MEAM 520 Final Project Report:
2D Potential Guided Rapidly-Exploring Random Trees (P-RRT)

Group 10: Zachary Fisher & Michael Woc

December 9, 2019

Abstract

The goal of our project was to implement a 2D potential guided rapidly-exploring random trees (P-RRT) planning algorithm on NAO Robots from SoftBank Robotics in simulation. These robots are used in the RoboCup Standard Platform League, which is a robotic soccer league. The planner and environment were simulated in MATLAB. Five different experiments were created to test the planner's performance with stationary and moving target positions and obstacles. The P-RRT planner's performance was also compared to the performance of using only a potential field planner and a 2D RRT planner in appropriate situations. The path planner maneuvered an individual robot within the boundaries of a miniature soccer field, avoided collisions with dynamic teammates and opponents, and required low computational ability. The P-RRT planner was also able to successfully navigate a general path planning problem environment.

I. Introduction

This project aimed to develop a planning algorithm for the NAO Robots from SoftBank Robotics used in the RoboCup Standard Platform League (SPL), which is a robotic soccer league. In the RoboCup SPL, there are five fully autonomous robots on each team that must cooperate to score points and win matches [1]. The robots are bipedal robots with video cameras and sensors such as sonars, force sensitive resistors, gyrometer, accelerometer, magnetic rotary encoders, and capacitive sensors [2]. The data collected from its stereo video cameras and sonars are used as inputs to define the current state of its environment. The data is used to identify obstacles and targets, which are delivered to the path planning algorithm. This presents a highly dynamic multi-agent environment where a fast planning algorithm with moving obstacle avoidance and moving target position is crucial to a team's success [3].

UPennalizers is the team from the University of Pennsylvania that participates in this league. Our project is an extension of the work previously performed by the UPennalizers. In this project, legged locomotion of the robots was not considered due to the complexities of working with legged locomotion and the time constraints. The path planning problem was abstracted out to simply apply for a 2D environment and a holonomic vehicle. The robots currently utilize a finite-state machine (FSM) and simpler sequential vector fields planner [4]-[5]. This planner is computationally expensive, so the end goal of this project was to devise a planner that was computationally less expensive without sacrificing performance.

The planner we designed is for one robot since the robots can communicate with one another and track the same target. Field information regarding enemy and friendly robots, the ball, and the goals, is treated as absolute knowledge. Even though a specific feature may be outside a single robot's vision, the team can communicate to define environment. The robot that ultimately gets tasked with heading to the target is the one whose path takes the least amount of time [4]. The planning algorithm we decided to implement was a 2D potential guided rapidly-exploring random trees (P-RRT) planning algorithm. A P-RRT planner is combination of a rapidly-exploring random trees (RRT) planner and an artificial potential fields planner. This allows us to take advantage of the two planner's strengths, which suit robotic soccer planning scenarios. Potential field planners can handle dynamic obstacles such as moving opponent robots. Finally, RRT planners rapidly create a path without an analysis of the path's length but is advantageous in escaping local minima scenarios due to its global nature.

II. Literature Review

Initial research focused on investigating the planning algorithms other teams in the RoboCup Standard Platform League utilized. Teams in other RoboCup leagues such the Small Size League were also examined. The teams that had published work on their planning algorithms mainly used variations of RRT and A* planners. Further research examined other types of planners used in dynamic environments. The following sections summarize important points, relating to path planning, from different papers we used to formulate our path planning algorithm and experiments, not written in any particular order.

“Fast Path Planning Algorithm for the RoboCup Small Size League” [3]

The STOx team participates in the RoboCup Small Size League, which uses small mobile robots instead of bipedal robots. Even though STOx participates in a different league, we found the information relevant since we were representing the NAO robots as 2D holonomic vehicles. The team developed a new planning algorithm that outperformed their previous RRT planner by 30%, on average, in all measured metrics, path length, path smoothness, and processing time [3].

In general, the planner generates straight trajectories between an initial position and a goal position, which are checked for collisions with obstacles [3]. A trajectory is first generated then checked for collisions [3]. If a collision is detected, a “subgoal” point is generated near the obstacle [3]. The “subgoal” is perpendicular to the initial trajectory estimate at a distance from the obstacle equal to the robot’s diameter [3]. Once the “subgoal” is determined, trajectories are drawn between the initial, “subgoal”, and goal positions [3]. The collision avoidance algorithm described above is performed repeatedly until a collision-free path is found between the initial and goal positions [3]. Finally, the “subgoal” generated can be perpendicular by 90° or -90°; both options are considered and the one that results in the shortest path is chosen [3].

“B-Human Team Report and Code Release” [6]

A team named B-Human, a RoboCup SPL team, utilized an RRT approach until 2014 with re-planning steps taken in each Cognition cycle [6]. Although the planner worked quite well, it had two major problems. One problem was the randomness inherent with a probabilistic planner sometimes resulted in suboptimal paths and oscillations [6]. Oscillations occurred when updating the path to account for changing obstacle and ball positions [6]. The second problem was that RRT creates quite an extensive set of paths to choose from, which did not seem completely necessary for this 2-D problem when a simpler and faster planner could be used [6].

