Deep Reinforcement Learning for Next-Best-View Planning in Agricultural Applications

Xiangyu Zeng Tobias Zaenker Maren Bennewitz

***Abstract*— Automated agricultural applications, i.e., fruit picking require spatial information about crops and, especially, their fruits. In this paper, we present a novel deep reinforcement learning (DRL) approach to determine the next best view for automatic exploration of 3D environments with a robotic arm equipped with an RGB-D camera. We process the obtained images into an octree with labeled regions of interest (ROIs), i.e., fruits. We use this octree to generate 3D observation maps that serve as encoded input to the DRL network. We hereby do not only rely on known information about the environment, but explicitly also represent information about the unknown space to force exploration. Our network takes as input the encoded 3D observation map and the temporal sequence of camera view pose changes, and outputs the most promising camera movement direction. Our experimental results show an improved ROI targeted exploration performance resulting from our learned network in comparison to a state-of-the-art method.**

I. Introduction

With the development of machine intelligence, mobile robots or robotic arms equipped with sensors can be deployed on farms for advanced agricultural applications, such as fruit picking and targeted crop spraying. Yet in practical applications, the regions of interest (ROIs), such as fruits, are possibly completely or partly occluded by leaves. While mature technology has been developed to infer the shape or reconstruct it from multi-view observations captured from fixed locations in the field of computer vision [1]–[3], these techniques can still fail in complex scenarios, where ROIs can be occluded from all fixed views. As already stated by Jayaraman and Grauman, visual perception should not only do inference from fixed observations but also make decisions about what to observe [4].

In this paper, we follow a similar line of research and present an exploration approach for 3D environments that changes the camera viewpoint to capture new observations that improve the information about the ROIs. Typical next-best-view (NBV) planners [5]–[12] work in two steps: First, they generate some candidate view poses, and, second they choose the NBV pose according to modeled information gain. These methods have proven to be effective, however, they have two general drawbacks: (1) Handcrafted metrics need to be designed to compute the information gain, which is often troublesome and difficult to migrate, and, (2) conventional viewpoint planning aims at reconstructing the environment or objects with the objective to improve the overall coverage rate. However, some applications, e.g., fruit size and position estimation, only require knowledge about certain regions of interest without the need for complete coverage of crops.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2070 – 390732324. All authors are with the University of Bonn, Germany.

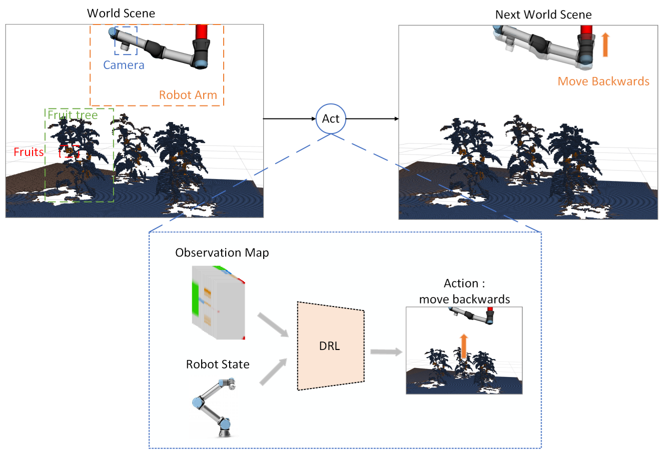


Fig. 1: The goal of our deep reinforcement learning model is to learn a strategy for effective exploration in a 3D scence containing certain regions of interest, e.g., clusters of fruit cells. Our model takes as input an encoded 3D observation map and a sequence of camera view pose changes, and outputs a camera movement.

Recently, deep reinforcement learning (DRL) based approaches have also been proposed for exploration. Some of them require high-dimensional observations to be handcrafted [13], others assume prior knowledge about the global environment to be available [14]. Motivated by these, we propose a DRL-based exploration approach to decide on the next best view in a 3D environment space. To represent the observations acquired with an RGB-D sensor, we construct an octree of the crops with labeled ROIs, i.e., fruit cells. For the encoded input to the DRL network, we do not only rely on known information about the environment, but also on information about the unknown space to guide the exploration. Our network takes as input an encoded 3D observation map, the temporal sequence of camera view pose pose changes, and outputs the action that should be taken (see Fig. 1). In our experiments, we evaluated our learned network in two simulated environments with crops using a robot arm equipped with an RGB-D camera. The results show an improved performance on multiple metrics that impact the subsequent agricultural manipulations actions, e.g., fruit picking or targeted spraying, compared to the state of the art in ROI targeted exploration [15].

