[[1]](#footnote-1)

ME547 Project Report

Nicholas J. Heersink, Anton Dolgovykh, and Mathieu J. Gagne

*Abstract*—This report analyzes the differences between two path planning algorithms, with the main metric to compare the two being the distance travelled. These path planning algorithms have been implemented on a salad-making robot “Saladin” – which ultimately can be extended to any pick-and-place application. The first “naïve” algorithm is the naïve implementation, which lifts the end effector high enough to guarantee collision avoidance. The second uses a probabilistic roadmap to avoid collisions. A full explanation of the derivation and assumptions made to run these simulations has been included in the report. Matlab simulations have been run to show the differences between the two algorithms. In addition, the Fanuc robot is used to demonstrate a naïve algorithm. After analysis, the probabilistic roadmap has been shown to provide the best results in terms of waypoints. Nonetheless, the naïve implementation yields the least standard deviation. Further improvements to provide more conclusive results have also been discussed.

# INTRODUCTION

Many people wish they could have nutritious food without the hassle of preparation and cooking. With thousands of units sold each year, robots become cheaper, easier to use, and generally more available to the public. These robots handle and address a variety of problems, such as the problem of food preparation. A fantastic example is the Moley cooking robot, which boasts a complex arrangement of actuators and sensors [1]. This robot is capable of cooking numerous different meals and performing the various required actions. As an emerging technical field, robotics is quickly progressing and new algorithms are being formed to assist in tasks such as pick and place operations, which are an integral part of these cooking robots. Path planning algorithms are very important in programming these robots, and have been explored from different papers when researching what kind of project the group should do. Notably, some path planning using collisions in 3D space were investigated[2]. Another paper delves into the optimal path when considering the control effort [3]. Both papers will be referred to throughout this report, although this project will have a narrower focus to the time limitations.

For the scope of this project, a simple robot, which could be used to make food, will be explored. Starting from simple salad, the idea for Saladin was formed. This robot would perform pick and place actions, combining salad components in a bowl, while avoiding surrounding obstacles and ingredients. Path planning algorithms were explored in this paper, mainly the naïve algorithm of raising selected ingredients above all others and one that uses Probabilistic Road Map path planning. These two algorithms were compared in performance, answering the question of whether complex planning algorithms are better than a simple overhead implementation that may add a bit of extra cycle time. While this was done with respect to Saladin, and making salads, this problem could easily be transferred to any other robotics applications, especially ones where cycle time and special efficiency is critical.

# System Details and Problem Definition

The bulk of the work related to this project was decided to be performed in a simulation in Matlab. The constraints of the simulations were initially configured to closely match the CRS robot available for use in the ME547 lab room. The problem, as stated in the introduction, was to see the effects on performance of using a conventional path planning algorithm, versus a simple overhead approach to pick and place food ingredients in a bowl. Around halfway through working on the project, the group switched to using the Fanuc robot instead of the CRS as the CRS was giving a lot of trouble in terms of startup and other errors. The environment was thus changed to match the Fanuc workspace. The Fanuc workspace is shown in Figure 1 below.

