Next Best View For Object Model Acquisition

Madalena Pombinho Caeiro Caldeira
Universidade de Lisboa - Instituto Superior Técnico
June 2024

***Abstract*—The Next Best View problem is a computer vision problem widely studied in robotics. Over the years, several deep learning models that successfully solve this problem have been proposed. Predictions obtained with the help of deep learning models naturally have some uncertainty associated with them. However, the standard models do not allow for the quantification of the uncertainty. Bayesian estimation theory contributed to demonstrate that Dropout layers can be used to estimate prediction uncertainty in neural networks. This thesis adapts the point-net based neural network architecture for Next-Best-View, PC-NBV, incorporating Dropout layers, thus allowing the computation of the uncertainty estimate associated with the predictions.**

The thesis aims to improve the network's accuracy in correctly predicting the next best viewpoint, proposing a way to make the 3D reconstruction process more efficient. Various accuracy uncertainty metrics capable of modeling prediction error and accuracy are studied. Two measures of uncertainty are obtained, these enable the reduction of the model's error and the increase in its accuracy from 30% to 80%, by identifying and disregarding predictions with high uncertainty. We also propose a method that directly uses these uncertainty metrics to improve the final prediction. However, this method presents very residual improvements.

***Index Terms*—Next Best View; Uncertainty Quantification; Robotics; Deep Learning Networks**

I. Introduction

DEVELOPMENTS in the fields of robotics and artificial intelligence have been allowing several improvements in other scientific and non-scientific areas, revolutionizing everyday tasks in the process. Areas like healthcare, agriculture and manufacturing undergone great evolution with the integration of intelligent robotics in their processes. This influence can also be visible in the domains of archaeology, conservation and restoration, where their strategic integration has become an invaluable asset, aiding men in tasks that require lots of precision and time investment. The work to be developed in this dissertation is inspired in a project [1] that focuses on the use of intelligent robotics in this area, with the goal of making the reconstruction process of Pompeii frescoes more efficient.

Some methodologies for reconstruction rely on computational help - it is possible to use computer algorithms that, just like solving a jigsaw puzzle, are able to perform the reconstructions. For this they require the 3D models of the pieces, which is the task category on which this dissertation falls onto - the study of an algorithm for the efficient 3D model acquisition of the fresco pieces.

The 3D data collection method must be as efficient as possible, which in this case means that the algorithm needs to find a way of getting the most information of the piece's surface with as few scans as possible. To achieve this goal the algorithm has to find the camera viewpoints (i.e. camera pose with respect to the object being scanned) that will get the most new information on the object being scanned. This is known as the Next Best View (NBV) problem, which the algorithm must adeptly solve to achieve its intended outcome.

There are several already existing algorithms that use artificial intelligence to solve the NBV problem. Some, such as [14], [15], [2] and [16], showed very good results in performing 3D model reconstructions in an efficient manner. However, they do not account for the uncertainty of their predictions, which in real world scenarios, can be vital to guarantee a good reconstruction performance, since a prediction with high uncertainty will most likely be a wrong prediction that can potentially jeopardize the reconstruction efficiency. Therefore, in this dissertation a NBV regression algorithm that regresses the views coverage scores - PC-NBV [2] - is analyzed in a more detailed way, modifications are made to its architecture to make it account for uncertainty. We aim to use the uncertainty of the estimated coverage score, using the idea of Monte Carlo sampling to estimate the uncertainty of the output of a Neural Network. To do this, the deterministic output becomes probabilistic through the application of Dropout layers. The goal is for the model to output a prediction along with an uncertainty measurement that reflects how sure the model is of its prediction, in an attempt of enhancing precision with the information given by this measurement, and subsequently improving the efficiency of the model acquisition process. We use two types of metrics that give us two different kinds of uncertainty measurements. One is error related, as it estimates the prediction uncertainty related to the closeness of the prediction to the groundtruth. Another is accuracy related, as it estimates the prediction uncertainty related to the ability of the network in choosing the right NBV.

II. State of the Art

A. The NBV Problem

The NBV problem is an important research topic in robotic 3D reconstruction. In its essence, it aims to find the optimal viewpoint for a camera (or sensor) for it to acquire data of an object in observation, in order to maximize the information gained and enhance its overall understanding. This is useful for object 3D model reconstruction since it enables to build the object's 3D model using a minimal number of camera scans. The common methods for the resolution of this problem can be broadly divided into two general groups: synthesis methods and generate-and-test methods.

Synthesis methods directly calculate the pose of the next best view by means of combining several pieces of heuristic

information such as visibility, information gain, uncertainty reduction and others. This type of methods are mainly designed for a very specific task with specific constraints, as they form a solution using logical reasoning and domain-specific knowledge rather than exhaustive exploration. As such, although having low computational cost, they are not robust and lack flexibility - being only able to excel in situations where clear rules or constraints are available.

On the other hand generate-and-test methods, as the name suggests, form a solution by creating a set of several possible solutions for the problem and then evaluating each one to determine the best one based on specific evaluation criteria. Since finding the next best camera position in the continuous space by evaluating each possible position is very costly these methods reduce the search space by sampling some candidate views - therefore defining a discrete search space as an approximation of the continuous one. They then look for the next position in that discrete space, turning the optimization problem into an evaluation problem - a set of evaluation criteria is defined, all the viewpoints in the search space are evaluated and ranked, and the best performing one is selected. Although being more computationally demanding, these types of methods are very adaptable and explore a wider range of possibilities, finding solutions that might not be immediately apparent from domain knowledge alone and showing good results in situations where heuristic rules are not well defined.

Recent works have began to propose solutions using deep learning to predict the NBV. These methods learn from data on past experiences and can predict which viewpoints yield the most informative observations.

B. Deep Learning for the NBV Problem Resolution

The vast majority of the methods that use deep learning to find the NBV are generate-and-test methods. So, as explained above, they approximate the continuous space by creating a discrete set of possible camera viewpoints, referred as candidate viewpoints. Then each method has its own set of evaluation criteria to measure the quality of each candidate view, that is usually translated in an evaluation score - a score that calculates the information gain that each view point brings. After ranking all the candidate viewpoints according to their score, the best classified one is selected as the NBV. Deep learning enters this process by learning how to approximate the function of the evaluation score for each candidate viewpoint, being then able to predict it.

