University of Pennsylvania
Fall 2018

MEAM 520: Introduction to Robotics

Final Project: Manipulating potential fields for dynamic robot path planning

Group 30: Akhilesh Bhat, Samarth Kalluraya, Sonitabh Yadav

---

1 Introduction

Potential field path planning has good computational speed and generates smooth paths. But however, it has a few other problems like getting stuck in a local minimum or not being able to reach goal points that are close to obstacles. In our project we aim to tailor various parameters, used in setting up artificial potential fields, to a specific application . We plan to alter the repulsive forces acting on the robot, vary the distance at which these repulsive forces act on the robot depending on the position of the robot and add extra artificial repulsive fields to guide the robot on a desired path.

To implement these ideas, we defined the problem statement as follows. A few colored blocks will be placed in a jumbled order in predetermined positions. The goal is to use the Lynx robot to rearrange these blocks to a goal color sequence. To do this a camera will determine the color sequence of the blocks and will provide this information to the program. The program will then find the optimum sequence of rearranging the blocks. Then the potential fields will guide the robot to pick up the blocks and place them in the desired positions.

Some of the problems that we expect to occur and that we plan to overcome are local minimum occurring due to the non-convex shapes of the cubes, difficulty in placing the blocks between two other blocks due to the

2 Method

2.1 Detecting Sequence of Objects using CV

We implemented computer vision to detect the order of the color of blocks in front of the user (as goal configuration) and the order of current position of different color blocks in front of the Lynx (as current configuration).

We start by acquiring the image, first of the current order of the blocks and then of the goal order. Since the Lynx is just copying the configuration of the blocks and not the exact motion of the user, a continuous tracking is not required. So just the final image is acquired and processed. To determine the order, we needed to look at the different pixel values after the image is acquired and try to determine the color of that pixel and extract it. The system needs to know what colors to look for in an image. Here we chose the Red, Green and Blue. Also, the resolution of the image is also reduced to decrease the total number of

pixels being processed. This helps reducing the computation time, since pixel-wise accuracy is not a major factor here, but speed is. Once the pixels of the known colors are identified, we need to read the RGB values for all the pixels. Due to various reasons like lighting and image noise, these pixels may not represent the correct value corresponding to R, G & B planes. Hence, we set a threshold to select the pixels in the R, G&B planes and then blow them up to their maximum value.

We then use color segmentation to detect the colors in the reinforced image. Here, we turn the planes of R, G and B into binary images so as to have distinct values from the rest of the image wherever the given color is present. Next, we use the MATLAB function greythresh() to implement Otsu's thresholding method to generate a black and white image for each of the colors, representing the colors position in white.

After determining this, we go on to reconstruct the image. We start by cleaning the image for any stray pixel value and cleaning the border pixels from stray values. This is done for all the three RGB images in black and white format obtained in the previous step. We label the pixel position as 1 if that color is present there, otherwise zero.

After the cleaning and labeling, we simply combine the three black and white images to for the final image, which would just show the colors on a black background.

The red layer will have a label of 1 on position where Red is present.

The green layer will have a label of 1 on position where Green is present.

The blue layer will have a label of 1 on position where Blue is present.

The original and reconstructed image is shown in figure 2.1

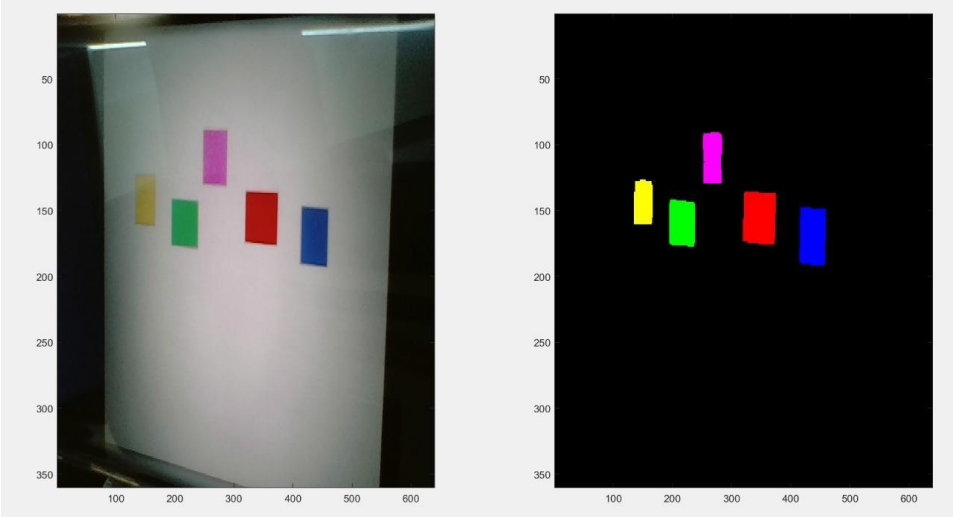

Figure 2.1 Output of CV.m

Once we know where the 5 colors are on the image map, we extract the position of each pixel where the 5 different colors are present and compute the median of the value to get the position of the center pixel for