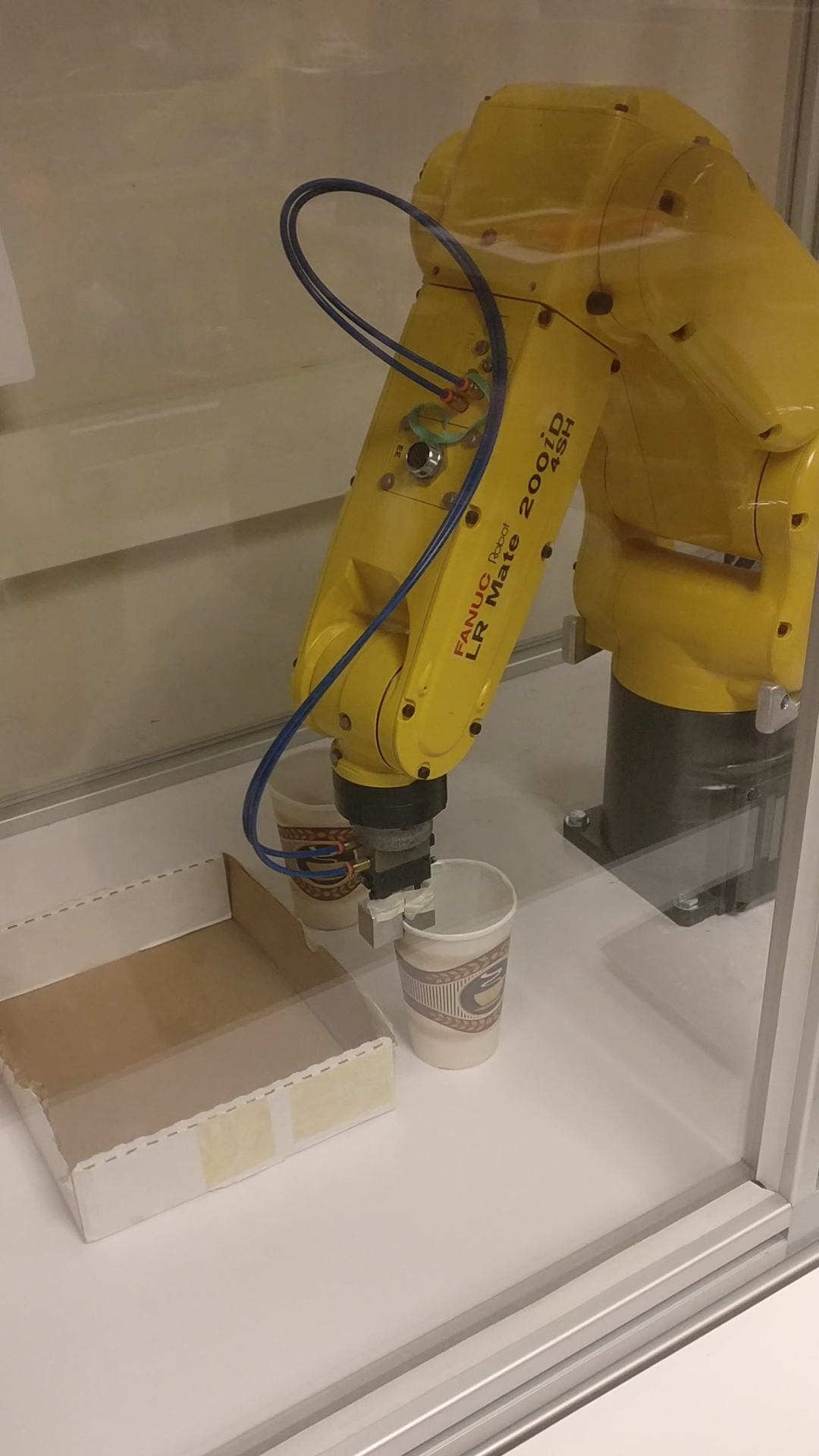


Figure : Fanuc Workspace

The next section outlines the methodology behind the simulation and live verification.

# Implementation

First, the naïve method simulation was implemented, where the initial step was to create an environment containing various containers of ingredients at specified points. The environment also featured a bowl placed in the space. Each object was simulated as a square, for simplicity and the robot was assumed to have the knowledge of the placement and size of these objects. The robot is simulated as a three-member 3D object. A sample environment is shown in Figure 2 below.