For the networks to be able to make these predictions they need to be fed with training data containing the ground-truth information. Therefore, a simulated reconstruction process can be performed with objects whose complete 3D models are previously known. During that process data like the partial model, the view's scores and the selected NBV is saved, forming the training pairs. Later a network can be trained on these data points and learn to predict the evaluation scores. Deep learning algorithms have shown to improve the efficiency and efficacy of NBV planning, since they achieve higher object coverage scores with less inference time.

1) Point Cloud Based Next-Best-View Network (PC-NBV): PC-NBV is a deep learning network that has shown to have good performance, compared to other common NBV deep learning networks, since it uses point cloud for the model representation. Point cloud representation is less computationally costly compared to other common representation methods like voxels and triangular meshes. This network uses deep learning not only to predict the NBV but also for point cloud processing, as deep learning has proven to be quite successful in this area [3], [4].

The network receives a raw point cloud and predicts what is the NBV out of a set of 33 candidate views. The candidate views are defined by uniformly sampling view points on a sphere around the object.

$$C(P) = \frac{1}{|P_o|} \sum_{p \in P} U(min_{p_0 \in P_0} ||p - p_0||_2 + \varepsilon) \qquad (1)$$

It learns the utility function that measures the improvement of the object's surface coverage, (1), where U is the Heaviside step function, ϵ is a distance threshold, P_0 is the complete point cloud of the object and P is the object's partial point cloud - the point cloud of the object's reconstruction. Essentially this score (from now on referred to as coverage score) quantifies the amount of new points a viewpoint captures in relation to the object partial point cloud (points of the object's total point cloud that have not yet been included in the partial point cloud), and presents the value as a ratio between the number of new points and the total number of points of the object's full point cloud.

The network's architecture is represented in the Figure 1. It consists firstly of a feature extraction unit proposed in [5] - this unit is composed by a sequence of dense blocks, and it is used to extract structure-aware features. [5]'s ablation study showed that this unit improved the network's capacity, contributed for a significant reduction of the number of parameters, and showed that it extracted better features from the point cloud. The feature extraction unit is then followed by a max pooling layer and a self attention unit proposed in [6]. According to this study, attention mechanisms have become an integral part of models that have to capture global dependencies. This unit enables the network to learn how to allocate attention according with similarity of color and texture, it helps to model multi-level dependencies across image regions. This unit is then followed by a shared multilayer perceptron, a max pooling layer and another multilayer perceptron.

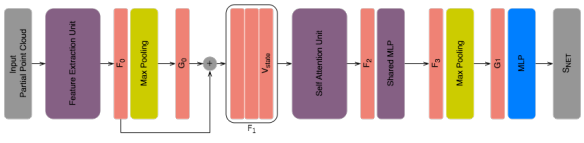

Fig. 1. PC-NBV Architecture. Input and Output: Gray; Larger Units: Purple; Pooling Layers: Yellow; Fully Connected Layers: Blue; Features: Pink.

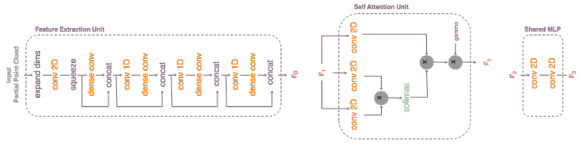

Fig. 2. PC-NBV Units Architecture. Convolutional Layers: Orange; Activation Layers: Green; Inputs: Gray; Features: Pink

The total loss of the model is given by Equation (2): the sum of the MSE between the coverage score prediction and the groundtruth with the L2 loss of the weights. The MSE between the coverage score prediction and the groundtruth measures how accurate the model is. The L2 regularization loss measures how well the model adheres to the regularization constraints.

$$\mathcal{L} = \frac{1}{n_s}\sum_{i=1}^{n_s}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{M} w_j^2 \tag{2}$$

This network shows great results compared with other NBV methods. It quickly reaches high surface coverage levels both with objects similar to the ones analyzed in the training data, as with objects that the network has never seen before. Also, the inference time of PC-NBV is smaller than in other methods, because the point cloud representation does not require any complex data transformations like ray projection operations that other methods require.

C. Uncertainty in Deep Learning

Deep learning models are not always certain about their predictions. Taking for example a model that is fed with a test point that is out of the distribution of the set it was trained on: it is forced to output a prediction for this point. However it does not have enough training data to support the decision. The probabilities of it resulting in a wrong prediction are high, the model makes a prediction with high uncertainty. Hence, in some applications it can be important to know the model's prediction with an extra value that measures how certain the model is of its output.

The standard tools used to create deep learning models, like feedforward neural networks and convolutional neural networks, cannot usually access the models confidence in its predictions. However, in Bayesian networks [13] this becomes possible - this type of networks represent probabilistic dependencies between variables, here the weights of the networks are defined as probabilistic distributions, which offers a mathematically grounded framework to reason about model uncertainty. [7] explains that Bayesian networks are able to reason under uncertainty and introduces a method that uses dropout layers [12] to approximate neural networks to Bayesian networks, and the Monte Carlo sampling method to enable the network to infer the uncertainty of the predictions. [7] proves that a neural network with dropout applied before every weight layer is mathematically equivalent to a Bayesian Network.

In Bayesian networks, in each forward pass the network assigns a different value for the weights, sampled from the weight's distribution. So, in practice, in each forward pass there is a different learned function f. Hence, with the same data a different prediction $y' = f(x, \theta)$ is made. If several forward passes are made with the same data, there are several different predictions as well, and in the end the network's final prediction can be given as an average of all the predictions, (3).

$$y' = \frac{1}{N}\sum_{i=1}^{N} f_i(x, \theta_i) \tag{3}$$

Having several predictions, it is now possible to measure the prediction's uncertainty by calculating the variance.

