Coverage Planning for Part Inspection

Sahil Sindhi - A*STAR

Abstract

My task for this research project involved improving state of the art algorithms used for part inspection, more specifically, capturing a set of images of the part that would then be used to inspect the surface of the part. The setup involves a camera mounted on a robot arm and a manufactured part for inspection, as shown in Figure 1. The part inspection task is an instance of the coverage planning problem(CPP). The key aim is to develop better algorithms to improve inspection coverage, cycle time and quality of images obtained. The goals of this work are also to find more general purpose algorithms that are able to carry out inspection tasks in complex environments. This is important problem as there are a variety of use cases where viewpoints for inspection or other sensor positions need to be chosen in a complex scene with occlusions and obstacles. This is a technical report outlining various aspects of the internship work including the work completed, research carried out and findings from selected papers accompanied by thoughts and evaluations.

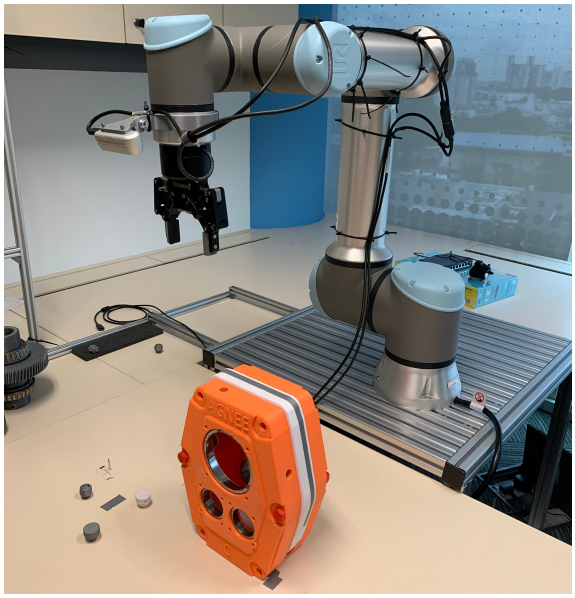

Figure 1: Part inspection Setup

1 Problem set up

The coverage planning problem is obtaining a path for the robot to follow such that all regions of interest can be scanned. In the context of part inspection this involves generating a path such that the surface area of the part visible from the images taken by a camera on the robot is maximised. Additionally the task may be formulated to achieve this while also minimising other objectives such as the time taken to complete the inspection cycle. There are two general types of approaches to the task, the first involves selecting a sequence of viewpoints prior to the actual inspection task, this is the offline approach. The alternative class of solutions follow a next best view strategy, this consists of selecting viewpoints in an online manner one after another based on the images captured already.

Existing literature on the task formulate the offline problem as follows, we start with the 3D mesh of the object as a set of polygons and a set of predefined viewpoints and for each viewpoint we deduce the set of visible polygons from that viewpoint. The coverage planning problem can then be broken down into a set covering optimisation problem(SCOP), finding the smallest subset

of viewpoints which provide visibility of all the polygons, followed by a travelling salesman problem(TSP), obtaining an order to visit these viewpoints to minimise the overall travel time. Both of these problems are NP-hard and hence we must seek alternative solutions to these problems. There are a few key limitations with this formulation of the problem. Firstly the problem is split up which means that solutions that are optimal with respects to the two sub problems may not be jointly optimal in some sense. For example, there may be a subset of viewpoints that give complete coverage but include a greater number of viewpoints but actually happen to lead to a faster cycle time to visit all the viewpoint and this solution would be missed. Hence we seek a scheme to jointly optimise the objectives of coverage and time. Also these schemes rely on accurate information about the coverage obtained by each point however these may be inaccurate due to uncertainties in the true pose of the object or the uncertainty in the environment.