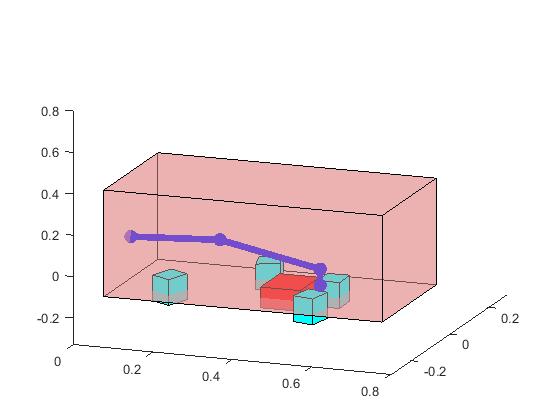


Figure : Sample Simulation Workspace

The naïve method called for the robot to navigate to each ingredient while maintaining a z-axis height that would be higher than all objects. This was a very simply implemented method. The robot navigation was done by calculating the inverse kinematics of a desired point, seeing what joint angles would get the robot to that point. The joint angles were then discretized and put through the simulation to simulate the robot moving as it would reach the specified point. The sequence of the naïve approach was as follows. The first waypoint would be directly above the cup. The second would be placed such that the gripper can grip the container from overhead. Originally the waypoint was alongside the object, however it was changed due to the small grippers of the Fanuc robot when the group switched to using the Fanuc robot. The third waypoint is placed at the same point as the first, directly above the object. A constant offset in the z-direction is used in each case for lifting up the objects above the others to avoid possible conditions when moving the arm and gripped object. Each ingredient is then poured into the bowl. This is done by first navigating to a waypoint above the bowl and then tilting the final joint of the robot so that the cup empties its contents into the bowl. Finally, the container, or cup, is returned to its original position using the same logic as to pick one up. All joints are moved at the same time to avoid strange behavior that may result in dips and the robot going below the required height.

With inverse kinematics, the waypoints are input as (x, y, z) coordinates and the link lengths are known quantities and have been denoted as and . Note that the inverse kinematic equations have already been generated and can be seen below.

Where

The second, more intelligent algorithm implemented is based on a probabilistic roadmap (PRM). In this method, it was assumed that the robot arm would still travel above all the objects; however, when it would reach a cup, it would only lift it off the ground by a minuscule amount, instead of above all the other objects like in the naïve method. Due to this, the end effector would constantly be a threat to collide with other cups, the bowl, or obstacles.

To avoid these collisions, PRM is used to find a path for the arm. Concerning the algorithm, the PRM is first populated with the goal points and the vertices of the objects. In this method, random obstacles were also included. All objects were increased in size based on the radius of the end-effector, meaning any point not touching an object in the simulation would be a safe location for the entire end-effector. Considering the “footprint” allows the end-effector to be assumed as being a point in the PRM algorithm. Figure 3 illustrates the surrounding areas of both the object and end effector.

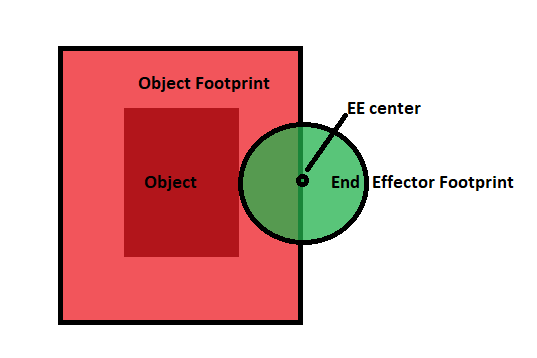


Figure : Object and End Effector "Footprints"

After this step, random points should be generated. Any points which lie within the area bounded by the object vertices should be removed from the set. Straight lines connecting the points can then be easily added. To avoid collisions, any lines that pass through the area of the objects should be removed. After this, A\* can be used to determine the shortest path from your starting position to the desired end goal. A\* uses a combination of Dijkstra’s algorithm and Greedy Best-First-Search to come up with an optimal solution [4]. If the goal is the bowl, the algorithm will calculate the shortest distance to one of the four sides of the bowl. The closest side of the bowl will then be used as the end position. The possible side waypoints of the bowl are shown in the figure below.

