Motion Planning for an Automated Pick and Place Robot in a Retail Warehouse

Sharath Jotawar
TCS Innovation Labs
Robotics Group
Bengaluru, Karnataka, India
sharath.jotawar@tcs.com

Manish Soni
TCS Innovation Labs
Robotics Group
Noida, UP, India
manish.soni1@tcs.com

Swagat Kumar
TCS Innovation Labs
Robotics Group
Bengaluru, Karnataka, India
swagat.kumar@tcs.com

ABSTRACT

This paper describes the use of Moveit motion planning software for implementing an articulated robot based automatic pick and place system for a retail warehouse. The proposed system is expected to automatically pick things from a rack and place them in a tote and vice-versa, based on an order file. Currently, these tasks are carried out by humans leading to higher cost of operation. The motion planning methods are demonstrated through both simulation and real world experiments. We believe that the details provided in the paper will act as a tutorial for beginners and reference manual for experienced researchers and practising engineers.

KEYWORDS

robot manipulator, pick and place, motion planning, inverse kinematics, TRAC-IK, Moveit.

ACM Reference format:
Sharath Jotawar, Manish Soni, and Swagat Kumar. 2017. Motion Planning for an Automated Pick and Place Robot in a Retail Warehouse. In *Proceedings of AIR '17, New Delhi, India, June 28-July 2, 2017,* 6 pages.
DOI: https://doi.org/10.1145/3132446.3134904

1 INTRODUCTION

In the recent years, there has been an increase in the availability of affordable commercial service robots[1] encouraging the development of applications around service robots. Service robots refer to the robots that assist humans to carry out tasks that are simple, dull and repetitive in nature[2]; such robots can be utilized in performing manual work to increase the efficiency of industries. Big corporations like Amazon employ thousands of people for handling of goods stored in racks in their warehouses. These warehouses are usually huge in size and can be as big as nine football pitches[3]. Daily operations in a warehouse involve stock assessment, picking, stacking, packing, unpacking of goods for order fulfilment and transfer of goods from one place to another. Such operations involve lot of manual work making a person walk over miles in a day around the warehouse. As shown in left image of Figure 1, Amazon was able to automate the process of transportation of goods around the warehouse by the use of mobile robots called Kiva systems[4] that carry the racks around warehouse to a dedicated place where the items are picked from or stacked into the rack. Still the process of carrying an entire rack just to pick or stack few items is not an efficient procedure as it makes the robot consume more energy and time to transfer the rack carefully. Hence, Amazon wants to automate the process of picking and stacking of items using robot manipulators so that the mobile robots just have to carry the required items instead of the rack. In this regard, Amazon has been conducting an annual robotics challenge called 'Amazon Picking Challenge'[5] for the advancement of automation of picking and stacking of items in a rack.

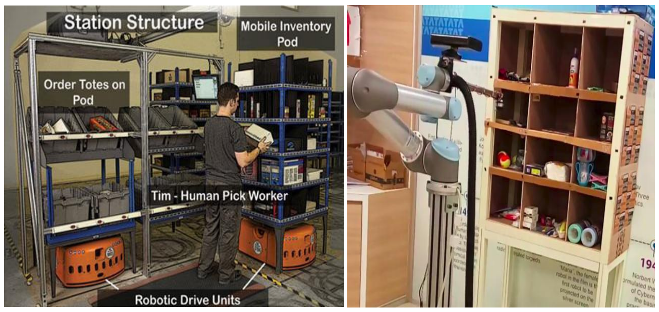


Figure 1: Left image shows Kiva systems carrying rack to human for picking objects[1]. Right image shows the robot manipulator system used for picking objects.