each color. This position is then read and helps us determine the order by simply sorting. The output will thus be the goal order and the current order which will be used as input by the arrange.m function.

2.2 Sorting

The arrange.m function takes the original order and the goal order as detected by the CV.m function as the input. Here, original order is the order of the blocks that need to be reordered while the goal order is the order in which the blocks that need to be reordered.

The CV.m function returns the original order and the final order in terms of the colors detected by the camera. The arrange.m function first orders the goal order in numbers which will make it easier to guide the Lynx from one block to the other. We then order the original order according to the number of the respective color in the goal order. We thus convert it into a number sorting problem.

Since we are adding an extra position in the simulation to help with the rearrangement, we add an extra element in the original order and goal order as well. We use structures to store the arrangement order to be used by the main function to arrange the blocks.
The main principle used for arranging the blocks is a greedy approach. We first find the block number which is in the position that is at the farthest distance from its goal position. We place this block in the extra position that we have defined. We then check which block is supposed to be in the position that was vacated by the block that we moved to the extra position. After placing the correct block in the vacated position, we carry out the same process till the blocks are in their correct position.
For example, suppose we have [2, 4, 5, 1, 3] as the original order and [1, 2, 3, 4, 5] as the goal order. After adding the extra position, the original order and goal order are [2, 4, 5, 1, 3, 6] and [1, 2, 3, 4, 5, 6] respectively. The vector having the differences in the positions is given by [1, 2, 2, -3, -2, 0]. Here, we can see that the absolute value of the maximum difference is 3. So we move block 1 to the extra position, which then makes the intermediate order to be [2, 4, 5, 6, 3, 1]. Since the position 6 is empty, we transfer that to the new position that has been vacated. Now, on checking which block is supposed to be in the vacated position, we find that the block we want is block 4. So the new intermediate order becomes [2, 6, 5, 4, 3, 1]. This continues till we get the goal order or if we get to the case where 2 of the blocks are swapped.

In such a case when 2 of the blocks are swapped, we pick any 1 of these blocks to be placed in the extra position and place the other block in its final position before placing the former block in its goal position. The code takes care of the case when there are multiple such cases of swapped positions by checking for them every time in the outer loop.

2.3 Potential Fields

2.3.1 Approach

Every time the Lynx has to pick up an object, the program would generate an attractive field at the center of the object and set up repulsive fields around all other objects. Thus, the robot will travel to that object and will then grip the object. Once it has gotten hold of the object, the potential fields will be updated and

the location at which the object is to be placed will generate an attractive field. Therefore, for every new motion, that is every time a new block must be moved, a new map of the obstacles has to be generated.

2.3.2 Modified $F_{repulsive}$

During our initial tests we saw that the robot used to travel very close to the surface of the obstacle and sometimes if the step size was large, the robot joint would enter the volume of the obstacle. In order to understand the behavior of the robot under repulsive forces we plotted a graph of $F_{rep,i}(q) = \eta_i * \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_o}\right) * \frac{1}{\rho^2(o_i(q))}$ against $\rho(o_i(q))$. (Note that the $\nabla\rho(o_i(q))$ term is ignored because it is a unit direction vector and we can assume that the repulsive force is from a flat plat acting on a point). Two curves were plotted with value of η as 1000 and 10,000 and ρ_o was chosen as 50 mm. The graph is shown in the figure below. We can see that the repulsive force is 0 for most of the distances below 50mm but then drastically increases over a very short span of distance. Thus, the robot does not feel any repulsive force till it comes very close to the obstacle and thus travels very close to the obstacle. Although increasing η helps increase the gap between the robot and the obstacle, the effective force still acts on a very short span (in this case from approximately 12mm to 3mm the force goes from 0 to near infinity).

