Marker-Based Localization of Robots in Simulation

Osayame Gaius-Obaseki Thaddeus I. Madison Jory Denny Aditya Mahadevan Nancy M. Amato

***Abstract*— Localization is one of the key components of practical robotics. Localization is easy in an environment whose features are known a priori, as the agent does not have to simultaneously map the environment. One scheme for localization in a known environment is to use unique visual markers that denote specific coordinate positions. This paper details our implementation of a particular localization scheme. Our technique utilizes a simulated vision sensor to detect a robot's distance away from landmarks, and its relative direction to them. With the provision of at least two markers, we are able to triangulate the relative position and orientation of a robot. We test the effect of estimation error on the accuracy of our method.**

I. INTRODUCTION

Localization is one of the key components of practical robotics. Localization is easy in an environment whose features are known a priori, as the agent does not have to simultaneously map the environment. We assume that there are beacons in the environment whose positions are known and whose distances from the pursuing robot are measured at every time step in the simulation.

This paper details our implementation of a particular marker-based localization scheme. Our technique utilizes a simulated vision sensor to detect a robot's distance away from landmarks, and its relative direction to them. Perfect simulation is referenced as a way to explain what our approach is and how it achieves a given position. We also test the robustness of our method to variations in the estimation error and results are obtained and examined as well.

II. RELATED WORK

In this section, we review some research relevant to the marker-based localization discussed in this paper. We first highlight some work in motion planning, and then review some interesting work in localization and group behaviors.

A. Motion Planning

Motion planning is a term used in robotics for the process of detailing a task into discrete motions. There are countless situations in the real world that involve moving from point A to another point B. For this reason, computer simulations can be used to model events in nature. Concepts that are describable by an algorithmic process are modeled on computers so that the environment can be controlled. These simulations can be implemented on real robots to view events in a semi-controlled environment.

An example of this is our prior work, [9] and [8], that uses a roadmap-based approach to motion planning. This paper builds on the prior work.

B. Localization

Research on mobile robot localization using landmarks assumes that the environment the robot enters has landmarks in them. M. Betke and L. Gurvitis [5] introduce noisy input to their simulation to simulate real world error. Their algorithm estimates the robot's position and orientation with respect to the map of the environment. M. Betke and L. Gurvitis's [5] algorithm represents the landmarks by using complex numbers. The algorithm runs in time linear in the number of landmarks.

M. Betke and L. Gurvitis [5] also address a position estimation problem. They define the problem of estimating a robots position and orientation in its environment given a global map of the environment and bearings of landmarks measured relative to each other at the robots position. Due to distributed errors along the distances away from the robot, the triangulation process will not result in exact measuresments. For this reason, in [5], triangulation with perfect data and with noisy data are presented and compared. Using "noisy data" can help determine how flexible the localization algorithm is.

Sutherland and Thompson [10] present an approach to how error varies. Their study focuses on how the amount of error from the theoretical position to the actual depends on the orientation of the landmarks and how making "wise choice[s]" in the landmark configurations can significantly decrease error.

Different approaches are presented in [7] and [6] for handling position uncertainties associated with navigating robots. These papers differ in terms of how the environment is structured and some present results of hardware implementations of navigating robots.

We address localization using a software simulation with a semi-random path and percentage based random error to model sensing error in hardware.

C. Group Behaviors

Research on group behaviors focuses on the behavior of agents in multi-agent systems. In other words, group behaviors involves the behavior of agents as individuals within a group, as well as the group as a collective system. Group behaviors arise in many scenarios ranging from search and rescue operations, to hunting, to soccer.

Covering is one type of group behavior. Covering involves an agent attempt to traverse an environment, whilst performing a particular task, i.e. localization. In [1], a new class of covering problems is introduced where the objective is to generate an optimal route for a snow-blower in a polygonal environment. Furthermore, [2], [3], [4], the benefits of integrating roadmap-based path planning techniques with flocking techniques were explored, and a variety of group behaviors including exploring and covering were simulated utilizing an underlying roadmap.

III. EXISTING FRAMEWORK

In this section, we review some relevant sources that are necessary to building to the marker-based localization discussed in this paper. We review some existing framework to simulation and sensors.