The challenge involves two tasks, namely pick and stow task. In pick task the competitors are provided with a computer generated JavaScript object notation (JSON) task file containing information of names of the items placed in the tote, names of the items in each bin of the rack, and names of target items to pick from each bin. The robot manipulator has to read the task file and identify location of each target item inside the corresponding bin. Once a target item is found the manipulator has to carefully pick and place the target item into a tote without damaging any item or the rack. Similarly in stow task the competitors are provided with a JSON task file that contains information of the names of the items that are placed in the tote, names of the items in each bin of the rack, and names of target items in tote to be placed in rack. Here as well the manipulator has to identify the target items' location in the tote. Once the target item is found the manipulator has to pick the item from tote and place it any bin of the rack. At the end of each

[1]Image Courtesy: http://tech.firstpost.com/

task the robot has to generate an output JSON file specifying each item's final location in the rack and tote.

In the competition the teams are provided with a rack similar to the racks used in the Amazon warehouses. The rack consists of 12 bins arranged in a 4 rows by 3 columns grid structure, the bins can be one of two different sizes $27cm \times 30cm$ and $22cm \times 30cm$, with all the bins extending upto a length of $35cm$. The teams are provided with 40 general household objects, some of the objects can have multiple copies. The objects can be of different sizes, shapes, appearance, weight and hardness. The objects can be placed in any orientation or position in the rack and tote, such a placement of objects can have partial or complete occlusion of some of the objects. In the pick task 12 target objects has to be picked from the bins and placed in the tote, while in the stow task 12 target objects has to be picked from tote and placed in the rack. Each task has to be completed within a duration of 15 minutes. Point are awarded for each right pick and place of an object, while points are deducted for a wrong pick and place or damage of an object.

The competition provides us with real world problems which can be broadly classified into four main categories

- Motion planning for robot manipulator to carefully manoeuvre around the rack and tote for pick and place of the objects.
- Identification and pose estimation of objects with partial or complete occlusion, variation in lighting conditions, varying position of camera.
- Calibration of camera, 3D sensor, laser sensor with robot manipulator.
- Detection of rack, tote or any other nearby collision objects within the workspace of the robot. This is needed so that the robot is aware of collision with the environment.

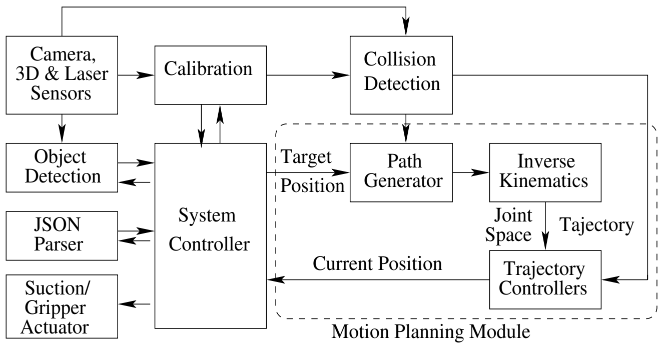


Figure 2: Block diagram of robot manipulator system for autonomous pick and place of objects

Figure 2 shows the block diagram of the robot manipulator system providing individual modules for the problem categories explained above. In Figure 1, the right image of shows our setup consisting of UR5 arm with vacuum pipe suction based grasping system. The steps involved in pick and place of objects from rack is shown in Figure 3.

The focus of this paper is on the problem of motion planning for the manipulator inside bins of the rack and outside the rack. The details of the motion planning module is shown in the Figure 2. The motion planning is divided into two fold, online motion planning inside the bins of the rack and offline motion planning outside the rack. The online motion planning is more challenging in nature since the planning has to be made within the confined space of a bin which can be as small as $22cm \times 30cm$ in size, the problem becomes more complicated considering the situation of target object being present along with the other objects inside the bin. The algorithm has to take into consideration of space that has to be provided for the suction system of volume $7cm \times 4cm \times 35cm$ to move through the bin to pick and bring the object out of bin without damaging any object or the walls of the bin. Offline motion planning refers to the motion planning for movement of the arm for taking bin view for object detection and moving arm to the tote for dropping the object. These motions are pre-defined and repetitive in nature, the challenge here lies in selecting the shortest path between the current and target positions without colliding with any part of the rack or tote setup.

