SPATIOTEMPORAL QUERY STRATEGIES FOR NAVIGATION IN DYNAMIC SENSOR NETWORK ENVIRONMENTS

Gazihan Alankus, Nuzhet Atay, Chenyang Lu, O. Burchan Bayazit

{gazihan,atay,lu,bayazit}@cse.wustl.edu
Department of Computer Science and Engineering, Washington University in St. Louis, MO, USA

ABSTRACT

Autonomous mobile agent navigation is crucial to many mission-critical applications (*e.g.*, search and rescue missions in a disaster area). In this paper, we present how sensor networks may assist probabilistic roadmap methods (PRMs), a class of efficient navigation algorithms particularly suitable for dynamic environments. A key challenge of applying PRM algorithms in dynamic environment is that they require the spatiotemporal sensing of the environment to solve a given navigation problem. To facilitate navigation, we propose a set of query strategies that allow a mobile agent to periodically collect real-time information (*e.g.*, fire conditions) about the environment through a sensor network. Such strategies include local spatiotemporal query (query of spatial neighborhood), global spatiotemporal query (query of all sensors), and border query (query of the border of danger fields). We investigate the impact of different query strategies through simulations under a set of realistic fire conditions. We also evaluate the feasibility of our approach using a real robot and real motes. Our results demonstrate that (1) spatiotemporal queries from a sensor network result in significantly better navigation performance than traditional approaches based on on-board sensors of a robot, (2) the area of local queries represent a tradeoff between communication cost and navigation performance, (3) through in-network processing our border query strategy achieves the best navigation performance at a small fraction of communication cost compared to global spatiotemporal queries.

1. INTRODUCTION

Awareness of the environment plays an important role in mobile robot navigation. Until recently, the robots mostly relied on the on-board sensors. However, as the technical challenges of sensor networks are being solved, a new interest raised to employ them in the robot navigation task. There are several advantages of using a sensor network in this task. Perhaps the most important one is a sensor network's ability to relay information from not only the robot's vicinity but also from distant regions of the environment. Reduction in the cost can be another advantage, since several distributed cheap sensors can replace expensive on-board sensors. Also, once a network is deployed, it can be used by multiple agents, and even help separated agents coordinate their movements.

Sensor networks are successfully utilized for robot navigation [1, 2]. However, most of these methods use all the sensors in the network increasing power consumption. Also, they distribute the path finding task to sensor nodes hence reducing the flexibility of using a different path finding algorithm. They may also fail to adapt multiple robots if the initial network was deployed for a single robot.

Path planning algorithms developed in the robotics community are capable of navigation in complex environments [3]. In particular we note the *roadmap methods* which can quickly answer many diverse path planning queries in the same environment using a map, typically constructed during preprocessing, containing a network of representative feasible paths in the environment. In essence, these maps function similarly to driving maps in that one plans a route by first locating their initial and final positions and then selecting a route connecting them from the roads and highways shown on the map.

In this work, we investigate how the addition of spatiotemporal information through the sensor network can be used to build a roadmap of the environment which enables more sophisticated navigation. Our goal is to navigate safely in a danger field, i.e., reach a goal while avoiding the dynamic dangerous regions. This dynamism requires the robot to modify its route continuously to avoid the danger. In order to build a roadmap we use a Probabilistic Roadmap Method (PRM) [4]. Probabilistic roadmap methods are shown to be probabilistically complete and they are successful where deterministic algorithms failed due to time complexity of the navigation problems. They are very fast, and can be applied in the dynamically changing environments. In our integration of sensor networks with PRMs, sensor networks pass the spatiotemporal information to the robot. The information can be partial (e.g., only local vicinity of the robot), or global (e.g., all sensors response to a query). The robot uses this information to update its roadmap. If it discovers that the current route goes through a dangerous region, it finds alternative routes on the roadmap. Please note that our work can easily be applied to other areas where the robot navigates based on the dynamically changing environmental information. For example, in a scenario where oil is spilled to the environment, a cleaning robot can gather the pollu-

[1]This research was supported in part by NSF under Grant No. CCR-0325529.

tion information through sensor network and can follow a path that will collect the oil. In an exploration scenario, the motes can store the information about whether the region was visited or not.

In earlier work, Bayazit et.al. [5] showed that roadmap algorithms capturing the global spatiotemporal information about the environment performs better than other commonly used navigation algorithms in multi-robot scenarios. In a sensor network, the global spatiotemporal information can be captured by querying all sensors. However, this global query approach consumes significant energy and may cause network congestion. Furthermore, it may not be necessary or beneficial to query sensors far away from the robot when sensor data change rapidly in dynamic environments. To overcome such drawbacks we suggest two new query strategies to facilitate efficient navigation, *local query* and *border query*. In local query, a robot only queries its spatial neighborhood, where the size of the query area can be tuned to achieve desired tradeoff between energy/communication cost and navigation performance. In border query, only the sensors in the border of a danger responds to a given query.