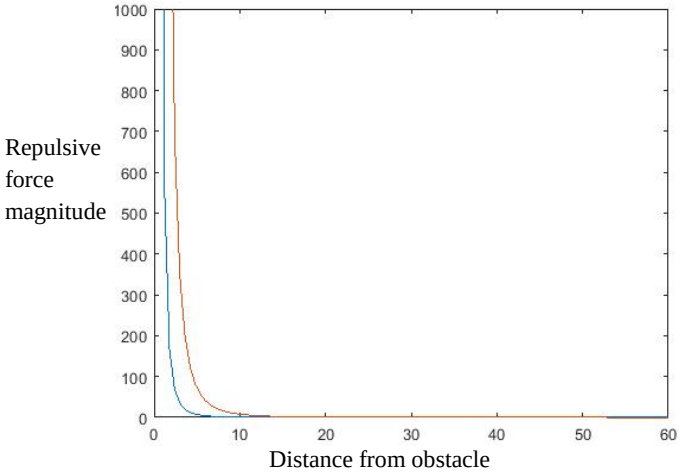

Figure 2.2

In order to create more gap between the robot and the obstacle and have a gradually increasing curve of repulsive force, we decided to use a slightly modified version of the repulsive force equation.

$$F_{rep,i}(q) = \eta_i * \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_o}\right) \dots\dots \rho(o_i(q)) \leq \rho_o$$

$$F_{rep,i}(q) = 0 \qquad\qquad \dots\dots \rho(o_i(q)) > \rho_o$$

As you can see the term $\frac{1}{\rho^2(o_i(q))}$ has been omitted from the equation. This helps to increase the repulsive force gradually. Hence the obstacle will push the robot even when it is far from its boundary. This can be seen by the graph shown below where the two new curves for $\eta = 1000$ and $\eta = 10{,}000$ have been plotted along with the previous two curves.

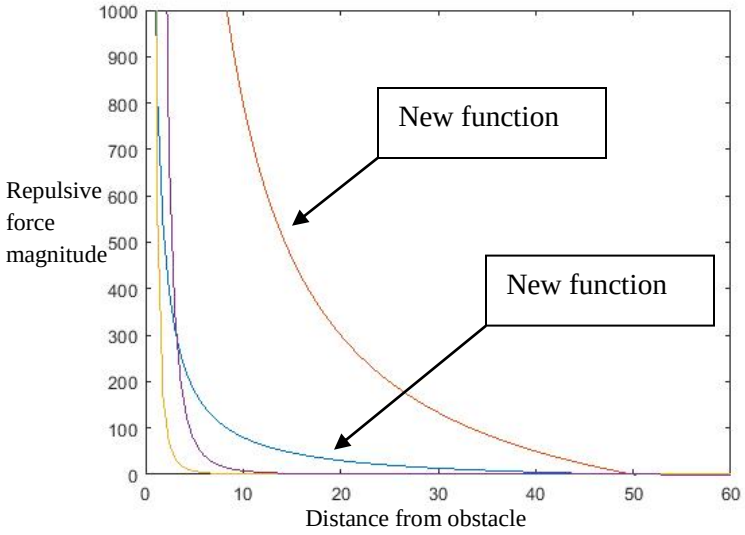

Figure 2.3

2.3.3 Direction and affective distance of repulsive force

Generally, the repulsive forces from the obstacle is taken in the direction perpendicular to the surface of the obstacle. This becomes a problem if the surface is flat as there are high chances of the robot getting stuck in some local minima. To avoid this, we treated each obstacle as a sphere, that is, the direction in which the repulsive force acted was the direction vector joining the center of the obstacle to the joint of the robot. The minimum distance however was still taken to be the distance between the joint and the closest point on the obstacle. This method helped eliminate local minima due to the shape of the obstacle.

Since the objects will be placed close together, having a large ρ_o, which is the distance up to which the repulsive force will be effective, will cause two repulsive fields of adjacent blocks to overlap each other. Figure 2.3 shows the boundary of influence of the repulsive force around two adjacent blocks in a 2D plane. Thus, if a robot intended to place another block between these two blocks, it would find it very difficult to travel between these two blocks and chances are that it will be stuck in a local minimum. On the other hand, if we reduce the value of ρ_o, the effective field will become very small and the distance above the block till which the repulsive force acts reduces. Thus, there are greater chances of collision when the robot is transporting blocks above these blocks.

