Design and Development of Path Planning Techniques for a Tennis Ball Retriever Robot

M.F.Ali[1], C.K.Lam[1], K.V.Ch'ng[1], K.Sundaraj[2] and W.-H.Tan[1]
[1]*School of Mechatronic Engineering, Universiti Malaysia Perlis (UniMAP), Kampus Pauh Putra, 02600 Pauh, Perlis. Malaysia.*
[2]*Faculty of Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka (UTeM), Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia.*
lckiang@unimap.edu.my

***Abstract*—During a tennis solo training, players usually train using an automatic ball launcher machine. After some time, they are required to collect all the balls scattered all around the court themselves to refill the launcher machine. This is a physically challenging procedure, which is generally loathed by keen tennis players and may cause unwelcome injuries. This study aims to design of an autonomous tennis ball retriever that will discard all the unnecessary energy and time wasting in traditional ball picking up method. This robot will sweep all the balls using a suitable path planning technique. After this, a few path planning methods such as Coverage Path Planning (CPP) U-Turn, CPP ISS, and Probabilistic Roadmap Method (PRM) were integrated into the tennis ball retriever robot for comparison. After comparison between all the experiment done, CPP U-Turn is proven the best path planning method among the three tested algorithms to be integrated into a tennis ball retriever robot.**

***Index Terms*—Coverage Path Planning; Mobile Robot; Probabilistic Roadmap Method; Tennis.**

I. Introduction

In this era, the whole world concentrates on modernisation, automation, industrialisation and development. Some industries are currently trying to replace humans with robots to improve efficiency, accuracy and time. Robots in industries can be separated into arm robot (arm manipulators) and a mobile robot. Arm robots usually used to handle products while mobile robots are used to carry, move, collect, etc. the products. Now, there are the even humanoid type of mobile robot developed that can do various type of task [1]. Arm robots' movements are limited to their joint, kinematics and dynamics while mobile robots are limited to its geometry and Degree of Freedom (DOF).

This research focuses on the design and development of path planning techniques for a tennis ball retriever robot, which is to help tennis players collect the balls during their training. During a tennis solo training, players usually train using an automatic ball launcher machine. After the training section finished, they are required to collect all the balls which are scattered all around the court themselves in order to refill the launcher machine. This is a physically challenging procedure, which is generally loathed by keen tennis players and may cause unwelcome fatigue. This procedure, not just energy consumption, but also takes an extended period of time.

To help them, an autonomous tennis ball retriever robot is developed to discard all the unnecessary energy and time wasting in traditional ball picking up method. This robot can sweep all the balls by integrating a suitable path planning technique and uses appropriate sensors to detect and avoid obstacles.

II. Literature Review

When talking about a mobile robot, the significant problems that will pop up are path planning and motion control. According to Strandberg [2], robot path planning is about finding a collision-free motion from one position to another. In planning a path for a mobile robot, a few factors should be considered which are the surrounding of the robot workspace, the path planning algorithm, and types of path planning techniques.

A. Robot Locomotion

A robot can be divided into two types: stationary and mobile robots. Stationary robots are a robot that cannot move or in other words, has fixed position. Then, what is a mobile robot? Lima and Ribeiro [3] described mobile robots as a device with considerable flexibility within its surroundings. Mobile robots also must have a system with the following functional characteristics; mobility, which means it has total freedom of movement relative to the surroundings, need very little of human monitoring to make it less human-dependent and perception ability which is sensing and reacting in any situation. To summarise, the primary keys of the mobile robot is it is capable of moving around its environment and also capable of being autonomous in navigating itself. These two characteristics are the very opposite of industrial robots which are fixed to its station and depend on its hard-coded codes to perform its repetitious tasks.

As the demand for mobile robot increasing in this era, many types of the mobile robot are design and developed to be used in various fields. However, the most common and widely developed by researchers are legged robots and wheeled robots.

Wheeled robots, or also well known as Wheeled Mobile Robot (WMR) is mobile robots that move around an environment using powered wheels (usually with motors) to drive themselves. WMR is widely and most commonly used by researchers and engineers as they are easy to design, execute and efficient for robots that require speed. They also have more excellent stability in static and dynamic motion than legged robots as their centre of gravity does not change when the move or standing still [4].

WMR does not require complicated and challenging algorithms and designs. However, the most wheeled mobile robot is not reliable in rough terrain as they became uneven and unstable and also not useful in the very smooth surface as they tend to skid and slip. WMR may have a various number of wheels, but for static and dynamic balance, three wheels are considered adequate.

