# Fully automatic color angle recognition robot

Zeng, Zhuo[1]
ucabz30@ucl.ac.uk

Yuxuan, Dong[1]
ucabyd6@ucl.ac.uk

Bowen, Jiang[1]
zczlbj0@ucl.ac.uk

***Abstract*— As human assistance, the robotic arm must frequently look for and retrieve target things from the clutter of shelves and tables. "Pick and place" is a simple talent for humans, but it is extremely difficult for robots to master. In real life, individuals frequently need to pick up partially visible things. However, reasoning about entire objects and avoiding potential accidents is extremely challenging for robots. In this paper, we simulate a simplified version of the 'Pick and place' environment to test our algorithm, where the tasks are primarily colour recognition and stack creation. Our robot arm can easily perform the recognition and grasping tasks, but the algorithm has some issues due to inaccurate orientation measurements.**

## I. INTRODUCTION

The "Pick and place" challenge for robotic arms has been a source of debate in the scientific community. In simulated scenarios, current algorithms for picking up and putting down things have worked admirably. Many algorithms employ Convolutional neural network(CNNs) to extract picture information and are based on deep learning. There are still some issues to be handled for complicated circumstances, such as picking up items that overlap in a natural environment. Robotic arms are already in use in fruit picking and production lines, and they will soon be in our life as well.

The RGBD camera will be used to scan the surroundings and produce point clouds, which will aid in the identification of colours and angles. The construction function in the PCL library is mostly used to process the point clouds.

In this paper, several approaches to reconstructing cubes in different colours will be introduced via ROS and Panda robot in Ubuntu. In this task, cubes will be randomly generated on a green map, including both single cubes and stacks of cubes in various colours. This challenge has been divided into parts, which will be explained in the following sections. Including:

- Detecting the position and orientation of the stack of cubes, outputting a list of the colour sequence.
- Creating a stack of objects according to the given colour order at the given position and orientation.
- Scanning and detecting the only coloured stack of cubes, and reconstructing another identical stack with the same orientation at the given position.

This experiment will be carried out with some black cubes as a distractor, as well as using similar colours to increase the difficulty of detecting colours.

[1],All authors contribute equally in this coursework for both report and coding.

## II. RELATED WORKS

### *2.1 Robotic arm kinematics*

The initial stage in operating the robot arm is determining the relative location of the manipulator and the target item. Coordinates of the target item are converted to base coordinates by getting the transformation matrix of each axis joint using the D-H coordinate approach. Forward and reverse kinematics of the robotic arm are utilised to characterise the item in motion. Forward kinematics is used to modify the coordinates of the end effector by changing the rotation angle of each joint. In contrast, reverse kinematics is used to determine the angle of rotation required for each axis based on the end-effector location. Forward kinematics is the foundation of inverse kinematics and may be used to verify if inverse kinematics is accurate. Finally, two considerations are singularity analysis in inverse kinematics and computing the mass centre calculation while grabbing.[1]

### *2.2 Grasping*

The primary aim of coursework3 is to grasp objects using robotic arms. Our approach obtains the coordinates and orientation of objects of a fixed shape (small squares) in 3D space. However, owing to flaws in the perception and control systems, grabbing items with a robotic arm is a more difficult job in real life. This is because many factors impact the stability of the grasp, such as mass distribution, object shape, surface friction, etc. Our conventional algorithm is sluggish, prone to perceptual mistakes, and can only grasp things with a predetermined shape[2].

Model-based algorithms ( provide a geometric model of the surroundings and the item[3][4][5]) and raw video-based algorithms (without a specific model[6][7][8]) are the two algorithms for grasping objects. Recent research has proposed using CNN models to extract characteristics from pictures and approach robotic arm gripping as a supervised learning issue[9][10][11]. There has been a significant quantity of research on open-loop prediction, closed-loop continuous control, and grasping of veiled objects. The most challenging task is grasping occluded objects [12].In a practical scenario, a robot would be instructed to grasp a certain shaped item from a cabinet containing a variety of shaped objects. The target item must be gripped by the robot arm without clashing with other objects or the surrounding environment. Creating a kinematically plausible and effective gripping route for the robot arm might be difficult[2].