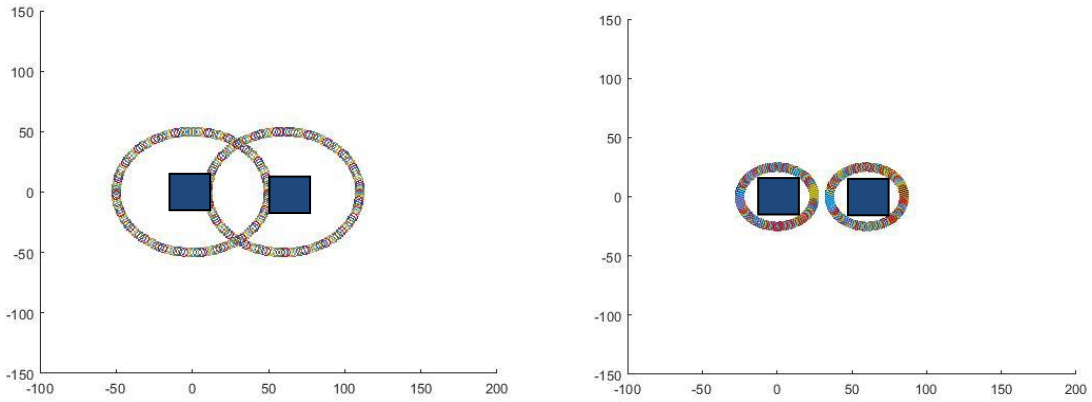

Figure 2.4

To avoid this situation, we made the effective repulsive distance as a function of the position of the joint. Our target was to use a function that will have a large value of ρ_o when the robot is above the block and a small value of ρ_o when the joint is on the sides of the block. Another advantage of such a field would be that it would help guide the robot to the position between the two blocks thus helping the robot to maintain its orientation.

The value of the new distance $\rho_o{}'$, is given by the equation,

$$\rho_o' = \frac{\rho_o}{0.2 * \sin(tz)}$$

Where, 'tz' is the angle made by the line joining the joint and the center of the block with the positive Z axis.

Thus, as the robot moves above the object, the angle 'tz' decreases to 0 degrees. Therefore, the value of ρ_o' becomes five times the value of ρ_o. On the other hand, when the robot joint comes closer to the side of the robot, the angle 'tz' increases and hence the value of $\rho_o{}'$ decreases to a minimum value of 0.833 times of ρ_o. The figure below shows the area of influence of the repulsive force due to this change.

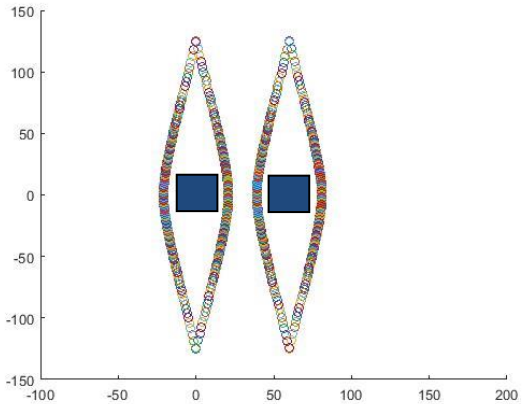

Figure 2.5

2.3.4 Guiding obstacles

We added a few extra obstacles such as walls and the table platform to guide the motion of the robot. With the help of these obstacles we intend to limit the space in which the robot can move. In a normal potential field, the robot tends to take a long and round about path. However, by adding a few extra obstacles we can force the robot to follow a straight-line path which also helps reduce the distance travelled by the robot. In the figures below, we can clearly see that while the traditional potential field makes the robot take a roundabout path around the obstacles, our modified field forces the robot to follow a straight line path without much deviations (the path traced by the end effector is shown in the figure).