A. Simulation

The Parasol Laboratory has developed a framework that allows us to easily develop real time simulations of different types of behavior in mobile agents, including robots. In these simulations, each agent is equipped with behavior modules that determine how the agent moves and reacts to its environment and other agents in the simulation. The simulation can resolve collisions among agents and between agents and the environment. This framework is currently being extended to incorporate physical robots, namely iRobot Creates. In this paper, we utilize the simulation framework for our experiments.

Below is an outline of the algorithm which represents the simulation loop.

Algorithm 1 General Simulation Loop

```
Input: simulator $sim$, environment $env$
1: $groups_{all}$ = sim.getAllGroups()
2: for $g \in groups_{all}$ do
3:     $g \rightarrow$applyBehaviorRule($env$)
4: end for
5: for $g \in groups_{all}$ do
6:     updateState($g$)
7: end for
8: ResolveCollisions($groups_{all}$,$env$)
9: Evaluation
```

At each step of the simulation, every agent updates its sensory information. The agent's ability to detect other agents in the environment can be affected by agent capabilities (the view radius and angle), the presence of obstacles in the environment, and (in 3D environments) the surfances and obstacles that constitute the environment itself.

B. Sensors

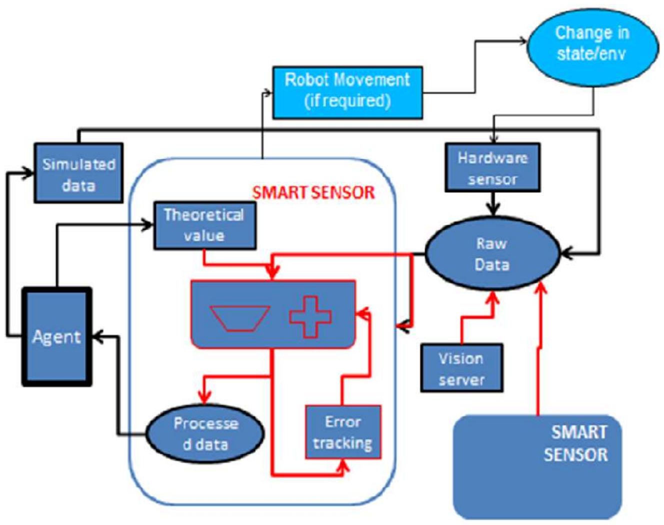

Fig. 1. A diagram detailing intercommunication between sensors and the agent

Figure 1 is a diagram of our sensor suite. The sensor suite is a collection of sensors used to gather information from the environment. This diagram does not refer to any specific sensor, but a generic representation of how sensors are implemented. These sensors can gather values from simulation, hardware, etc. The given sensor, denoted as a "smart sensor", should be able to work coherently with other sensors. Lastly, these sensors should model mearsurement error in simulation.

C. Specific Sensors

The vision sensor from the Parasol Lab is a basic implementation of a sensor that checks for agents within a view radius that are not blocked by any other obstacle.

We have developed a more generic sensor that not only checks for agents, but also checks for markers, other nodes, or whatever the robot is trying to see in the environment.

IV. Our Approach

In this section, we describe the algorithm used to solve localization problems given a small amount of knowledge about the environment.

A. Localization

Our approach, called "marker-based localization" uses a process called triangulation [5]. Given two markers, we assume that the vector that connects the two markers and positions of the markers in relation to the environment are known a priori. The robot uses these two markers to form a triangle (using itself as the third point on the triangle). The way triangulation works is:

- The distances from the markers to the robot and the facing direction of the robot are calculated.
- The angle that lies at the tail of the vector that connects any two given markers formed is calculated using the law of cosines.
- This vector is rotated by this angle to point in the direction of the robot.
- The head of this vector is the position of the robot.

This is the way the basic triangulation works. It is computationally simple in that it only requires the rotation of one vector.

B. Obtaining an unique solution

Approaching triangulation from a geometric standpoint creates an interesting problem.