In 2015, B-Human developed a new planner that is a visibility-graph-based 2-D A* planner [6]. Each obstacle is bounded a circle and tangential straight lines connect the obstacles [6]. For every pair of non-overlapping obstacle, four tangent lines exist; for a pair of overlapping obstacles, only two tangent lines exist [6]. With up to nine other robots, four goal posts, and the ball, the number of edges in the visibility graph can increase drastically [6]. Thus, the planner creates the graph while planning, i.e. it only creates the outgoing edges from notes that were already reached by the A* planning algorithm [6]. The A* heuristic (which is an Euclidean distance) not only speeds up the search, but also reduces the number of nodes that are expanded. When a node is expanded, the tangents to all other visible nodes that have not been visited before are computed [6].

“Path Planning in Dynamic Environments” [7]

The report focused on planning in different types of dynamic environments such as known dynamic environments, partially known dynamic environments, unpredictable dynamic environments, and repetitive dynamic environments [7]. Based on the descriptions for each category, the dynamic environment in the Robocup SPL is a partially known dynamic environment. The locations of static obstacles, such as soccer nets and soccer field boundaries, is known, but future locations of dynamic obstacles, such as other robots, is not precisely known [7]. A planner designed for this type of environment must be able to update its path based on new information in a timely fashion [7]. The planner developed combines deterministic planning algorithms with probabilistic sampling approaches [7].

The first step of the planner is to generate a probabilistic roadmap (PRM) in the planning space containing all known static obstacles [8]. The PRM takes into account robot motion constraints and costs for navigating different areas of the environment [8]. Then, an initial path in the configuration space is generated based on the roadmap and is incrementally updated as the robot moves towards the goal [8]. The path is generated from the goal to the start using Anytime D* (AD*); AD* is similar to D* and D* Lite but operates in an anytime fashion [8]. This allows a non-optimal path to be quickly generated before progressively optimizing it and the path should become better the longer it runs [8].

Furthermore, AD* utilizes two heuristic values: minimum possible time to move between the current vertex to another vertex in the PRM and the minimum cost associated with moving between these vertexes [8]. These two heuristics are used in the heuristic estimate of the overall cost of traversing from the current vertex to another vertex in the PRM; this heuristic estimate has the same purpose as the *f*-value in A* [8]. Finally, the last step is to update the path if changes to the environment have been identified such as a dynamic obstacle is detected or a previously static obstacle has moved [8].

“Potential Guided RRT*” [9]

The planner developed is a potentialized rapidly-exploring random trees star (RRT*) planner with a single tree based at the start node. The tree grows from the start node and a new random node (*x*) is generated with each iteration; this random node is then moved a fixed distance along the direction of the potential gradient [8]. The new location of this node, *z*, is added to the tree and connected to another node in the tree in a way that the distance from the start node to the *z* is minimized [8]. Lastly, previously calculated distances between the start node and other nodes are updated if they have changed due to the new node [8].

“Bidirectional Potential Guided RRT* for Motion Planning” [9]

This paper improved upon the potential guided rapidly-exploring random trees star (P-RRT*) planner concept by applying a greedy strategy to design a bidirectional P-RRT* called P-RRT*-Connect [9]. In P-RRT*-Connect there are two trees: one based at the start node and one based at the goal node. The two trees alternate in generating new random nodes until the nodes of the two trees meet; the final path is one that connects the start node to the goal node using a combination of the two trees passing through the “middle node” [8]. In addition, a target region is defined in the free space as the goal for the attractive potential field. The driving force in the planner is still the RRT* algorithm that generates the initial random node, which is then modified, moved using attractive potential fields to attract it to the target region [9]. The attractive force decreases the closer the random node is to the goal node [9].

Observations

We concluded from “Fast Path Planning Algorithm for the RoboCup Small Size League” the planner they implemented was similar to a potential fields planner. The STOx planner always estimates a path directed towards the goal, similar to how the attractive potential on the goal node in potential fields affects path estimation. The obstacle avoidance measures can be compared to repulsive forces on obstacles. The “B-Human Team Report and Code Release” highlighted issues related to using only RRT for path planning. The A* planner B-Human developed is very efficient and effective. The AD* planner developed in “Path Planning in Dynamic Environments” appeared to be the most efficient planner while also providing the most optimal path, based on qualitative observations.

Our conclusion about the STOx planner and their ability to work well with dynamic obstacles led us to research variations of potential field planners. Research revealed that 2D RRT planners and potential field planners have been combined before and can be described as “potential guided RRT” or “potential function RRT” planners. The two P-RRT* planners reviewed above, from “Potential Guided RRT*” and “Bidirectional Potential Guided RRT* for Motion Planning”, utilize an RRT* planner guided by potential fields to the goal. Due to our previous experience with 2D RRT and potential fields, we decided to implement a 2D P-RRT style planner. Instead of using RRT as our main algorithm for path planning, we used 2D potential fields, without random walk, since the soccer environment is dynamic a majority of the time. We utilized the 2D RRT algorithm to help escape any local minima the robot might encounter to ensure a higher rate of success than with random walk for escaping local minima in potential fields.

