A procedural approach to stacking multi-oriented coloured objects using basic geometry and heuristic-based algorithms

Daniel Chipping[1], Mahmud Huseynov[2], Chetan Kalsi[3]

***Abstract*— This paper covers the issues of object segmentation, localisation and grasping of multi-oriented coloured objects in an environment with obstacles. A series of three tasks are defined of increasing generality, and a set of efficient heuristic-based solutions are proposed. The pick and place experiments are simulated in Gazebo on a Franka Emika Panda arm using MoveIt!, OctoMap and the Point Cloud Library to detect coloured cubes and stack by colour whilst avoiding obstacles.**

Simulation results show that we have been able to successfully extract object colours and stack cuboid objects in correct orientations. Finally, we show how this work lays foundation for future approaches that optimise for time taken, grasping of irregular objects and generalised reinforcement learning policies.

I. INTRODUCTION

Object detection, localisation and manipulation are necessary abilities for robots to be able to interact with the world. Current research is looking at how robots can be of assistance in daily tasks through independent completion and through task collaboration. Tasks in the home environment are complex due to the unstructured nature of how items are kept. This involves being able to find objects that could be obscured with obstacles, determine current and required orientation and being able to correctly determine grasp positions for manipulation. Research has been undertaken that tries to solve these complex issues through the development of a robotic systems that can unload items from a dishwasher and fetch specific objects via voice commands [1].

Within the industrial setting, robots are being looked at to support conventional building techniques such as brick laying [2]. They have the advantage of being able to lay more bricks than a person can per day. To be able to successfully complete these tasks, they have to be able to correctly grasp each brick and stack them with the correct alignment and orientation. For increasing productivity within industry, robotic systems are well placed to be able to carry out monotonous pick and place tasks with accuracy and speed.

Robots that are already used in warehouses for pick and place problems are designed for very specific tasks in a particular environment. There is a huge challenge in developing inexpensive robotic systems that have the ability to grasp different products in changing environments [3].

This paper presents 3 simulation tasks that build on each to solve a pick and place problem of finding and stacking multicoloured cubes in a precise colour order. This is done through the use of the Point Cloud Library (PCL) and storage of cube RGB colour data, co-ordinates and orientation. The cubes in question are of both uniform size and shape. The solution is evaluated via simulation using Gazebo.

Department of Computer Science, University College London, Gower Street, WC1E 6BT, UK. [1]ucabdc4@ucl.ac.uk, [2]ucabmh5@ucl.ac.uk, [3]zcecf50@ucl.ac.uk

II. RELATED WORK

Successful object manipulation is a central skill that allows robots to carry out complex tasks. Lozano-Perez et al. [4] at MIT developed the Handey robotics system. The team broke down the pick and place problem into the following inter dependant steps:

- Choose a grasp on object
- Plan motion to grasp object
- Plan motion to new object location
- Plan motion to extract gripper

Visual perception is an import component that allows a robot to carry out the steps of a pick and place problem. Building a map of the environment allows the robotic system to detect target objects and obstacles to allow for path planning. Point Cloud Library (PCL) is an open source 3D image processing library which allows for building a detailed map of the environment. It allows for point cloud filtering, feature extraction, geometric registration, reconstruction, segmentation and model fitting. It has now "become the standard for unorganised point cloud processing among robotisicst" [5].

For vision systems, correct camera calibrations are essential to create accurate point clouds. Calibration feature templates are used [6] along with camera models like Tsai's model [7] to achieve required accuracy.

Obstacle avoidance is another important part of motion planning. Pick and place operations have a typical feature of smooth trajectories. Hyperbolic type trajectories have been suggested for robotic arm manipulators as a means of obstacle avoidance. [8]

Kabutan et al. [9] have developed a "robotic intelligent space (RIS)" sensor through the use of Kinect devices (RGB-D cameras), that can obtain point cloud data. The cameras are not attached to the robot arms but are instead located around the robot work space preventing occlusions. Point cloud data is collected from multiple Kinect devices and merged for object detection. A payoff is required between point cloud density and number of Kinect devices to reduce merge computational time. Object detection steps from point cloud data is as follows:

- Extract Euclidean clusters from point cloud data.

- Project 3D point cloud data set to 2D x-y plane.
- Reduce point cloud dimensions to determine convex hull.
- Search minimum bounding rectangle.
- Obtain rotation and center of gravity.
- Determine 3D space bounding box.

