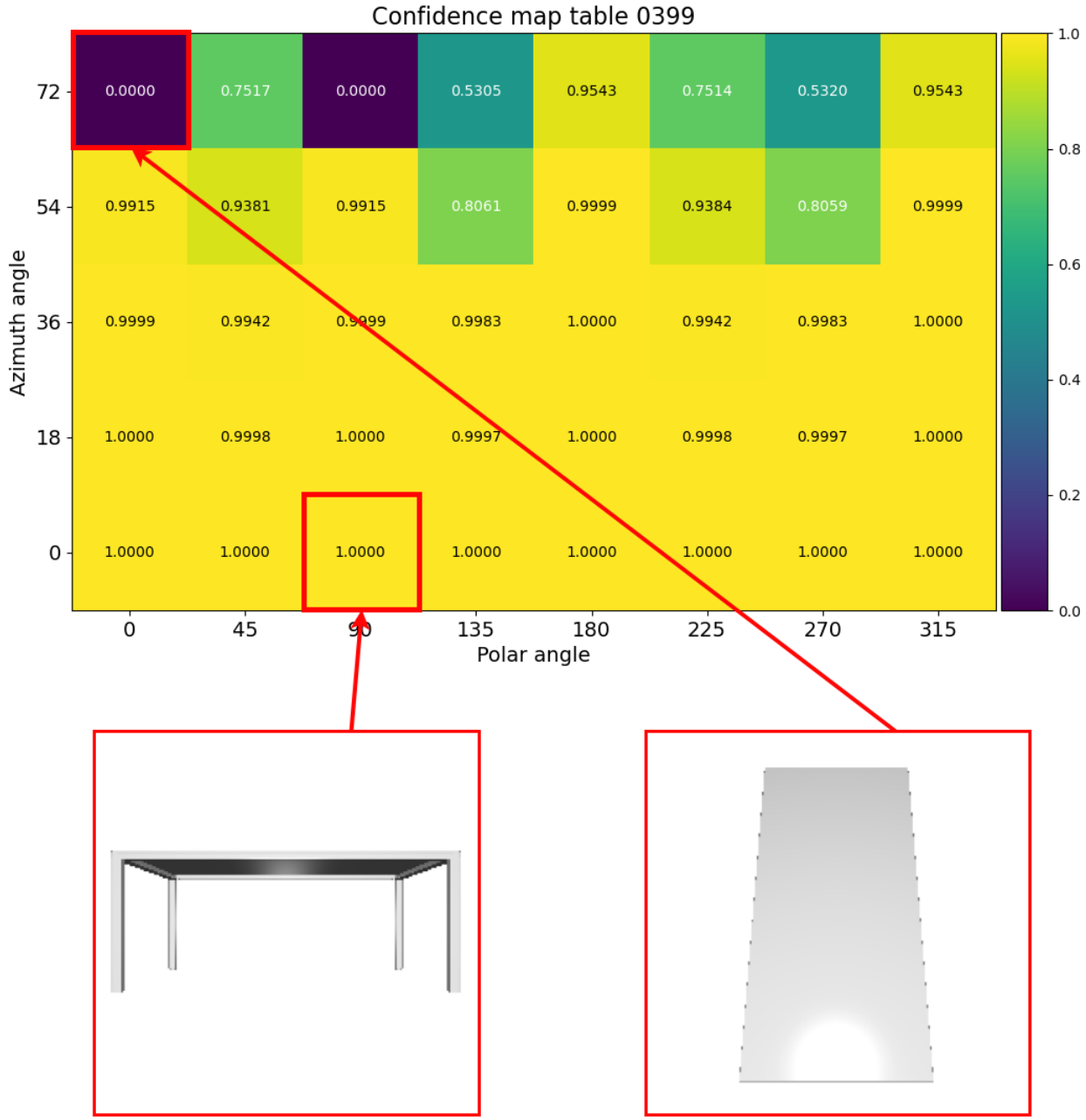

Figure 24: Confidence map of table 0399 with the images shown of the views with the lowest and highest confidence score in the map.

A.8 Map prediction performance per class for depth-entropy approach on ModelNet10

labels	$d = 0$	$d <= 1$	$d < 2$	avg_loss
bathtub	0.2560	0.7305	0.8080	0.0166
bed	0.3590	0.7618	0.8905	0.0109
chair	0.4400	0.7808	0.8430	0.0245
desk	0.2157	0.5959	0.6709	0.0319
dresser	**0.6273**	0.9410	0.9692	**0.0104**
monitor	0.5573	0.9323	**0.9830**	0.0468
night_stand	0.3919	0.7651	0.8238	0.0197
sofa	0.3105	0.6825	0.8127	0.0153
table	0.3222	**0.9655**	0.9712	0.0208
toilet	0.5080	0.8552	0.9050	0.0216

Table 7: Performance per class for each threshold for LEM10 tested on ModelNet10

A.9 Prediction accuracy per class and NBV method for ModelNet10

Label	LCM10	LEM10	furthest	random	unidirectional
bathtub	0.912	0.89	0.92	**0.922**	0.888
bed	**0.978**	0.976	0.984	0.975	0.969
chair	0.989	0.988	**0.99**	0.985	0.989
desk	**0.7779**	0.7674	0.764	0.7651	0.757
dresser	**0.8256**	0.8058	0.7663	0.793	0.7907
monitor	0.976	0.977	0.974	**0.982**	0.976
night_stand	0.7709	0.7698	**0.7977**	0.7884	0.7419
sofa	0.966	0.965	**0.968**	0.955	0.96
table	0.832	**0.864**	0.822	0.788	0.716
toilet	0.994	0.993	0.994	**0.997**	0.994
Avg class acc	**0.9021**	0.8996	0.898	0.8951	0.8782
Class std	0.0865	0.0878	0.0935	0.0932	0.1084
Avg instance acc	**0.9067**	0.9056	0.9024	0.8988	0.8829

Table 8: NBV accuracy per class with best confidence threshold for each method tested on ModelNet10.