III. Method

Design Considerations

Creating a path planner for RoboCup required a strong understanding of the gameplay, strategy, field layout, and physical capabilities of individual robots. Some key parameters used in our simulation were the top-down view of a NAO Robot's maximum radius (200 *mm*), the field dimensions (6000 *mm* by 9000 *mm*), the maximum translational speed of a robot (50 *mm/s*), and the number of robots that would typically occupy the field simultaneously (10). Obstacle robots are characterized as any robot that is not operating under the guidance of the path planner. Therefore, friendly and enemy team robots will be treated equally as obstacles to avoid.

Potential Field Planner

This planner models the system incrementally with respect to the robot's current position. A local approach is valuable in dynamic environments or when information about the entire environment is absent. Every point in the environment can be represented by a two-dimensional vector which has an orientation that is found through the resulting forces imparted by the environment. Our robot can be under the influence of a single attractive field and three repulsive fields at any one time.

A. Attractive Potential Field

An attractive field exists to draw the robot towards the target position. Attraction is accomplished through a gradient descent approach in which the robot's current position and desired target position are used to direct and scale the magnitude of the field vector. A conic well defines the attractive vector towards the target as a unit vector to standardize the magnitude and is described by:

$$F_{att,i}(q) = -\frac{(o_i(q) - o_i(q_f))}{\|(o_i(q) - o_i(q_f))\|} \tag{1}$$

Once sufficiently close to the target position, attraction is defined with a parabolic well description that proportionally scales the vector size down with the remaining distance as seen here:

$$F_{att,i}(q) = -\zeta_i(o_i(q) - o_i(q_f)) \tag{2}$$

B. Repulsive Potential Field

Repulsive fields exist to redirect the robot's motion away from obstacles along a vector from the closest point on the obstacle's surface to the robot. When the robot is outside of the specified range of an obstacle ($p_i(q) > \rho_0$), where ρ_0 is 900 *mm* for dynamic obstacles and 400 *mm* for static obstacles, the repulsive force from that specific obstacle should be negated to remove unnecessary path changes and is described by:

$$F_{rep,i}(q) = 0 \tag{3}$$

Once within the repulsive zone of an individual obstacle, the following equation amplifies the force as the distance decreases:

$$F_{rep,i}(q) = \eta_i \left(\frac{1}{\rho(o_i(q))} - \frac{1}{\rho_0}\right) \frac{1}{\rho^2(o_i(q))} \nabla\rho(o_i(q)) \tag{4}$$

$$\nabla\rho(o_i(q)) = \frac{o_i(q) - b}{\|o_i(q) - b\|} \tag{5}$$

This model includes three sources for repulsion to be created:

1. *N*-1 repulsive fields from the other robots in the environment, where *N* is the total of robots.
2. *M* repulsive fields from the *M* number of fixed obstacles in the environment.
3. A repulsive field from the boundary.

Dynamic obstacle repulsion begins when 3.5 robot radii are left between the bounding zones of the robot in the planner and the obstacles. Static obstacles and the boundary only begin to repel the robot when 2.5 robot radii remain as clearance. A larger repulsion zone encapsulating dynamic bodies was chosen to better compensate for unpredictable and occasionally conflicting movements of these objects. A smaller zone was not used since collisions have the potential to severely damage the NAO Robot. While falls are manageable, we would still like to prioritize collision avoidance. The boundary is also modeled as repulsive so that a dynamic obstacle does not redirect our robot outside the field bounds, resulting in a gameplay penalty.

The resultant unit vector incurred by the summation of attractive and repulsive influences dictates the step direction of the robot. The magnitude of the unit vector is scaled based on the desired walking speed of the robot which is typically 50 *mm/s*. When sufficiently close to the target position, the step size is decreased to reduce overshooting.

The robot is classified as "stuck" when its position does not leave a thresholded radius across 10 iterations. This model uses a "stuck" thresholding radius of 50 *mm* (the distance possible to traverse in one second) over the course of a duration that would allow the robot up a travel distance of up to 500 *mm*. A spherical representation of obstacles was employed to help reduce local minima; however, it can reduce the motion capabilities of a guided robot that is approaching head-on. Additionally, narrow passages between dynamic obstacles will result in rapidly changing vector field orientations that slow the robot by continuously redirecting it. The small "stuck" thresholding radius chosen allows for the possible alleviation of narrow passage problems due to the dynamic nature of the environment.

RRT Planner

This planner is designed to probabilistically search through an environment incrementally until a connection from the start to goal locations is established. Planning is done in the configuration space so that every point along the NAO robot can be referenced in comparison to known obstacles. RRT is useful when it is difficult to describe the free configuration space of a robot's environment, but easy to describe the configurations in collision. Within probabilistic planning, RRT is unique because it builds a roadmap incrementally for a single-query search.

RRT is an inherently unideal global planner in terms of path length. The path generation can be erratic, counter-intuitive to humans, and unnecessarily complex. One adaptation we made to the classic RRT planner prioritizes examining the initial robot position and desired target position. Immediately, these two points and the points that can be interpolated along a straight-line between them are checked for collisions. If no collisions are found, RRT is averted and the path is defined by a single line.