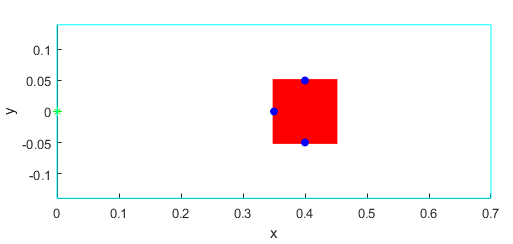


Figure : Waypoints on the Sides of the Bowl

In this method, it is assumed that the cup is picked up from the front, and that the grippers are wide enough to accommodate this. This allows the end effector to dump contents from the cups into the bowl from the side and introduce optimality in where it can dispense its contents. For the simulation, once the arm picks up a cup, it disappears from the map, allowing a trajectory through the previous position of the cup, which makes sense as one would be carrying the cup. The other objects in the map would also further increase in size as the cup’s footprint is bigger than the end effector, requiring a bigger “no-go” zone to avoid objects. Note that this method is implemented in 2D to reduce computational complexity and due to the nature of the problem.

# Novelty

While the idea of having robot path planning has been done in numerous applications beforehand, this specific project outlines the benefits of implementing this algorithm in a specific, constrained environment configuration. This configuration is one where there is limited overhead space for the robot to traverse and the robot will be subject to performing as many operation cycles as it can in a given amount of time. Thus, this project examines the ability to save travel time for the robot end effector for each cycle performed by path planning through objects versus the naïve approach. This project will also highlight whether implementing more complex path planning algorithms is worthwhile in a small-scale application such as salad making.

# Results

The result of implementing the naïve and the more sophisticated PRM approaches are shown below in Table 1. The table summarized the total distances travelled for both the naïve approach and the PRM method. Five different configurations have been used to prove the results for these two algorithms.

Table : Naive and PRM results

|  |  |  |
| --- | --- | --- |
| **Configuration** | **Total Distance Travelled Naïve (m)** | **Total Distance Travelled PRM (m)** |
| 1 | 3.55m | 2.13m |
| 2 | 3.70m | 3.45m |
| 3 | 3.78m | 2.78m |
| 4 | 2.82m | 1.94m |
| 5 | 3.32m | 2.12m |

As previously discussed, the results given as total distance can be directly correlated to the amount of time the object would take to complete a cycle of operation. These results provide a very important conclusion regarding the value of using complex path planning algorithms versus simple overhead obstacle avoidance. They highlight if it is worth implementing path planning algorithms to avoid possible extra travel distance for the robot that would be associated with lifting each object above all other objects and the travel time associated with lowering that object back to its starting position. In this project, it is assumed that all distance travelled is weighed equally. This comparison and analysis of performance is discussed in the discussion section. As a sample, configuration 2 and configuration 5 are shown in the figures below from both the 3D naïve simulation and the 2D PRM simulation. In the naïve configuration, obstacles are not seen as it is assumed the robot arm is always moving above them.