[7] states that dropout layers can emulate this behavior. In a network with dropout layers each neuron has some p probability (dropout rate) of being switched off. In each training step a different neuron is turned off, according to the probability p. It is as if we get a slightly different network each time, which means we can get several different predictions as well. Most commonly dropout layers are only used during training since they are being used to reduce overfitting [12]. However, in the method proposed by the authors, dropout layers also have to be used during inference. This way we get several predictions to average and are able to calculate the variance. This method is referred to as the Monte Carlo Dropout Method (MCDM).

Many articles such as [10] and [11] have already successfully used the MCDM to modify already existing deep learning networks to make them account for uncertainty. Both works achieved positive results. In [10] modeling uncertainty improved segmentation performance by 2-3%, and in [11] it enabled an improvement in localization accuracy.

III. Methodology

To address the NBV deep learning models limitation of not being able to quantify the prediction uncertainty, and to improve the performance of the models, this dissertation proposes modifying PC-NBV as presented in [7] - making it account for uncertainty. The modification effects on the prediction will be examined, and two uncertainty-based metrics will be studied to improve view selection. The uncertainty measurements will be analyzed by evaluating their relevance as decision-support measures, and analyzing their potential ability to improve the model's final coverage score prediction.

In brief, this thesis aims to enable the model to predict an uncertainty measurement along with its regular prediction, that reflects how sure the model is of each prediction. This uncertainty measurement could then be used to assist the NBV choice process, and could ideally be used to improve the model's outcome, making it closer to reality.

A. Dataset

The PC-NBV's authors did not provide the partitions of the training/testing set they used to train the model on, but made available the process they followed to generate it. Therefore, the first critical task was to follow the same process to create the training and testing datasets.

The adopted approach to generate the dataset was by making a simulated 3D reconstruction of objects whose full 3D

models are previously known. These models were obtained from ShapeNet [9] - a dataset with hundreds of 3D models organized in several categories (called classes) of objects.

For the training set 4000 models from 8 categories (airplane, cabinet, car, chair, lamp, sofa, table and vassel) were randomly chosen (500 models from each category), and another different 400 models for the validation set (50 models from each category). Regarding inference two sets were created - one with 400 models selected from these 8 categories known in the training/validation sets, and another set - the novel set - with 400 models from other 8 categories not learned by the model. This last testing set was created to evaluate the performance of the model on unknown objects.

The selected ShapeNet models were then processed to create the NBV data. An artificial reconstruction of the 3D object is made for each model. At each viewpoint from the candidate view space (Figure 3), it is simulated the point cloud acquisition with a virtual camera in the 3D environment. Then a first viewpoint is randomly selected and the search for the NBV begins. All the views are evaluated calculating the surface coverage score they would achieve if selected (1). The one with a higher score is selected as the NBV. This scanning process occurs for 10 scans. The training data consists of 3 types of information that is stored in each scan, during the reconstruction process: the selection state of each viewpoint, V_{state} - when a viewpoint is selected as the NBV its selection states changes from 0 to 1; the coverage score of each viewpoint - whose value belongs to the interval $[0, 1]$; the accumulated point cloud, P_{part} - the point cloud of the reconstructed object until that point of the scanning process. This point cloud starts empty in the beginning of the process, and in each scan the point cloud from the selected NBV is appended to it. In the last scan P_{part} will be the complete point cloud of the reconstructed object. Each scan of this artificial reconstruction is fed to the model as an independent sample.

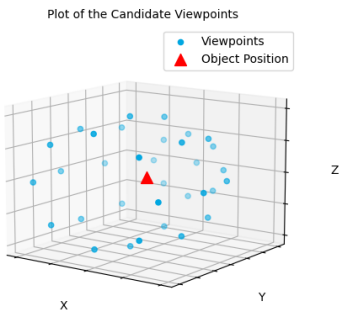

Fig. 3. Candidate View Space - $n_v = 33$ viewpoints with direction pointing towards the origin, from each point in the plot.

B. Training and Model Architecture

1) Baseline: To be able to evaluate the results of the network that is going to be created by modification of the PC-NBV model it is first important to get a baseline to compare the results to. This baseline is given by the results of the original model. So it is important to train and test this model on the created dataset.

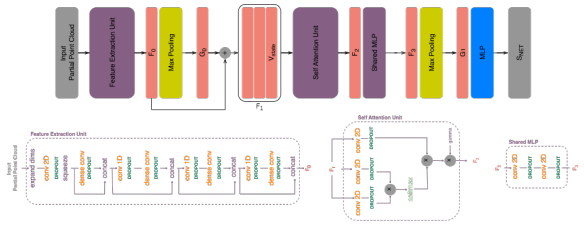

Fig. 4. Bayesian PC-NBV architecture. Input and Output: Gray; Larger Units: Purple; Pooling Layers: Yellow; Fully Connected Layers: Blue; Features: Pink; Convolutional Layers: Orange; Activation Layers: Light Green; Dropout Layers: Dark Green.

The model was trained in the training set for 300 epochs, with a batch size of 32, a 0.01 learning rate and a 10^{-6} weight decay. The trained model of the epoch that achieves the lowest training loss was then used to evaluate the model performance on the two test sets.

2) Architectural Changes: To enable the model to account for uncertainty by implementing the MCDM dropout layers need to be applied after each convolutional layer. The new model architecture is presented in Figure 4, where the added dropout layers are signalized in dark green. This new architecture will be referred to as Bayesian PC-NBV.

When setting the dropout layers, a dropout probability needs to be defined. Usually, the dropout probability falls between 0.2 and 0.5. This is an hyperparameter that could be optimized, however, due to time constraints, it was decided to set it to 0.5, as done in [10], [11], and leave that optimization to future work.

The model was then trained with the same parameters used in the PC-NBV training.

3) Evaluation metrics: In this dissertation the performance of the models is evaluated with two metrics besides the loss (2). These two measurements are the error and accuracy of the predictions. The error measures the difference between the model prediction and the groundtruth, it differs from the loss since it does not account for the regularization loss. The accuracy measures if the model's predicted NBV matches the groundtruth's NBV.