In order to validate our approach, we have tested our system both on a real sensor network with a real robot and a simulated robot with a simulated network. Our sensor network simulator is built on top of NIST Fire Dynamics Simulator [6]. Using our software, we can simulate a sensor network which can relay real-time temperature information from a spreading fire. Combining our sensor network simulator with a robot simulator, Player/Stage [7], we found that when they are supplied with real-time temperature data by a sensor network, PRMs can successfully navigate a robot in a fire. We also found that using a border query strategy, we can capture the spatiotemporal information at the reduced cost. Our experiments with the real robot showed that we can use our algorithm on a real scenario as well.

In the next section, we give a summary of related work. In Section 3 we briefly describe our system. Section 4 discusses different query strategies we investigate. Section 5 describes our navigation strategy. We present our experimental results in Section 6 and Section 7 concludes our paper.

2. RELATED WORK

Recently there were successful applications of using sensor networks to navigate a robot to a goal [1, 2]. These algorithms use sensor networks to compute a path for the robot. They use wavefront expansion to update the path information which may result in a flood of messages in the network increasing the communication overhead and power consumption. Also since they need continuous update, there will be less time for nodes to sleep. In contrast, we propose a new strategy where nodes only send messages when they are near the robot and within the range of a danger. So if a node is outside of the robot's query range it could stay in sleep mode. We also believe that our approach is more flexible. For example, if the network is deployed for one robot and if two robots are required to move to two different goals, waves originating from the two goals would create problems. In our approach, the path finding is done by the robot, hence reducing the computational cost over the network and adding some flexibility. For example, two robots can still use the same network without message congestion.

There are other applications of sensor networks for robot navigation task. In [8], a sensor network gives a path to help an autonomous agent to reach a goal. The path is found using the wavefront expansion of [1]. Similarly, [9, 10] use a potential field based on value iteration to find a path and direct the robot. Hence all of these algorithms also suffer from the drawbacks discussed above. In addition to using the sensor network for navigation, mobile agents are also used to increase connectivity of a sensor network [11, 12, 13]. In order to reduce the network response time and power consumption, MobiQuery [14] utilizes prefetching so that a robot can have the sensor data ready when it reaches a destination.

3. SYSTEM OVERVIEW

Our system consists of three components that enable safe navigation in a dangerous region: (i) a sensor network to collect real-time information about the environment, (ii) a robot with a mote connected to it, and (iii) a controller which navigates the robot based on the information from the sensor network and on-board sensors (see Figure 1). In our implementation we have both physical and simulated components, i.e., we can replace a real robot with a simulated robot or replace the real sensor network with a simulated network. The details of the simulators are described in Section 6.1.

We have some assumptions about our system. First, we assume the robot knows its location. The motes in the sensor network are also assumed to know their locations. These assumptions are realistic since: (a) robot may have on-board odometry and floor plan, (b) the positions of the motes can be assigned during the deployment or they can be computed later using the localization algorithms suggested in [15, 16]. The environment coverage of the sensor network is uniform and we assume symmetric radio links between neighboring motes, which can be achieved using an approach similar to [17]. Please note that since our goal is to validate our approach, we haven't implemented issues such as sensor noise, sleeping schedule [18, 19], network congestion control [14] or border detection [20] in our experiments. We are working on addressing them in our system with the addition of algorithms related to those issues. The communication between the robot controller and the sensor network is done through multi-hop communication between a mote connected to the controller and nearby motes of the sensor network. The system has two kinds of messages: a query message, where robot requests the danger information from the motes, and a data message where a mote returns the danger information. The query message is broadcasted once by the robot and propagated in the sensor network, and the motes receiving the query message sends the data messages until some predefined time is expired.

Our system integrates its components in the following way. The controller uses a PRM algorithm to safely navigate in the environment (see Section 5). It uses the danger information of the environment gathered through the sensor network. The controller periodically queries the sensor network through the mote that is connected to the robot. The query dissemination and data collection components of our system run on all of the motes in the network. When the controller sends the query message to nearby motes, the query dissemination component in those motes forward this message to the motes in the environment. As a response to this query, motes generate data messages and using the data collection component they send their data back to the mote on the robot over multiple hops. The mote on the robot forwards these messages to the controller and it plans the path and moves the robot accordingly.

We address the power consumption and network congestion using different query strategies. We use both general spatiotemporal queries and queries that specify the areas of interest. We have classified our queries in two types: spatiotemporal and border-response. We further classified the queries into two types based on their ranges (i.e., vicinity of the robot or entire network): global and local. We will discuss these strategies in the next section, however a brief overview of them is described below.

Global vs local query. The purpose of global query is to let the robot know about the whole environment to plan a good path, while in local query the information about the environment is limited. All the motes in the network respond to a global query, whereas in local query only a group of motes that are close to the robot respond. This would reduce the number of messages generated by the sensor network.

Border-response vs spatiotemporal query. Spatiotemporal query is a regular query where each mote in the query range responds to query. Even in a local query, this may generate an extra amount of messages. In order to balance the trade-off between safety and fair usage of network resources, we suggest a new query response strategy where only the sensors which are on the borders of dangerous areas respond. The other nodes act only as routers to propagate data messages from border nodes to the robot. Similar to spatiotemporal query, a border-response query can be done in local or global level.