To summarize, the contribution of our work is a novel viewpoint planning approach for 3D environments based on DRL, which aims at detecting and covering as many as possible regions of interests in a given time limit.

II. Related Work

Traditional exploration approaches rely on various information gain metrics to select the next best view, usually applying ray casting. The metrics are often designed based on statistics using the number of unknown voxels [12], information entropy [9], [10], or additional constraints or trade offs related to occlusion, safety, or movement cost [8], [11], [16]. Delmerico *et al.* [17] proposed and evaluated different metrics to estimate the volumetric information gain and to calculate the utility of views, taking also into account the move costs, to determine the best next view. Furthermore, Palazzolo *et al.* [7] presented an efficient exploration approach based on the cost of reaching a new viewpoint and information gain. The authors adapt a hull to fit the explored building or object to select candidate viewpoints. However, such an approximation can perform less efficient, e.g., on L-shaped objects. Monica *et al.* [6] presented a NBV method that samples viewpoints from frontiers to unknown space to observe the border of incomplete objects. All those NBV methods heavily rely on carefully designed, hand-crafted metrics and are often task- or environment-specific. Thus, they are not able to easily adapt to different environments. Most of those approaches aim at improving the coverage rate of an entire environment, and are not oriented towards special regions of interest. In contrast, our work based on DRL is able to automatically learn the best exploration strategy by taking the poses that lead to higher rewards.

Recently, Hepp *et al.* [18] proposed to learn information utility functions by using deep learning to determine the next best view for 3D scene exploration. Furthermore, Wang *et al.* [5] designed a convolutional neural network to maximize the information gain and predict a direction for the next best camera move. This approach checks the resulting volumetric gain by moving the current camera pose to six directions and chooses the direction with maximum gain as the motion label. Both are supervised learning methods that require environment models or a synthesized scene in advance to generate training data.

Viewpoint planning has also been studied in the agricultural field. Bulanon *et al.* [19] analyzed the visibility of citrus fruits with different sets of fixed camera configurations. Hemming *et al.* [20] compared the average fruit detectability of pepper plants from multiple camera angles. Both these approaches perform an analysis on viewpoint effects, but do not provide an active planning strategy. Kurtser and Edan [21] suggested an approach to select the next best viewpoint based on analyzing the fruit occlusion in the current image. Sukkar *et al.* [22] proposed to evaluate viewpoints based on a weighted sum of the number of visible voxels that have not been previously explored and the visibility of ROIs and used this evaluation metric to plan a sequence of viewpoints for multiple robot arms in an integrated path planner. In our previous work [15], we presented a viewpoint planning approach in unknown environments, which chooses viewpoint candidates that increase the information gain around the fruit regions and evaluates them using the expected information gain. As shown, the position and size of the detected fruits can be estimated after sufficiently covering the crops. This work was subsequently combined with the local 3D Move-to-See approach [23] to include additional occlusion avoidance [24]. In the experimental section, we provide a comparison to our previous work.

Some DRL-based algorithms have already been proposed to address exploration problems. For example, Tai and Liu [25] developed a method to explore room or corridor environments that do not contain cluttered objects. As a result, a mobile robot traveled collission-free in the environment using the depth images from an RGB-D camera as input to the Q-network. Sampedro *et al.* [14] combined a classifier trained for target classification and a DRL-based method to perform image-based visual servoing tasks. A model-based object recognizer is utilized to obtain target ROIs in the environment and the positional error to these targets is then used to train velocity commands for the robots. The approach of Niroui *et al.* [13] was designed for 2D input data and computes the location of frontier cells and the 2D occupancy grid of the explored environment, and utilizes DRL to learn actions that allow a robot to autonomously explore unknown cluttered environments in urban search and rescue applications. The observations from the 2D world are more suitable as the input to the deep neural network, while from 3D observations it is more difficult to obtain a simple and effective representation as input to the network. Jayaraman and Grauman [4] proposed a learning based approach that processes input views to predict the observation for viewpoints stored in a viewgrid. Viewpoints are selected to improve the predicted views, which can be applied for scene or object shape completion.