The issue is with the robot's perception. Parts of the target item may be obscured by other objects in complicated

surroundings. Even if the object's shape is known, conventional algorithms struggle to generate a faultless motion trajectory.Furthermore, 3D reconstruction with great accuracy utilising RGBD cameras [13] or multiple views [14] is challenging. This is because scanning the area needs the robotic arm to continually shift position, which is impractical in a tiny space[2].In order to enable continuous completion of grasp tasks in the environment, hierarchical models based on reinforcement learning have been extensively researched[15][16]. Multi-level models are computationally intensive and can only conduct simulations or simple physical tasks[17]. According to a recent study by Adithyavairavan and Arsalan, although the accuracy rate for grasping novel objects in complicated situations is 80.3%, this conclusion only applies to simulated environments, and their motion planning may fail[2].

The MIT-Princeton ARC team, Berkeley Deep Drive, Stanford Vision  Learning Lab, Berkeley AI Research, and other world-leading laboratories are now working on reinforcement learning research. It is supported by large technology firms like Google, Amazon, NVIDIA, etc[2][12][18].

### *2.3 The development of robot arm and robot manipulator*

The manipulator is especially necessary for the robot to accomplish complicated tasks since it adjusts the direction of an object while retaining a grasp. Manipulators capable of conducting sophisticated grasping actions follow these three paths. Designing manipulators by simulating the human hand, however, they are not only costly to produce but also complicated because of the great degree of flexibility. In addition, grippers with active surfaces (conveyors), which enable items to be gripped without altering posture, restrict the mobility of the manipulator due to the fixed direction of motion. The Stanford Artificial Intelligence Laboratory created the Roller Grasper, which employs three spherical rollers to replicate in-hand handling. Their design does not rely on sophisticated mechanics like multi-fingered dexterous hands but instead relies on a spherical framework to accomplish rolling and reorientation.[19]

Rasoul and his colleagues have presented a novel form of a manipulator that employs three different grasping systems: pumpless vacuum suction, radially deployable claws, and Nano Polyurethane sticky gels. This robot is mostly used for garbage sorting; it addresses the issue of various shapes of recyclable materials and grasps contaminated objects.[20] "Blue" robot presented by the University of California, Berkeley, is the most significant item to introduce because of its inexpensive cost. Each robot costs 5,000 dollars and contains two robotic arms with seven degrees of freedom.[21]

## III. PROBLEM STATEMENT AND HYPOTHESES

### *A. Task 1*

In Task 1, we are expected to detect and localise a stack of cubes. In this task, we are not given any prior information of the environment except the initialised height of the ground which is 0.01m and there will be a stack consists of 3 to 5 cubes with side length of 0.04m to be localised. In other words, the location of the stack and the number of cubes of the stack are unknown. To accomplish this task, we need to scan the environment by the RGB-D camera to find the coordinate of the stack in the world coordinate system, the number of cubes and the colour of each cube in order, and then return these results in the response.

### *B. Task 2*

In Task 2, there are individual cubes scattered on the ground and we are expected to use these cubes to construct a stack. In this task, we are given the information about the expected coordinate, orientation of the stack and the colours of the cubes from bottom to top. To accomplish this task, we need to scan the environment first to get the location, orientation and colour information of each single cube, and then place some of them to form a stack as required.

### *C. Task 3*

Task 3 can be seen as the combination of the previous two tasks. In this task, we are given the a stack of cubes in different colour, some individual cubes and several black stacks, which are generated in random positions and with random orientations. We are expected to scan the environment and then use the individual cubes to build an identical stack as the only coloured stack generated in the environment at the required position given in the request message. To accomplish this task, we need to first scan the whole environment to construct a point cloud. Then, filtering the point cloud to filter out the black stacks. The filtered point cloud can be clustered and the centroid of each cluster can be computed. Since the black stacks are all removed from the point cloud, the example stack is the only object that is taller than others. Therefore, the example stack can be extracted according the z value of its centroid. After obtaining the example stack, the rest work to do is just the same as in the previous two tasks.