A.10 Travel distance per class and NBV method for ModelNet10

Label	LCM10	LEM10	furthest	random	unidirectional
bathtub	3.6434	**3.3333**	7.1318	3.9853	1.9767
	(1.9235)	**(1.8215)**	(4.4518)	(2.5618)	(1.1889)
bed	3.2881	**3.1017**	6.4294	3.878	1.565
	(1.6207)	**(1.9831)**	(3.8327)	(2.7621)	(0.9871)
chair	**2.4306**	3.4444	4.7917	2.6377	1.1806
	(1.0724)	(1.7753)	(3.1217)	(1.894)	(0.5393)
desk	**3.2279**	3.5318	7.7906	4.5301	1.8727
	(1.8413)	(2.1666)	(4.8607)	(2.8978)	(1.0963)
dresser	4.259	**3.8648**	10.1059	5.4762	2.2443
	(2.7736)	**(2.6441)**	(6.0374)	(3.6025)	(1.1991)
monitor	**3.1864**	4.661	8.4407	4.7667	1.6102
	(2.232)	(3.6037)	(5.3988)	(3.1319)	(1.0003)
night_stand	4.5689	**4.3502**	10.0475	5.4505	2.3281
	(3.0796)	**(2.7167)**	(5.561)	(3.5442)	(1.2719)
sofa	**3.1846**	3.2385	5.6308	3.4094	1.6385
	(1.2927)	(1.6973)	(3.1107)	(2.4924)	(1.0268)
table	**3.335**	3.7512	8.3284	4.9325	2.3416
	(1.5317)	(1.8403)	(5.3652)	(3.1251)	(1.2796)
toilet	**2.9706**	3.3824	5.9265	3.7500	1.4118
	(1.496)	(2.6147)	(4.2683)	(2.8251)	(0.8679)
Avg class travel	**3.4095**	3.6659	7.4623	4.2816	1.8169
	(0.5421)	(0.4357)	(1.5967)	(0.8063)	(0.356)
Avg instance travel	**3.7414**	3.7889	8.5512	4.8447	2.0848
	(2.3576)	(2.3662)	(5.427)	(3.2717)	(1.2145)

Table 9: NBV average travel distance per method tested with their respective best thresholds. The travel is calculated when there is traveled during a prediction run. It is the accumulated travel distance for traveling to multiple views with a maximum of five views.

A.11 Map prediction examples for ModelNet10

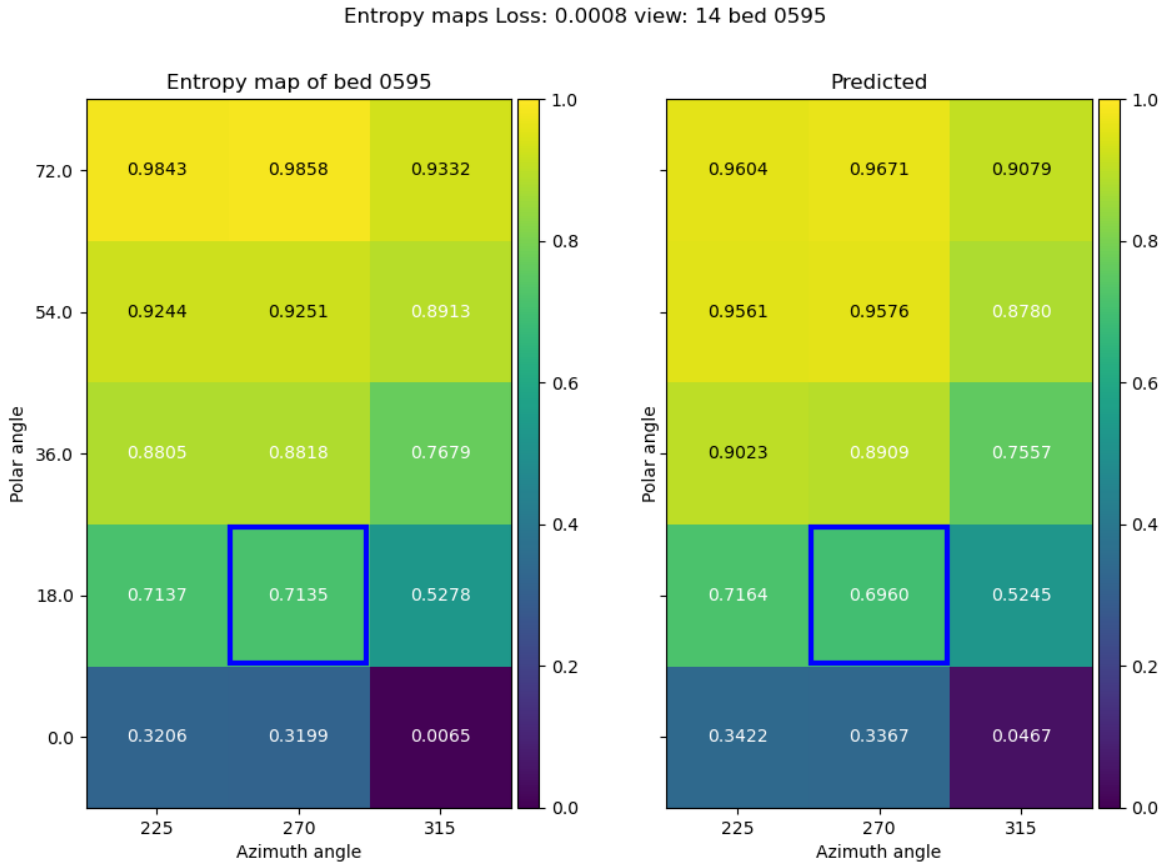

Figure 25: Ground truth and predicted depth-entropy map for bed 0595 in ModelNet10 with a low loss.