Various error metrics were evaluated, in order to pick the one that would be the most relevant. Equation (4) measures the error by calculating the Euclidean Distance between the groundtruth (gt) and the model prediction (fp). Equation 5 measures the sample error by calculating the sum of each view Squared Difference between gt and fp, which is the square of the Euclidean Distance. The model error is then given by the mean of the sample errors.

$$Euclidean\ Distance = |gt - fp| = \sqrt{\sum_{i=1}^{n_v} (gt_i - fp_i)^2} \quad (4)$$

$$Squared\ Error = \sum_{i=1}^{n_v}(gt_i - fp_i)^2 = Euclidean\ Distance^2 \tag{5}$$

Despite the fact that PC-NBV is a regression model, the scores it regresses are used to choose a NBV, i.e - in a real world problem the closeness of the coverage score predictions to the groundtruth is not 100% important, the most important aspect is that the model predicts the highest coverage score on the right viewpoint. And for this is important to evaluate if the model has the ability to predict the right NBV viewpoint or not.

In most of the cases, in the groundtruth, there is only one NBV - the viewpoint with the highest coverage score. However, up on further examination, when visualizing the data, it is possible to understand that in some cases more than one view achieve coverage scores close to the NBV coverage score value. In practice, if we assume independence between a sequence of selections, it wouldn't make a lot of difference to choose one of these views over another, therefore in some cases there is more than one possible NBV. To calculate the accuracy of each sample this circumstance is taken into consideration. We analyze if the predicted NBV belongs to a set of views whose coverage scores lay in an interval given by Equation (7), i.e. a set of top views with close coverage scores between each other. If the model chooses a viewpoint that is inside this set we assume the prediction is right, if not the prediction is set as wrong - Equation (6). In the end the model accuracy is given by the ratio between the number of right predictions and the total number of samples - Equation (8).

$$accuracy(fp) = \begin{cases} 1, \text{if } fp_{NBV} \in \{\text{NBVS}\} \\ 0, \text{otherwise} \end{cases} \tag{6}$$

$$\begin{aligned} \{NBVS\} = \{v_{i^*}\}, \{i^*\} = \{i^* \in arg(C), \\ C \in [max(fp) - 0.15\alpha, max(fp)]\} \\ \alpha = max(C) - min(C) \end{aligned} \tag{7}$$

$$model\ accuracy = \frac{\sum_{i=1}^{n_s} accuracy(fp_i)}{n_s} \tag{8}$$

C. Bayesian PC-NBV Inference and Uncertainty Measurements

1) Inference: The usual behavior of dropout layers is to be on during testing and off during inference. However, in this case, to get several different predictions for the same test sample, it is desired that the dropout layers are on during inference as well.

The MCDM requires a somewhat large number of inferences of the same test sample. Based on the conclusions of the works [10], [11] this number (n_{mc}) was set to 40. The 40 MC samples (mc) are then averaged to obtain the model's final prediction - Equation (9).

$$fp = C\hat{(P)} = \frac{\sum_{i=1}^{n_{mc}} mc_i(x)}{n_{mc}} \tag{9}$$

2) Uncertainty Measurements: The uncertainty measurements are given by the differences along the MC samples. In theory, samples that show larger differences between themselves reflect a higher model uncertainty.

The model's prediction is composed of a coverage score for each of the 33 viewpoints. So it is possible to calculate the standard deviation for each viewpoint, having an uncertainty metric for each view. However, it would be more interesting if the model could return a single value that reflects how certain the model is of its prediction. So, to address this question several standard deviation metrics were studied: Standard Deviation by View, σ_v - Equation (10); Mean Standard Deviation, $\overline{\sigma}$ - Equation (11); NBV Standard Deviation, σ_{NBV} - Equation (12); Standard Deviation as a Whole, σ_{whole} - Equation (13).

$$\sigma_{v_j} = \sqrt{\frac{1}{n_{mc}}\sum_{i=1}^{n_{mc}}(fp(v_j) - mc_i(v_j))^2}, \quad j = \{1, ..., n_v\} \tag{10}$$

$$\overline{\sigma_v} = \frac{1}{n_v}\sum_{j=1}^{n_v}\sigma_{v_j} \tag{11}$$

$$\begin{aligned} \sigma_{NBV} = \sigma_{v_{i*}} \\ v_{i*} = argmax(fp) \end{aligned} \tag{12}$$

$$\begin{aligned} \sigma_{whole} = \sqrt{\frac{1}{n_{mc}}\sum_{i=1}^{n_{mc}}|fp - mc_i|^2} = \\ = \sqrt{\frac{1}{n_{mc}}\sum_{i=1}^{n_{mc}}\left[\sum_{j=1}^{n_v}\left(fp_j - mc_{i_j}\right)^2\right]} \end{aligned} \tag{13}$$

All of these metrics were evaluated to understand which one could better translate the uncertainty and better adapted to this problem.

IV. Results Analysis

A. Training and Testing Results

The PC-NBV model was trained in the training set with the parameters described in the previous section. Its learning curve is shown in Figure 5, it is visible that the model was able to learn the data without overfitting it.

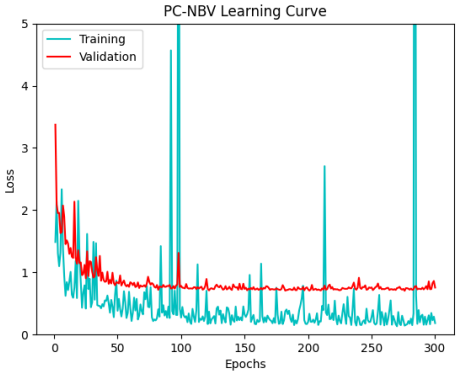

Fig. 5. PC-NBV Learning Curve. Loss given by Equation (2)

TABLE I
PC-NBV MODEL LOSS (2), MODEL ERROR - EUCLIDEAN DISTANCE (4) AND SQUARED DIFFERENCES (5)), AND MODEL ACCURACY (8)

PC-NBV	Model Loss	Model Error (Euclidean Distance)	Model Error (Squared Differences)	Model Accuracy
Train	0.144	-	-	-
Valid	0.691	0.070	0.018	29.76%
Test	0.662	0.069	0.017	29.95%
Test Novel	1.160	0.093	0.033	26.59%

TABLE II
BAYESIAN PC-NBV MODEL LOSS, MODEL ERROR - EUCLIDEAN DISTANCE (4) AND SQUARED DIFFERENCES (5), AND MODEL ACCURACY (8)

Bayesian PC-NBV	Loss	Error - Euclidean Distance	Error - Squared Differences	Accuracy
Train	0.287	-	-	-
Valid	0.704	0.072	0.018	28.66%
Test	0.662	0.071	0.017	29.02%
Test Novel	1.217	0.097	0.035	25.66%

The loss, error and accuracy results are presented in Table I, these are the metrics that are going to be used to compare model performances.