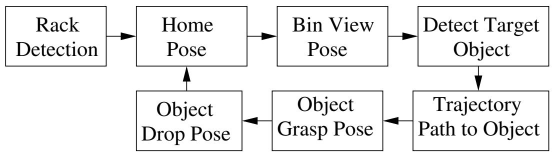


Figure 3: Flowchart for pick and place of target objects from rack. Rack detection is a one time procedure, rest of the steps are performed in continuous loop until all the objects are pick and placed.

An overall view of the rest of this paper is provided as follows. In the next section details of related work in the field of motion planning is provided. In section 3 the individual components of motion planning required for moving manipulator from its current to target position has been described. The simulation and experimental results have been provided in section 4. A detailed summary and direction for future work is provided in section 5.

2 RELATED WORK

A lot of research work has been done in the field of collision avoidance based motion planning, an overview of these methods can be found in [6]. Motion planning can be classified into two groups - local and global. Local methods consists of planning using limited information of the planning environment, while in global methods a comprehensive model of the entire robot workspace is used for finding a global optimal trajectory. Local methods based motion planning can be implemented using potential field approaches [7] [8] [9]. Probabilistic roadmaps (PRM) [10] [11] and probabilistic cell decomposition [12] [13] are approaches used for global optimization of trajectory. Rapidly exploring random tree (RRT)[14] is one of the widely used PRM motion planning algorithm. Motion planning algorithms have been made available through open source libraries like open motion planning library (OMPL)[15].

3 DESCRIPTION OF COMPONENTS OF MOTION PLANNING

Motion planning refers to generation of a trajectory in joint space (in terms of joint angle position or angular velocities) that a manipulator has to trace to reach a target pose from its initial pose. The components that are involved in generation of a motion plan are 1) generate a path (set of way points) in Cartesian space to reach the target pose, 2) convert the the Cartesian way points to joint space angular position or velocities using inverse kinematics algorithm, 3) a controller to transfer the joint space trajectory into actions onto the real robot manipulator, 4) input from collision detection system that enables to generate path plan without any collision of the robot with its environment.

3.1 Automatic Path Generator

The storage rack consists of 12 rectangular bins with front opening for each bin. The objects could be placed lying on the base of a bin or leaning on the walls of the bin. It is assumed that all the objects are within the workspace of the robot manipulator, objects are static in the rack with no relative motion with respect to the rack. A straight line path generation algorithm that exploits the geometry of the bins has been implemented.The placement of objects in the bin enables the consideration one of three default orientations for picking any object. The default orientations for picking objects are picking from the top surface, picking from the left side and right side of the object. The challenges in this method are i) figure out the proper orientation of the target pose and ii) planning a path that manipulator has to trace to pick the object without collision with any part of the rack.

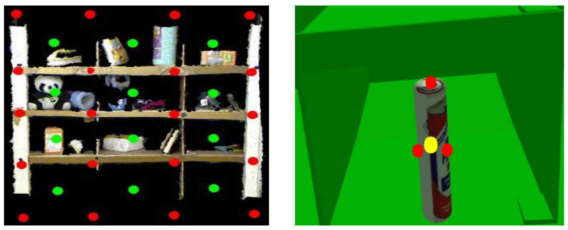


Figure 4: Left image shows bin corners (red color) and centroids (green color) which form the bin information (b_{info}). Right image shows the object information (o_{info}) top, left and right corners (red color) and object centroid (yellow color). b_{info}, o_{info} are used in the auto path generation algorithm.