2 Prior Work on project

Prior to my work the current pipeline employed at A*STAR consisted of randomly sampling viewpoints on a sphere as shown in Figure 2. Next, these viewpoints are filtered based on reach-ability and visibility. A mesh is created of the scene and the object and then ray tracing methods are used to decide if the viewpoints have any visibility of the object and if not they are discarded. Next the reach-ability of a viewpoint is determined by inverse kinematics and trajectory planning using bi-directional RRT. If no solution is obtained to the motion planning the viewpoint is deemed unreachable and filtered out. After these steps we are left with a smaller set of viewpoints. During the trajectory planning the travel times between every pair of viewpoints is calculated and during the visibility filtering a visibility matrix is constructed which is a matrix of ones and zeros, one dimension corresponds to the number of viewpoints and the other the number of mesh triangles of the CAD model, a one in the matrix corresponds to a particular mesh triangle being visible from a particular viewpoint. Next a Monte Carlo Tree Search(MCTS) is used to find a subset of the selected viewpoints and a sequence for these viewpoints in order to maximise the visible coverage of the part surface while minimising total travel time.

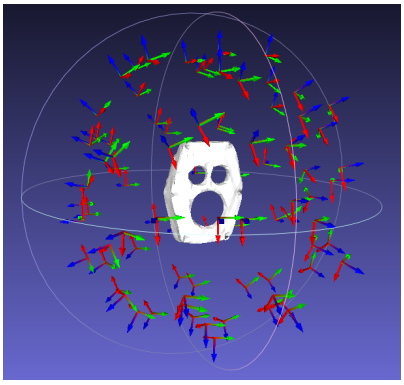

Figure 2: Randomly sampled viewpoints on a sphere

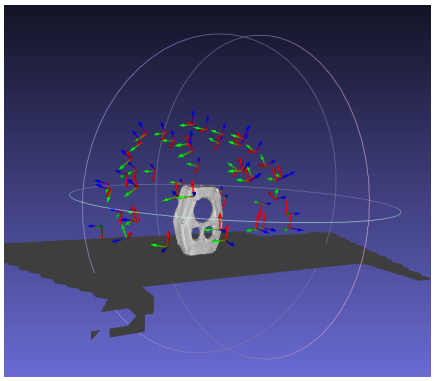

Figure 3: Viewpoints remaining after filtering based on reach-ability and visibility

3 Background reading

In order to prepare for the research task I had a look at the existing literature relevant to the topic in order to better understand the problem, the approaches that have been explored already and most importantly the limitations with them. I started with reading into what constitutes a good viewpoint, traditional methods to solve the problem use heuristics which give some measure about the 'goodness' of a viewpoint for the algorithms to optimise over and use in their search. Developing an understanding of this topic is also valuable for the reinforcement learning based approach because the reward function will need to be shaped in a way that meets the true objective for the inspection criteria. More specifically, two views can capture the same number of polygons from a surface mesh of an object but one can still be 'better' than the other since capturing a less skew image of the surface would lead to better surface inspection subsequent to the image capture. Another example is that more detailed features may be more prone to damage and hence

these should be inspected preferentially and so views that capture these details are now 'better' viewpoints.

[1] served as a good summary of the existing ways to quantify the quality of viewpoints, the survey explores these measures in the context of object recognition however the concepts could be used to draw inspiration from. The most important and relevant ideas from the paper are presented followed by a critical assessment of the approach in the context of part inspection. The first measure is the number of mesh triangles visible from a viewpoint, the advantages of this approach are that meshes that are generated non-uniformly tend to have more triangles to model more detailed parts of the object hence more triangles could correlate to capturing more detailed and descriptive regions. The next is projected area, taking the area of the 2D projection of the model onto the viewpoint, this is probably a strong metric for the inspection task since it is desirable to have non-skew views of a surface to detect defaults from the image. A combined metric including both the number of triangles and projected area is also proposed which could incorporate the benefits of both when tuned properly.

The next measure is based on information theoretic concepts. The viewpoint quality measure is given by:

$$H(v) = -\sum_{z \in \mathcal{Z}} \frac{a_z(v)}{a_t(v)} \log(\frac{a_z(v)}{a_t(v)})$$