If this step is not possible, sample nodes are randomly generated within a subset of the maximum environment dimensions that prevents node placement near the boundaries. Encouraging the guided robot to stay away from the boundaries is an important gameplay consideration, especially later when combining potential fields with this global path. Successful node creation occurs when the point is identified as within the free configuration space. On the iteration after successful node generation, the planner attempts to connect the node to the starting position and goal positions through a straight line. If the node can be connected to just one of these root points (by validating that no collisions occur between or at the points), it is added as a branch of this now growing tree. If the node can be added to both trees, then a connection is formed that results in the termination of the planner. The RRT branching strategy between two trees shown in this YouTube video we created: https://youtu.be/N9eSqOP8lHg. The path between the start and end is extracted by removing branch "tips" that are not continuations of a path. This is done by removing path nodes that exist between two other nodes that can be connected through the interpolation and collision finding technique.

P-RRT Planner

Our path planning algorithm of focus is P-RRT, a combination of global and local planning strategies. Many variations on this style of planner exist; in this model, the motion of the guided robot is dictated solely by a potential field at the beginning of the simulation. The traditional RoboCup style environment of nine dynamic obstacles rarely resulted in local minima that trapped our robot and so it is more efficient to employ a pure potential field in hope of bypassing the need for a global planner.

If the “stuck” criterion described in the potential fields section is met, the robot has most likely arrived in a local minimum that did not resolve itself naturally. Using RRT, a path is created from this current stuck position to the desired global target destination. In *Experiments 1-4* discussed in the next section, the dynamic obstacles were modeled to cause collisions; however, in *Experiment 5*, only the stationary rectangular block was modeled as an obstacle. This change in obstacle definition alleviated issues with narrow passage problem associated with RRT. If dynamic obstacles are blocking the only passage to the target, it is more effective to still define the existence of a path through them with the expectation that the moving obstacles will create enough room for the guided robot. The path created by RRT is a sequence of points that are typically spaced relatively far from one another in the RoboCup style environment, due to the low number of obstacles. When fewer obstacles exist, the interpolated points between long-distance nodes are less likely to collide with these obstacles.

Potential fields are applied incrementally to each line segment created with RRT. Since the environment is dominated by freespace, the number of these segments is typically low (at less than 20) which reduces the number of unique potential field descriptions that are created. Within each segment, the robot is operating under the influences of the attractive force to the specific line endpoint and the changing environmental repulsive forces. In this way, RRT creates subsets of the environment that are individually and sequentially solved with fields that ignore the presence of the original global final target position. As the robot approaches a segment’s endpoint, a thresholding distance determines if the path is complete. During the first execution of the potential field planner, before RRT is needed, this distance is 30 *mm*. If RRT is needed, the satisfactory distance to points along the path is significantly increased to 600 *mm*. This value is intentionally large to bypass some of the inefficiency of traveling vertices along sometimes jagged paths and instead shortens the total distance traveled by “cutting corners.” This process can also be described as sequential funneling in which the wide open entry area at the top of the funnel is designed to allow for easy entry so that the robot can be redirected sooner to the narrowing mouth at the bottom of the funnel (next RRT endpoint). When the robot arrives at the last leg of the global path, it returns to a distance threshold of 30 *mm* to improve accuracy.

An additional method of bypassing unnecessary RRT path vertices is implemented that checks if the space between the robot’s current position and the global target position is entirely free of obstacles at that instant. If the space is collision-free, the sub target position (incremental RRT vertex) is abandoned in favor of moving directly towards the global target. This strategy is stronger in static environments but is still effective in dynamic situations since a local minimum is not likely to form at the exact time the robot moves through this space.

If, while operating under the influences of a potential field to move between the RRT vertices, the robot is trapped in an additional local minimum, the current global path is abandoned and a new one is computed with RRT from this current stuck position. The pseudocode below depicts the simple nature of this algorithm that operates until a *stuck* event occurs and repeats the path creation if ever *stuck* subsequently after.

P-RRT Pseudocode

```
stuck ← PotentialField(initial position, global target position, environment, parameters)
While stuck
    RRT(current position, global target position, environment, parameters)
    For each line segment in RRT
        stuck ← PotentialField(segment start, segment end, environment, parameters)
        If stuck
                Break for loop
```

Note: Pseudocode for RRT and PotentialField are in Appendix A.

IV. Evaluation and Results

The following section presents results from the five experiments conducted to test the P-RRT planner; each experiment was performed three times to ensure results were reproducible. *Experiment 1* is an environment with 9 randomly generated static obstacles while *Experiment 2* is an environment with 19 randomly generated static obstacles. *Experiment 3* is an environment with 9 randomly generated dynamic obstacles while *Experiment 4* is an environment with 19 randomly generated dynamic obstacles. *Experiment 5* is an environment with 9 randomly generated dynamic obstacles and 1 static block obstacle. All the circular obstacles have diameters equal to the width of a NAO robot.

Experiment 1 and *3* are intended to simulate environments seen in the Robocup SPL; the nine obstacles generated in *Experiment 1* and *3* are used to represent both team and opponent robots. *Experiment 2* and *4* have double the number of total robots, twenty, as *Experiment 1* and *3*, ten. Having twenty robots on the field is also representative of human soccer, ignoring goalies. This allowed us to artificially create local minima scenarios since we were unable to generate any local minima in *Experiment 1* and *3*. *Experiment 5* simulates a general path planning problem involving a static obstacle and dynamic obstacles. The circular obstacles still have the same diameters as before.