Motion planning is carried out through the creation of a 3D occupancy grid map for obstacle avoidance. Kabutan et al. [9] modelled their system using "MoveIt!". Their experiments resulted in a 83% success rate, with most failures occurring due to errors in motion planning. For successful motion planning three dimensional models of the environment are required. Point clouds are not memory efficient due to to the storage of large number of measurement points. Cloud points also do not differentiate between what areas are obstacle free or unmapped [10]. A common approach that overcomes these issues concerning point clouds is through the use of the OctoMap framework as described by Hornung et al. [10] which uses a tree based representation of the mapped area. Occupancy is estimated through probabilistic means. It works through the use of a hierarchical data structure known as an octree that represents cubic volumes of space that are called voxels. The fact that a probabilistic model is used to determine occupancy, sensor noise can be dealt with and sensor fusion can also be carried out.

Within the pick and place problem, the task of grasping is a significant area of research. Of particular issue is grasping novel objects and grasping in cluttered environments. For novel objects, grasping poses have been determined with the use of synthetic data and deep neural networks [11]. Systems have been developed to improve the speed of determining grasp positions through the use of two-step cascade deep learning systems [12].

For grasping in cluttered environments, many methods have been suggested and tested. For example Zeng et al. [13] employ "object-agnostic grasping" which isolates objects from clutter and then try to recognise the object. Jiang et al. employ a system that rearranges and pushes objects out of the way to recognise a grasp target amongst the clutter.

Lobbezoo et al. [14] carried out a review of pick and place operations in robotics specifically with regards to reinforcement learning and provided a detailed summary table of 22 papers that have looked into this. Reinforcement learning "is becoming a popular alternative to task-specific programming" to allow robotic operations to take place in a less constrained manner [14]. They highlight that pick-and-pace implementations differ drastically and there is no industry standard. All the key tasks involved in pick and place are brought together as shown in Fig. 1.

III. PROBLEM STATEMENT AND HYPOTHESES

The overall goal is to generate a solution to a simulated pick-and-place problem using the the Fraka Emika Panda 7 DoF manipulator. This problem involves colour detection, grasping, orientating and stacking while avoiding obstacles. The task is broken down into 3 sections that build on each other. The environment is simulated in Gazebo and

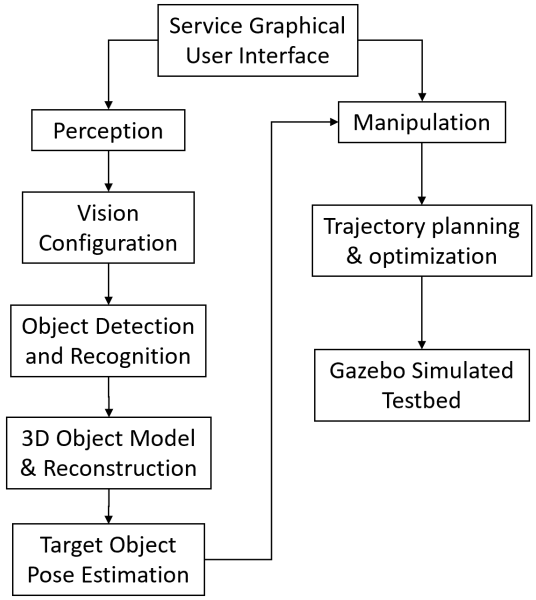


Fig. 1: Pick and Place System Flowchart [3]

MoveIt! to carry out motion planning, manipulation, inverse kinematics, control, 3D perception and collision checking. For 3D perception, Point Cloud Library and OctoMap are used. Fig. 3 shows the simulation environment for each task.

As mentioned previously, camera calibration is crucial for accuracy of 3D perception via point cloud data. Within our simulation we can use the knowledge that the cube objects are 4cm in length to confirm if there are any inconsistencies with the point cloud data. Fig. 2 demonstrates how this is done, by allowing the robot arm to get close to a cube form overhead.