where v is a viewpoint, $\mathcal{Z}$ is the set of all polygons that make up the mesh, $a_z(v)$ is the projected area of polygon z from viewpoint v and $a_t(v)$ is the projected area of the model from viewpoint v. This metric captures the amount of information provided by a single viewpoint, hence the viewpoint with maximum information is the one in which all faces have the same projected area. While this metric might be great in terms of gaining a viewpoint that provides visibility of all the mesh polygons somewhat fairly it may not be the best approach for part inspection for a few reasons. Areas represented by larger polygons will tend to be viewed from angles far from normal incidence for views with larger entropy, this is not desirable for inspecting the surface. Also we would much rather take multiple images and have better views of the surfaces given by multiple images rather than try and capture as much as possible about all of the surfaces in one image and have slightly worse views of all the surfaces. A measure that mitigates this and is virtually unaffected by the way the mesh is formed is the viewpoint Kullback Leibler distance(VKL) given by:

$$VKL(v) = \sum_{z \in \mathcal{Z}} \frac{a_z(v)}{a_t(v)} \log(\frac{\frac{a_z(v)}{a_t(v)}}{\frac{A_z}{A_t}})$$

where A_z is the area of polygon z and A_t is the surface area of the model. Minimising this corresponds to a viewpoint which has a projected area distribution which is equal to the actual area distribution. This has the advantage of making sure to not encourage skew views of larger polygons.

The paper also presents some measures based on silhouette lengths of an object from an object but this class of measures didn't seem as relevant to the inspection task. Next the paper presents some measures based on depth of polygons from the viewpoint, these were more for viewpoints of terrains since the view with largest projected area is from above, this is less relevant for object inspection. After this some measures based on view stability are given, the stability of viewpoint refers to how similar views are to neighbouring viewpoints. The most stable viewpoints correspond to viewpoints with the largest number of similar views. This class of measures could be useful since unstable views are desirable in some sense for the inspection task since the robot arm is able to capture two different views without moving too far, so the robot arm should visit unstable regions and take images within these.

Next I looked into the existing literature on applying reinforcement learning to the task. There were two papers that looked at applying reinforcement learning methods to view planning. [6] tackles the view planning problem and is only concerned with obtaining a set of view poses using reinforcement learning methods. This work solely concentrates on obtaining a good set of viewpoints and pays no attention to optimising for other criteria such as inspection cycle times or controller effort. They explore using RL algorithms for both discrete and continuous action spaces. The formulate the inspection task as an RL problem in the following way. The action space consists of picking the next view pose. The observation is made up of the view point pose, the cumulative point cloud formed of the scene so far and the information gain. This observation space however is too large so they only use the viewpoint pose as the state, this however violates

the Markov property that is required. The state space is non-Markovian meaning the state does not encode the information of all the has occurred up to that state, hence the robot does not know if it has visited a view pose already given its current state. Appending the cumulative point cloud would encode this information however the dimentionality is too large, condensed representations of the point cloud could be a viable way to incorporate this information into the state.The reward is based on the extra surface area scanned as a result of the viewpoint. The point cloud is used to calculate surface areas for the reward. They investigate using Q-learning, DQN and PPO. Important to note that they needed to fix some of the degrees of freedom for the PPO since the continuous action space would otherwise be too large. Ultimately the PPO did perform worse than DQN for small numbers of viewpoints but started to close the gap for larger number of viewpoints, the restricted view pose space could be the reason. They choose an episode to be a fixed number of viewpoints, this could be useful in a different formulation where you can end an episode early if you meet a certain coverage threshold or reach the max number of viewpoints to prevent the bot from searching indefinitely in the continuous action space setting. While this paper makes some progress towards applying RL to the problem it is far from an ideal solution, but presents some ideas which could be useful.

The other paper, A reinforcement learning approach to the view planning problem [5] also only considers the view planning problem and not the path planning to generate a sequence and hence it also does not deal with generating a optimal sequence of viewpoints with regards to additional optimisation objectives and instead just a set with regards to coverage. Acting greedily in terms of the fraction of coverage provided by a viewpoint is able to provide good coverage in most situations. The true difficulty of the problem lies in selecting views that are able to provide new information even if they do not provide large coverage in order to provide maximum coverage across the set of views. The idea behind the method proposed by this paper is to selectively act greedily, so that at the start you can just take the views with the largest coverage but then choose the subsequent views more intelligently. They provide a a family of functions to act greedily with respect to:

$$f_\lambda(X) = \frac{\mathcal{A}(X)}{\mathcal{L}^\lambda(X)}$$

where $\mathcal{A}(X)$ denotes the total surface area covered by the submesh X, $\mathcal{L}(X)$ denotes the total boundary length of the area covered by X. The RL formulation differs greatly to the previous paper, here they aim to limit the size of the action space and not have it depend on the number of view poses. So instead of an action corresponding to a view pose they give the action space as a discrete setting of the parameter λ for the function above. The next viewpoint is then selected to be the one which maximises the above function with the particular value of lambda picked at that time step. This way it doesn't always act greedily with respect to the area visible from a particular viewpoint. We can understand the behaviour of picking viewpoints greedily for settings of $\lambda > 0$ to be the poses which correspond to the tightest packing of visible area. Finally they run the algorithm which picks viewpoints in the described manner until a threshold coverage is met. The reward function is -1 every additional viewpoint, hence maximising the reward corresponds to minimising the number of viewpoints used.

So far the approaches have only focused on the view planning problem and not the joint optimisation of the view planning and the travelling salesman problem. [4] takes a step towards solving the joint optimisation problem. The approach used in this paper uses a MCTS in order to tackle the joint optimisation. The MDP formulation of the problem presented in this paper consists of cost function made up of inspection cost and a travelling cost, the action space is selecting a viewpoint. The state space is made up of the current robot pose and a vector of the coverage of the part so far. This is the current implemented work.

4 The initial approach

In order to make improvements over the current pipeline I was tasked with trying to collapse the pipeline down into something that worked end to end and didn't consist of several individual processes. As a first step I worked on using a supervised learning approach to filter the viewpoints based on visibility and reach-ability. The current method requires time consuming path planning between each of the viewpoints in the set to then use in the MCTS, the idea is to have a neural network that is able to filter out the unreachable points to minimise the path planning required for the MCTS. This takes a step towards collapsing down the individual stages and potentially also be a stepping stone towards a neural network that is able to provide some value function for viewpoints for an RL algorithm to use as heuristic during the training. The approach was to

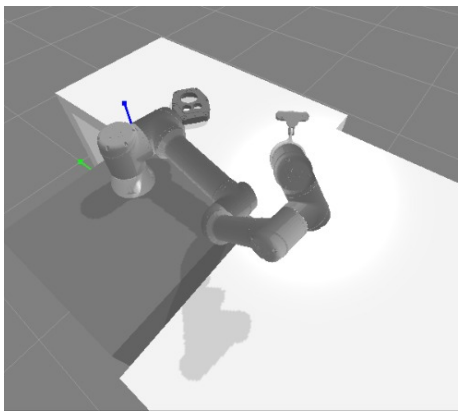

Figure 4: Capturing image of scene in PyBullet

have the robot take an image of the scene as shown in 4 and use this image of the scene together with a particular view pose as input and then the classifier outputs a binary classification for this viewpoint, the viewpoint is classified as 1 is it is both reachable and there is some visibility of the object from the given viewpoint.

In order to train the model the first step was to build a training data set. The existing code base for the part inspection project consisted of various scripts to run the different parts of the pipeline. This consisted of scripts that would take the object for a given pose scan the scene with and without the object to form meshes that would be used for later scripts, as shown in Figures 5 and 6. The next parts would then compute visibility matrices and the path time matrices between viewpoints and then run a MCTS. To build the data-set the process of randomising the object in the scene and then running the pipeline needed to be atomised. The pipeline would generate a sphere of viewpoint and then filter them based on the visibility and reach-ability matrices computed above and then would append each viewpoint to the data set with the classification label. Additionally for the classifier an image for the scene is necessary so an identical scene needed to be setup in PyBullet and then a photo taken for the data point. The existing code base consisted of Python 2 and 3 scripts and so I made use of sub processes to automate the whole procedure and made sure to route the information from one process to the next.