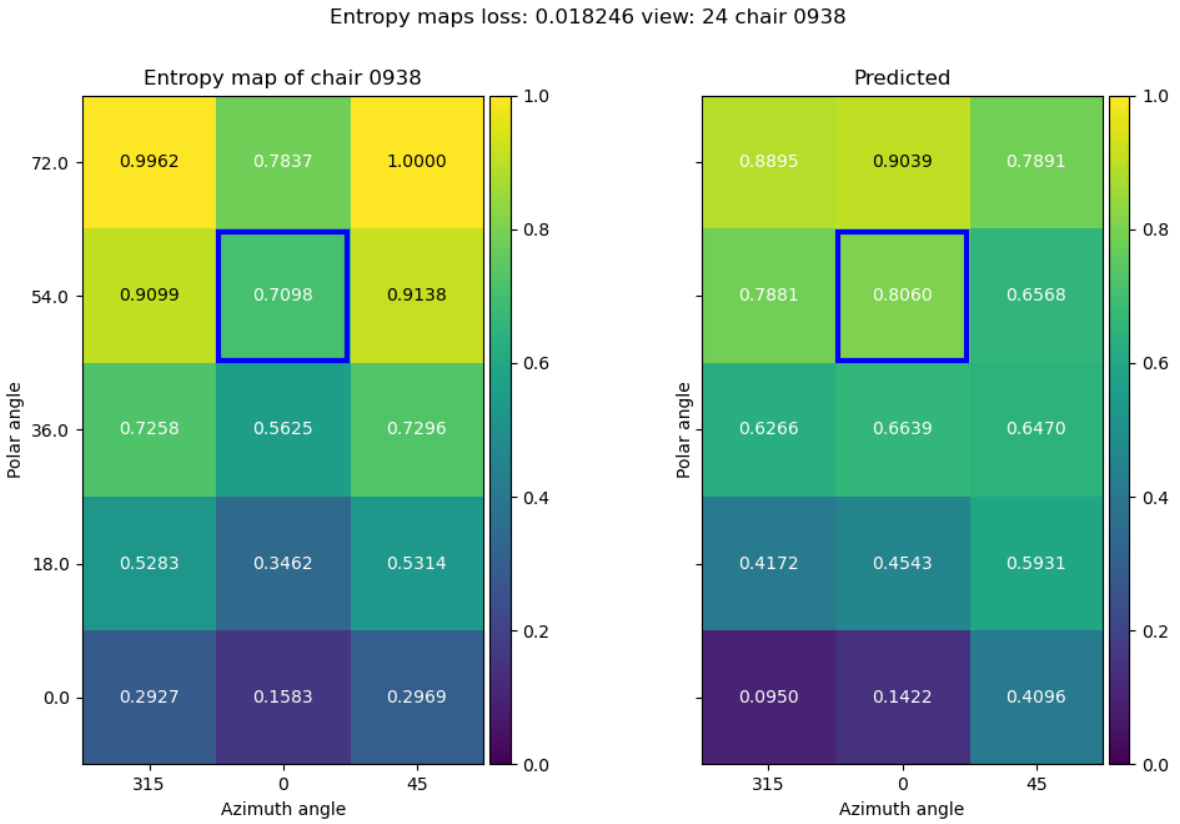

Figure 26: Ground truth and predicted depth-entropy map for chair 0938 in ModelNet10 with a medium loss.