In following sections, the potential field planner will be referenced as APF (artificial potential fields). Each experiment has a link to a YouTube video showing P-RRT in simulation. Fig.1-4 represent the results from the RRT and P-RRT planners for *Experiment 1* and *2*. The APF results for *Experiment 1* and *2* are in the Appendix B as Fig. B1-B2. Fig.5-10 represent the results from the APF and P-RRT planners for *Experiment 3-5*. Finally, Table I contains the average performance metrics, run time and path length, used to compare the performance of our planners. The values "DNF" indicates the planner was unable to complete a path and "N/A" indicates the planner produced inconclusive results.

Experiment 1: RoboCup Style Static Environment

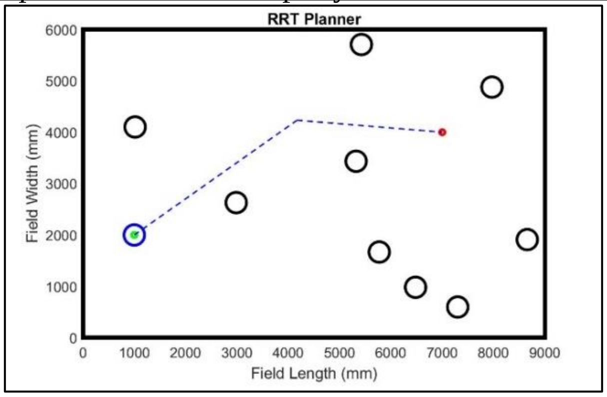

Fig. 1. RRT planner result for *Experiment 1*.

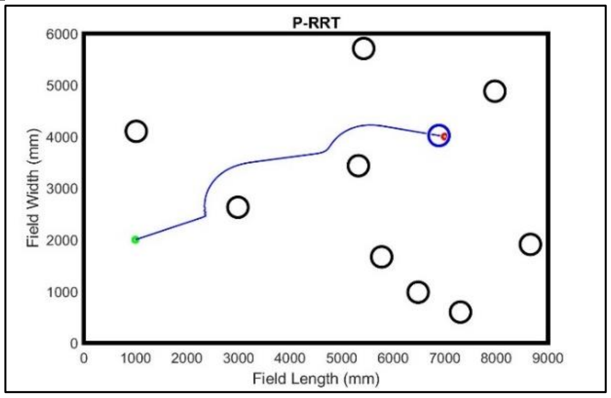

Fig. 2. P-RRT planner result for *Experiment 1*.

Experiment 2: Challenging Static Environment

https://youtu.be/9-mgKkdGxM8

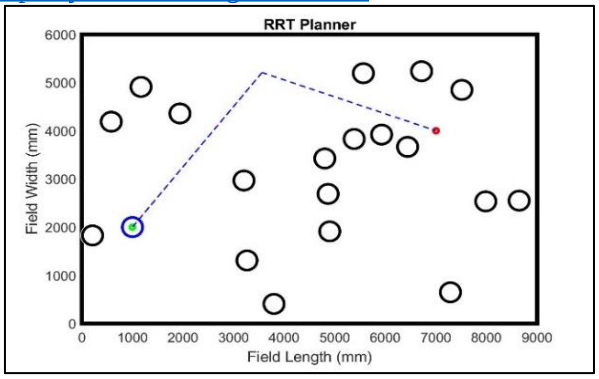

Fig. 3. RRT planner result for *Experiment 2*.

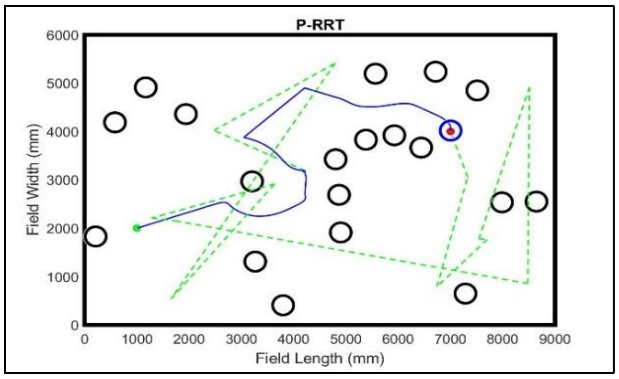

Fig. 4. P-RRT planner result for *Experiment 2*.

Experiment 3: RoboCup Style Environment
https://youtu.be/uOfbyxXJxao

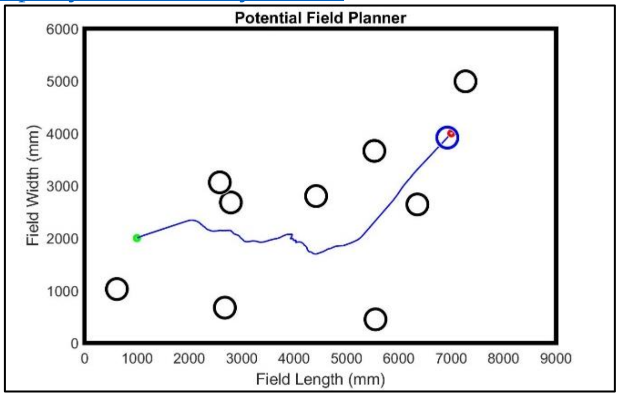

Fig. 5. APF planner result for *Experiment 3*.