The model error as well as the model accuracy in the valid and test sets are quite similar. This was expected, since both of these sets have similar sizes and are composed by models in which the network was not trained on but of the object classes the network knows. In the test novel set, as expected as well, these values worsen: the model error increases 35% for the Euclidean Distance and 94% for the Squared Differences; as for the model accuracy it decreases 3.36 percentage points.

The Bayesian PC-NBV model was trained with the same parameters used for the PC-NBV model. The model's learning curve is presented in Figure 6. It is visible that also here the model did not overfitt the data. However, compared to the previous training process, the loss seems now slightly more unstable. This is an expected behavior caused by the randomness dropout introduces during the training process, causing the loss to fluctuate more during training compared to when dropout is not used. Table II shows the network's loss, error and accuracy values.

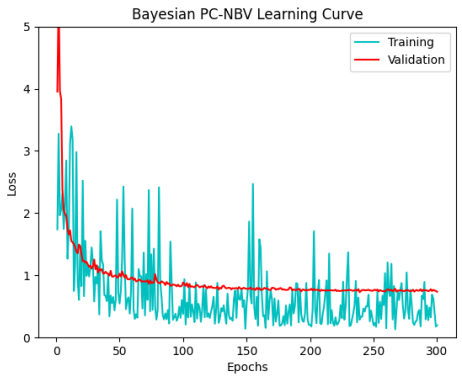

Fig. 6. Bayesian PC-NBV Learning Curve. Loss given by Equation (2).

Comparing the loss, error and accuracy values of the Bayesian PC-NBV model with the PC-NBV model there are very little performance shifts. The loss throughout the sets keeps quite similar values, with a small increase of about 0.1 in some sets. As for the model error values they virtually do not change between both models. When it comes to the model accuracy there is a slight decrease in the performance of the model, it was verified a decrease of about 1 percentage point across all the sets.

Despite the introduction of dropout layers and the inherent randomness they entail during training, the Bayesian PC-NBV model keeps presenting a good performance when implementing the Monte Carlo sampling method during inference. Although some marginal shifts in loss, error, and accuracy metrics were verified, when comparing to the PC-NBV model, these were quite small and are not considered as a big performance drop. These small changes are the price to pay to enable the model to predict an uncertainty measurement, which will be addressed next.

B. Uncertainty Quantification

The biggest challenge of uncertainty quantification is to find a measurement that correctly reflects the metric we want to predict. In this case it would be interesting to predict a measurement that could model the sample's error and could also model the samples accuracy, i.e - a large uncertainty value would mean that the prediction would have a large error and a higher probability of being wrong (of not having predicted the right NBV).

The relation between the uncertainty measurement and the error was studied. The desired relation between both metrics is a directly proportional one - when the uncertainty value is high the error value should be high as well. A purely horizontal or vertical pattern would not be good to differentiate between right and wrong predictions. So to study this and understand which uncertainty measurement - (12), (11) or (13) - best models which error metric - (4) or (5) - a plot of the error by the uncertainty of each sample for the different combinations was made, and is displayed in Figure 7. In addition, we want to study if the accuracy of the selected view can be modeled as a function of the uncertainty. To do that, we plot each sample with one of two colors: blue if presents a right NBV prediction and red if is wrong.

Looking at the different plots of Figure 7 we only see an approximation of the desired proportional relation when the error metric is given by Euclidean Distances. When the metric is given by the Squared Differences the distribution is quite

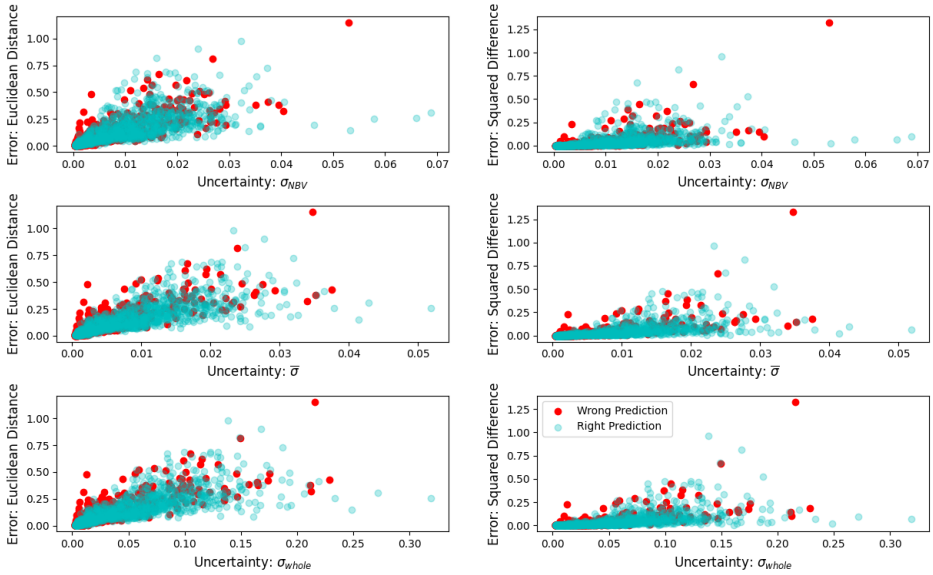

Fig. 7. Distribution of the sample's error in function of uncertainty - different combinations. From top to bottom - uncertainty: σ_{NBV}, (12); $\overline{\sigma}$, (11); σ_{whole}, (13). From left to right - error: Euclidean Distance, (4); Squared Differences, (5).