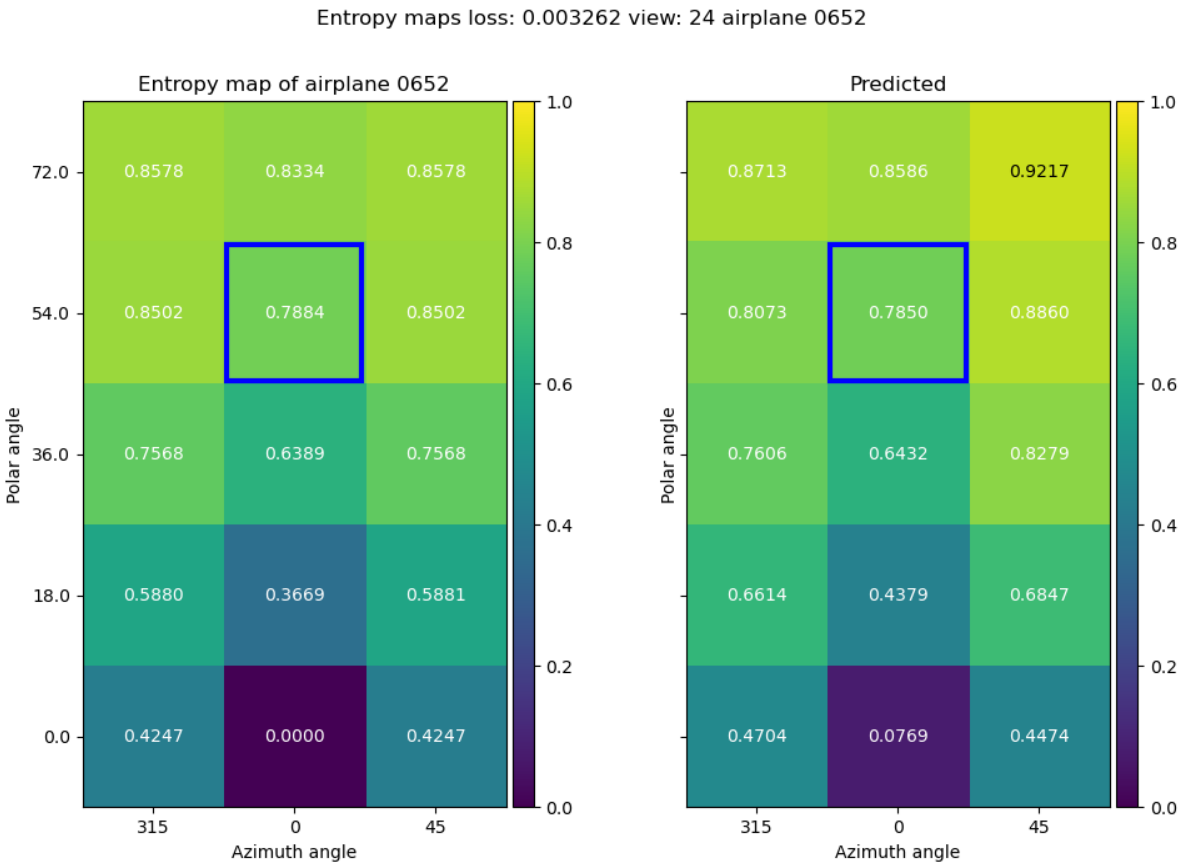

Figure 27: Ground truth and predicted depth-entropy map for airplane 0652 in ModelNet40 with a low loss.

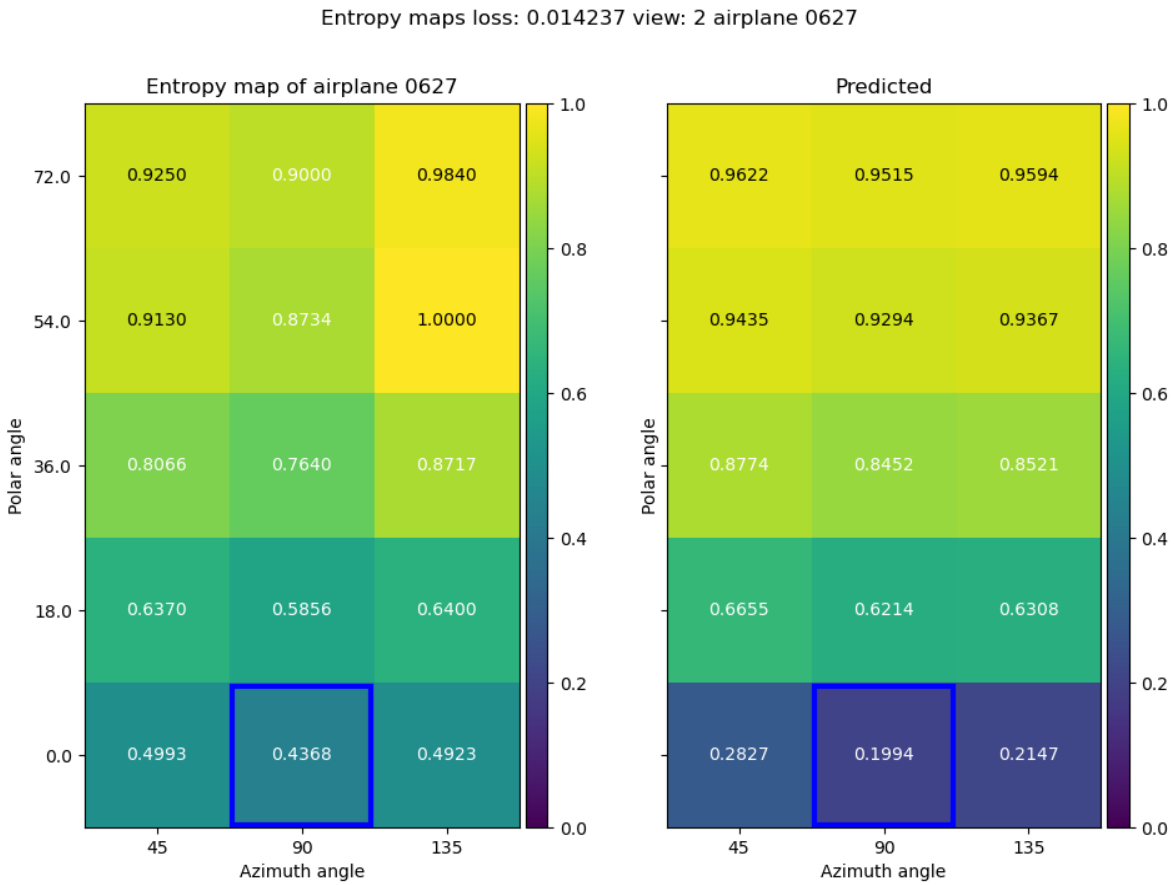

Figure 28: Ground truth and predicted depth-entropy map for airplane 0627 in ModelNet40 with a medium loss.