III. Problem Statement and Formulation

Our goal is to learn an exploration strategy that can effectively explore a 3D environment while prioritizing detecting and covering more relevant regions of interests in given time steps. We developed a deep reinforcement learning framework that takes as input an encoded 3D observation map and a sequence of camera view pose changes. The output of our model is an action in 3D space such as moving forward or pitching up.

We denote the robot state at time step t as s_t. At each time step t, the robot performs an action a_t, from which we get the next robot state s_{t+1}, the next observation, for which a 3D observation map m_{t+1} is encoded, and the reward r_t. Formally, the objective of our framework is to learn a policy function π, which takes the two inputs, i.e., the time-series state change Δs_t,...,$\Delta s_{t-|N|+1}$ in the past $|N|$ time steps and the encoded observation map m_t, and produces a probability distribution $\pi(a_t|\Delta s_t, ..., \Delta s_{t-|N|+1}, m_t)$ over the action space A_t. The optimal policy is the one which maximizes the state-action value function $Q(\Delta s_t, ..., \Delta s_{t-|N|+1}, m_t, a_t)$ using the reward.

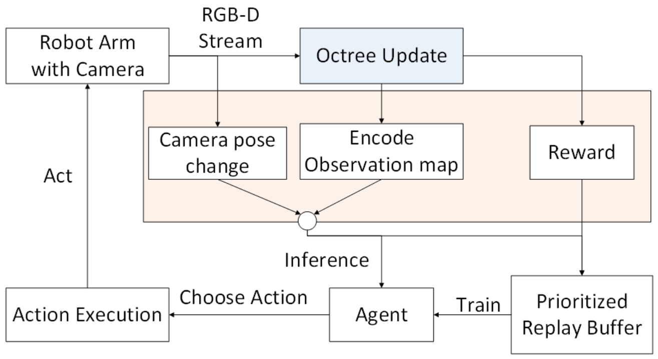


Fig. 2: Illustration of our learning framework. Based on the RGB-D stream, the octree is updated, which enables the computation of the 3D observation map and reward. The encoded observation map and the camera view pose changes in the last N time steps are taken by agent to do inference, and are also stored in a prioritized replay buffer with the reward as an experience for training. In the navigation module, we use MoveIt [26] to plan the arm motions to the next viewpoint corresponding to the chosen action, which is then executed.

IV. Deep Learning for 3D Exploration

In this section, we first describe our learning framework. Then we detail our world representation, the learning setup including the state representation and action space, the network structure used for training, and the reward design.

A. Framework

Deep Q-learning [27] was proposed to solve the difficulty of traditional Q-learning methods in maintaining a large-scale state set. It approximates the value function and uses the experience replay buffer technique to update the targeted Q value. Double deep Q network (DDQN) learning [28] eliminates the overestimation problem by decoupling the current action selection and the estimation of the future maximum approximated action value. Prioritized Experience Replay DQN [29] calculates the experience weights based on the absolute value of the temporal difference (TD) error of each experience sample, and stores their priorities in the experience replay buffer.

Our proposed DRL-based exploration framework is presented in Fig. 2. At each time step, the robot executes the current action a_t and moves to the next pose s_{t+1}. The new robot state s_{t+1} can be computed under the assumption that the robotic arm performs the action accurately. The RGB-D camera attached to the robotic arm generates the RGB-D stream, which we use to update the octree to store the newly obtained information about occupied, free, and ROI voxels in the limited field of view (FOV). Then, the reward r_t is calculated based on newly detected ROI and free voxels and the next observation m_{t+1} is encoded.

Our agent uses a Double DQN to predict the expected value of each action and generate the next action a_{t+1}, taking as input a sequence of camera view pose changes and the current observation. Meanwhile, the robot state Δs_{t+1} and observation map m_{t+1} are stored in the Prioritized Replay Buffer, as well as the reward r_t. The agent chooses the action, a motion planning module based on MoveIt [26] plans the corresponding arm motion, and the robot arm executes it.

B. World Representation

Our world representation maintains the information of the environment in form of an octree generated from the data of an RGB-D camera. The voxel values are categorized into free, occupied, ROIs, and unknown cells.

The camera has a limited frustum-shaped field of view, with horizontal angle range ψ_{fov}, vertical angle range θ_{fov} and distance range d_{fov}. When the camera moves and obtains a new observation, the voxels in the field of view are updated based on the new observations.