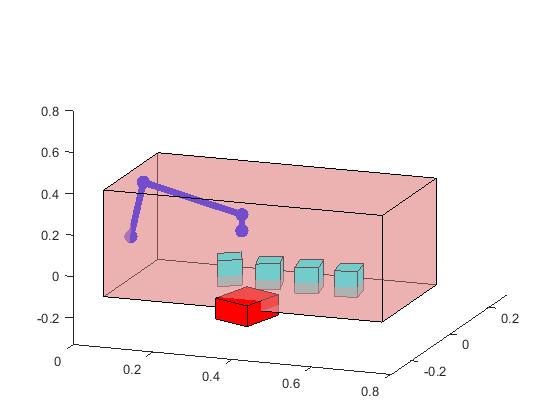


Figure : Configuration 2 Naïve Method

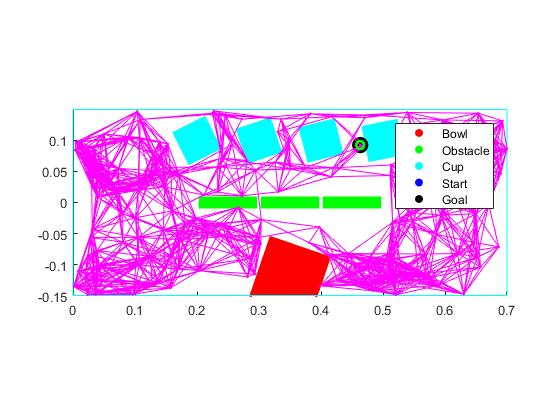


Figure : Configuration 2 PRM Method

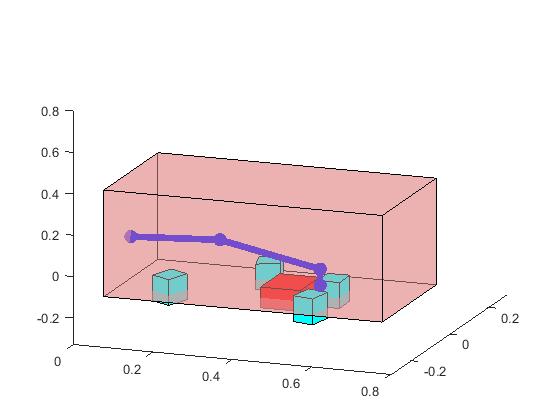


Figure : Configuration 5 Naïve method

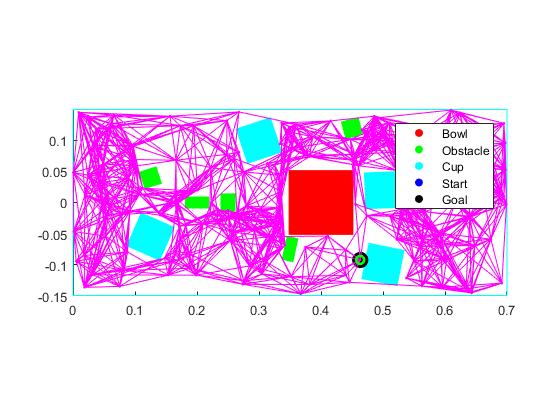


Figure : Configuration 5 PRM Method

Videos have been attached with this document of the robot arm simulation navigating and performing its tasks through each configuration space for each method. In the videos, one can see the disappearing objects for the PRM configuration as the arm reaches the pickup point for the object. The current waypoint and desired waypoint are highlighted in black and blue. The PRM configuration has the legend shown below in Figure 9 for all attached media when differentiating between objects.



Figure : Simulation Legend

A live video is also attached to show the naïve method working in practice with a real robot arm. Note that this video only has two cups instead of the simulated four.

# Discussion

By looking at the results of the simulations, a conclusion may be drawn regarding the value of using path planning algorithms over a simple naïve overhead approach in pick and place applications, specifically here being the makings of a salad. Summarized results have been included in

Table : Summarized Results

|  |  |  |
| --- | --- | --- |
|  | **Naive** | **PRM** |
| **Average Distance (m)** | 3.434 | 2.484 |
| **Average Standard Deviation (m)** | 0.345 | 0.561 |
| **Percent Difference (relative to other method)** | - | -27.7% |

As can be seen from the aggregated results, the PRM implementation yields much shorter travelling distances, with an average difference of 27%. On average, the distance is about 1 metre shorter. Assuming a consistent travel time, this may be a significant amount of time. For any industrial engineer, this reduction in cycle time may be critical.

Note the average standard deviation is larger for the PRM method versus the naïve implementation. This may imply that a more constant run time is possible with a naïve implementation. The total distance may spike when using the PRM method, depending on the configuration of obstacles and cups.

While the method used in this project allowed for a reasonable preliminary conclusion regarding the worthwhileness of using a naïve approach versus a path planning algorithm, it includes many assumptions and cannot yield a definite answer. Each scenario must be carefully considered for the potential benefits and pitfalls.

For example, assumptions such as all travel distances have an equal weight might not be accurate in real life. For instance, lateral movement for the robot may be more expensive in terms of actuation time and control effort when compared to vertical motion - the main method of motion within the naïve implementation. In addition, the methods offer alternate ways of picking up objects, where one is overhead, and one is in line with the object. In actuality, this may not always be feasible depending on the end effector gripper or the size of the object. More configurations could have also been tested to get better results. Finally, both methods assume that, other than the end effector, the robot arm is above all the objects. This may not be possible in many scenarios and may introduce many more complex issues in path planning. Path planning for each of the joints may need to be checked to avoid collisions.