## IV. PROPOSED TECHNICAL SOLUTION

### *A. map scan*

In order to observe the cubes and stacks on the map, scanning the environment with a camera on the robot arm is needed, as well as filtering out the unwanted point clouds thus getting a better output result.

*1) path planing:* During the experiment, it is noticed that the faster the robot arm moves, the lower the point cloud quality will be. Therefore, the moving path of the robotic arm has been planned to cover the whole front map. Multiple stopping points are set up in between to slow down the moving speed thus observing a better quality of point cloud.

*2) pass though filter:* In addition, a z-axis pass-through filter is required to filter out the point cloud of the ground, thus only the cubes and stacks are left, which is a benefit for the following jobs(i.e. centroid detection, point cloud clustering).

### B. point cloud storage

while scanning the map, the point clouds observed by camera for each frame need to be merged together and stored. This is achieved by creating an empty global point cloud variable – *g_add_cloud* to start with and keeps adding the new point cloud to it, which can then be published so that can be observed in Rviz.

*1) point cloud clustering:* In order to separate the point cloud of each cube or stack, the point cloud needs to be clustered, which is beneficial for finding the centroid of cubes. This can be achieved by clustering in terms of colour and the density of point cloud.

### C. detecting centroid

Centroid detection can be achieved by using *pcl::ComputeCentroid* function, given the *g_add_cloud* as the input. This built-in function returns *pcl::PointXYZRGB* type points and this is convenient for the color detection, as the point contains not only the position information, but also the color of the centroid.

### D. finding orientation

In order to find out the orientation of the cube, the point cloud at the z-direction can be ignored. Therefore, the point cloud is projected onto the z-axis, in other words, only the top plane of the cube remained.

The orientation angle can be derived by subtracting the angle between any side of the square and the x or y-axis. The vertex of cubes can be extracted by observing the minimum and maximum coordinate of the given point cloud. Normally, taking the average value computed by the four sides gives the best result. However, during the experiment, it is investigated and discovered that only the side connected by the maximum y coordinate and the minimum x coordinate produces a more accurate angle due to the imperfection of the point cloud scan.

### E. colour detection and storage

In Task 1, before detecting the colour of each cube, the number of cubes needs to be figured out. The height of the stack intuitively indicates the number of cubes. In order to find the height of the stack, its centroid needs to be computed by the function mentioned in section IV_C. As the number of cubes will always be 3, 4 or 5, the z value will stay at some threshold. If the z value is greater than 0.1m, there will be 5 cubes, and if it is less than 0.08m, then there will be 3 cubes. Otherwise, there are 4 cubes in the stack.

After knowing the number of cubes, the point cloud will be clustered according to different heights. The clustering is done by applying Pass Through filter to the whole point cloud with different z-intervals. To make sure that each cluster only contains the points belonging to one single cube, the interval of the Pass Through filter is set to be 0.02m. To extract the colour information of each cube, the centroid of each cluster is computed by the function *pcl::computeCentroid* which gives a *pcl::PointXYZRGB* point as output. The resultant point not only contains the centroid and orientation of that cluster but also indicates the colour of that centroid. All the colour information obtained is stored in a vector and returned in the response. In task 3, this can also be achieved by directly reading the colour by changing the z value of the centroid of the stack for each layer.

In Task 2, the basic idea to handle the colour of the cubes is quite similar to that in Task 1, which is always clustering the point cloud from the sensor message to separate each cube first. After that, the centroid and orientation of each cluster can be computed as well as its colour can be obtained. To store the centroid and orientation of these cubes, six empty vectors corresponding to six different colours are created as a space holder. A colour filter is applied to each cluster's RGB values to store the information of the cubes.

In the Task 2 callback function, the order of colour cubes is requested, which can then extract the centroid and orientation from the six colour vectors. These can then be transferred to the *pickAndPlace* function.