The robot's design and requirements should be focused on to choose the best and effective wheel for the robot. Fixed wheels are decent for merely connecting wheels to a motor and steering. Spherical and orientable wheels are right at balancing a robot. Omni wheels are an excellent choice for both steering and driving, but they are costly with reduced efficiency. Slippage of wheels may occur when they are used for positional control making it unreliable.

Four-wheeled robot. Most WMR is of this type. This configuration is most straightforward to design and build. The first drawback of this configuration compared to the three-wheeled configuration is the extra cost of the fourth wheel and sometimes an extra motor to propel them, but their advantages surpass these drawbacks.

This type of robot can be divided into three configurations; 1) two powered, two free rotating wheels, 2) two-by-two powered wheels for tank-like movement, and 3) two-by-two powered wheels car-like steering. Most of the projects regarding tennis ball collector review in this thesis use this kind of configuration [5-8]. Figure 1 shows locomotion of two powered wheels with two free turning wheels.

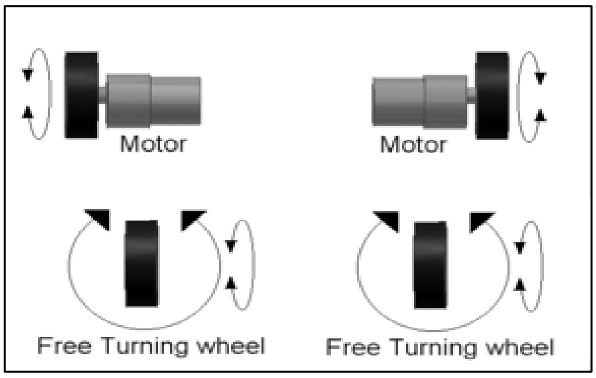

Figure 1: Two powered, two free-turning configurations

B. *Path Planning Techniques*

Before going further into path planning techniques and its applications, researchers usually need to know what path planning is, why is path planning important in mobile robots and what are the key aspects in developing a path planning. According to Strandberg [2], robot path planning is about finding a collision-free motion from one position to another. A path planning is considered not sufficient when the robot bump with obstacles, stray away from the path or takes too much time to reach the destination. In planning a path for a mobile robot, a few factors should be considered which are the surrounding of the robot workspace, the path planning algorithm, and types of path planning techniques.

The surrounding or the environment of the workspace can be divided into two; static and dynamic. Static is when all object or obstacle in the workspace is not moving while dynamic is when the obstacles can move for example another mobile robot sharing the same workspace or human moving around. Path planning algorithm also can be separated into two; global and local. Global path planning algorithm is when all the obstacles in the workspace are static and known to the robot before it even starts. A slight change in obstacle location may disrupt the robot motion. On the contrary, a robot with local path planning algorithm does not know the workspace obstacles has to create its own path while moving. With the help of sensors, the robot will try to do its task while avoiding obstacles [9].

According to Cai [10], the intelligent robot path planning can be divided into two aspects; the one is point-to-point optimisation path planning, the other is complete coverage optimisation path planning. Currently, many projects and researchers are about point-to-point optimisation path planning, on the other hand, complete coverage optimisation path planning is not that popular among researchers. Enric and Marc [11] defined Coverage Path Planning (CPP) as a task finding a path that goes over all points of a region or any specific space while avoiding obstacles. CPP plays a significant role in robots like collector robot, vacuum cleaning robot, lawn mowing robot and so on.

1) *Coverage Path Planning (CPP)*

According to Enric and Marc[11], Coverage Path Planning (CPP) is the function of finding a path that goes through all points of a region or specific spaces while avoiding obstacles. This task is vital in many robotic applications, such as vacuum cleaning robots, autonomous mobile robot underwater, a paint sprayer robot and many more. The primary concern in this technique is usually the time taken for the algorithm to cover all the free space in the configuration space. Apparently, it may take a long time to generate a path that covers all the possible points in the configuration space. Moreover, if the configuration space has moving obstacles, this problem becomes much more complicated and computationally demanding if CPP is used.