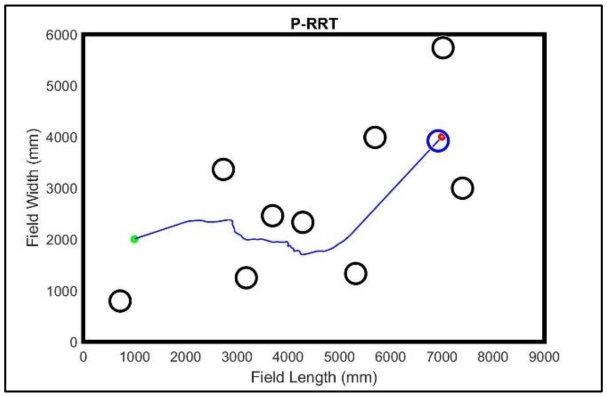

Fig. 6. P-RRT planner result for *Experiment 3*.

Experiment 4: Challenging Dynamic Environment
https://youtu.be/bq9kIwkzppg

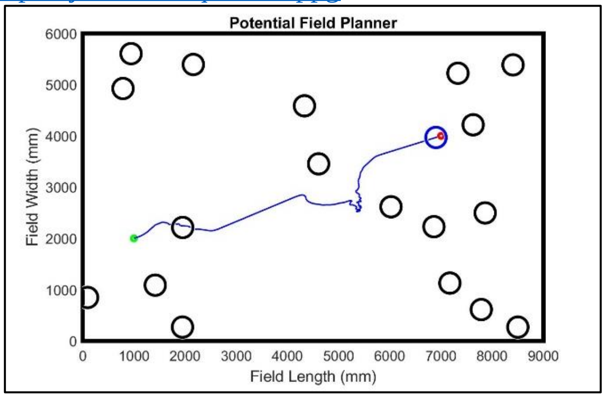

Fig. 7. APF planner result for *Experiment 4*.

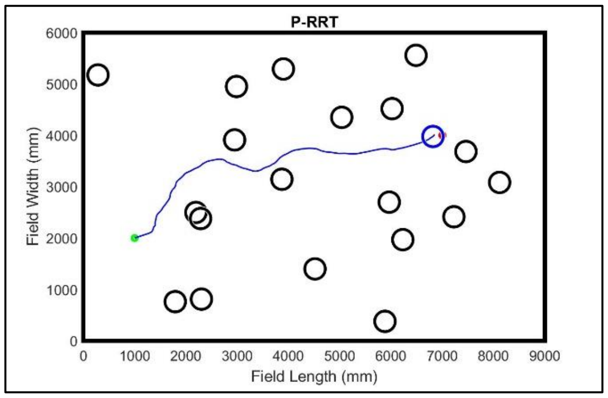

Fig. 8. P-RRT planner result for *Experiment 4*.

Experiment 5: Dynamic Environment with a Fixed Local Minimum
https://youtu.be/HgxBtRtbqV8

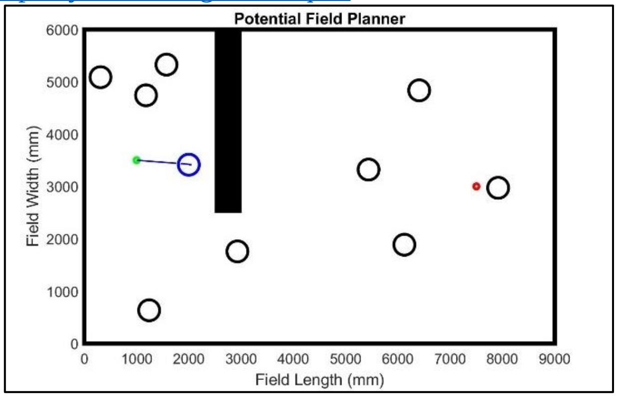

Fig. 9. APF planner result for *Experiment 5*.

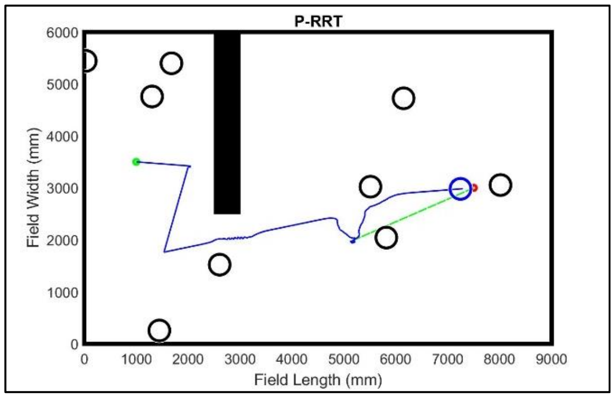

Fig. 10. APF planner result for *Experiment 5*.