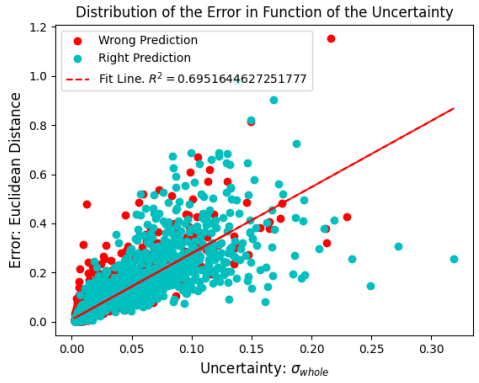

Fig. 8. Distribution of the sample's Euclidean Distance error in function of the σ_{whole} uncertainty and its approximated fit line (in red) with $R^2 = 0.695$.

horizontal, with no clear distinction in error as uncertainty increases. Hence, the chosen error metric to model is going to be the Euclidean Distance (4).

Looking at the left side of the Figure 7, where the used error metric is the Euclidean Distance (4), it is possible to verify that the uncertainty metrics that better allow the desired distribution are $\overline{\sigma}$ (11) and σ_{whole} (13). Both of these metrics allow quite similar distributions. However, σ_{whole} has a larger range of uncertainty values. While the values of $\overline{\sigma}$ only vary in the interval $[0, 0.06]$ the σ_{whole} values vary in the interval $[0, 0.4]$ making it a more suitable metric.

Figure 8 shows a closer look at the distribution of the (4) error in function of the the (13) uncertainty, as well as the line that best fits the data. This fit is evidently not perfect, having a $R^2 = 0.695$. However, it serves to highlight the tendency of the data distribution.

Despite this uncertainty metric being able to successfully model the prediction error given by the Euclidean Distances, it is visible that it is not capable of modeling the accuracy of the sample (6). The desired behavior would be for the blue dots (the right predictions) to be distributed in the lower range of the uncertainty values and the red dots (the wrong predictions) in the higher range. However, this is not visible - the σ_{whole} uncertainty metric does not model the accuracy of the samples.

This happens because the accuracy of a sample (6) is not correlated to its error (4). The accuracy measures if the highest coverage score was attributed to the groundtruth's NBV while the error measures how close the coverage scores are from the groundtruth's coverage scores. As the σ_{whole} uncertainty calculation is based on the MC samples $Euclidean\,Distance$ error, it is natural that it can correctly offer an insight of the error, not being able to do so for the sample's accuracy.

Inspired by this, an uncertainty metric capable of modeling the sample's accuracy (6) can be obtained if it is calculated based on the the MC samples accuracy. Thus, we computed the accuracy of every MC sample for uncertainty estimation as in Equation (15). This accuracy checks if the sample's NBV is the same NBV of the final prediction (9). The uncertainty is calculated as the Standard Deviation of the MC sample's accuracy, $\sigma_{accuracy}$ (14).

$$\sigma_{accuracy} = \sqrt{\frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} (1 - accuracy(mc_i))^2} \tag{14}$$

$$accuracy(mc_i) = \begin{cases} 1, NBV_{mc_i} = NBV_{fp} \\ 0, otherwise \end{cases} \tag{15}$$

$$i \in \{1, ...40\}$$

Figure 9 shows an histogram that quantifies how many samples in each uncertainty bin get the NBV prediction right and how many get it wrong. It is visible that most of the samples in the low range of the $\sigma_{accuracy}$ uncertainty

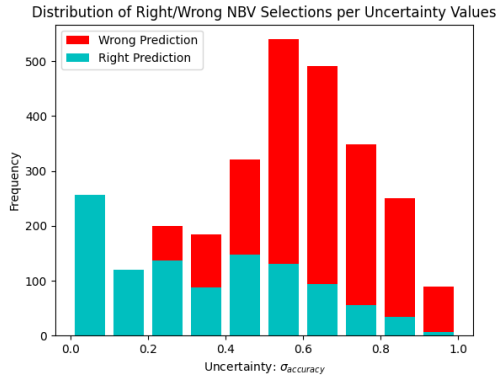

Fig. 9. Distribution of right/wrong NBV predictions per uncertainty $\sigma_{accuracy}$ values (14).

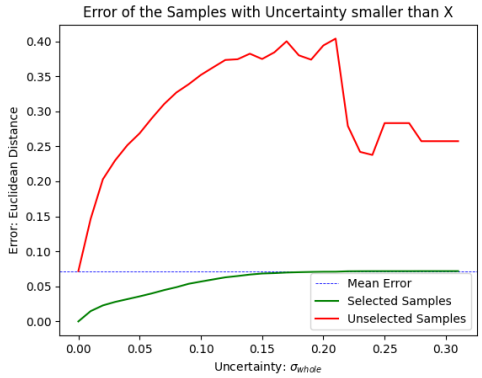

Fig. 10. Model Error (Euclidean Distance, (4) in function of uncertainty σ_{whole} (13).

values are samples that got the prediction right, while most of the samples with higher uncertainty values are samples that weren't able to do so. This was the desired behavior for accuracy modeling, and validates the proposition that in order for uncertainty to reflect accuracy it needed to be computed from values that measured this quantity.

By now we have two uncertainty metrics capable of estimating uncertainty measurements that give insights into the prediction's error - σ_{whole}, (13) - and into the prediction's accuracy - $\sigma_{accuracy}$, (14). However, an important question arises: How can these metrics help improve the model's performance?