In Figure 2, it has been noted that there are two potential places that the robot can lie. To help prove this, notice how the distances d_1 and d_2 here act as radii to their respective circles. At any given time step in our simulation, where a time step is the process of the robot moving from one position to another, the distance from the node to the robot can be interpreted as a radius for that particular marker. If this is the case, then it is a safe assumption to say that robot can lie on the circumference of circle one. Now after this same process is applied to the other marker, there exists two circles, whose intersections are the solution to where the robot lies. When the circles touch, there is only one solution. When the circles overlap, as depicted in Figure 2, there are two solutions to the system of equations.

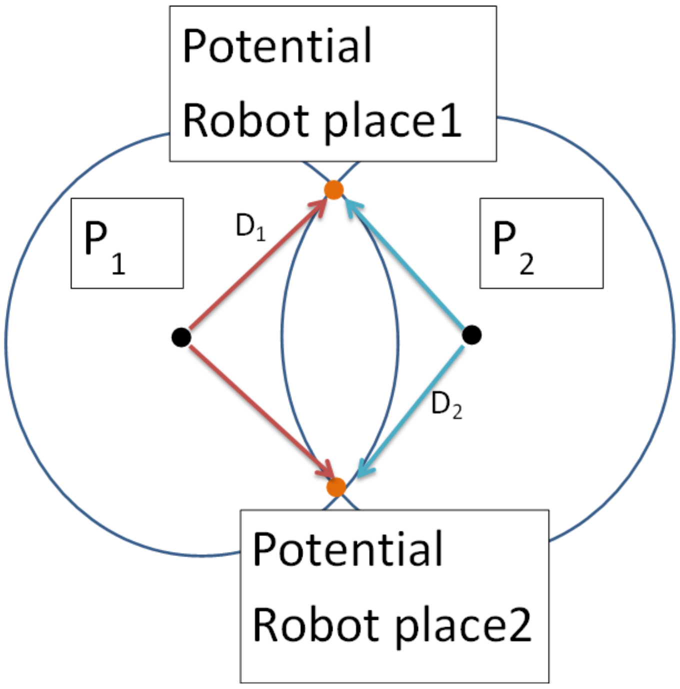

Fig. 2. An example of localization with no error

The triangulation results in two positions. The robot needs to differentiate between the two positions. This is the position-error problem. We will have to implement a way for the robot to determine if the robot lies on the rightside or the leftside of any of our markers. After this is satisfied, then it is possible to always calculate the correct triangulation.

Figure 3 depicts the right versus left decision process. This process works by using the vector of the robot's facing direction and calculating the value of the angles that exist between the facing direction and the vector constructed from the robot to any one marker. The two angles that are calculated are solved for in a range from -90 to 90 degrees. This is the determining factor in our implementation. If the angle's value is positive, this the marker is on the right. If the angle's value is negative, then the marker is on the left.

C. Accounting for measurement error

Measurement error is the error that was imposed on the distances from the markers to the robot. This creates two cases: an underestimate case and an overestimate case. The underestimate case occurs when the error imposed is a negative value. When this error grows large, the markers' circles could possibly not intersect, creating a no solution result. To address this problem,

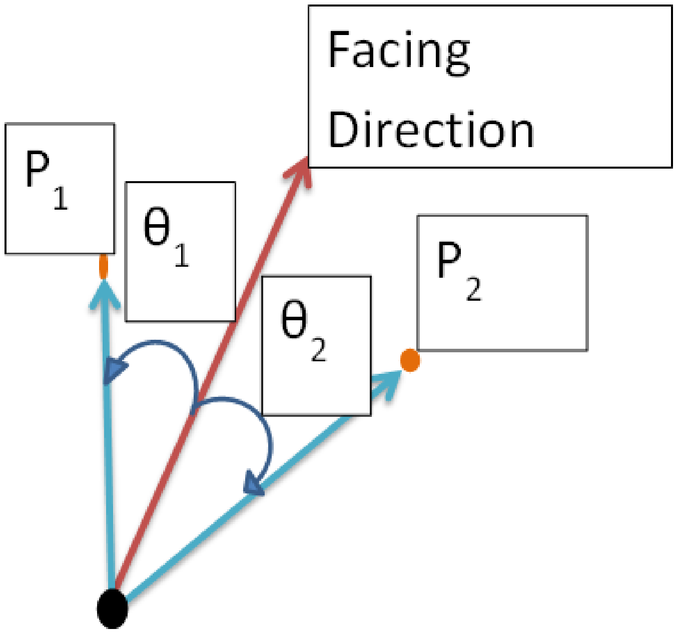

Fig. 3. The Right Versus Left Decision Process

the robot incrementally increases the distance estimates until it finds an intersection.