In the earliest research related to CPP by researchers in [12], they listed down the requirements and criteria a robot must meet to perform coverage operation. Below is the list of requirement and criteria mentioned in Journal of Robotic Systems [12]:

i. A robot must navigate itself through all the points in the target area completely
ii. The robot needs to move in the region without overlapping its own taken paths
iii. Constant and orderly operation without any repetition of paths is requisite
iv. The robot needs to avoid all obstacles
v. A simple motion such as straight lines and circles should be used to simplify the control
vi. An “ideal" path is preferred under existing conditions

Choset, in his survey [13], has classified CPP into two algorithms; off-line algorithm and on-line algorithm. Online algorithms rely only on immobile information, and the environment is presumed to be known beforehand. However, sometimes, in some cases, assuming full prior knowledge of the environment might be unlikely. Contrariwise, on-line algorithms will not presume full knowledge of the surroundings and depend on real-time sensor measurements to navigate through all the target space. These algorithms are now also called as sensor-based coverage algorithms. Figure 2 shows a typical zigzag or U-turn path design by Enric and Marc [11]. Shaded area indicates the covered path (darker) and the uncovered area (lighter) when the robot completed trailing the zigzag path.

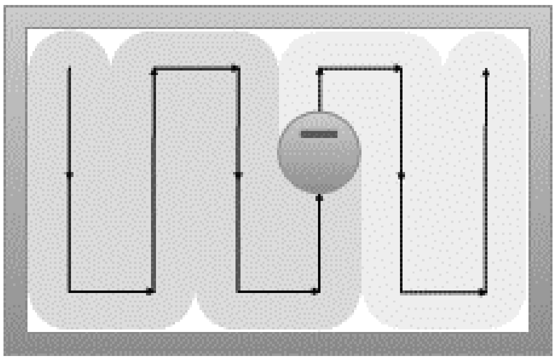

Figure 2: Typical zigzag path. Covered path indicated darker colour than the uncovered region [11]

2) *Probabilistic Roadmap Method (PRM)*

For the past two decades, many researchers tried to come up with new techniques of path planning, but only a few of them are valid especially for a robot with many DOF. Among all the methods available, probabilistic roadmap method, also known as PRM really stands out and used by many researchers to computes collision-free paths for a mobile robot in a static environment. PRM is particularly related to a robot with many degrees of freedom. Some of the variants of PRM, such as lazy PRMs and visibility based PRMs, are examined by Charpin in [14]. Geraerts and Overmars [15] showed some studies and compared a few methods in PRM but it is difficult to justify the results as the testing spaces and hardware used is different.

According to Kavralu, Svestka and Latombe [16], PRM can be separated into two kinds of phases; a learning phase and a query phase. In the learning phase, a graph with nodes that are collision-free configurations and the edges collision-free paths, called a roadmap, is built by repeating two steps. The first step is to pick a random configuration and then test it for collision and repeat the step until it is collision-free. The second step is to connect the previous configuration to the roadmap using a fast-local planner. In Figure 3 shows an example of a roadmap generated using PRM algorithm in a two-dimensional Euclidian space while Figure 4 is an example of the shortest path solved using the roadmap in Figure 3.

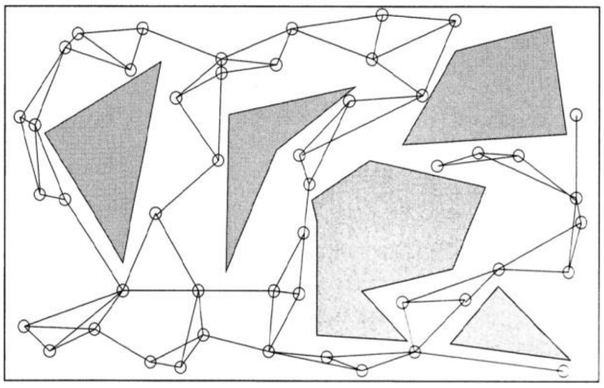

Figure 3: An example of a roadmap for a point robot in a two-dimensional Euclidean space from [17]

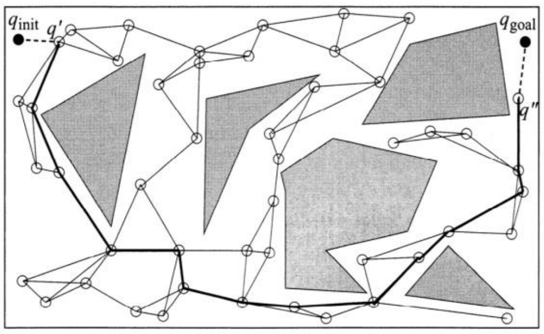

Figure 4: An example of how to solve the query in Figure 3

III. Methodology

The primary purpose of this research is to design and develop path planning techniques for a tennis ball retriever robot. The robot will help tennis player by discarding all the unnecessary energy and time wasting in traditional ball picking up method. The robot will sweep all the balls scattered all over the court during solo training. While sweeping the balls, the robot will avoid any obstacle and try to stay and complete its path.