To do that, rays are cast from the camera origin to the points extracted from the depth image. The occupancy probability of hit voxels is updated to be more likely free, while the probability of the voxels corresponding to the points are updated to be more likely occupied. To obtain the target voxels for fruit coverage, fruits can be detected as ROIs either with color thresholding or an object detection network. For our simulated scenarios we used color thresholding to filter the red pixels, since the used plants contain only red peppers. Details on creating such an octree can be found in [15].

C. Learning Setup

The input to the network consists of the temporal relative camera pose in the last N time steps and an observation map at time step t as described in detail in the following.

1) State Representation: We indicate the state $\boldsymbol{s_t}$ of the robot as the position and viewing direction of the RGB-D camera attached to the robot arm:

$$\boldsymbol{s_t} = [x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t]$$

where $[x_t, y_t, z_t]$ is the position $\boldsymbol{p_t}$ and $[\alpha_t, \beta_t, \gamma_t]$ is the view direction $\boldsymbol{r_t}$ of the camera at time step t respectively.

The relative camera movement of two consecutive time steps is defined as:

$$\boldsymbol{\Delta s_t} = \boldsymbol{s_t} - \boldsymbol{s_{t-1}}$$

If necessary, the action is adapted according to the space boundaries, dynamics limitations, and collision with occupied cells (fruits and leaves) in the environments.

2) Observation Map: When the robot arm moves to the next viewpoint, it will obtain a new observation, from which we construct an observation map. The complex 3D data provided by the octree is not suitable as input to the neural network, especially for unsupervised learning. Therefore, it is necessary to design a simpler representation. We generate an egocentric observation map m_t based on the octree. Originating from the camera, we cast rays up to a distance d_{obs} through the voxels, with an azimuthal angle step $\Delta\psi_{obs}$ and a polar angle step $\Delta\theta_{obs}$, and divide the ray into $|D|$ segments in depth to encode distance information. We then count the number of unknown, free, occupied and ROI voxels along the segmented rays. Then, we construct an observation

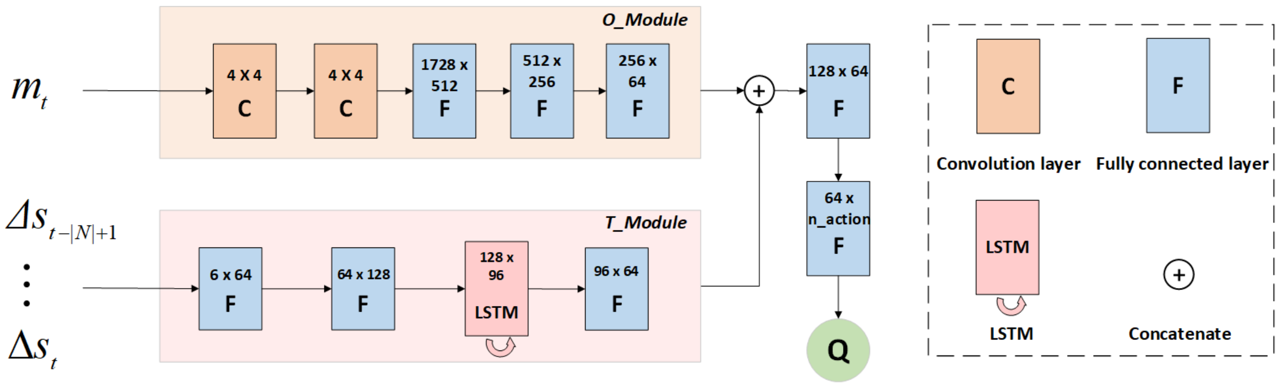


Fig. 3: Illustration of our network architecture for training the DDQN. Details are described in Sec. IV-D.