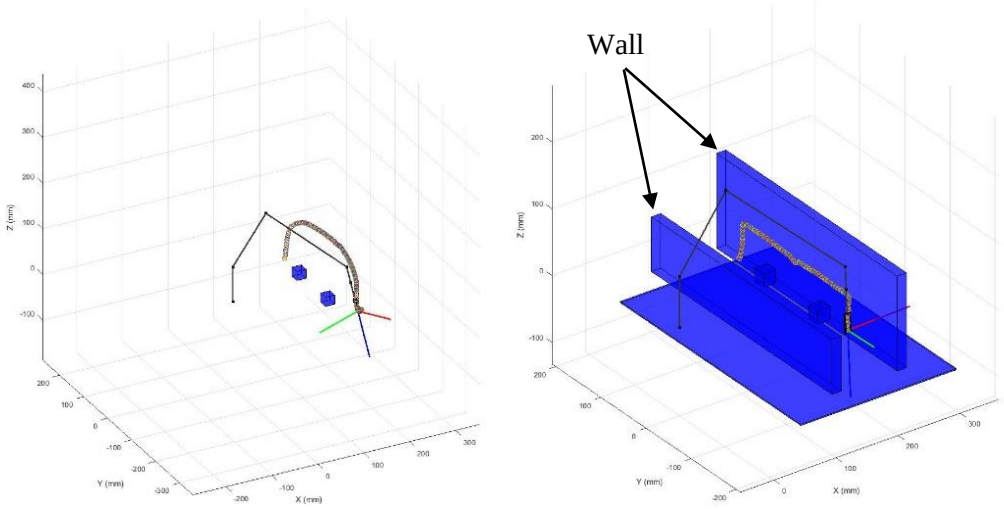

Figure 2.6

3 Evaluation

The main function integrates the results of all the different modules and then carries out the simulation on the Lynx as well as a virtual simulation. All the movements, including the transition between the goal position of one movement and the start position of the next movement, have been implemented using potential fields. For each set of movement, we have 2 different motions. For example, for the motion [1 6], there are two motions. The first from the previous position of the end effector to the position of the first block and then from the first block to the 6th position. Therefore for ‘n’ motions for arranging the original order into the goal order, we get ‘2n+1’ movements. For each of these ‘2n+1’ movements, the set of obstacles need to be different. The blocks that need to be picked and the position where the block needs to be placed cannot be listed as obstacles. So, for each movement, we generate different maps which are stored in a structure and are numbered corresponding to the order of movement. The potField.m function is then called with the correct q_start, q_goal and the map to generate the required motion. Sufficient pauses have been given while performing the experiment on the Lynx to prevent the joint angular velocities from exceeding their limits.

To evaluate the new potential field function that we generated, we compared it with the traditional potential field that we used in Lab 5. We evaluated the two potential fields based on the following parameters:

- Distance travelled by the Lynx after picking up one block to the position where it places the block.
- Minimum proximity of the end effector to the obstacles

The tests were run in a simulation environment for the same start and end points on the two fields and field parameters were kept constant,

$\boldsymbol{\zeta} = [1, 1, 1, 10, 10] * 10^4$ …(Attractive field strength (N/m))

$\boldsymbol{\eta} = [1, 1, 1, 1, 1] * 10^7$ …(Repulsive field strength (Nm3))

d = 30 …(distance from the goal point within which the parabolic potential starts to act (mm))

$\boldsymbol{\rho}$ = 10 …(distance of influence around the object within which the repulsive force acts (mm))

We obtained the following readings.

Table _ shows the readings taken when the potential field from Lab 5 was applied to the predefined positions of the blocks as compared to the observations from the new potential fields.

Start Coordinates	Goal Coordinates	Traditional Potential Field		New Potential Field	
		Distance Travelled	Minimum distance to obstacle	Distance Travelled	Minimum distance to obstacle
[160 -90 10]	[160 150 10]	349.4983	7.9967	300.7098	1.8436
[160 30 10]	[160 -150 10]	301.3719	7.9967	242.8310	1.3707
[160 150 10]	[160 -150 10]	467.1232	7.888	358.7384	2.3104
[160 -90 10]	[160 -30 10]	61.3650	7.9967	53.8642	2.5297
[160 -90 10]	[160 30 10]	253.5065	7.9967	167.0942	1.8436

Table 1

From the table above, we can clearly see that the distance travelled by the end effector in the new potential field is significantly lesser than in the potential field from Lab 5.

We then executed the code of rearranging the blocks on the Lynx robot. We performed the experiment on a set of three, four and five blocks and noted down the following observations.