This section focuses on the design of the experiment and how it is conducted. The experiment was conducted on an actual tennis court to test the robot more in its physical way. The robot movement is varied by few factors such as the robot locomotion, robot geometry, and type of wheels and so on. These factors are considered based on the literature review. The robot consists of a few sensors such as sonar sensors, and proximity sensor to detect obstacles. A few path planning methods were implemented into the robot and data was collected from the experiments.

A. Tennis Ball Retriever Robot

Figure 5 shows the isometric view of the developed robot. The body of the robot is made of stainless steel making the robot durable enough to withstand being hit by a tennis ball.

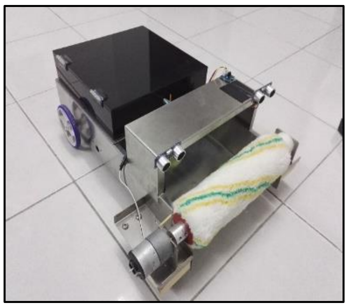

Figure 5: Isometric view of the actual tennis ball retriever robot

After the robot is built, the specifications of the robot such as turning radius, velocity and total weight were defined by using experiments and calculation. The specifications of the tennis ball retriever robot will be shown in this subtopic.

Table 1
Robot specifications

Specifications	Value
Weight	5.9kg
Velocity	$0.28ms^{-1}$
Turning radius	28.77cm
Dimensions	0.46m x 0.36m x 0.18m

B. Mechanism of the System

The developed tennis ball retriever robot is programmed by following the pseudo code below:

```
Step 1:  Get a map from the knowledge base
Step 2:  Convert the map to 2D grid map
Step 3:  Generate goal point from the target areas
Step 4:  Use path planning algorithm to find a path
Step 5:  Robot moves along the path while marking the
         covered path
Step 6:  If the ball storage is full, go to unload area and repeat
         step 3
Step 7:  If the robot reaches the destination, go to step 8. If
         not repeat step 5.
Step 8:  Calculate a number of the target area, if more than
         0, repeat step 3. If target area equal to 0, go to the
         base point, terminate the program.
```

1) Changing the court into court map

The grid method is a method that is changing the entire map into a grid. This grid allowed the robot to compute and plan its path in 2D, making the computational much simpler. These grids will be marked to represent the target area which is the uncovered area by the robot, the obstacle which is the grid that the robot does not need to cover, and the covered area which is the area that already passed by the robot. This robot need cover all target area while avoiding the obstacles.

To ensure smooth coverage, the grid of the map must consider the robot dimension. If the grid dimension is bigger than the robot dimension, some region of the map will be left out. However, if the grid dimension is too small compared to the robot dimension, it will cost more time and more energy to the process.

In the robot design section, the proposed robot dimension is 46cm (or 0.46 metres) in length and also 36cm (or 0.36 metres) wide. With this two information, the number of the grid in the grid map can be measured. The actual full dimensions of a standard tennis court are approximately 37 metres in length and 18 metres wide. However, in this project, the robot only has to cover one-half of the court making it cover 18m x 18m of the area.

Number of grid in length= 18m /0.46m =39.13~= 40 grids
Number of grid in width= 18m/0.36m = 50 grids
Total grid = 40grids x 50 grids=2000grids

Based on the calculation above, the robot needs to cover 2000 grids, assuming that no obstacles are laying on the court.

2) Changing the court map into grid map

To represent the target area or the uncovered area, 0 is used. The already covered area is marked with 1, and the obstacle area will be marked as 2. As the robot moving through the target area, the 0 value in the grid will change into 1 to ensure no path overlapping or repetition.

The grid in the environment is defined as g(x, y), where x(x=1,2,…,n) is the line number of g's grid, y(y=1,2,…,n) is the column number of g's grid. The sampling information of this environment is a series of continuous grid set G(g1,g2,…,gi), i(i=1,2,…,n) is the serial number of the grid sampling.

The system uses the grid method to model the actual court into 40 x 50 environment map in this plane and extract the environment into a 2D grid.

C. Generating Path

After a goal point is picked. The robot must find its way to the point while avoiding the obstacles around the map. Few path planning algorithms such as CPP U-Turn algorithm, CPP ISS and A* algorithm is put into tests. These algorithms will be compared and analysed to record their reliability in path planning, especially for a tennis ball collector robot. These algorithms will be compared based on their execution time to generate a path, time is taken to cover all target area entirely, and the accumulated path length after all target areas are covered.