TABLE I
AVERAGE PLANNER RUN TIMES AND PATH LENGTHS

Experiment	**RRT**		**APF**		**P-RRT**	
	Time (s)	**Length (mm)**	**Time (s)**	**Length (mm)**	**Time (s)**	**Length (mm)**
1	0.0226	6718	0.0398	7025	0.0233	7025
2	0.0527	7748	DNF	DNF	0.0989	10700
3	N/A	N/A	0.0694	7550	0.0474	7475
4	N/A	N/A	0.1006	7250	0.11	7050
5	N/A	N/A	DNF	DNF	0.154	10425

V. Discussion and Analysis

Overall, Fig. 1-10 accurately represent each planner's success in the experiment. Fig. 1-4 show that both RRT and P-RRT were able to navigate an environment with static obstacles. APF was also able to find a path for *Experiment 1*, but not for *Experiment 2* due to local minima, as seen in Fig. A1-2. The solid blue line in Fig. 4 represents the final path, while the dashed green lines represent the path made by RRT to escape local minima. The robot deviates from the RRT path due to an optimization in which it prioritizes direct line-of-sight paths to the target. In this case it was extremely effective and reduced the path length substantially. RRT results were not reported for *Experiment 3-5* since the results were inconclusive, planner either found no solution or took a very long time to run. Fig. 5-6 show that APF and P-RRT generate the same path for a simple dynamic environment.

In *Experiment 4*, APF and P-RRT generated different paths for the same environment which is not seen in the results. In Fig. 7, the beginning of the path appears to go through an obstacle, but that is not the case. It is a dynamic obstacle that moved to that spot after the robot had already passed. Furthermore, Fig. 7-8 represent environments from different test runs; the results shown had the least number of obstacles obscuring the final path. This was done for clarity, to clearly show the path generated. *Experiment 5* showed that P-RRT, Fig. 10, is able to handle a mix of static and dynamic obstacles unlike APF, Fig. 9.

In general, P-RRT had a longer run time and path length compared to the other planners. P-RRT did have a shorter path length for *Experiment 3* and 4, and a shorter run time for *Experiment 3*, compared to APF. As seen in Table I, APF was unable to find a path for *Experiment 2* and 5 which contain local minima. Table II contains percent differences in run time and path length of RRT versus P-RRT and APF versus P-RRT. P-RRT had the largest time difference, about 61%, compared to RRT for *Experiment 2*; the smallest time difference, 2.76%, was for *Experiment 1* when compared to RRT. The largest path difference was 32.0% when compared to RRT and RRT for *Experiment 2*; the smallest path difference, 0%, was for *Experiment 1* when compared to APF.

TABLE II
PLANNER RUN TIMES AND PATH LENGTHS PERCENT DIFFERENCES

Experiment	**RRT versus P-RRT**		**APF versus P-RRT**	
	Time	**Length**	**Time**	**Length**
1	2.76%	4.47%	52.5%	0%
2	61.0%	32.0%	DNF	DNF
3	N/A	N/A	37.8%	0.998%
4	N/A	N/A	8.95%	2.80%
5	N/A	N/A	DNF	DNF

Thus, while P-RRT did not outperform RRT and APF in the performance metrics every experiment, P-RRT was able to find a path to the goal in each of the experiments.

VI. Conclusions & Future Work

These experiments provide only the start of modeling for the RoboCup League's behavior. Abstracting out many of the many of challenges with team coordination and simplifying the topic to a single robot helped us focus on path performance; however, a robust planner for a team will require exploration into role assignment and strategy. Randomly perturbed obstacles also likely made the planning process easier, as the enemy robots were not actively attempting to interfere with this model's path generation efforts. Lastly, regarding simplifications, even while a RoboCup team's robots are communicating with one another to characterize their environment, erroneous sensor readings, conflicting robot opinions, and failures in self-localization can dismantle a team's field awareness and effectively make many planning attempts obsolete when considering worst-case hardware implementations.

At first, a local planner like potential fields appeared straightforward and powerful in this type of environment; however, we found that it can result in non-smooth paths that result from "bouncing" between opposing forces that result in oscillation. It appears that more foresight may be required in in the planning scheme instead of reacting to the gradient vector at a single location. This led us to believe that there might be value in recomputing a global planner continuously to produce better results in terms of path smoothness and length. Some of the planning approaches seen during the literature review phase showed this possibility, but the planner needs to be significantly optimized to prevent computational delays. Therefore, we would like to explore the possibility of using an informed RRT* planner or even BIT* which both use a subset of their environment to place random nodes and iteratively optimize the cost of their paths. Efficient, but still natively random planners like these could be ideal for RoboCup if the complexity from changing between new global paths does not create unintended behavior

In the future, we would first like to attempt a purely global planning strategy that recomputes the ideal path over the course of the robot's movement. This could be compared with P-RRT to see which is more viable for the actual implementation within the UPennalizer's codebase. Additionally, our current planner of P-RRT could be further improved to incorporate clustered repulsive fields around obstacles that are close to each other as well as tangential potential fields to guide our robot more smoothly around obstacles. We would like to test planning strategies on the NAO robots to develop a better understanding of their movement limitations and how their paths could be optimized specifically for them. Many of the parameters in our current model have been tuned based on the simplified simulated behavior and will likely need to be recalibrated. Incorporating any strategy into the UPennalizer's codebase will also mean reconfiguring the input/output structure of some functions.

VII. References

[1] RoboCup Technical Committee, “RoboCup Standard Platform League (NAO) Rule Book,” 22-May-2019. [Online]. Available: http://spl.robocup.org/wp-content/uploads/downloads/Rules2019.pdf. [Accessed: 18-Dec-2019].