Sr. No.	Current order	Goal order	Time required	Result
1.	["blue", "green", "red"]	["green", "red", "blue"]	203.83s	Success without collision
2.	["blue", "green", "red"]	["red", "green", "blue"]	220.52s	Success without collision

3.	["blue", "red", “green”, "pink"]	["red", "green", "blue", “pink”]	367.39s	Collided
4.	["blue", "green", “pink”, "red"]	["red", “pink”, "green", "blue"]	495.27s	Success without collision
5.	["blue", "pink", "green", "red", "yellow"]	["green", "red", "blue", "yellow", "pink"]	1057.86s	Collided
6.	["blue", "pink", "green", "red", "yellow"]	["yellow", "red", "green", "pink", "blue"]	1472.59s	Collided

Table 2

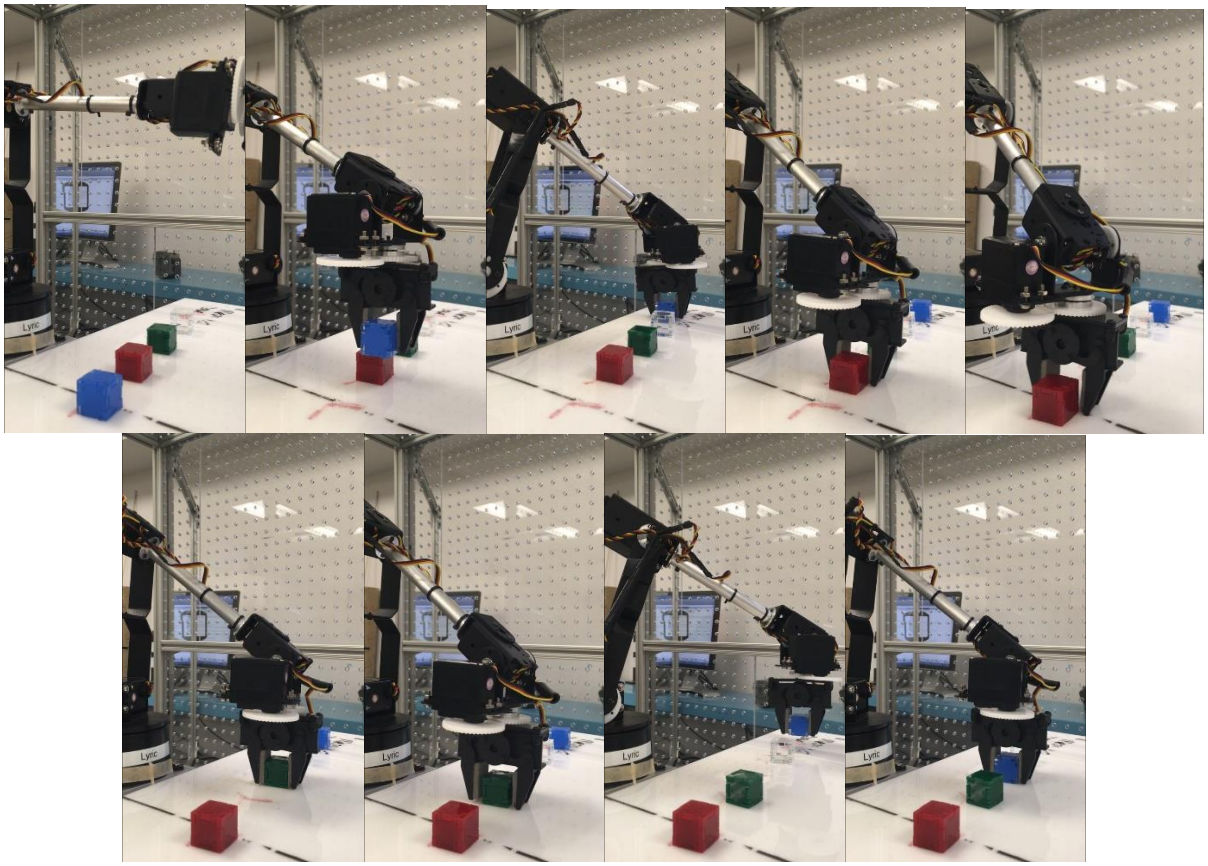

Figure 3.1 Series of photos showing the robot arranging the blocks (Observation 4). Link to video: https://drive.google.com/file/d/1S7Vy-WXv0lnNyU_JlajGLU83aHhv2Uwm/view?usp=sharing (14x speed)

Video link for observation 1 : https://drive.google.com/file/d/1YvgGO6XndLBHX9eolEkQkZ-mm_TtbTBL/view?usp=sharing (14x speed)

4 Analysis

4.1 New potential field

The potential field worked very well in navigating between the two closely placed blocks and placing the third block between them. Also, no local minima were observed when executing the program either on the Lynx robot or on the simulation. We were also able to achieve a near straight line trajectory using the artificial walls (figure 4.1 shows the path without the walls and figure 4.2 shows the path followed with the walls, we can also see that the path in figure 4.1 goes outside the X-Y plane which does not happen when the walls are present) . This helped in simplifying the path travelled by the end effector.