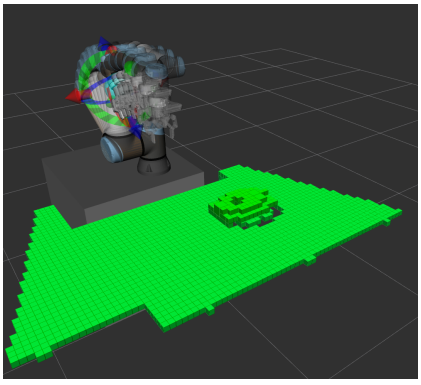

Figure 5: Scene scanned for mesh creation

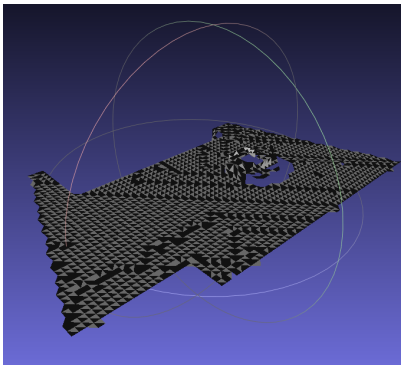

Figure 6: Scene mesh

I also put together a script to perform augmentations on the images of the scene to increase the robustness of the neural network. The aim was to increase the realism and allow the network to remain robustness to irrelevant variations in the image such as colours. The augmentations included changing colours of the object and table, having a variety of backgrounds in the scene and textures for the table by using segments of the image. An example can be seen in Figures 7 and 8. Note these images are from later work where a box is also included into the scene, the purpose of which is discussed later.

Next I set out to develop the neural network to carry out the classification task. Since it was not possible to generate a large enough data set for training a neural network from scratch it made

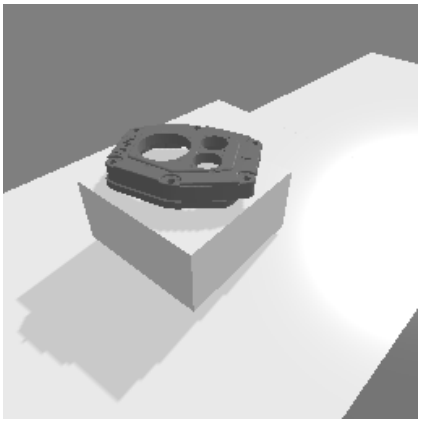

Figure 7: Grayscale image

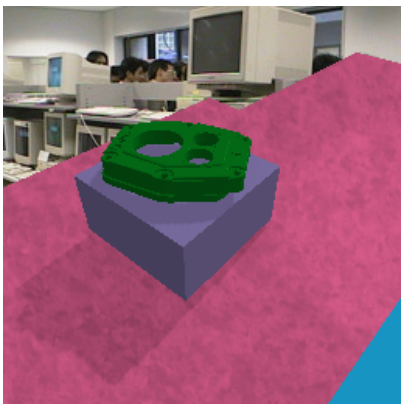

Figure 8: Image with augmentations applied

sense to look into a transfer learning approach. The high level approach was to extract features from the image and use these features to determine if a viewpoint pose would be good or not. Since this is fundamentally a spatial reasoning task I felt that feature vector should encode the spatial information of the objects in the scene. For the transfer learning I decided to experiment with both features extracted from pretrained classification networks on torch hub such as ResNet18 and also object detection networks such as Faster RCNN [8]and DETR [2] . In order to understand where would be the most appropriate place to extract features from these pre-trained networks I decided to research to get a deeper understanding of these networks. Just experimenting on the images with the pretrained networks shows that the object detection networks are in fact able to pick up the object as shown in figure 10 The features extracted from the network are then concatenated with the view pose and the pose of the object. In order to make sure the image feature didn't dominate the pose of the object and view point positional embeddings are used for these in the same way as poses for NeRFs [7].