Algorithm 1 presents the pseudo code used for the path generation in Cartesian space and trajectory in joint space. The path generation procedure detects the maximum space available to pick the target object and then chooses the orientation to pick the object. The algorithm takes input o_{info} target object information, b_{info} bin information, min_{gr} minimum gap required for the suction pipe. o_{info} contains object's topmost, leftmost, rightmost and centroid information found through object detection algorithm as shown in right image of Figure 4, similarly b_{info} contains the bin's topmost, leftmost, rightmost and centroid information found through rack detection algorithm as shown in left image of Figure 4. An estimate of the space available between the object and bin in three directions is made, the side having maximum space for the suction pipe is chosen as the picking direction. Based on the picking direction a pre-defined orientation is combined with each way point in the trajectory path. The rack and object detection algorithm is not discussed as it is not within the scope of this paper.

Algorithm 1 Trajectory Planning

```
1: procedure TRAJECTORY_PLANNING($o_{info}, b_{info}, min_{gr}$)
2:     $(o_{left}, o_{right}, o_{top}, o_{cent}) \leftarrow parse(o_{info})$
3:     $(b_{left}, b_{right}, b_{top}, b_{cent}) \leftarrow parse(b_{info})$
4:     $o_{lg} \leftarrow |o_{left} - b_{left}|$
5:     $o_{rg} \leftarrow |o_{right} - b_{right}|$
6:     $o_{tg} \leftarrow |o_{top} - b_{top}|$
7:     if ($o_{tg} > o_{lg}$ & $o_{tg} > o_{lg}$ & $o_{tg} > min_{gr}$) then
8:         path ← Create set of way points in straight line at interval of 2cm from current position to top end point of the object
9:         trajectory ← IK(path, top surface pick)
10:    else if ($o_{lg} > o_{tg}$ & $o_{lg} > o_{rg}$ & $o_{lg} > min_{gr}$) then
11:        path ← Create set of way points in straight line at interval of 2cm from current position to left end point of the object
12:        trajectory ← IK(path, left surface pick)
13:    else if ($o_{rg} > o_{tg}$ & $o_{rg} > o_{lg}$ & $o_{rg} > min_{gr}$) then
14:        path ← Create set of way points in straight line at interval of 2cm from current position to right end point of the object
15:        trajectory ← IK(path, right surface pick)
16:    else
17:        return with no trajectory generated
18:    end if
19:    return trajectory
20: end procedure
```

3.2 Inverse Kinematics

Kinematics of robot manipulator constitutes of two components namely forward kinematics and inverse kinematics. Forward kinematics refers to solving kinematic equations to find the end-effector pose of manipulator based on the current joints of the manipulator. Similarly inverse kinematics is a procedure to solve for joint position or velocities using set of robot kinematic equations to achieve a desired end-effector pose for the manipulator.

Mathematical expressions for forward and inverse kinematics is given as

$$\boldsymbol{x_{eef}} = f(\boldsymbol{q})$$
$$\boldsymbol{q} = f^{-1}(\boldsymbol{x_{eef}})$$

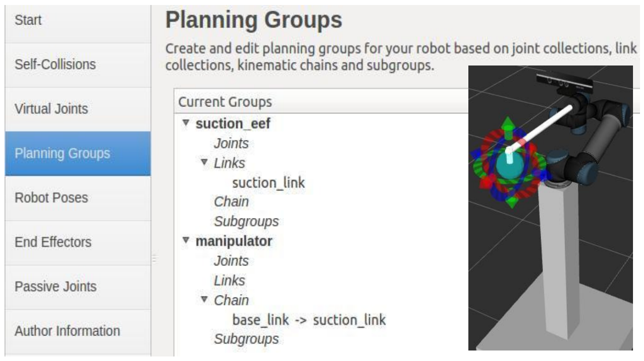
where $\boldsymbol{x_{eef}} = (x, y, z, \theta_r, \theta_p, \theta_y)$ is the end-effector pose and $\boldsymbol{q} = (q_0, q_1, \cdots, q_{n-1})$ is the joint positions of a n-degrees of freedom (DOF) manipulator. The orientation of end-effector pose is specified by θ_r roll, θ_p pitch, θ_y yaw. In algorithm 1, pre-defined values for

$(\theta_r, \theta_p, \theta_y)$ are used for the target end-effector pose based on the direction chosen to pick an object.