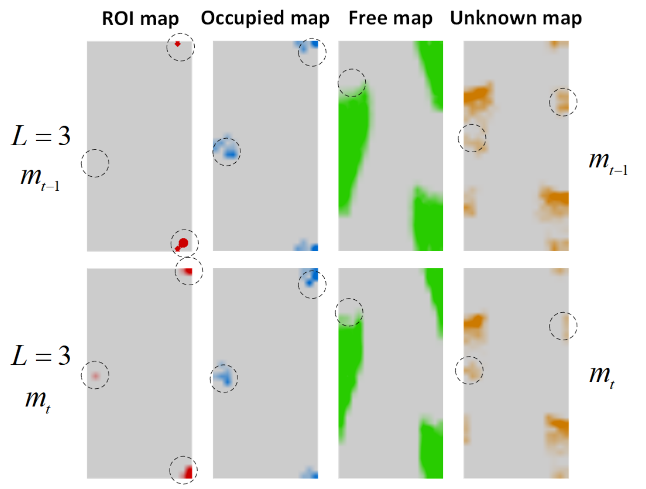


Fig. 4: Example of two consecutive observation maps at depth $L = 3$. The red pixels indicate ROI voxels in that area, with saturation corresponding to the count, yellow means unknown voxels, green corresponds to free voxels, and blue indicates occupied voxels. This observation map is the one with the highest reward from one of the episodes. Note the change between the observation maps of two time steps $t-1$ and t, where m_t has more red pixels compared to m_{t-1}. If a cell has no value for any count, it lies outside of the camera's workspace.

map of size $|W| \times |H| \times |D| \times 4$, where $|W| = \frac{2\pi}{\Delta\psi_{obs}}$, $|H| = \frac{\pi}{\Delta\theta_{obs}}$ (see Fig. 4).

3) Actions: The agent has six categories of translations, i.e., move forward / backward, move left / right, move up / down and four categories of rotations, i.e, pitch up / down, yaw left / right, thus, the action space size is 10. The rolling rotation is omitted since it does not change the field of view for camera much.

4) Reward: We aim to improve the exploration ability prioritizing looking for ROI cells appearing in clusters in the initially unknown environment. The reward is therefore given by weighting the number of newly found ROI voxels and newly found occupied voxels, and additionally rewarding the coverage of the unknown area:

$$R = \alpha * N_{roi} + \beta * N_{occ} + \gamma * N_{visit}$$

where N_{roi} and N_{occ} are the number of newly observed ROI voxels and occupied voxels, respectively, and α, β, γ are weighting factors.

Rewarding the coverage of unknown area encourages the general exploration. A coverage map is stored, which divides the whole environment space into several parts, with $16 \times 16 \times 16$ size for each parts. Once the camera arrived at a new voxel, the voxel and its neighbors are marked as 1 in this coverage map. The reward N_{visit} is given by how many new voxels are marked as 1. The larger the N_{visit} is, the more unfamiliar this part of area is.

D. Network Architecture

As shown in Fig. 3, our network architecture contains multiple modules.

1) O_Module: This module has a convolution layer followed by two fully connected layers. It takes as input the observation map of $|D| \times |W| \times |H| \times 4$, where the depth channel and the voxel category channel are concatenated. In the agriculture application, fruit voxels typically appear in clusters. Thus, it is likely that if a ROI voxel is observed, multiple ROI voxels are around it. The *O_Module* module is able to learn by training that the unknown area is possibly more worthwhile than the known area to visit, and the unknown area near a ROI area is more likely to obtain more rewards. Besides, the observation map contains the information of which parts are unknown, which parts are known, and which parts contain ROI voxels in the known environment. With this input, the module is able to extract effective information for guiding the action selection.

2) T_Module: This module contains a long short-term memory (LSTM) unit which encodes the temporal information of the relative camera movements in the past N time steps. By using relative position and rotation, our network is able to infer where the camera has been recently and where it came from to guide the action selection. It takes as input a sequence of the past $|N|$ camera view pose changes $\mathbf{\Delta s}_{t-|N|+1}, ..., \mathbf{\Delta s}_t$.

All convolution layers and fully connected layers are followed by a ReLU activation function [30] in our network structure. The outputs from these two modules are concatenated to the last two fully connected layers and output the Q values for each action.

V. Experiments

In this section, we first describe our experimental scenarios and setup. Afterwards, we evaluate our learning approach and compare it to a state-of-the-art method on crop exploration using metrics relevant to agricultural applications.

A. Simulation Environments and Training

We designed two simulation environments to train our networks.

1) P3D Environment: The first environment was designed to enable fast training in a randomized scenario to evaluate the general exploration behavior of our approach. It is based on a voxel grid with $400 \times 400 \times 150$ cells, consisting of four crops with fruits as regions of interest. We assume a cell resolution of 1 cm, making the plants approximately 0.85 m high. A visualization of the environment with Panda3D [31] is shown in Fig. 5(a). Within the voxel grid, we simulate the depth readings of a camera, which can be placed at an arbitrary position that does not have to be aligned with the cells. As described in the previous section, we consider ten actions to move the camera. We set the translational step size to a fixed value of 2 cm, and the angular step size to 20°. The horizontal FOV ψ_{fov} of the simulated camera is 80°, the vertical FOV θ_{fov} 60°, the sensor range d_{fov} 300 cm, and the used camera resolution is 80 $\times$ 60. From these values, the visible cells are computed after each camera movement. For this environment, we specified the fruit cells as the ROI cells and the leaf cells as the occupied cells.