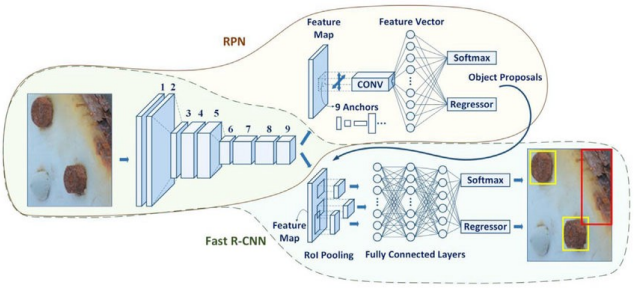

Figure 9: Faster-RCCN

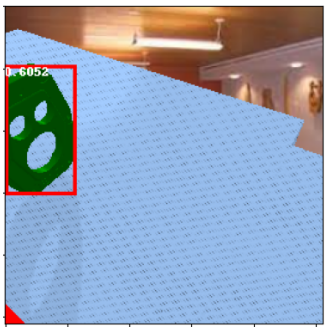

Figure 10: Object detection applied to scene images

5 A Critical Analysis

After some experimentation with the network architecture, using a pretrained Resnet18 as feature extractor the network was achieving 88% test accuracy on the training data. This seemed quite promising but it was important to do a more thorough analysis. To carry out more thorough testing I setup some simulations to create more complicated scenes and a harder task for the classifier. In order to do this I introduced a box into the scene, as shown in 7, this would act as an obstacle which would affect visibility of the object and reach ability of the viewpoints. The scenes were created by dropping the object into a scene in a random location on the table to create a more complex randomised scene such as that shown in 11. As an initial test running the network trained on the simple data on the more complex data yielded 70% accuracy. This prompted me to do a more thorough investigation. I decided to remove components of the input to the model and retrain the model and see how vital each was to the classifier. In fact removing the image didn't affect the classification performance too much but removing the object pose from the input had a

detrimental affect. If the object pose is removed the classifier is not able to do much better than 50 % test accuracy. It was clear that the model was learning simple relationships in the data and exploiting these rather than actually generalising to the true task. When viewpoints are generated they are generated on a sphere around the object as shown in figure 2 this means that a large fraction by default are underneath the table the case is placed on, also there is a rough extent to where the robot arm is able to reach which would also cut off a large fraction of viewpoints is the case is placed too far or too close to the robot arm. Since these patterns occur in the data the neural network is perhaps able to create simple boundaries based on the plane of the table and distance from the arm that would lead to a high classification accuracy.

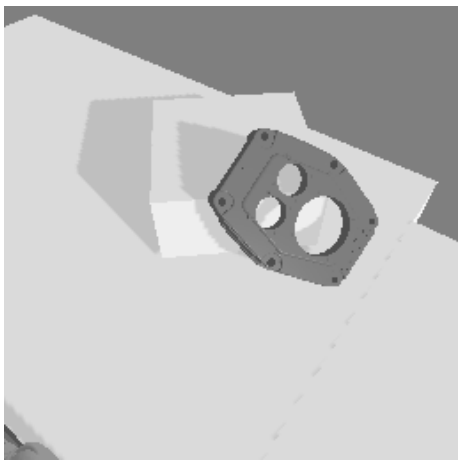

Figure 11: Complex scene

At this point I decided to take a step back to decide how to proceed at this point. It was clear that in order to obtain any meaningful results a more robust data set would need to be generated and it would also need to be conducive to train the network to learn the task. The true goal of the network is based on an image of the scene alone to be able to reason about if a viewpoint is reachable by the robot arm and the object is visible from this viewpoint all without the true object pose as this is not known to high accuracy. Both of these require an understanding of the 3D spatial relations of the items in the scene to decide if a trajectory to a point is possible without object or obstacle collision and if the viewpoint provides a view of the object without obstacle occlusion. This always seemed like too demanding a task with only a 2D image of the scene. The camera used is in fact a stereo camera and so some level of 3D information is ex-tractable. I explored making use of the depth image in the classifier also. Here the problem still remains about not have sufficient data to train a network from scratch, unfortunately not many pre-trained networks exist for depth images. One approach is use HHA encoding which takes a depth image and turns it into an RGB image with the three channels representing encodes height above ground, angle with gravity for each pixel and horizontal disparity. Using this as an extra feature did not lead to much improvement, I suspect these features are not fundamentally helpful for this task.