A.12 Examples of NBV baseline trajectories

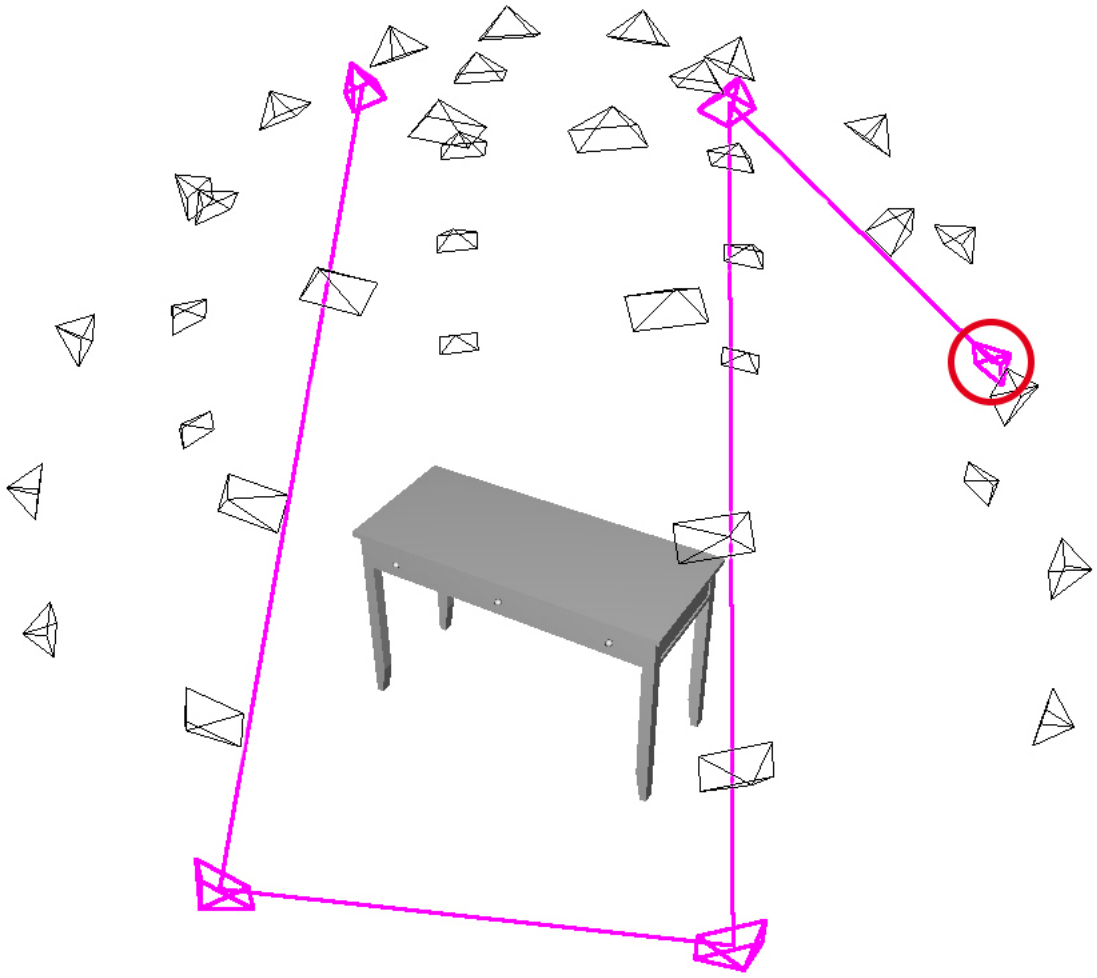

Figure 29: Viewpoint trajectory for using furthest NBV selection on table 0444. The trajectory goes through viewpoints 13, 38, 7, 0, and 33. The object is correctly predicted with a confidence score of 0.7817. The starting point (viewpoint 13), is indicated by the red circle.

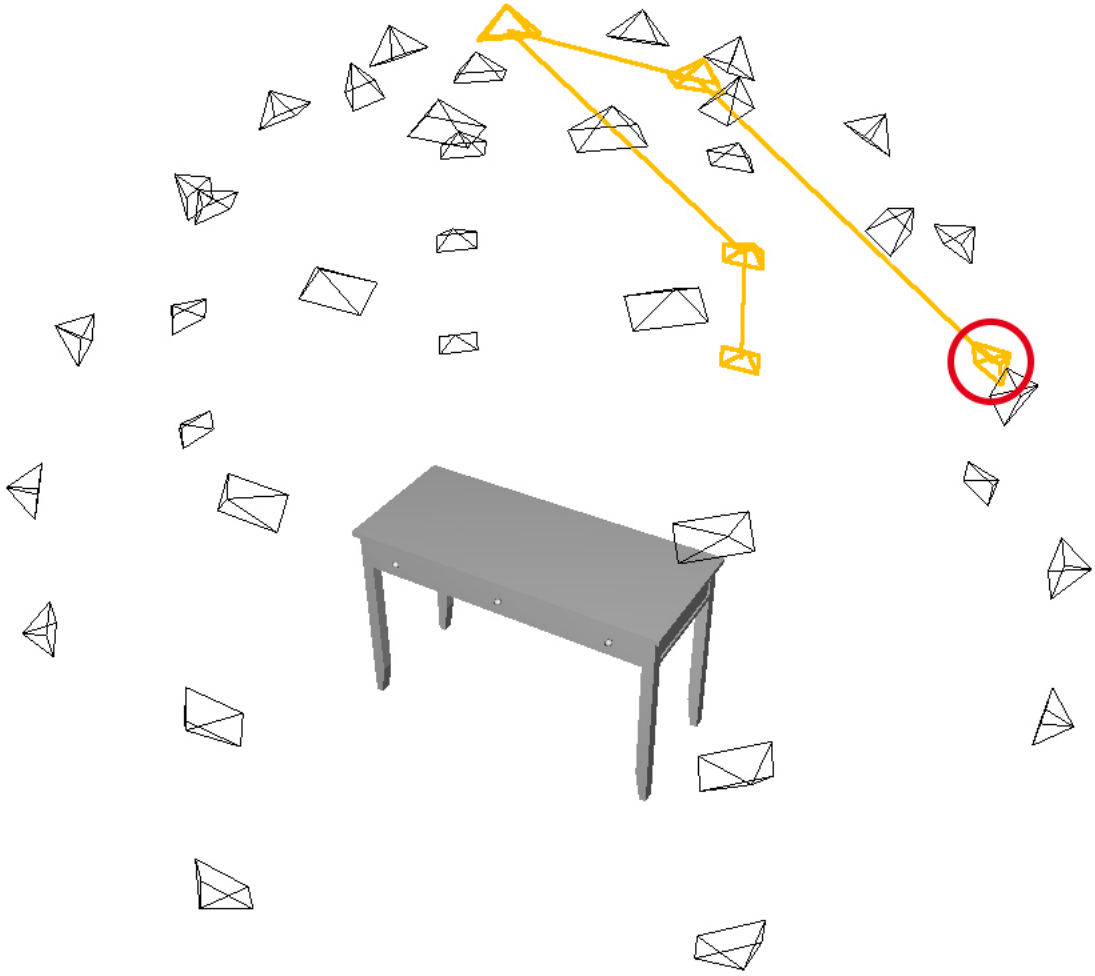

Figure 30: Viewpoint trajectory for using random NBV selection on table 0444. The trajectory goes through viewpoints 13, 28, 35, 12, and 4. The object is correctly predicted with a confidence score of 0.9719. The starting point (viewpoint 13), is indicated by the red circle.