1) Pathfinding via Coverage Path Planning CPP (U-Turn Algorithm)

Based on literature review, CPP comes with many types of approaches, for example, U-turn shape algorithm and spiral-shaped algorithm. U-turn algorithm is an algorithm proposed by Zengyu Cai[10]. This algorithm is used to complete all-region on a map while avoiding an obstacle in a U-turn manner. To use this algorithm, many factors should be considered such as robot maximum angular turn, robot geometry and robot degree of freedom.

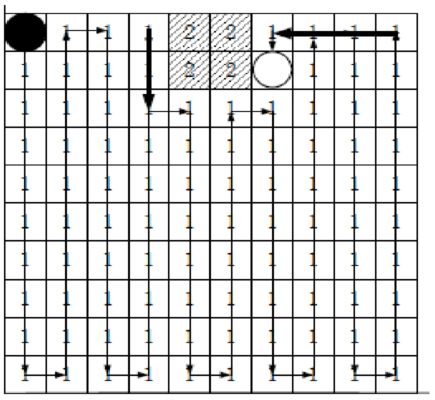

Figure 6: U-turn algorithm with an obstacle [10]

Figure 6 shows what happen if an obstacle is added to the grid map. The thin arrow line shows the robot path while the bold arrow line represents a path that was passed by the robot twice. This path is called repetition path. In this algorithm, the robot will try to minimise the repetition rate as low as possible to cut waste in energy and time.

Steps in CPP U-turn algorithm:

```
Step 1:  Get target area
Step 2:  Move in a straight path
Step 3:  If the front of the robot is the obstacle grid, check
         the side grid of the robot, if the left side is an
         obstacle or already covered grid, turn 180 degrees
         to the right and vice versa
Step 4:  If the robot comes into a dead situation where all
         side of the robot is an obstacle or covered grid,
         trace back the path until found an opening. Repeat
         step 2 until reached goal target area
```

2) Pathfinding via Coverage Path Planning CPP (ISS Algorithm)

Based on the literature review, Zengyu Chai[10] also introduce coverage path planning in shape resembling a spiral call Internal Spiral Shape (ISS). This algorithm will try to cover all target area while avoiding an obstacle spirally. Just like its counterpart algorithm, U-turn, it also requires factors such as robot maximum angular turn, robot geometry and robot degree of freedom to be considered.

The basic idea of the internal spiral algorithm is robot navigates this map area in a specific direction. When the front of the grid is not covered, the robot moves forward. If there are obstacles or the front of grid covered already, and then the robot turns right 90 degrees, as shown in Figure 7.

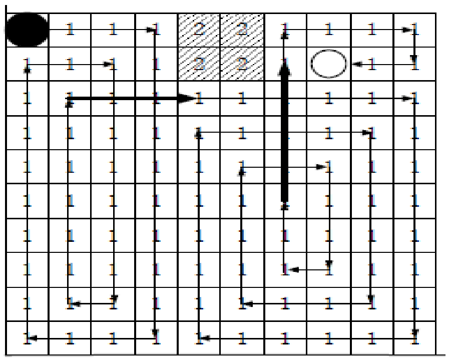

Figure 7: ISS algorithm with an obstacle [10]

Steps in CPP ISS algorithm:

Step 1: Get target area

Step 2: Move in a straight path

Step 3: If the front grid has an obstacle or covered grid, turn 90 degrees to the right

Step 4: If the robot comes into a dead situation where all side of the robot is an obstacle or covered grid, trace back the path until found an opening. Repeat step 2 until reached goal target area

3) *Pathfinding via Probabilistic Roadmap Method*

There are a few factors that may affect the path generated using PRM. The most crucial part is the deciding the number of nodes that will be used to get a suitable path. A higher number of nodes will result in a better path, but it will also increase the computational time. In the simulation, the number of nodes tested is 50 and 100. The first simulation with 50 nodes did not come with a path while 100 nodes do. Thus, 100 nodes are considered suitable for the project.

Steps in PRM algorithm:

Step 1: Get target area

Step 2: Get number of nodes

Step 3: Algorithm then attempts to connect all pairs of randomly selected nodes, if two nodes can be connected with a straight line, it will be added as an edge.

Step 4: After any all possible pairs are connected, the shortest combination of the path will be generated

Step 5: Robot moves according to the path. Repeat step 1 until all area is covered.

D. *Experiment Design*

In this title, there will be three path planning techniques that will be implemented into the robot; CPP U-turn, CPP ISS and PRM. To test the effectiveness of these algorithms, the robot must cover all target area on the court in two conditions; without obstacle, known as Condition A and with obstacle, known as Condition B. The time taken for the robot to cover all target area and the algorithm to generate the path is recorded for comparison data.