Of course, this study has completely failed to account for computation time. This is obviously a huge factor in determining the suitability of a certain algorithm for path planning. The naïve implementation has a large benefit in this regard, as it does not need to check for collisions. On the other hand, the PRM-based method must check between each waypoint. This may present additional computation time, offsetting any otherwise saved time in traversing to each of the waypoints.

Another large assumption for this whole project is that the configuration of the workspace would change from time to time while also being known to the arm. In reality, this information may not be known *a priori*, meaning that an obstacle detection algorithm such as PRM may be the only suitable method.

Finally, only one method of path planning was tested. Many more exist that may have yielded even better results than the PRM implementation. In the future, this project could be extended by testing other path planning methods such as rapidly-exploring random tree (RRT) or flood fill. It could also be extended by using joint control effort or actual execution time instead of using distance travelled as a measure of cost and performance for the arm.

# Future Work

For the future, many of the aforementioned assumptions may need to be addressed to provide a much more conclusive result. From a control perspective, the command input required is of utmost importance. However, this study failed to consider the difference in necessary joint torques. Going forward, this can be a topic of further study.

In addition, the physical Fanuc robot can be used to run more comparisons. The Fanuc robot will provide a more useful sense of which algorithm will take more computation time within an industrial setting. Also, any assumptions relating to the orientation of the end effector or other links may need to be reworked to provide a more utilitarian perspective on this problem. Other robot configurations, such as the CRS robot, may also be studied in an effort to determine if there are any differences between arm manufacturers.

Another metric that has not been optimized is the required height to lift the cup under the naïve method. Other cup heights may be used in different settings, meaning that the naïve method will need to lift the cup less. At a certain cup height, the naïve implementation will perform better than the PRM method. Therefore, this could be studied to find the critical points for each configuration.

# Conclusion

In conclusion, this project was performed to analyze the validity of using path planning algorithms versus a naïve overhead approach when using a robot manipulator to navigate and perform pick and place operations in a workspace. The immediate concept robot discussed in this project is a salad making robot that would pick and place salad ingredients and place them in a bowl while avoiding obstacles and other ingredients.

Simulation results for the two algorithms were compared. Total travel time is measured between the naïve and the PRM path planning method, as a function of the total travel distance required. A basis conclusion that the PRM method is faster was formed by realizing that the average required distance required is less than that of the naïve method. Nevertheless, the naïve implementation had less standard deviation than the PRM one, meaning that a naïve algorithm may be better suited to avoiding large spikes in travel time.

A live demo of the naïve method is also performed to make sure that the simulation is transferrable to the actual Fanuc robot arm. This validated our ability to make salads, or otherwise perform proper pick and place operations without hitting any other objects when employing offline waypoint generation. This project provided an extremely important insight not only for the case of a salad making robot, but also any sort of pick and place associated robot as to whether it is worth considering spending time and resources to come up with and program path planning algorithms for the robot, versus just going above all the objects if this is possible.

# References

|  |  |
| --- | --- |
| [1] | Moley Robotics Inc, "The World's First Robotic Kitchen," 2015. [Online]. Available: http://www.moley.com/. [Accessed 4 April 2018]. |
| [2] | C. Qin and D. Henrich, "Path Planning for Industrial Robot Arms - A Parallel Randomized Approach," University of Karlsruhe, Karlsruhe. |
| [3] | J. A. Cascio, "Optimal Path Planning for Multi-Arm, Multi-Link Robotic Manipulators," Naval Postgraduate School, Monterey, 2008. |
| [4] | A. Patel, "Introduction to A\*," Stanford University, 2 March 2018. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. [Accessed 4 April 2018]. |

1. [↑](#footnote-ref-1)