The overestimate case creates intersections, but the theoretical triangulated position becomes further away as the error imposed is increased. We addressed this situation by using many pairs of markers in the robot's view radius to triangulate and the results of the individual triangulations are averaged.

Since the view radius of the robot is 360 degrees it is assumed that error imposed on markers in front of the robot will be decreased by markers behind the robot and averaging the results. If at least three markers are in the robot's view radius, as the error increases it reaches a certain limit where the amount of error is always counteracted by averaging multiple pairs of marker triangulations.

V. EXPERIMENTS

A. *Experimental Assumptions*

Before we describe our setup, we will present some assumptions made during our experiement.

One of the most important assumptions about our environment is that the objects in our environment are static. We assume that the markers do not move as the simulation progresses. Our implementation is a simple answer to a localization problem. we are assuming that when implemented on real hardware the markers will be replaced with unique landmarks which do not typically move.

Another assumption is that there is more than one marker in sight at a time. If this assumption is not satisfied, the robot cannot localize using our scheme. We give the robot a probabilistic roadmap to traverse in the cases that it cannot localize itself. Essentially, this roadmap gives the robot a default random path to travel along. If, at any given timestep, the robot sees more than two markers, then it localizes.

B. *Experimental Setup*

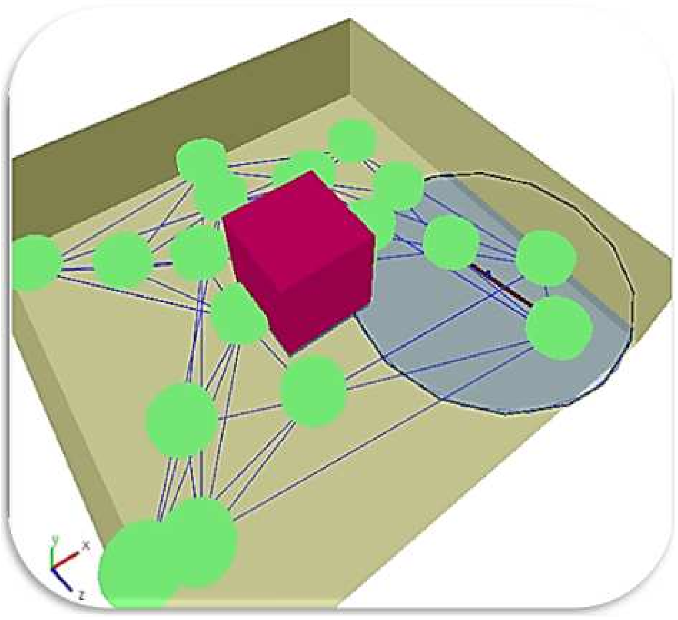

Fig. 4. The environment with an obstacle in the middle and green dots as markers

Figure 4 is a picture from one of our experiments. It is important to note that we assume a static environment for our experiment. This means that nothing in the environment moves at any given moment in time other than the robot that is localizing.

The large open box represents our "bounding box". This box represents the space available for the robot to traverse. It is important here to note that our algorithm is not space specific, but the bounding box allows for the user running this experiment to arrange the size to a comfortable viewing size. The size of the bounding box is 48 units wide and 48 units long.

The view radius for the robot is denoted by the large transparent circle. The markers that the robot detects are the small cylinders in the environment.

The lines interconnecting the markers denote where the robot can move in the space allowed, so without a domain generating these lines take a long time. Since generating this map is outside of the scope of this paper, we will not discuss map generation but mention that a set domain size makes this process quick.

The block in the center is an obstacle. This obstacle presents a few issues that call for close attention. One is that if two markers are seen but an object is blocking the interconnections between the markers then how can the robot still localize? The answer is that the robot can still localize because our schemes uses the sight distances or the distances from the robot to the markers. The connection between any two markers is used for calculation purposes but is optimized to only use the computational distance for that connection. Also, what happens if the marker is in the robot's view radius but is blocked by an obstacle. All markers that have an obstacle in the way are not seen and therefore are not used to localize the robot.