Inverse kinematics can be found by solving kinematic equations using different approaches like algebraic method, Jacobian inversion, non-linear optimization. Various open source libraries are readily available for solving inverse kinematics like Jacobian inversion based kinematics and dynamics library (KDL) [16], analytical kinematics solver IKFAST [17], [18]. In the current application inverse kinematics library TRAC-IK is used which runs two algorithms, an extension to KDL Jacobian inversion method and sequential Quadratic programming (SQP) non-linear optimization approach which is presented in paper [19]. TRAC-IK is better compared to the traditional IK algorithms in terms of its success rate and the time required for computation[2].

3.3 MoveIt!

MoveIt! is a set of software packages used for motion planning, environment monitoring, 3D perception, kinematics, trajectory control of robot manipulators[20]. It is a user-friendly platform that can be used to develop industrial, research and commercial robot manipulator applications. Moveit uses thread based architecture and parallelizes the motion-planners and collision checking. As specified in [20], Moveit provides options to interface with multiple motion planning libraries like OMPL[15], search based planning library (SBPL) [21], stochastic trajectory optimization for motion planning (STOMP) [22], optimization based library CHOMP[23]. It is easily configurable with multiple kinematics libraries like KDL[16], IKFAST[17], TRAC-IK[19]. It provides a graphical user interface (GUI) application called moveit setup assistant, the GUI as shown in Figure 5 can be used to generate a moveit configuration package, in this package the plugin for motion planning and kinematics libraries to be used can be specified. For collision avoidance moveit



Planning Groups

Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.

```
Current Groups
▾ suction_eef
    Joints
  ▾ Links
      suction_link
    Chain
    Subgroups
▾ manipulator
    Joints
    Links
  ▾ Chain
      base_link -> suction_link
    Subgroups
```

Figure 5: Moveit setup assistant GUI application to create configuration package and the URDF model of UR5 manipulator with suction cup as end-effector. Image shows how to specify a manipulator planning group in the form of a kinematics chain.

facilitates adding collision objects, octomaps into robot work environment. The collision objects are used as a way to include the rack into planning environment. After the detection of the rack as shown in Figure 4, the bin corners information is used in adding rectangular plates as collision objects to represent four walls of each bin in the rack as shown in Figure 6. Octomaps are 3D volumetric representation of environment which can be generated using the input from a 3D sensor [24], Figure 7 shows the octomap of the rack. Octomap is used in avoiding collision with nearby objects while picking a target object. In the current application moveit is used to

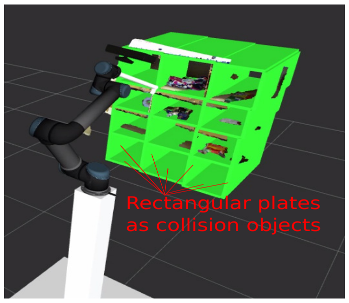


Figure 6: Rectangular plates added as collision objects for four walls of each bin in the rack.

combine the path generation algorithm described in algorithm 1, collision avoidance procedure, inverse kinematics and transferring the joint space trajectory to the real robot. Moveit provides visualization of trajectory for the motion planning in simulation using robot operating system (ROS)[25] and Rviz visualizer.

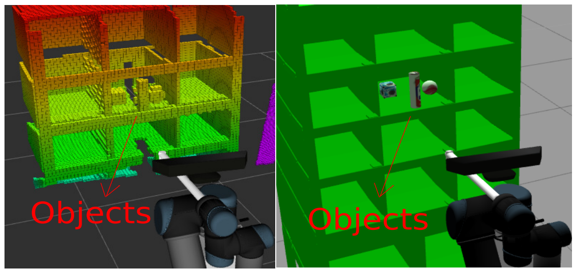


Figure 7: Left image shows octomap generated from the 3D sensor input. Right image shows rack with 3 objects placed in a bin.