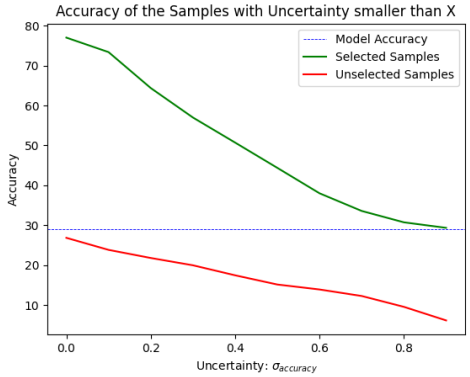

Fig. 11. Model accuracy (8) in function of uncertainty $\sigma_{accuracy}$ (14).

C. Leveraging Uncertainty for Model Performance Improvement

1) Coverage Score Improvement: The first analyzed approach for performance improvement using the uncertainty measurments was by incorporating the uncertainty predictions with the coverage scores predictions, in an attempt of generating a new improved coverage score. This coverage score would ideally be more similar to the groundtruth, reducing the $Euclidean\ Distance$ error (4). It could also improve the coverage score of the real NBV, increasing that view's coverage score, therefore improving the sample's accuracy (6).

To do this several different combinations incorporating the uncertainty measurements with the coverage prediction were evaluated. We concluded that simply using the error related uncertainties by adding or multiplying them with the final prediction did not improve the model error nor the model accuracy. The formulas that presented better results, being able to slightly increment the model's accuracy, were more complex. These multiplied σ_{view} uncertainty (10) by a factor and added that to the final prediction values. Here we were trying to find a factor that could maximize the coverage scores while minimizing the uncertainty of the predictions. The two best factors obtained in this study were: $\sigma_{normalized} \times 0.8$ and $\sigma_{normalized} \times \sigma_{accuracy}^{0.4}$, where $\sigma_{normalized}$ is given by $\sigma_{view}/max(\sigma_{view})$.

$max(\sigma_{view})$ is the maximum measured uncertainty σ_{view} throughout all the test samples, for each view. Therefore, to use this factor we need to have prior information of all the view's uncertainty. To achieve this a calibration set can be used to obtain an estimate of the $max(\sigma_{view})$ factor. Using a set like the validation set, we can compute all the prediction's σ_{view} uncertainties and then find the highest one for each view, obtaining a vector that can be used as a regularization factor on the other sets.

Table III presents the formulas that achieved the best results and the according improvements.

2) Decision Support: Another possible way of using the uncertainty measurements would be as decision support metrics. By rejecting predictions with an uncertainty value higher than a set limit we would be discarding low quality predictions ensuring a better model performance.

The Figure 10 shows in green the plot of the model Euclidean Distance error (4) in the test set when considering only samples with σ_{whole} uncertainty (13) smaller or equal than the value in the x axis. In red it plots the model error considering all the other samples that were not used, i.e. that had a higher uncertainty value. The dotted blue line marks the model's error when considering all the samples. This graph shows that in fact when we only account the samples with low uncertainty the model achieves lower error values. In this case if we consider only samples that have an uncertainty lower than 0.15 it is possible to reduce the model's error.

TABLE III
NEW COVERAGE SCORE FORMULAS - BEST RESULTS

		Error (4)	Error Change	Accuracy (%) (8)	Accuracy Change
$fp \pm (\sigma_{view} \times (\sigma_{normalized} \times 0.8))$	Valid	0.073	+0.001	28.45	-0.22
		0.072	0	28.71	+0.05
	Test	0.072	0	29.16	+0.14
		0.072	0	29.21	+0.19
	Test Novel	0.097	0	25.80	+0.14
		0.097	0	25.64	-0.02
$fp \pm (\sigma_{view} \times \sigma_{normalized} \times \sigma_{accuracy}^{0.4})$	Valid	0.073	+0.001	28.45	-0.21
		0.072	0	28.74	+0.08
	Test	0.072	0	29.21	+0.19
		0.072	0	29.18	+0.16
	Test Novel	0.096	-0.001	25.77	+0.11
		0.097	0	25.66	+0.00

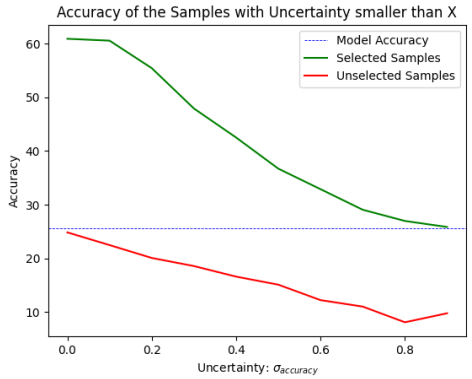

Fig. 12. Model accuracy (8) in function of uncertainty ($\sigma_{accuracy}$, (14) on the test novel set.

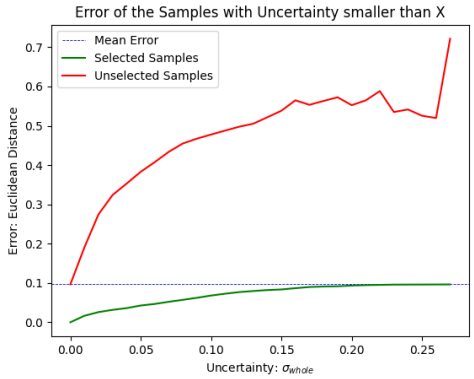

Fig. 13. Model error (Euclidean Distances, (4) in function of uncertainty (σ_{whole}, (13) on the test novel set.

Figure 11 shows the same behavior but for the model's accuracy. The model achieves high accuracy levels when samples with high uncertainty are discarded, and is visible with the red line plot that the samples that are discarded always achieve an accuracy lower than the model's accuracy when accounting all the samples. It becomes possible to achieve extremely good accuracy percentages comparing to the original model accuracy simply by discarding predictions with high uncertainty. In this case if we want to achieve a model accuracy of 60% we need to discard the samples with an $\sigma_{accuracy}$ uncertainty higher than 0.2.

To use this approach there is a need to set an uncertainty limit that will classify a prediction as acceptable or not. This limit can vary depending on the desired model performance and on the used data. This limit should then be calibrated on a calibration set, where one would examine which uncertainty values allowed the desired model performance.