We tested our implementation with varying error for 300 timesteps. At every timestep, the robot localizes itself using multiple pairs of points in its view radius. After, the localizations that were calculated in that time step are summed and averaged. We then record the deviation between the theoretical position and the actual position from every time step and average these values. These final results are plotted against the maximum value of error that can be imposed on the distances from the markers to the robot.

VI. Results

A. Zero Error Case

When the system undergoes no error, then no matter how long the simulation runs the theoretical position of the robot matches that of the acutal position of the robot. The zero case helps prove that our implementation is correct.

B. Cases With Varying Error

In Figure VI:

Measurement Error= Percent of actual distance from marker to robot imposed as error

Localization Error=distance between estimated and actual robot position

The estimated error increases with measurement error. This result is intuitive, but more interesting is what happens as the percentage-based measurement error increases to a high percentage of error. With small error, the amount of nodes does not seem to affect the localization error. With high measurement error, it is clear that localizing with more nodes results in lower localization error.

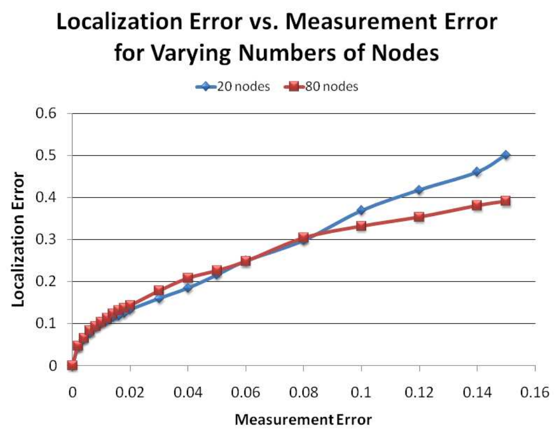

Fig. 5. A graph comparing estimated error versus measurement error

VII. Conclusion

The simulation was implemented successfully and can be adjusted to encompass error for testing. The results show that there is a trend that exists in our implementation. This example is a simple implementation that produces results that can applied to a wide variety of alterations to this experiment. It can be concluded that with enough visible markers, high error does not effect the robots ability to localize by some radical degree.

VIII. Future Work

The concepts presented in this paper will be implemented on real robots with real sensors where the effects of machine error will be observed. Our implementation will need to test scarce areas where there are not that many markers/ square units and excessively cluttered areas. If the trend still exists at this point, then our algorithm will be quantified.

References

[1] E. M. Arkin, M. A. Bender, J. S. B. Mitchell, and V. Polishchuk. The snowblower problem. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2006.

[2] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better flocking behaviors using rule-based roadmaps. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 95–111, Dec 2002.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better group behaviors in complex environments using global roadmaps. In *Artif. Life*, pages 362–370, Dec 2002.

[4] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Roadmap-based flocking for complex environments. In *Proc. Pacific Graphics*, pages 104–113, Oct 2002.

[5] M. Betke and L. Gurvits. Mobile robot localization using landmarks. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 135–142, 1994.

[6] T. S. Levitt, D. T. Lawton, D. M. Chelberg, and P. C. Nelson. Qualitative navigation. *DARPA Image Understanding Workshop*, 2:319–26, 1988.

[7] M. J. Mataric. A distributed model for robot environmentlearning and navigation. Technical Report 1228, M.I.T., Cambridge, MA, 1990.

[8] S. Rodriguez, J. Denny, J. Burgos, A. Mahadevan, K. Manavi, L. Murray, A. Kodochygov, T. Zourntos, and N. M. Amato. Toward realistic pursuit-evasion using a roadmap-based approach. *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1:173845, May 2011.

[9] S. Rodriguez, J. Denny, A. Mahadevan, J. C.-T. Vu, J. Burgos, T. Zourntos, and N. M. Amato. Roadmap-based pursuitevasion in 3d structures. May 2011.

[10] K. T. Sutherland and W. B. Thompson. Inexact navigation. *IEEE Int. Conf. Robot Automat.*,, pages 1–7, 1993.