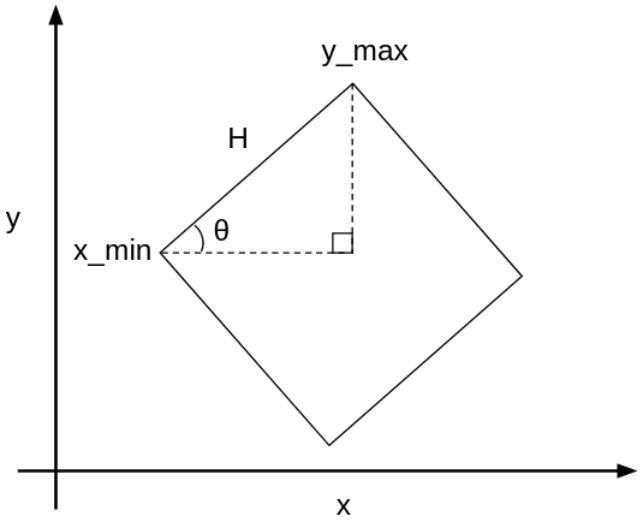
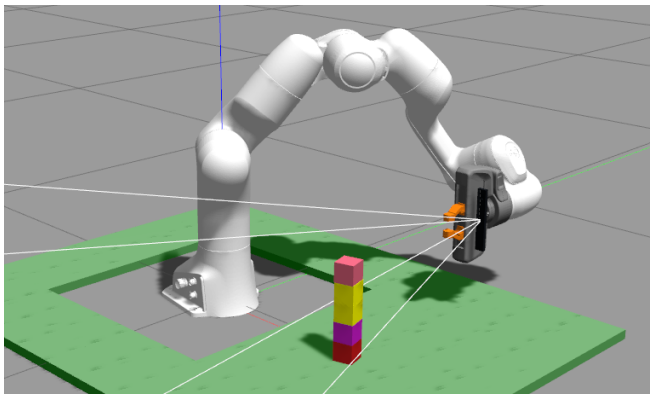


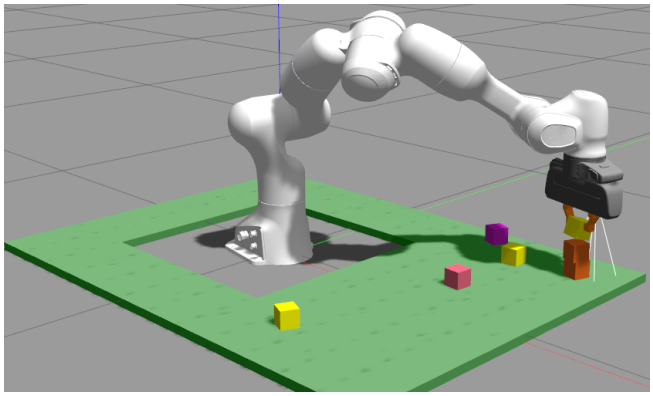
Fig. 2: Cube hypotenuse verification

$$\mathrm{H} = \sqrt{(x \text{ of } y_{\max} - x_{\min})^2 + (y_{\max} - y \text{ of } x_{\min})^2} \quad (1)$$

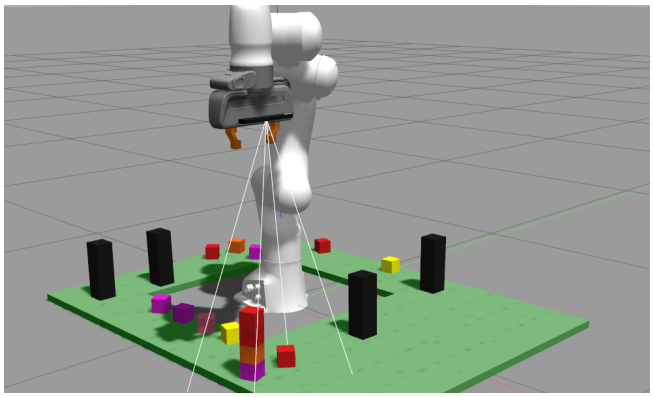
$$\theta = cos^{-1}((x \text{ of } y_{max} - x_{min})/H) \quad (2)$$



(a) Task 1



(b) Task 2



(c) Task 3

Fig. 3: Task Simulation Environments

The hypotenuse is verified by equation 1. If there were no point cloud precision errors, the hypotenuse would read as 4cm. If there is any error, this hypotenuse value will be scaled accordingly with the point cloud data.

IV. PROPOSED TECHNICAL SOLUTION

The proposed solutions for the simulation experiment is discussed in more detail in the following subsections:

A. Task 1

The first task involves detecting and localising a stack multicoloured cubes. To solve this initial problem, the task is broken down into the steps presented in Algorithm 1.