Assuming that the set used to generate the Figures 10 and 11 is our calibration set. If we would like to achieve a model accuracy not lower than 60% we would have to discard samples that had an $\sigma_{accuracy}$ uncertainty value higher than 0.2. The same can be calibrated for the error - taking the example of Figure 10 if we want to achieve a better model error we need to discard predictions with an σ_{whole} uncertainty lower than 0.15.

By testing these limits on the test novel set, we can see on Figures 12 and 13 that the error related uncertainty measurement with values under 0.15 truly achieved a model error smaller than the error when all the samples were accounted for. As for the accuracy related uncertainty measurement the achieved model accuracy with samples with uncertainty lower than 0.2 actually achieved values lower than 60%. But having into consideration that the possible maximum coverage for this set was 60%, the coverage achieved by the samples with uncertainty values up to 0.2 did not worsened the accuracy that much, having these samples achieved an accuracy of around 55%. In this way it is possible to calibrate the uncertainty measurements on a appropriated set to achieve a desired model performance.

To use these metrics as decision support measurements it would be, however, necessary to implement what to do when a prediction is discarded during the 3D acquisition process, since the model would not provide any information of what the NBV would be. Then tests to show the efficiency of the implemented method in comparison to the efficiency of simply choosing a wrong predicted NBV needed to be run. This is an essential analysis to evaluate the usage relevance of these as measurements decision support metrics. An example of what could be done to avoid not having a NBV to go to when a prediction is discarded is to choose the viewpoint that least interferes with the prediction's uncertainty.

V. Conclusion and Future Work

The Monte Carlo Dropout method was successfully implemented in an already existing model that showed good performance - PC-NBV. By applying dropout layers after all the convolutional layers of the model and sampling several model predictions for the same input during inference, a functional framework to obtain the prediction's uncertainty was implemented. Several uncertainty metrics were studied with the goal of finding the ones that better reflected the prediction's error and accuracy. It was not possible to find a metric that combined the two, but two separate metrics that successfully reflected these quantities were found. Additionally it was noted that to obtain an uncertainty metric that measures a certain quantity it is necessary to measure that quantity in the MC samples and compute that metric variation between the MC samples and the mean of the samples. In that way, to obtain an uncertainty measurement that reflected the sample's error the error between the MC samples and the mean is calculated. To get an uncertainty measurement that reflected the sample's accuracy the accuracy of the MC samples in relation to the mean was compared.

Moreover, the usage potential of the uncertainty metrics to leverage the model's performance, was studied. The first suggested approach was to use them to improve the model's final coverage score prediction, by implementing a new coverage score formula that incorporates the uncertainty measurements with the final prediction. We were able to obtain small model error and accuracy improvements with this approach, by multiplying the uncertainty by a factor and adding it to the final prediction. Another proposed method was to discard any sample that presented an high uncertainty value. This approach seems to be able to improve the model's error, but most important, the model's accuracy. By discarding samples with high uncertainties it was possible to improve the model's accuracy from 30% to 60%-80%. However, this method faces some obstacles that need further studies, as it would need another NBV decision approach to use when the model discarded a prediction. We propose a decision approach to use in this circumstance: choosing as the NBV, when a prediction is discarded , the view that least interferes with the prediction uncertainty. This strategy would, however, require testing to study it's affects on the efficiency of the 3D reconstruction. This was not possible to study during this dissertation, so it is an important limitation that can be studied in a later work.

Besides these future studies that explore the limitations of this dissertation, future works could also analyze if different dropout layer placements could achieve better results. The reviewed works [11] and [10] showed that in fact applying dropout after each convolutional layer could work as a too strong of a regularizer, and stated that only applying dropout layers in certain parts of their networks showed better results. That analysis was not done in this work, but could also be an interesting study. As well as the optimization of the dropout probability, that in this work was left at 0.5. However it is possible that another probability could achieve better results. Also, another study not covered in this dissertation was the study of the ideal number of Monte Carlo dropout samples, which is a parameter that could also be optimized. Lastly, another important study to run in the future is the use of this model on a real life scenario.

References

[1] *RePAIR Project*. https://www.repairproject.eu/

[2] Zeng, R., Zhao, W., & Liu, Y. *PC-NBV: A Point Cloud Based Deep Network for Efficient Next Best View Planning*. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7050-7057, 2020.

[3] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2016

[4] Qi, C. R., Yi, L., Su, H., & Guibas, L. J. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017

[5] Yifan, W., Wu, S., Huang, H., Cohen-Or, D., & Sorkine-Hornung, O. *Patch-Based Progressive 3D Point Set Upsampling*. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5951-5960, 2019.

[6] Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. *Self-Attention Generative Adversarial Networks*. 2018

[7] Gal, Y., & Ghahramani, Z. *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*. *international conference on machine learning*, 1050-1059, 2016.

[8] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research*, 15, 1929-1958, 2014.

[9] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. *"Shapenet: An information-rich 3d model repository"*. 2015

[10] Kendall, A., Badrinarayanan, V., & Cipolla, R. *Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding*. *CoRR* (abs/1511.02680), 2015.

[11] Kendall, A., & Cipolla, R. *Modelling Uncertainty in Deep Learning for Camera Relocalization*. *CoRR* (abs/1509.05909), 2015.

[12] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research* (15), 1929–1958, 2014.

[13] Radford M. Neal. *Bayesian Learning for Neural Networks*. *Springer New York, NY*, 1996.

[14] Mendoza, M., Vasquez-Gomez, J., Taud, H., Enrique Sucar, L., & Reta, C. *Supervised learning of the next-best-view for 3d object reconstruction*. *El Sevier, Patter Recognition Letters* (133), 224-231, 2020.

[15] Vasquez-Gomez, J., Troncoso, D., Becerra, I., Sucar, E. & Murrieta-Cid, R. *Next-best-view regression using a 3D convolutional neural network*. *Machine Vision and Applications* (32), 2021.

[16] Han, Y., Zhan, I., Haozhe, Zhao, W. & Liu, Y. *A Double Branch Next-Best-View Network and Novel Robot System for Active Object Reconstruction*. *2022 International Conference on Robotics and Automation (ICRA)* 7306-7312, 2022.