We trained our policy in this environment with four crops at random positions on the ground, and a randomly initialized camera position. An episode ends when it reaches the maximum step 300.

2) ROS Environment: The second environment also consists of four crops, two of which have a total of 14 fruits, and a robotic arm with an RGB-D camera at its tip is placed on a pole (see Fig. 5(b)). The environment was designed using Gazebo and a simulated robotic arm with an RGB-D camera in *ROS*. The ten output commands of the neural network are converted into relative pose changes, which are used to move the arm through its MoveIt interface. Here, we used an angular step size of 15°, but increased the translational step size to 10 cm. We used this environment to evaluate our DRL-based approach in comparison to a state-of-the-art method [15] on relevant metrics that impact subsequent fruit picking or targeted spraying.

Here, we trained only in the scenario shown in Fig. 5(b) with no randomization, so the trained model is not generalized. Therefore, this model is suited for repeated coverage of a known environment, e.g., in a glasshouse where the position of the crops stays the same. A model trained on recorded data can be used to get better coverage in these scenarios.

B. Evaluation

1) P3D Environment: We evaluated our models with respect to coverage rate of occupied and ROI cells, averaged over 30 episodes, and compared with two following baselines.

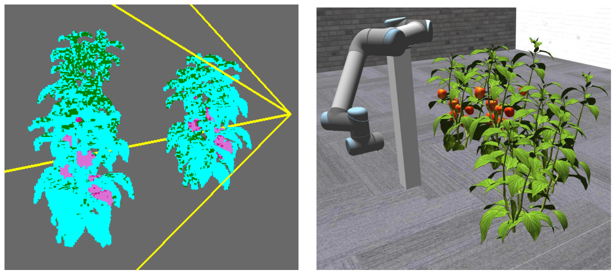
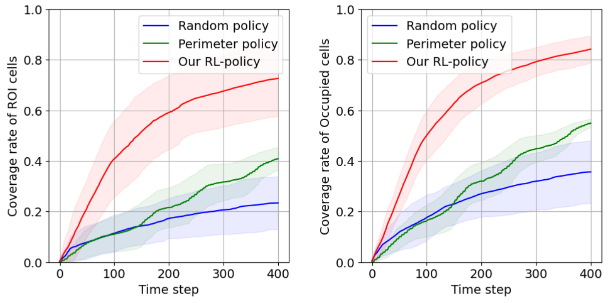

(a) P3D Environment (b) Ros Environment

Fig. 5: Visualization of the two simulation environments, both containing four crops with some fruits (indicated by magenta and red). (a) *P3D Environment* where the yellow frame represents the field of view of a free floating camera. The plant positions are randomized in each episode for both training and evaluation. (b) The *ROS Environment* contains a robot arm mounted on a pole equipped with am RGB-D camera.



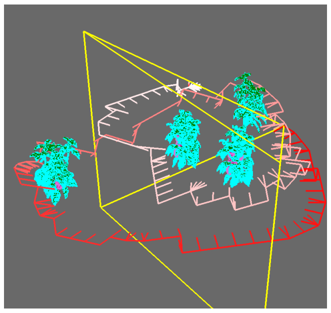
(a) Coverage rate of ROI cells (b) Coverage rate of occupied cells

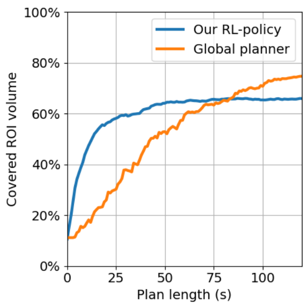
Fig. 6: Average results over 30 episodes within 400 time steps in the P3D Environment. (a) Coverage rate of ROI cells: ratio of observed ROI cells to observable ROI cells. (b) Coverage rate of occupied cells: ratio of found occupied cells to observable occupied cells. As can be seen, our policy found about 73% of the ROI cells and 85% of the occupied cells.

- *Random policy*: The camera starts from the center of the scenario, and randomly executes one of the 10 actions.
- *Perimeter policy*: The camera moves around the complete perimeter of the rectangular space while facing towards the inside. For each shift, it performs two random rotation actions and then rotates back.