4 SIMULATION AND EXPERIMENTAL RESULTS

The algorithms have been implemented on Dell Latitude E7450 laptop with Intel i7 2.6GHz quad-core (used single core) processor and 15.6GB available RAM. All of the programs are implemented on ROS Indigo platform. The experiments have been performed in simulation using Gazebo. The pick and place system using universal robot UR5 manipulator and Barrett WAM manipulator are provided in link [26]. UR5 is a six-axis lightweight, flexible and collaborative industrial manipulator with motor driven joints and payload capacity of 5kgs. Barrett WAM is a seven-axis cable driven light weight manipulator with its actuators placed at its base, it has a payload capacity of 3kgs. In this section the results of auto path

[2]https://bitbucket.org/traclabs/trac_ik.git

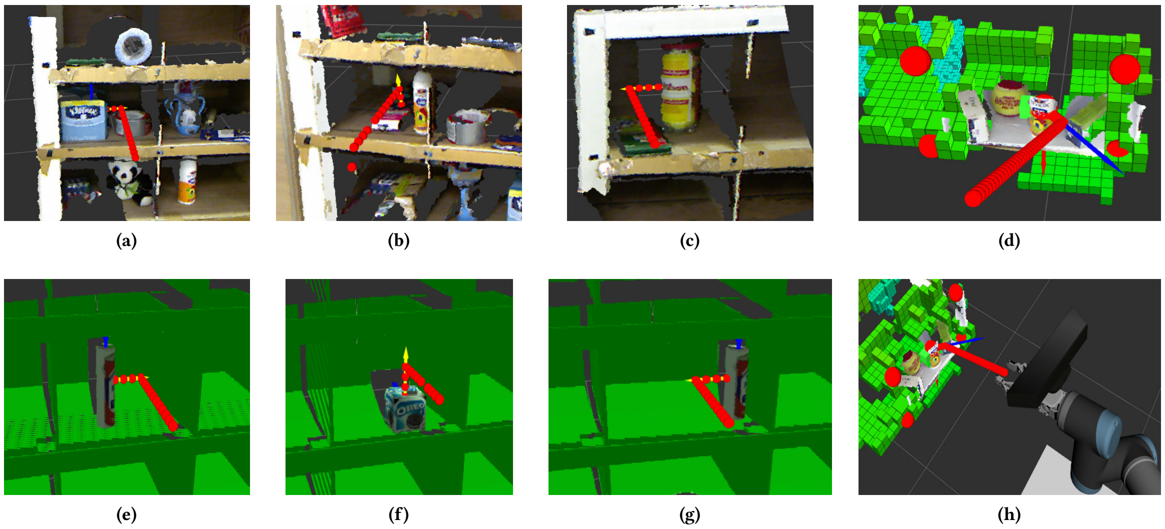


Figure 8: Images (a-c, e-g) show the way points (red color) of auto path generation algorithm for picking objects using suction system. Images (a-c) are for real objects and (e-g) are for objects in simulation. Images (a), (e) shows path for picking from right side of the object; (b), (f) shows path for picking from top side of the object; (c), (g) shows path for picking from left side of the object. Images (d), (h) shows the way points (red color) of path generated using octomap for collision avoidance for picking object using gripper. The normal vector (blue colored arrow) in image (d) shows the approach direction for gripper.

generation, time required for different approaches of path planning have been provided.

4.1 Auto path generation

As described in section 3.1 a straight line path is generated to reach a target object inside the bin. The path generation algorithm generates the path which extends 10 cm away from opening of the bin till the target object. The set of way points along the path are equally spaced at interval of 2cm. The same path is used for both entering the bin and retrieval. The minimum gap min_{gr} required for the suction cup is set at 8cm. The results of path generation for the three different picking directions for suction based grasping system are shown in images (a-c,e-g) of Figure 8. It is observed that the number of way points in a path varies from 10 to 30.