Figure 8 below shows how the experiment is conducted the tennis court. The green box represents the base or the robot start point. The grid representation of the base is g(0,0). The red dotted box with a dimension of 18m x 18m represents the area of experiment meaning that the robot has to cover all the grid within the red line. For Condition B, a box with 0.5m x 0.5m x 0.2cm is put right at the baseline to act as the obstacle. Yellow circle with their grid representation represents balls.

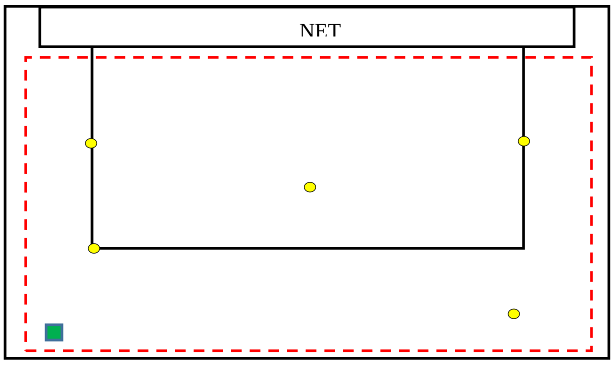

Figure 8: Layout of the experimental setup for Condition A

IV. Results and Discussion

A. *Experimental Results*

The experiment was completed on UniMAP's Sports Complex's tennis court. Even though the recommended dimension of half a tennis court by International Tennis Federation is 18m x 18m, the court that was used as experiment location has 14m x 15m dimensions. To generate the grid map, some calculation has been made:

Number of grid in length= 14m /0.46m =30.43~= 31 grids
Number of grid in width= 15m/0.36m =41.667~=42 grids
Total grid = 31grids x 42 grids=1302 grids

Two set of experiment was conducted; court without obstacle (Condition A) and the court with an obstacle (Condition B). For each of the experiment set, three algorithms were tested for data collection. Also, each of the algorithms was tested three times to gained better result to get better analysis and comparison.

Table 2 shows the result of CPP U-Turn algorithm in Condition A while the following Table 3 shows the result of CPP U-Turn algorithm in Condition B.

Table 2
The Result of Algorithm CPP U-Turn in Condition A

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	24.10	5	~94.00
2	24.30	5	~97.00
3	24.20	5	~95.00
Average	24.20	5	95.30

Table 3
The Result of Algorithm CPP U-Turn in Condition B

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	24.50	5	~94.00
2	24.45	5	~95.00
3	24.50	5	~94.00
Average	24.88	5	94.30

Table 4 shows the result of CPP U-Turn algorithm in Condition A while the following Table 5 shows the result of CPP U-Turn algorithm in Condition B.

Table 4
The Result of Algorithm CPP ISS in Condition A

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	27.20	5	~94.00
2	27.35	5	~95.00
3	27.10	5	~92.00
Average	27.22	5	93.67

Table 5
The Result of Algorithm CPP ISS in Condition B

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	28.05	5	~92.00
2	27.80	5	~91.00
3	28.10	5	~92.00
Average	27.98	5	91.67

Table 6 shows the result of PRM algorithm in Condition A while the following Table 7 shows the result of PRM algorithm in Condition B.

Table 6
The Result of Algorithm PRM in Condition A

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	33.20	5	~94.00
2	36.25	5	~95.00
3	31.45	5	~91.00
Average	33.63	5	93.30

Table 7
The Result of Algorithm PRM in Condition B

Trial	Time taken to finish algorithm (min)	Balls collected (/5)	Area covered (%)
1	31.55	5	~91.00
2	35.25	5	~94.00
3	33.30	5	~93.00
Average	33.37	5	92.67

From all the results above, the data collected can be shown in graph form to make the analysis much more explicit and accurate. Figure 10 shows a graph of time taken for the robot to cover all targeted area in Condition A while Figure 11 shows a graph of time taken for the robot to cover all targeted area in Condition B.