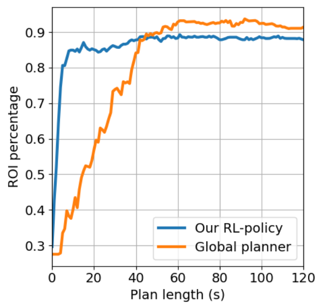
Performance in a randomized environment: As shown in Fig. 6, our policy found about 73% of the ROI cells and 85% of the occupied cells at time step 400 in an environment where the positions of both plants and camera randomly changed for each episode, with an ϵ-greedy policy ($\epsilon = 0.15$). A completely random policy covers only about 22% of the ROI cells and 35% of the occupied cells at that time step, and the perimeter policy reaches about 40% and 55% respectively.

Qualitative Result: Fig. 5(a) visualizes the camera trajectory resulting from the trained policy applied in one of the random environments. The yellow frame line represents the final camera FOV. The camera path starts from white and ends at red. The short lines attached to the path indicate the camera

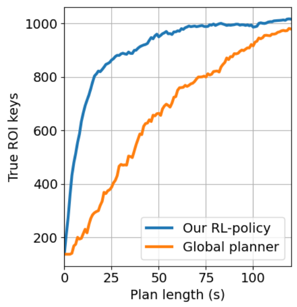

(a) Camera trajectory



(b) Covered ROI volume



(c) ROI percentage



(d) True ROI keys

Fig. 7: (a) Example camera trajectory resulting from our learned policy in a random P3D Environment. The camera path starts at the white spot and ends at red with the indicated field of view. The short attached lines indicate the camera viewing direction. As illustrated, our agent learned to rotate around the crops and keep looking towards them, while also exploring unknown areas. (b-d) show results for the ROS Environment. For each approach, 20 trails were performed with a duration of two minutes. The plots show the average results. As can be seen, our learned model performs better in finding more ROIs in a shorter time compared to our previous approach [15].

viewing directions. As can be seen, the system learned to rotate around the crops and keep looking towards them, while also exploring unknown areas.

Generalization: The trained policy generalizes to scenes outside the training setting. For example, we can apply the policy trained in a *P3D Environment* with 4 crops to one with 2-8 crops in an ϵ-greedy way ($\epsilon = 0.15$), and still receive similar ROI coverage ratios (Tab. I).

	Number of crops			
	2	**4**	6	8
ROI coverage	0.745	**0.726**	0.722	0.675

TABLE I: Generalization with respect to the number of crops in the environment. As can be seen, our policy achieves a high ROI coverage rate also outside the training setting.

2) ROS Environment: In the ROS environment, we evaluated our approach, with a sequence length of 5 for the LSTM layer, against our previous work [15] based on three metrics. This approach performs global viewpoint planning and samples potential viewpoints either at the frontier of partially detected fruits or at more general frontiers for exploration. Then, it computes the utility of the viewpoints taking into account the information gain and movement cost, and moves the camera to the viewpoint with the highest utility. As metrics we used

- *Covered ROI volume*: Percentage of the total volume of the ground truth ROI volume (fruits) that was detected, based on clustering the ROI cells and approximating the clusters with bounding boxes.
- *ROI percentage*: Percentage of ground truth fruits that were found, i.e., that can be matched with found clusters, which means that their center distance is smaller than 20 cm.
- *True ROI keys:* The amount of detected ROI voxels that are also in the ground truth.

Fig. 7 (b-d) illustrate the average performance of three metrics over a planning time of two minutes. ”Global planner” refers to the approach presented in [15]. As can be seen, our trained model takes shorter time to cover more of the ROI volume and discover a higher number of the fruits compared to the global planner. In addition, our network leads to the highest number of true ROI keys. These results show that our learned policy performs better especially at the start of an episode. In the beginning, there is little information about the environment, so traditional approaches like the presented baseline can struggle to find the best moves. Here, using learned models can be beneficial.

VI. Conclusions

In this work, we proposed a novel approach to exploration of 3D environments containing specific regions of interest with a robotic arm. We process the observations of an RGB-D camera into an octree and generate 3D observation maps, which contain information about unknown, free, occupied, and ROIs voxels to guide the search. Our network takes as input the encoded 3D observation map and the temporal sequence of camera view pose changes, and outputs the camera movement, which is then translated into a movement of the robotic arm to obtain the next observation. We trained and evaluated our model in two different simulated environments. The results demonstrate the capabilites of our approach to perform ROI targeted exploration and show an improved exploration performance resulting from our learned network compared to a state-of-the-art method. The source code of our framework as well as the environments are available on GitHub[1].