### F. pick and place with orientation

The main structure of the *pickAndPlace* function is similar to the function in the first coursework. Within this function, the *moveArm* function is called to move the end-effector to the desired position, and the function *moveGripper* is called to open or close the gripper to grasp the object. The difference is that in this coursework, the orientations of the cubes need to be taken into account for both grasping and placing. Therefore, an extra input is added to this function, which is the placing orientation. The grasping orientations are global variables, which can be called in this function. What's more, as it is expected to construct an identical stack as the example, we need to place the cubes one by one gently. Otherwise, the cube may bounce when colliding with the tile or other cubes and this will result in an error in its final position and failure in stack construction.

To deal with this problem, one more parameter was added to the *pickAndPlace* function, which is the placing height factor. This factor increases as the number of cubes being placed increases. In other words, the height of the gripper when dropping the cube will increase by 0.04m every time for a new inputting cube. Before releasing the cube, a slightly moving down the releasing position is also required to limit the dropping height.

### G. adding and removing collision

To make sure the robot arms will not collide with the required cubes as well as the building stack, collisions need to be added and removed before and during grasping. This is achieved by naming the cube in its order. Since by removing the collision, the individual name of the cube needs to be called.

## V. HYPOTHESIS AND EVALUATION

All of the algorithms do not employ octomap since there is no way for the algorithm to entirely eliminate the collision generated by octomap. In this section, the processes and results of each task will be discribed.

### A. TASK1:

In task 1, the programme explores the whole region for the point cloud because the position of the formed stack is unknown. The programme then computes the coordinates of the centre of the filtered point cloud and determines the number of cubes that the stack contains based on the height of the z-axis. Finally, the colour of cubes is stored in order as well as the position and orientation of the stack.

The main issue is to get an accurate orientation. The identification of angles is prone to certain inaccuracies due to the imprecise output of the point cloud, as indicated in the preceding section. Many attempts have been attempted to overcome this problem, such as using the diagonal angle to determine the orientation of the cube, switching the camera on/off using a global variable, and slowing down the movement of the robot arm movement. These procedures will help to increase the quality of the point cloud, but the angle is still not accurately determined especially when the orientation is around zero. Therefore another judgement is made, by finding the length between maximum and minimum on the y-axis. If the difference is smaller than a certain value the orientation is set to be zero.

This algorithm effectively recognises the angle, colour, centroid, and number of cubes, as shown in Fig. 1.

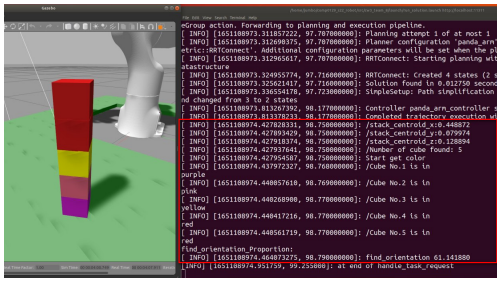

Fig. 1: task 1

### B. TASK2:

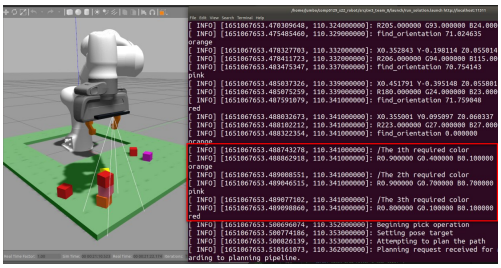

Fig. 2: task 2

In task 2, we discovered that if the camera scan through a region twice, the cube would be scanned as a rectangle. This is solved by moving the arms to the side before activating the camera, as well as adding a switch to control the storing point cloud. This gives a better quality of point cloud, thus a more accurate centroid and orientation of cubes can be computed.

As stated in task1, because of the imperfect orientation for the same reason, the cube gripped by the arm might be unstable. The cube does not completely align with the cube below it when placed, and it may even roll to the ground.