6 Alternative approaches to explore

We really need a meaningful way to capture the scene, encode this into a feature space which the neural network is then able to create complex decision boundaries that somehow separate viewpoints from which the object is visible and that a trajectory exists to the viewpoint. While it is desirable to collapse the pipeline a more sensible approach is to keep these tasks separate. For the visibility task a potentially promising avenue to explore is pixelNeRF [10]. NeRFs go a long way towards having neural network with 3D scene representation capabilities which is what is required in the context of this problem also. Traditional NeRFs require many views of a scene in order to train a good network, this is an issue since we seek an approach that is adaptable to new environments fast and if many images are required to train the NeRF we would already have completed the inspection task to some extent anyways. pixelNeRF however goes some way in addressing this issue, instead of being trained solely on scene it is trained on a variety of scenes and then is able to produce novel images of a scene conditioned on a few images of the scene, even as few as one. This is a potential avenue to explore to filter the viewpoints based on visibility by

using the pixelNeRF to generate an image of the scene from a viewpoint conditioned on the image taken of the scene. This is provides an advantage over the current ray tracing method since the true pose of the object is not needed. The problem however is the quality of the views that are produced and also calculating how much of the surface is actually visible from pixelNeRF rendered image. For reach-ability classifications we need to replace the time consuming bidirectional RRT with a network that is able to determine if a given pose is reachable and a path is possible to it from the current location using the robot arm. There have been some prior works tackling this problem due to the computational and time savings this approach would provide. There is a survey paper titled neural networks for path planning [3] which describes a few methods to apply neural networks to motion planning. One of the approaches presented is Neural RRT*(NRRT*) which uses a neural network to generate the distribution of sampling non-uniformly to speed up converge and it has actually been shown to increase the optimality of the paths generated. It is possible to perhaps apply this to the current pipeline to improve speed and perhaps even the path time.

7 Reinforcement learning approach

The goal of the project is to devise a method to allow the part inspection task to be carried out in a complex environment while maximising performance in terms of coverage and cycle time without require time consuming computation, with also the ability to handle for object pose uncertainty. The reinforcement learning approach can be used to jointly optimise for the objectives and can be trained in a range of complex environments to learn to avoid obstacle collision. The advantage of this approach is that at use time the network is able to propose actions without any computation needed and hence can be used instantly at inference time, this is not possible with the Monte Carlo approach which needs to compute the pair wise trajectories between viewpoints. As a first step I have set up the training pipeline for a PPO RL agent and currently the agent is trained to generate a path between a discrete set of viewpoints while optimising for time and coverage. For training on one specific scene setup and object pose the RL agent was able to generate a faster path which provided maximum coverage than the Monte Carlo method. The next step is to increase the training to handle any arbitrary object pose but for a fixed scene. In order to do this the state must now include the pose of the object. One of the aims was to make the method robust to inaccurate poses of the object, one way to tackle this is to estimate the pose of the object from each image of the scene captured and refine this estimate as more viewpoint images are taken throughout the inspection. Once the system is able to handle this the next step is to train the agent for an arbitrary object pose in a arbitrary scene configuration. The key complexity here is to find a state space representation to encode the 3D scene configuration and object pose. One option is to generate a 3D point cloud of the scene and use auto encoding methods as suggested by [6]. Another avenue I spent some time looking into an exploring is Meta-Learning where instead you have the object pose and scene be the variations of tasks to meta-learn over. An example of this is in [9], this paper demonstrates using an RL agent for a family of insertion tasks and then adapt to a new environment with few trials.

References

[1] Xavier Bonaventura, Miquel Feixas, Mateu Sbert, Lewis Chuang, and Christian Wallraven. A survey of viewpoint selection methods for polygonal models. *Entropy*, 20(5), 2018.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[3] Salim Janji and Adrian Kliks. Neural networks for path planning, 2022.

[4] Wei Jing, Chun Fan Goh, Mabaran Rajaraman, Fei Gao, Sooho Park, Yong Liu, and Kenji Shimada. A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning. *IEEE Access*, 6:54854–54864, 2018.

[5] Mustafa Devrim Kaba, Mustafa Gokhan Uzunbas, and Ser Nam Lim. A reinforcement learning approach to the view planning problem, 2016.

[6] Christian Landgraf, Bernd Meese, Michael Pabst, Georg Martius, and Marco Huber. A reinforcement learning approach to view planning for automated inspection tasks. *Sensors*, 21:2030, 03 2021.

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[9] Gerrit Schoettler, Ashvin Nair, Juan Aparicio Ojea, Sergey Levine, and Eugen Solowjow. Meta-reinforcement learning for robotic industrial insertion tasks, 2020.

[10] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020.