Algorithm 1 Colour Observation (Task 1)

1) Look at scene using RGB-D camera via the scoutFront function.
2) Localise x,y co-ordinates of stack with examineTop function.
3) Move robotic arm closer to stack to improve point cloud resolution.
4) Find orientation of stack of cubes using trigonometric function and correct for resolution errors.
5) Rotate robotic arm 90deg to stack.
6) Segment point cloud based on colour.
7) Record RGB values of each cube at each cube mid-point.

B. Task 2

The second task involves building a stack of objects in a specified colour order. The proposed solution is presented as Algorithm 2.

Algorithm 2 Building a Colour Ordered Stack (Task 2)

1) Define stack build location (taken from request)
2) Look at scene from top using RGB-D camera via the scoutFront function and segment out ground plane.
3) Confirm requested stacking space is clear on the x,y co-ordinates.
4) If not clear, grasp cubes in stacking location and move to free space.
5) Localise co-ordinates of cubes, centroids, orientation and colour. Save in vector.
6) If colour of cube matches colour request, execute pick and place function.
7) Define grasp via cube centoid and orientation.
8) Grasp cube.
9) Move cube to the requested orientation and location.
10) Repeat steps 6-9 until stacking is complete.

C. Task 3

The third task combines tasks 1 and 2, where a stack of objects need to be detected and localised. Then an identical stack needs to be built using objects in the simulation environment while avoiding stacked obstacles (shown in Fig. 3c as black stacks).

V. EVALUATION

A. Task 1

Simulations demonstrate that the RGB value colours of each cube in a stack were successfully identified and saved.

In carrying out simulations of Task 1, it was found that the segmented point cloud data did not show the cubes in high resolution and colours were found to bleed between the cubes. This could then result in a incorrect RGB value being recorded for the cube. To overcome this issue, the RGB data was collected from the center of the cubes on the side face at specific heights as shown in Fig. 4. This was

Algorithm 3 Building a Colour Ordered Stack with Obstacles in the Environment (Task 3)

1) Look at scene using RGB-D camera and segment out ground plane.
2) Localise x,y co-ordinates, centroids and colours of objects in the environment - Save to vector.
3) Loop through vector and find black cubes/stacks (RGB 0.1, 0.1, 0.1)
4) Assign black cubes as obstacles in the environment.
5) Move robot arm 90deg to side of coloured stack.
6) Record RGB values of each cube at each cube mid-point from bottom to top.
7) From Task 3 service request, determine location and orientation of new stack.
8) Confirm stacking location is clear and if not move objects that are in the way.
9) Move required colour cubes to the requested orientation and location using OctoMap occupancy grid to avoid obstacle stacks.
10) Continue to repeat steps 9 until stacking has been completed.

the location where the cube colours were not impacted by the adjacent objects. This method also overcame the issue of edge cases where cubes adjacent to each other are the same colour. By using specific heights to define the recording of the RGB values, the results are robust in ensuring a cube is not potentially missed due to being the same colour as neighbouring cube.

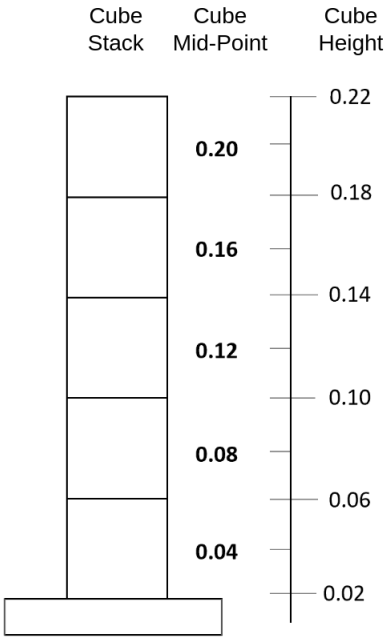


Fig. 4: Task 1 setup with data on cube heights to determine RGB values. (Height values in cm)

B. Task 2

We were able to demonstrate the stacking of coloured blocks through simulation with minimal deviations in centering and orientation. Speed was an issue when carrying out task 2. The problem was setup to analyse and save data on all the cubes in the scene irrespective of their requirement to be stacked. This resulted in increased processing times and data storage of objects that wouldn't necessarily need to be grasped or moved. This task could have been improved by filtering the objects by colour as per the stack building requirement. All other objects can then be ignored if not in the way of the stacking task.