Simulation of generation of trajectory path using octomaps as collision avoidance method have also been performed. The path generation results for picking objects using a gripper based grasping system is shown in images (d), (h) of Figure 8. The inclusion of octomap for path generation helps in avoiding collision with the nearby objects while picking the target object.

4.2 Time requirements

The different poses that the robot arm has to take for pick and place of an object from rack are shown in Figure 9. Home pose is a pose that arm takes before going for picking an object. At bin view pose 3D image of the target bin is captured to detect the target object pose. Joint space trajectory for arm is generated to reach the object which constitutes as grasp pose, then the arm takes the tote drop pose to drop the object. The time required for performing these steps are provided in table 1. The average time required for

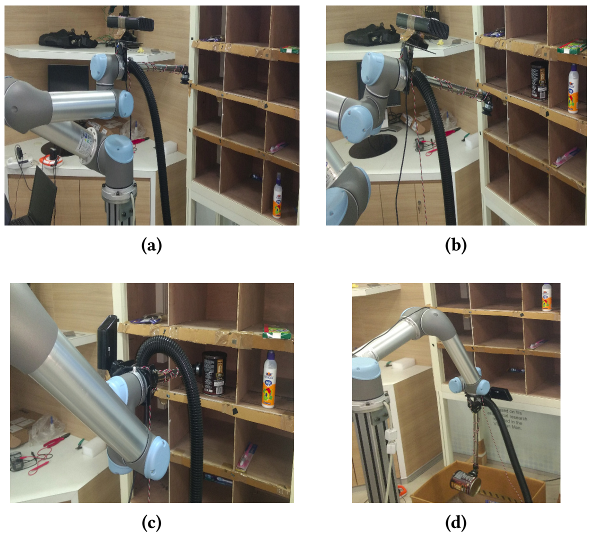


Figure 9: Image shows different poses involved in pick and place of coffee box in a rack (a) home pose (b) bin 4 view pose (c) object grasp pose (d) object drop pose.

generating rectangular plates as rack collision object and the time required for trajectory generations using octomap for collision avoidance have also been tabulated.

Table 1: Time requirements.

S. No.	Component	Description	Time (seconds)
1	Motion 1 (real robot)	Home pose to bin view pose	3.5
2	Auto path generation	Creation of way points	0.0154
3	Joint space trajectory	Inverse kinematics for all the way points	0.015
4	Motion 2 (real robot)	Enter bin to grasp object and coming out of bin	9.11
5	Motion 3 (real robot)	Reach the object drop pose	4.97
6	Motion 4 (real robot)	Motion from object drop to home pose	3.41
7	Rack as collision object	Setting up of rectangular plates as collision objects for motion planning	0.49
8	Octomap collision avoidance	Path generation using octomap for collision avoidance	1.31

4.3 Useful tricks to work with Gazebo simulation and Moveit

Following are some of the useful tricks that can be used to get things working in Gazebo simulation and Moveit.

- In the universal robot description format (URDF) file for robot always specify the world frame link at place where the robot connects with the ground plane. In case the world link is not attached to ground then the robot keeps sliding in Gazebo simulation.
- Use the argument "sim" in moveit planning execution launch file to switch between sending the trajectory joint angles to real robot and robot in Gazebo simulation. This is useful as it avoids creating a separate procedure to send joint space trajectory to trajectory controllers in Gazebo.
- For trajectory path planning using octomap RRTConnect has been used as the default setting in the OMPL config file. RRTConnect is the fastest in finding the trajectory path, other libraries fails to produce the trajectory path most of times.
- In moveit sensor manager launch file the octomap resolution has been set to 2cm, this reduces the computation time for path generation using octomap.