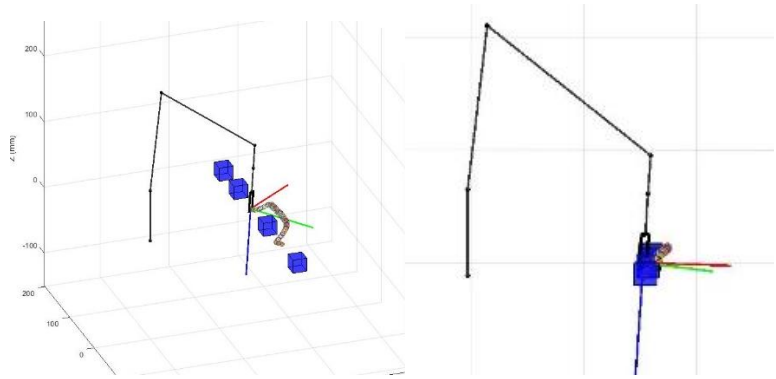

Figure 4.1 Potential field without walls

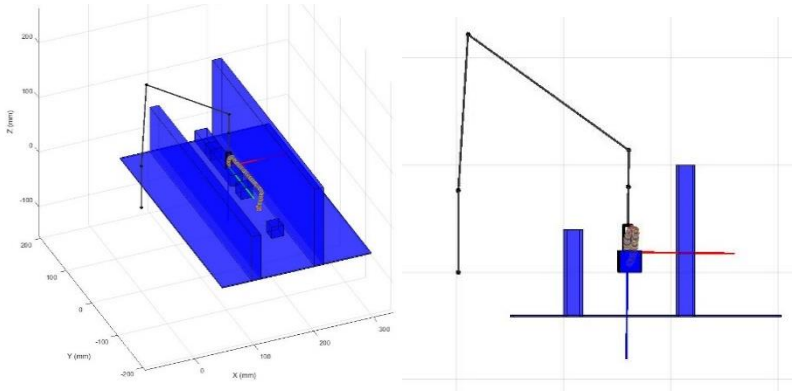

Figure 4.2 Potential field with walls

It is important to note that our approach and the modified potential field that we used is extremely problem specific and it is hard to generalize the potential fields for multiple applications. However, if a robot is supposed to be used for a specific application a large number of times, suitable fields can be set up that would satisfy the requirement even if the actual position of the blocks are changing.

4.2 Performance

The simulation of the program ran smoothly and the actual execution of the program on the robot also ran smoothly to some extent. We had to run the Lynx robot slowly because there were a large number of

intermediate points that the robot had to follow and if we reduced the time gap between execution of two points then the angular velocities of the joints would exceed its limits.

The major issue we ran into, when working the Lynx robot was the inaccuracy in positioning of the robot. This caused collisions of the block being transported with the stationary blocks. We particularly observed this to happen as we increased the number of blocks to be arranged. As the number of blocks increased the execution time also increased and hence error built up over time. Thus, although initially there were no collisions, towards the end of the program the blocks being transported would scrape the stationary block beneath it. This was the reason for the failure cases that we have reported.

4.3 Comparison

The evaluation validates that the distance travelled by the end effector and the proximity of the end effector to the obstacles will be much less in the new fields as compared to the fields from Lab 5. This when considered on a larger scale will help reduce energy consumption as energy consumption is directly proportional to the distance travelled by a robot. Such fields can be used in places where space is a constraint and we don't want the arm to wander away too much from a certain path.

5 Future Scope

One area of work would be to integrate the camera on the robot to help in positioning of the objects and the obstacles in the work environment. Our work on potential fields can be further expanded to various other scenarios like arranging objects in a circle or stacking objects on each other. In each case new attractive or repulsive fields would be needed to be designed that would suite that requirement. Also, in our case the blocks to be moved were stationary. We could thus apply the methods used in this project to applications where the blocks are moving. For example, such potential fields could be used to rearrange items that are moving on a conveyor belt. Defining the distance of repulsive force as a function of position can help robots navigate into narrow spaces. This could have many applications in mobile robot path planning.

6 Acknowledgement

We would like to thank project group 7 who shared their setup of platform and colored blocks with us.