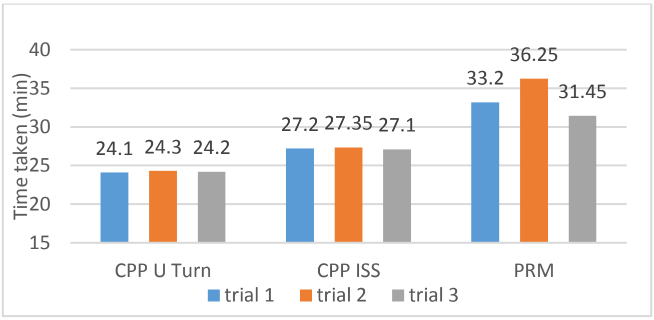

Figure 10: Graph of time taken for the robot to cover all targeted area in Condition A

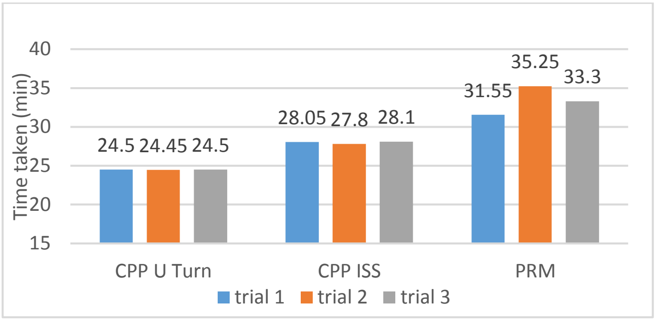

Figure 11: Graph of time taken for the robot to cover all targeted area in Condition B

B. Data Analysis

Based on Figure 10, it is also noticeable that result of using PRM was inconsistent throughout the three experiments. The difference between the slowest recorded time (31.55 minutes) and the fastest recorded time (35.25 minutes) is 3 minutes and 42 seconds. This is because the algorithm used a different path in each of the experiment. In PRM, after the nodes were put on the map, the algorithm will try to find the shortest path by connecting two nodes in a straight line. The algorithm making every path randomises this process is different in the experiments.

Based on the data collection, it is noticeable that the time taken for the robot to cover all target area for each experiment using CPP U-Turn and CPP ISS is not much different. Based on data in Figure 10, the fastest time recorded using CPP U-Turn was 24.45 minutes, and the slowest time taken was 24.5 minutes. The difference between the fastest time and the lowest time is just 3 seconds. This is because the robot used the same path for every experiment using CPP U-Turn.

However, every experiment using CPP ISS also came out using the same path but the difference of time taken recorded between the highest and the lowest is 18 seconds. This is probably because the initial position of the robot before start move around the court is slightly different. By experimenting three times, the average of time can be calculated to minimise any error during the experiments.

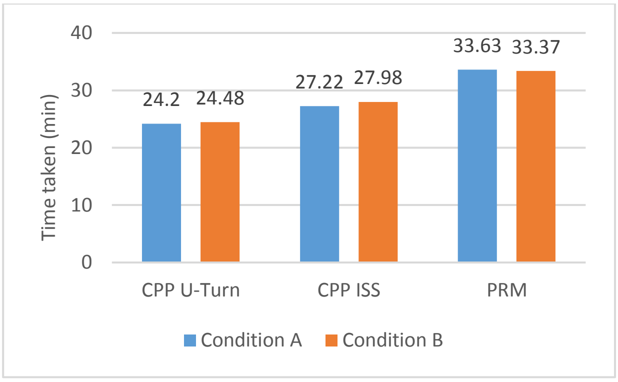

Figure 12: Graph of the average time taken in Condition A vs Condition B

Using average from data collection as shown in Figure 12, the result of three algorithms used in this experiment can be compared. When there is an obstacle on the map, the path generated by the algorithm will be more complicated than the path from a clear map. The complicated path requires more time to generate and may cause the robot to cover an area that has been a cover to get to the uncovered area as shown in Figure 7.

From Figure 12, for Condition A, the graph shows that CPP U-Turn average time taken to cover all target was 24.2 minutes, which is the fastest time among all the algorithms used in this experiment. CPP ISS took 27.22 minutes, 3 minutes and 1.2 seconds slower than CPP U-Turn. PRM recorded the slowest time to cover Condition A with 33.63 minutes, 9 minutes and 25.8 seconds slower that CPP U-Turn.

For Condition B, CPP U-Turn also recorded the best time, which was 24.48 minutes. CPP ISS algorithm recorded the second best average time at 27.98 minutes, which was 3 minutes 30 seconds slower than CPP U-Turn. By using PRM in Condition B, the average time taken was 33.37 minutes, which was 8 minutes and 53.8 seconds slower than CPP U-Turn.