[2] "Technical overview — Aldebaran 2.8.6.6 documentation", Doc.aldebaran.com, 2019. [Online]. Available: http://doc.aldebaran.com/2-8/family/nao_technical/index_dev_naov6.html. [Accessed: 18-Nov- 2019].

[3] S. Rodríguez, E. Rojas, K. Pérez, J. López, C. Quintero and J. Calderón, "Fast Path Planning Algorithm for the RoboCup Small Size League", RoboCup 2014: Robot World Cup XVIII, pp. 407-418, 2015. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-18615-3_33#citeas. [Accessed 11 November 2019].

[4] GRASP Lab, "The UPennalizers RoboCup Standard Platform League Team Description Paper 2017", Philadelphia, 2017.

[5] GRASP Lab, "The UPennalizers RoboCup Standard Platform League Team Description Paper 2018", 2018.

[6] T. Rofer et al., "B-Human Team Report and Code Release 2017", Bremen, 2019.

[7] Berg, "Path Planning in Dynamic Environments", 2007.

[8] R. Rahman and S. Sinha, "Potential Guided RRT*", 2014.

[9] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong and W. Rui, "Bidirectional Potential Guided RRT* for Motion Planning", IEEE Access, vol. 7, pp. 95046-95057, 2019. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8763966. [Accessed 18 November 2019].

VIII. Appendices

Appendix A

A. RRT Pseudocode:

```
If no collisions are detected between start and goal
    Link points
    Exit the RRT loop
For each iteration a path has not been found
    Generate a random node
    Find closest existing point within start and goal paths
    If no collision is detected when adding to the closest point
        Add the node to the start path
    If no collision is detected when adding to the closest point
        Add the node to the goal path
    If the node is added to both paths
        Merge the start and goal paths
        Exit the RRT loop
If (no path was created)
    Return an empty path
Else
    Optimize the path by skipping redundant intermediate node positions
```

B. Potential Fields (APF) Pseudocode:

```
While (distance to goal <  termination threshold distance)
    If (distance to goal < parabolic_conic_threshold)
        Attractive forces = Parabolic Well equation
    Else
        Attractive forces = Conic Well equation
    Loop through the environment's dynamic obstacles
        If (distance to obstacle > region of influence rho_0_dyn)
            Repulsive force = 0
        Elseif (distance <= 0)
            Repulsive force = 0
        Else
            Repulsive force = repulsive force equation
        Sum the repulsive forces caused by each of the obstacles
    Loop through the environment's fixed obstacles
        If (distance to obstacle > region of influence rho_0_stat)
            Repulsive force = 0
        Elseif (distance <= 0)
            Repulsive force = 0
        Else
            Repulsive force = repulsive force equation
        Sum the repulsive forces caused by each of the obstacles

    If (distance to the field boundary > region of influence rho_0_stat)
        Repulsive force = 0
    Elseif (distance < 0)
        Repulsive force = 0
    Else
```

```
        Repulsive force = repulsive force equation
    Sum the repulsive forces caused by each of the obstacles and the boundary
    Sum the attractive and repulsive torques
    qNext = qCurr + (the step size times the normalized resultant force)
    If (difference from qNext and a q from 10 steps previous < stuck_threshold)
        Terminate the while loop
```

Appendix B

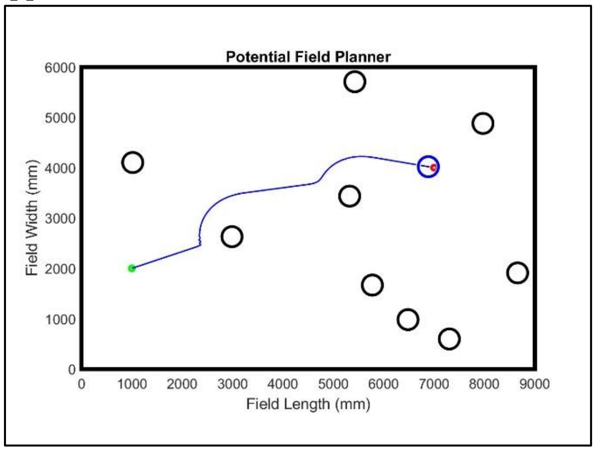

Fig. B1. APF planner result for Experiment 1.

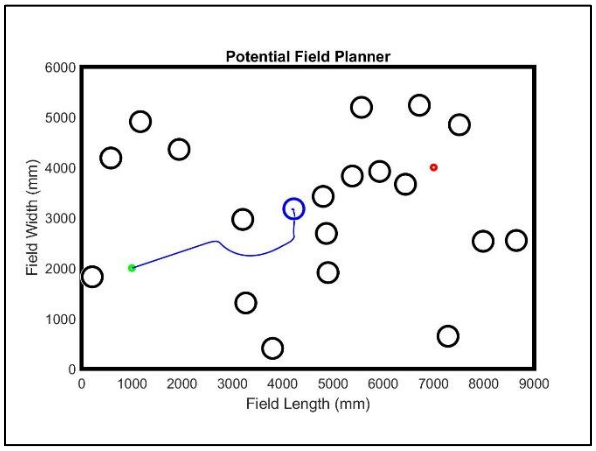

Fig. B2. APF planner result for Experiment 2.