This algorithm successfully constructs the stacks in the order of the input colours, with a slightly unstable picking orientation. Therefore, the target stack is not 100% aligned, which shown in Fig. 2

### C. TASK3:

Task 3 is the combination of Tasks 1 and 2 with a lot of black stacks as distractors. However, as the cubes are all over the map, the camera will sometimes not work as expected even with the planned path. The merged point cloud sometimes will have low quality, which is filtered using more filtering functions. In addition, due to the reflection of colour, the black stack is not 100% black. The point cloud of the black stacks cannot be completely removed so that the threshold while clustering is set to be higher to not treat these noises as a cluster.

Moreover, since the light of the simulated environment shines right in front of the robot arm, the arm casts a shadow. This shadow will have a significant impact on the ability of our algorithm to recognise colours for those behind the robot. As a result, we have created specific movement pathways and also govern the switching of the point cloud. These two measures limit the influence of shadow on the point cloud.

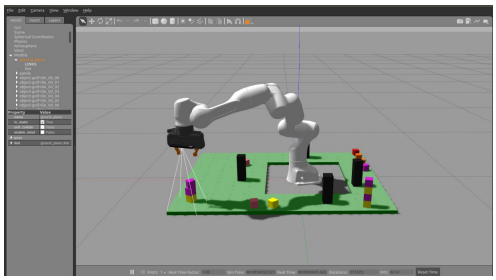

Fig. 3: task 3

## VI. DISCUSSION AND CONCLUSION

The report concentrates on coursework3 solutions. The accuracy of the point cloud is one of the important aspects impacting the pick and place task, and future studies might focus on enhancing the quality of the point cloud. According to recent research, object identification accuracy utilising deep learning algorithms has significantly increased, particularly in our instance where the cubes do not overlap. Reinforcement learning may be a superior option for objects that overlap. These could be the areas where we should focus our future study. Additionally, using four claws instead of two parallel claws for grasping may lead to a more stable result. To improve the quality of the point cloud we may use other cameras, as well as use a slower and smoother motion plan algorithm.

## REFERENCES


[1] R. Sadeghian, S. Shahin, and S. Sareh, "Vision-based self-adaptive gripping in a trimodal robotic sorting end-effector," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2124–2131, 2022.

[2] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6232–6238.

[3] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 189–196.

[4] M. Moll, L. Kavraki, J. Rosell, *et al.*, "Randomized physics-based motion planning for grasping in cluttered and uncertain environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 712–719, 2017.

[5] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 639–646.

[6] A. Saxena, L. L. Wong, and A. Y. Ng, "Learning grasp strategies with partial shape information." in *AAAI*, vol. 3, no. 2, 2008, pp. 1491–1494.

[7] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conference on robot learning*. PMLR, 2017, pp. 515–524.

[8] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation," *Autonomous Robots*, vol. 37, no. 4, pp. 369–382, 2014.

[9] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 202–216, 2020.

[10] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[11] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine, "End-to-end learning of semantic grasping," *arXiv preprint arXiv:1707.01932*, 2017.

[12] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical search: Multi-step retrieval of a target object occluded by clutter," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1614–1621.

[13] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[14] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[15] J. Sung, B. Selman, and A. Saxena, "Learning sequences of controllers for complex manipulation tasks," in *International Conference on Machine Learning*. Citeseer, 2013.

[16] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[17] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf

[18] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3750–3757.

[19] S. Yuan, L. Shao, C. L. Yako, A. Gruebele, and J. K. Salisbury, "Design and control of roller grasper v2 for in-hand manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9151–9158.

[20] R. Sadeghian, S. Shahin, and S. Sareh, "Vision-based self-adaptive gripping in a trimodal robotic sorting end-effector," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2124–2131, 2022.

[21] D. Gealy, S. M. McKinley, B. Yi, P. Wu, P. Downey, G. Balke, and A. Zhao, "Quasi-direct drive for low-cost compliant robotic manipulation," 04 2019. [Online]. Available: https://arxiv.org/pdf/1904.03815.pdf?