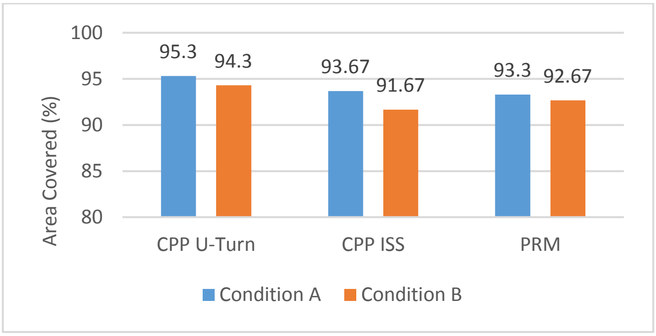

Figure 13: Graph of average percentage area covered in Condition A vs Condition B

From Figure 13, we can see the different percentage of area covered in both conditions by using three different algorithms. For Condition A, CPP U-Turn covered the most area with 95.3% and PRM covered the least percentage of area with 93.3%. For Condition B, CPP-U-Turn also recorded the highest result with 94.3% while CPP ISS recorded the least with 91.67%.

By giving the best result in both Condition A and Condition B, it can be concluded that CPP U-Turn is the most suitable algorithm to be implemented on the tennis ball collector robot.

V. Conclusion

The designed and developed path planning algorithms using CPP U-Turn, CPP ISS and PRM were integrated into the robot successfully. All three algorithms were validated in two different conditions to identify which algorithm is best suited for a tennis ball retriever robot at an actual tennis court. From the results, it concludes that CPP U-Turn is the best algorithm to be used on the tennis court. For the continuation of the project, the path planning of the robot can be improved by using more complex localisation methods with the location information from GPS or WIFI. The localisation method is believed to be able to allow the robot to know its surrounding much better and more efficient in term of time.

References

[1] S. A. Mnubi, "Motion Planning and Trajectory for Wheeled Mobile Robot," *International Journal of Science and Research (IJSR)*, vol. Volume 5, no. 1, pp. 1064-1066, 2016.

[2] M. Strandberg, Robot Path Planning: An Object-Oriented Approach, Stockholm, Sweden: Royal Institute of Technology (KTH), 2004.

[3] P. Lima and M. I. Ribeiro, "Introduction to Mobile Robotics," in *Mobile Robotics*, Instituto Superior Técnico/Instituto de Sistemas e Robótica, 2002, pp. 22-24.

[4] Y. Zhao and S. L. BeMent, "Kinematics, Dynamics and Control of Wheeled Mobile Robots," in *International Conference on Robotics and Automation*, Nice, 1992.

[5] D. Szondy, "Murata's Dancing Robotic Cheerleaders Showcase Advanced Group Control," New Atlas, Tokyo, 2014.

[6] I. Elamvazuthi and V. Singh, "Development of an Autonomous Tennis ball Collector," December 2015. [Online]. Available: https://www.researchgate.net/publication/283347206.

[7] M. Farrell, J. Hack and R. Irwin, "*Bear Claw: Tennis Ball Collector*," University of California, Berkeley, Carlifornia, 2009.

[8] D. Ford, "The BallBot Project," *Encoder: The Newsletter of Seattle Robotics Society*, pp. 1-12, August 1999.

[9] K. Leena. N, "A survey on path planning techniques for autonomous mobilerobots Leena.N1, K.K.Saju1," *IOSR Journal of Mechanical and Civil Engineering*, pp. 76-79, 2014.

[10] Z. Cai, "Research on Complete Coverage Path Planning Algorithms based on A* Algorithm," *The Open Cybernetics & Systemics Journal*, no. 8, pp. 418-426, 2014.

[11] G. Enric and C. Marc, *A Survey on Coverage Path Planning for Robotics*, Catalonia: University of Girona Press, 2013.

[12] Z. L. H. Y. a. H. E. L. Cao, "Region filling operations with random obstacle avoidance for mobile robotics," *Journal of Robotic Systems*, vol. 5, no. 2, pp. 87-102., 1988.

[13] H. Choset, "Coverage for robotics: A Survey of Recent Results," in *Annals of Mathematics and Artificial Intelligence*, Pennsylvania, Kluwer Academic Publishers Hingham, 2001, pp. 113-126.

[14] S. Carpin, "Randomized Motion Planning - A Tutorial," *International Journal of Robotics and Automation*, vol. 3, no. 21, pp. 184-196, 2006.

[15] R. G. a. M. Overmars., "A Comparative Study of Probabilistic Roadmap Planners," *Workshop on the Algorithmic Foundations of Robotics*, pp. 43-57, 2002.

[16] L. E. Kavralu, P. Svestka and J.-C. Latombe, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *Ieee Transactions On Robotics And Automation*, vol. 12, no. 4, pp. 568-571, 1996.

[17] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch and S. Thrun, "Theory, Algorithms, and Implementation," in *Principles of Robot Motion*, Massachusetts, MIT Press, 2005, pp. 121-123.