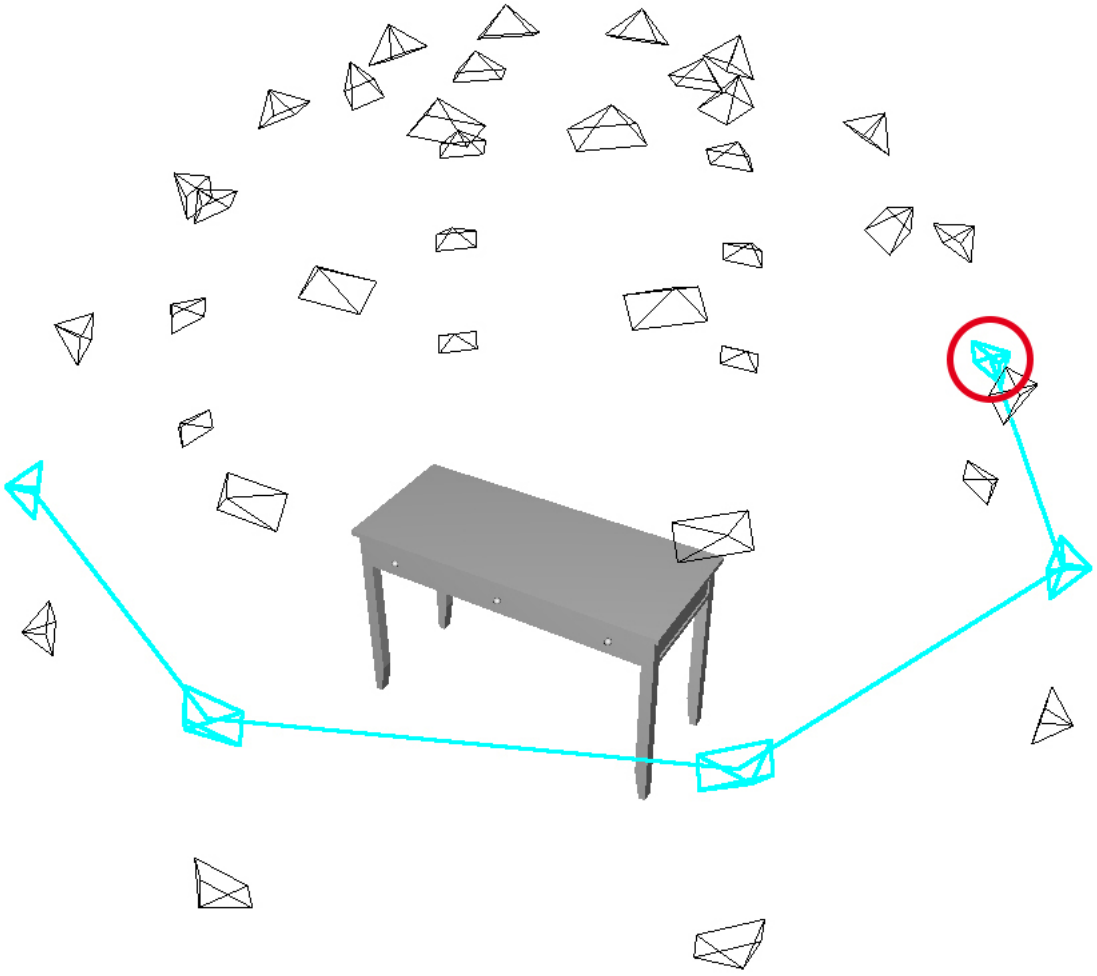

Figure 31: Viewpoint trajectory for using uni-directional NBV selection on table 0444. The trajectory goes through viewpoints 13, 14, 15, 8, and 9. The object is correctly predicted with a confidence score of 0.8877. The starting point (viewpoint 13), is indicated by the red circle.

A.13 Prediction accuracy per class and NBV method for ModelNet40

Label	LCM10	LEM10	LEM40	furthest	random	unidirectional
airplane	1.0	1.0	1.0	1.0	1.0	1.0
bathtub	0.872	0.876	0.876	0.876	0.86	0.84
bed	0.974	0.972	0.976	0.974	**0.984**	0.968
bench	0.83	**0.84**	0.83	0.81	0.8	0.81
bookshelf	0.9	0.896	0.894	**0.908**	0.904	**0.908**
bottle	0.932	**0.942**	0.94	0.936	0.928	0.936
bowl	0.85	0.8	0.8	**0.86**	0.81	0.83
car	**0.99**	0.994	0.988	0.992	**0.99**	0.994
chair	0.956	0.954	**0.96**	**0.96**	0.944	0.958
cone	0.95	0.94	0.94	0.95	0.94	0.95
cup	0.71	0.67	0.69	0.69	**0.75**	0.71
curtain	0.95	0.94	0.94	0.95	0.93	0.95
desk	**0.7558**	0.7326	0.7419	0.7465	0.7419	0.7326
door	0.92	0.93	0.93	0.91	**0.96**	0.91
dresser	0.7186	0.7209	0.7326	0.7279	0.7186	**0.7419**
flower_pot	0.18	0.18	0.2	0.23	0.17	**0.24**
glass_box	**0.932**	0.924	0.918	0.92	0.918	0.93
guitar	0.976	0.978	0.978	0.974	0.968	0.978
keyboard	1.0	1.0	1.0	1.0	1.0	1.0
lamp	0.82	0.84	0.83	0.85	**0.86**	0.81
laptop	1.0	1.0	1.0	1.0	1.0	1.0
mantel	0.908	0.916	0.91	0.914	**0.93**	0.91
monitor	0.932	0.946	0.946	0.944	0.936	0.946
night_stand	0.707	0.7163	**0.7326**	0.6953	0.7186	0.6977
person	0.98	1.0	1.0	1.0	0.98	0.99
piano	**0.94**	**0.94**	0.934	0.93	0.926	0.934
plant	0.89	0.892	0.89	0.896	**0.9**	0.88
radio	0.71	0.73	0.67	**0.77**	0.71	0.68
range_hood	0.894	0.884	0.896	0.892	**0.904**	0.894
sink	0.86	0.82	0.83	**0.89**	0.83	0.83
sofa	0.962	0.96	**0.964**	0.96	0.952	0.954
stairs	**0.86**	0.83	0.84	0.82	0.84	0.83
stool	0.73	0.67	**0.74**	0.73	0.68	0.67
table	0.688	0.762	**0.778**	0.744	0.714	0.73
tent	**0.91**	0.89	0.87	0.9	0.89	0.9
toilet	0.988	**0.99**	0.992	0.99	0.984	0.986
tv_stand	0.784	0.75	**0.786**	0.766	0.78	0.78
vase	0.798	0.8	0.78	**0.812**	0.806	0.79
wardrobe	0.46	0.45	0.45	0.45	0.42	**0.48**