References

[1] H. Chen, P. Guo, P. Li, G. H. Lee, and G. Chirikjian, “Multi-person 3d pose estimation in crowded scenes based on multi-view geometry,” in *European Conference on Computer Vision*. Springer, 2020, pp. 541–557.

[2] S. Fuhrmann, F. Langguth, and M. Goesele, “Mve-a multi-view reconstruction environment.” in *GCH*. Citeseer, 2014, pp. 11–18.

[3] A. M. Andrew, “Multiple view geometry in computer vision,” *Kybernetes*, 2001.

[1] `https://github.com/zengxyu/vpp-learning`

[4] D. Jayaraman and K. Grauman, "Learning to look around: Intelligently exploring unseen environments for unknown tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1238–1247.

[5] Y. Wang, S. James, E. K. Stathopoulou, C. Beltrán-González, Y. Konishi, and A. Del Bue, "Autonomous 3-d reconstruction, mapping, and exploration of indoor environments with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3340–3347, 2019.

[6] R. Monica and J. Aleotti, "Contour-based next-best view planning from point cloud segmentation of unknown objects," *Autonomous Robots*, vol. 42, no. 2, pp. 443–458, 2018.

[7] E. Palazzolo and C. Stachniss, "Effective exploration for mavs based on the expected information gain," *Drones*, vol. 2, no. 1, p. 9, 2018.

[8] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3d reconstruction," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3477–3484.

[9] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, vol. 10, no. 4, pp. 611–631, 2015.

[10] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 148–164, 2014.

[11] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3d object reconstruction with positioning error," *International Journal of Advanced Robotic Systems*, vol. 11, no. 10, p. 159, 2014.

[12] P. Quin, G. Paul, A. Alempijevic, D. Liu, and G. Dissanayake, "Efficient neighbourhood-based information gain approach for exploration of complex 3d environments," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1343–1348.

[13] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[14] C. Sampedro, A. Rodriguez-Ramos, H. Bavle, A. Carrio, P. de la Puente, and P. Campoy, "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, vol. 95, no. 2, pp. 601–627, 2019.

[15] T. Zaenker, C. Smitt, C. McCool, and M. Bennewitz, "Viewpoint planning for fruit size and position estimation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[16] J. Wettach and K. Berns, "Dynamic frontier based exploration with a mobile indoor robot." in *ISR/ROBOTIK*, 2010, pp. 1–8.

[17] J. Delmerico, S. Isler, R. Sabzevari, and D. Scaramuzza, "A comparison of volumetric information gain metrics for active 3d object reconstruction," *Autonomous Robots*, no. 42, 2018.

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

[19] D. Bulanon, T. Burks, and V. Alchanatis, "Fruit visibility analysis for robotic citrus harvesting," *Transactions of the ASABE*, vol. 52, no. 1, pp. 277–283, 2009.

[20] J. Hemming, J. Ruizendaal, J. W. Hofstee, and E. J. Van Henten, "Fruit detectability analysis for different camera positions in sweet-pepper," *Sensors*, vol. 14, no. 4, pp. 6032–6044, 2014.

[21] P. Kurtser and Y. Edan, "The use of dynamic sensing strategies to improve detection for a pepper harvesting robot," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 8286–8293.

[22] F. Sukkar, G. Best, C. Yoo, and R. Fitch, "Multi-robot region-of-interest reconstruction with Dec-MCTS," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[23] C. Lehnert, D. Tsai, A. Eriksson, and C. McCool, "3d move to see: Multi-perspective visual servoing towards the next best view within unstructured and occluded environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] T. Zaenker, C. Lehnert, C. McCool, and M. Bennewitz, "Combining local and global viewpoint planning for fruit coverage," in *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2021.

[25] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in *IEEE Int. Conf. on Real-Time Computing and Robotics (RCAR)*, 2016.

[26] Moveit.ros.org, https://github.com/ros-planning/moveit2/.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[29] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[30] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[31] M. Goslin and M. R. Mine, "The panda3d graphics engine," *Computer*, vol. 37, no. 10, pp. 112–114, 2004.