C. Task 3

Task 3 was implemented as highlighted in section IV, which combined the elements of Task 1 and 2. It ran into similar speed issues as seen in the implementation of Task 2, so object filtering by colour is something that should be considered for further improvements. Through the use of OctoMap, the occupancy grid and its integration with MoveIt!, path planning amongst obstacles was successful. A key performance indicator that will be useful to determine effectiveness of implementation is the speed and success rate of completing this task. This could be compared to task performance when using hyperbolic trajectories for obstacle avoidance and path planning as described in section II [8].

Further optimisation would be through the determination of the ideal voxel grid size. This is defined in the code by the variable 'g_vg_leaf_size'. Decreasing the size, would increase the resolution of the environment occupancy grid. This should reduce task failures but at a cost of increased computational time.

VI. CONCLUSIONS AND FUTURE WORK

This paper has demonstrated how colour data can be extracted from objects, how objects can be grasped, manipulated and stacked while navigating an obstacle filled environment. Point Cloud Library (PCL) and the OctoMap framework are powerful tools in allowing a robot to extract information and understand its environment. This is important as robotic systems are being developed to carry out tasks in less structured environments like a home or construction site. Camera calibrations are essential to confirm what a robot sees and maps matches reality, especially if it is required to manipulate small objects or navigate with high precision.

The task of stacking cubes by colour has been outlined and tested via simulation. Future development of this task includes trying to improve the speed of task completion and find efficiencies as described in section V.

Liu et al. [15] have developed a system that can stack irregular shaped stones. This problem carries greater task complexity as a stable pose needs to be determined for each object. Our work could be used as a foundation to deal with irregular shaped objects.

The implementation of a reinforcement learning approach would be another research area to build on, since it is considered that reinforcement learning "has the potential to replace traditional robotic control" [14] and improve task generalisation [3]. This would open up the robotic system to being more flexible to working in different environments.

REFERENCES

[1] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng, "A vision-based system for grasping novel objects,"," 2007.

[2] R. Bogue, "What are the prospects for robots in the construction industry?" *Industrial Robot: An International Journal*, vol. 45, 12 2017.

[3] P.-C. Huang and A. K. Mok, "A case study of cyber-physical system design: Autonomous pick-and-place robot," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 22–31.

[4] T. Lozano-Perez, J. Jones, E. Mazer, and P. O'Donnell, "Task-level planning of pick-and-place robot motions," *Computer*, vol. 22, no. 3, pp. 21–29, 1989.

[5] K. Zampogiannis, C. Fermuller, and Y. Aloimonos, "cilantro," in *Proceedings of the 26th ACM international conference on Multimedia*. ACM, oct 2018. [Online]. Available: https://doi.org/10.1145%2F3240508.3243655

[6] A. Sbnchez and J. Martinez, "Robot-arm pick and place behavior programming system using visual perception," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 4, 2000, pp. 507–510 vol.4.

[7] R. Y. Tsai, "An efficient and accurate camera calibration technique fro 3d machine vision," in *CVPR'86*, 1986.

[8] C. Muller-Karger, A. Leonell Granados Mirena, and J. Scarpati Lopez, "Hyperbolic trajectories for pick-and-place operations to elude obstacles," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 3, pp. 294–300, 2000.

[9] R. Kabutan, R. Tanaka, S. Oomori, M. Morita, E. Inohira, K. Yoshida, H. OHTAKE, and T. NISHIDA, "Development of robotic intelligent space using multiple rgb-d cameras for industrial robots," in *Proc. of ICT-ROBOT, ThBT3. 2*, 2016.

[10] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, 04 2013.

[11] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," 2018. [Online]. Available: https://arxiv.org/abs/1803.11469

[12] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015. [Online]. Available: https://doi.org/10.1177/0278364914549607

[13] 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.

[14] A. Lobbezoo, Y. Qian, and H.-J. Kwon, "Reinforcement learning for pick and place operations in robotics: A survey," *Robotics*, vol. 10, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2218-6581/10/3/105

[15] Y. Liu, J. Choi, and N. Napp, "Planning for robotic dry stacking with irregular stones," in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds. Singapore: Springer Singapore, 2021, pp. 321–335.