xbox	0.64	0.65	**0.66**	**0.66**	0.61	0.64
Avg class acc + (std)	0.8464 (0.16)	0.8431 (0.1619)	0.8458 (0.1576)	**0.8507** (0.1542)	0.8422 (0.1641)	0.843 (0.1544)
Avg instance acc	0.8763	0.8768	**0.8797**	**0.8797**	0.8755	0.8756

Table 10: Accuracies per class for different NBV methods tested on the Modelnet40 dataset

A.14 Travel distance per class and NBV method for ModelNet40

Label	LCM10	LEM10	LEM40	furthest	random	unidirectional
airplane	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
bathtub	2.85 (1.6655)	**2.7667** (1.7209)	3.0167 (1.4786)	6.35 (3.6677)	3.0185 (1.9666)	1.5667 (0.9088)
bed	**2.36** (0.9424)	2.98 (1.4497)	2.58 (1.3716)	4.86 (2.0801)	3.3793 (1.9897)	1.26 (0.7231)
bench	3.4167 (2.3204)	3.5 (2.2842)	3.0417 (1.459)	6.0 (3.73)	**2.9615** (1.5616)	1.5417 (0.9771)
bookshelf	3.1277 (1.8621)	3.3085 (2.0947)	**2.3404** (1.1963)	6.6489 (4.4546)	3.5 (2.6576)	1.5426 (0.9)
bottle	**2.45** (1.1972)	2.95 (2.0248)	3.4 (2.2165)	5.7 (3.5388)	3.3125 (2.4155)	1.65 (0.9213)
bowl	**3.1667** (1.9262)	3.3333 (2.0359)	3.2917 (1.5737)	7.4167 (5.6331)	3.4231 (2.4195)	2.1667 (1.274)
car	2.5263 (1.0203)	2.9474 (1.6824)	**2.4211** (1.6437)	4.7895 (1.9316)	2.8462 (1.2142)	1.3158 (0.5824)
chair	**2.7561** (1.3375)	3.0732 (1.2921)	4.6098 (2.4482)	5.6585 (3.6511)	3.2826 (2.3727)	1.4634 (0.8688)
cone	3.0 (1.0)	3.6 (1.5166)	2.8 (1.3038)	11.0 (6.442)	**2.2** (1.0328)	2.0 (1.2247)
cup	**2.0** (1.1304)	4.3514 (3.1904)	3.7027 (2.2094)	5.1622 (2.4778)	3.5862 (2.1633)	1.7838 (1.2502)
curtain	3.2 (1.7809)	3.7333 (1.9445)	5.2 (3.7455)	6.8 (5.1297)	**2.8824** (2.571)	1.7333 (0.9612)
desk	3.3202 (2.0485)	3.5112 (2.2379)	**3.1404** (1.6866)	6.4494 (3.5985)	4.0276 (2.9391)	1.7753 (0.9538)
door	2.7143 (1.496)	3.1429 (2.1157)	2.4286 (1.1339)	6.1429 (3.6253)	**2.125** (0.991)	1.4286 (1.1339)
dresser	4.0169 (2.628)	4.3586 (2.5911)	**3.0338** (1.7415)	8.4684 (5.1193)	5.038 (3.1947)	2.0675 (1.1841)
flower_pot	3.9091 (1.9979)	**3.1136** (2.1697)	4.5227 (1.9107)	7.8636 (5.3812)	4.9286 (2.8915)	2.1364 (1.1732)
glass_box	3.9041 (2.3403)	3.8356 (1.9792)	**3.1644** (1.4241)	10.3288 (6.2161)	4.7711 (3.198)	2.1507 (1.2323)
guitar	2.875 (1.9621)	3.125 (1.9279)	**2.8125** (1.9738)	5.625 (2.9183)	3.2143 (1.6257)	1.125 (0.3416)
keyboard	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
lamp	3.0 (1.1767)	3.8571 (1.8337)	3.0714 (1.859)	7.0714 (4.0661)	**2.75** (1.4222)	2.0 (1.4142)