5 SUMMARY AND FUTURE WORK

This paper provides the implementation details of motion planning for picking objects placed in a rack. An algorithm for auto path generation to pick objects using any one of three pre-defined orientations for the manipulator has been described. Moveit is used to combine the path generation with collision avoidance to generate the trajectory in joint space, and execute the trajectory commands on the robot manipulator. The experimental results for the path generation have been provided for robot in simulation and real robot. The use of octomap in planning scene has enabled generation of path avoiding collision with nearby objects. The future effort will be in generation of trajectory for picking objects in any orientation.

REFERENCES

[1] Roger Bostelman, Tsai Hong, and Jeremy Marvel. 2015. Survey of Research for Performance Measurement of Mobile Manipulators. *Journal TBD, March* (2015).

[2] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. 2016. Fetch and freight: Standard platforms for service robot applications. In *Workshop on Autonomous Mobile Service Robots.*

[3] Sarah O Connor. 2013. Amazon unpacked. *Financial Times* 8 (2013).

[4] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine* 29, 1 (2008), 9.

[5] Peter R Wurman and Joseph M Romano. 2016. Amazon Picking Challenge 2015. *AI Magazine* 37, 2 (2016), 97–99.

[6] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. 2010. *Robotics: modelling, planning and control.* Springer Science & Business Media.

[7] Daniel E Koditschek. 1989. Robot planning and control via potential functions. *The robotics review* (1989), 349.

[8] Jerome Barraquand, Bruno Langlois, and J-C Latombe. 1992. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics* 22, 2 (1992), 224–241.

[9] Yong K Hwang and Narendra Ahuja. 1992. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation* 8, 1 (1992), 23–32.

[10] Nick Malone, Aleksandra Faust, Brandon Rohrer, Ron Lumia, John Wood, and Lydia Tapia. 2014. Efficient motion-based task learning for a serial link manipulator. *Transaction on Control and Mechanical Systems* 3, 1 (2014).

[11] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.

[12] Francis Avnaim, Jean-Daniel Boissonnat, and Bernard Faverjon. 1988. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on.* IEEE, 1656–1661.

[13] Frank Lingelbach. 2004. Path planning using probabilistic cell decomposition. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 1. IEEE, 467–472.

[14] Steven M LaValle and James J Kuffner Jr. 2000. Rapidly-exploring random trees: Progress and prospects. (2000).

[15] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. 2012. The open motion planning library. *IEEE Robotics & Automation Magazine* 19, 4 (2012), 72–82.

[16] Ruben Smits, H Bruyninckx, and E Aertbeliën. 2011. Kdl: Kinematics and dynamics library. *Avaliable: http://www. orocos. org/kdl* (2011).

[17] R Diankov. 2010. Ikfast: The robot kinematics compiler. (2010).

[18] Rosen Diankov and James Kuffner. 2008. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34* 79 (2008).

[19] P. Beeson and B. Ames. 2015. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids).* 928–935. DOI:http://dx.doi.org/10.1109/HUMANOIDS.2015.7363472

[20] Sachin Chitta, Ioan Sucan, and Steve Cousins. 2012. Ros topics. *IEEE robotics and automation magazine* 19, 1 (2012), 18–19.

[21] B. Cohen and M. Likhachev. 2009. The Search Based Planning Library (SBPL) [Online]. Available: http://www.ros.org/wiki/sbpl. (2009).

[22] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. 2011. STOMP: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 4569–4574.

[23] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. 2009. CHOMP: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on.* IEEE, 489–494.

[24] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots* 34, 3 (2013), 189–206.

[25] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, 5.

[26] Sharath Jotawar and Nishant Kejriwal. 2016. Experimental video for pick and place using UR5 and Barrett WAM arm. Available: Link 1- https://youtu.be/paq2pk13CKY Link 2- https://youtu.be/CAssP1RjzkI. (2016).