laptop	**2.5714** (1.2724)	3.1429 (1.069)	2.7143 (1.2536)	4.5714 (0.5345)	3.0 (1.5275)	1.0 (0.0)
mantel	**3.4327** (2.1305)	5.0962 (3.1113)	6.9231 (3.7822)	7.4135 (5.0041)	4.4623 (3.1658)	1.5962 (0.8979)
monitor	3.0612 (1.6634)	4.0408 (2.2542)	4.6327 (2.7439)	4.8163 (2.0683)	**3.0208** (1.4364)	1.551 (0.8912)
night_stand	4.0615 (2.6516)	4.1423 (2.4191)	**3.1923** (1.9634)	8.5577 (5.0862)	4.8893 (3.1592)	2.0769 (1.1019)
person	**2.5** (1.1785)	3.8 (2.3944)	3.0 (1.4142)	5.0 (1.8257)	3.0 (1.0)	1.2 (0.4216)
piano	**3.15** (1.684)	3.83 (2.2476)	4.82 (3.3375)	5.52 (3.4126)	3.6068 (2.3451)	1.43 (0.7688)
plant	**3.1705** (1.6915)	3.8217 (2.7541)	3.6047 (1.9781)	7.6279 (5.3137)	5.1322 (3.3984)	1.907 (1.1755)
radio	3.3171 (1.8363)	2.9512 (1.5804)	**2.7317** (1.9239)	6.6585 (3.985)	3.5517 (2.9831)	1.4634 (0.7777)
range_hood	3.6241 (2.2817)	4.8947 (3.255)	**3.4962** (2.2684)	8.2105 (5.7709)	4.4333 (2.9113)	1.9398 (1.1597)
sink	3.4348 (1.996)	**3.2609** (1.8145)	4.5652 (2.0411)	5.3478 (3.588)	3.9091 (2.1802)	1.7391 (1.0098)
sofa	2.8936 (1.5912)	**2.4255** (1.4408)	2.8511 (1.4139)	5.6809 (3.9678)	4.0 (2.9683)	1.5532 (0.8799)
stairs	3.5 (2.4251)	3.5 (1.7661)	**2.5455** (1.3707)	6.1818 (4.2721)	2.913 (1.4114)	1.5 (0.9129)
stool	**2.6429** (1.3113)	3.0357 (1.4268)	2.7857 (1.1661)	5.5 (2.396)	3.1944 (1.4307)	1.7857 (1.0666)
table	**3.765** (2.2505)	4.5385 (3.1228)	4.2137 (2.6101)	6.8248 (4.3346)	4.3205 (2.8261)	1.7949 (1.0444)
tent	3.4706 (1.8411)	3.2941 (1.6494)	4.4706 (3.3)	5.0588 (3.1518)	**2.6429** (1.7368)	1.3529 (0.6063)
toilet	**2.3077** (1.4358)	3.1538 (1.1897)	3.6538 (1.9171)	4.4615 (1.9438)	2.9583 (1.4885)	1.2692 (0.4523)
tv_stand	3.4227 (2.4078)	3.5591 (2.2332)	**2.9045** (1.5245)	7.5136 (5.1236)	4.1388 (2.8681)	1.7091 (1.01)
vase	**3.1727** (2.0222)	3.8818 (2.9138)	3.6727 (2.4536)	5.6818 (4.223)	3.8085 (2.6285)	1.8182 (1.1667)
wardrobe	3.9296 (2.093)	4.6901 (2.8665)	**3.2958** (1.9228)	8.5634 (5.3338)	4.6981 (3.2557)	1.8169 (1.0462)
xbox	4.1515 (2.4254)	3.4848 (1.9704)	**3.3636** (2.0739)	7.4242 (4.2722)	4.8788 (2.9449)	1.8485 (1.0932)
Avg class travel	**3.0043** (0.8436)	3.4008 (0.9554)	3.3004 (1.1545)	6.2237 (2.0081)	3.6451 (0.778)	1.5765 (0.4525)

Avg instance travel	**3.4353** (2.1688)	3.8789 (2.5267)	3.5439 (2.317)	7.109 (4.7216)	4.165 (2.8669)	1.7801 (1.054)

Table 11: Travel distance per class for with standard deviation different NBV methods tested on the Modelnet40 dataset

A.15 Preliminary viewpoint selection test on ModelNet40

	ascending	descending	Best	Random
airplane	1.0	1.0	1.0	1.0
bathtub	0.78	**0.9**	0.98	0.89
bed	0.98	**1.0**	1.0	1.0
bench	0.8	0.8	0.95	0.78
bookshelf	0.89	**0.94**	1.0	0.97
bottle	0.93	**0.94**	0.99	0.94
bowl	0.75	0.75	1.0	0.84
car	1.0	1.0	1.0	1.0
chair	**0.97**	0.94	1.0	0.96
cone	**0.95**	0.9	1.0	0.95
cup	**0.7**	0.5	0.85	0.73
curtain	**1.0**	0.9	1.0	0.97
desk	**0.67**	0.65	0.99	0.76
door	0.9	0.9	1.0	0.94
dresser	0.69	0.69	0.95	0.68
flower_pot	0.2	**0.25**	0.5	0.22
glass_box	0.88	**0.94**	0.98	0.94
guitar	**0.99**	0.96	1.0	0.98
keyboard	1.0	1.0	1.0	1.0
lamp	0.75	**0.85**	0.9	0.84
laptop	1.0	1.0	1.0	1.0
mantel	**0.95**	0.86	0.99	0.97
monitor	**0.95**	0.94	1.0	0.96
night_stand	0.59	**0.67**	0.99	0.68
person	0.85	**1.0**	1.0	1.0
piano	**0.94**	0.82	0.99	0.96
plant	0.8	**0.89**	0.96	0.89
radio	0.7	**0.75**	0.95	0.77
range_hood	0.88	**0.89**	0.98	0.91
sink	**0.85**	0.75	1.0	0.9
sofa	0.91	**0.97**	0.98	0.96
stairs	0.8	**0.95**	1.0	0.92
stool	0.6	**0.7**	0.85	0.7
table	**0.84**	0.42	1.0	0.8
tent	**0.9**	0.85	0.95	0.92
toilet	**0.99**	0.97	1.0	0.99
tv_stand	0.74	**0.82**	0.98	0.85
vase	**0.8**	0.79	0.91	0.79
wardrobe	0.25	**0.6**	0.9	0.64
xbox	0.55	**0.6**	0.85	0.6

Table 12: Accuracy per class for four selection methods using five views