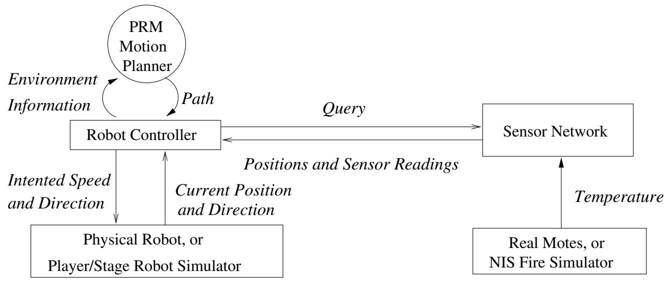

PRM Motion Planner
Environment Information
Path
Robot Controller
Query
Sensor Network
Positions and Sensor Readings
Intented Speed and Direction
Current Position and Direction
Temperature
Physical Robot, or Player/Stage Robot Simulator
Real Motes, or NIS Fire Simulator

Fig. 1. System overview.

4. QUERY STRATEGIES

As we have discussed before, our basic query, spatiotemporal query generates a tree from a query source. We have also extended this general query type to border-response in order to make fair use of the network resources without compromising safety. We are planing to incorporate MobiQuery [14] to our system.

Spatiotemporal query. In a spatiotemporal query strategy, whenever the robot needs to make a query, the mote on the robot broadcasts a query message $Q(q_c, r, t_r, \delta t, t_p)$, indicating the query center (q_c) and the query radius (r) in order to be able to identify the sensors to respond, query sent time (t_r), the query's validity duration (δt) which determines the lifetime of the query, the period of the query (t_p) which indicates data generation period of the motes. An internal parameter, a depth field is also attached to a query. The depth field is 0 for the initial query. Once the robot broadcasts a query, it is received by the motes that are in the communication distance. Those motes increase the depth field and broadcast it again. The mote that sent the query message with the least depth is set as the parent for the query by each mote. This way an implicit query tree is formed. If the query reaches a mote which is outside the geographic area specified by q_c and r, it stops propagating. Note that both global spatiotemporal query and local spatiotemporal query share this same infrastructure. The global spatiotemporal query is achieved by setting the query radius to ∞.

After the query is disseminated to the network, all the motes within the vicinity r create data messages every t_p. Data messages contain the a destination field and a data field. The destination is always the motes parent that was set before in the query phase. In addition to continuously sending the data messages every t_p, the motes also listen for data messages that are addressed to them (by their children). If a data message is received from a child, a mote forwards it to its parent. Since the root of the tree is the mote on the robot, every data message is eventually reaches the robot. This process continues until all the answers are send back to the robot (see Figure 2(b)). In our implementation, the robot is slow enough and stays in the communication distance while waiting for the data messages. In the future we will also add the possibility to specify a pick-up location as in MobiQuery, so that even if the robot is out of the broadcast distance of the original interface nodes it can get the answers from a nearby node in its path.

Border-response. There are several drawbacks of having all sensors respond a query, such as unnecessary power consumption and network congestion. In addition, when the robot controller is flooded with several responses, its processing time may increase. These drawbacks persist even if the response is restricted to some vicinity. In order to overcome such drawbacks, we have developed a query strategy where only the nodes with significant information respond to a given query. Other nodes do not generate data messages to be sent to the robot but forward the messages that are sent to them. Remember that our goal is to navigate the robot while avoiding the dangerous areas. As long as we know about the borders of the dangers, we can avoid them. In other words, we do not need the information from the motes that are not on the border of dangerous areas. Figure 2(c) shows how the border-response query works. In the situations like Figure 2(a) that have large dangerous areas, several (proportional to mote distribution and the area) motes do not generate data messages when a border-response query is used.

In order to implement border-response, we have added a *border* flag to the data messages exchanged between the motes. This binary flag represents if a mote thinks it is in the border of a danger or not. If the *border* flag is not set, the message is not forwarded by the motes to their parents. In order to determine a mote is a border mote or not, every mote overhears all data messages that it can, even though they are not intended for that mote to be aware of readings of its neighbors. By comparing its own readings to its neighbors', a mote can determine if it is on the border or not. If all the neighbors and the mote have low readings, the mote is in a safe zone. Similarly if all the neighbors and the mote have high readings then the mote is deep inside a danger zone. In both of these cases, the mote is not on the border, hence it turns off the *border* flag in its data messages. Since it continues broadcasting, its parent knows the mote is not in danger. If the mote suddenly stops broadcasting, the parent assumes the child is in danger and updates itself to on the border status.

This way, only the data messages of border motes are forwarded to the robot and the amount of messages are reduced in the network. Before determining if a mote is in the border or not, it sets the *border* flag in its messages to on, in order to keep the robot informed about the environment. This approach enables the robot to get readings from all the motes in the beginning. After learning about the readings of their neighbors, some of the motes start to set their *border* flags off. The robot has an idea of what would be the readings from silent motes, since it knows their old readings and they are on the same side of the threshold until they become border motes. Therefore, this method reduces the forwarding of the data messages with minimum information loss.

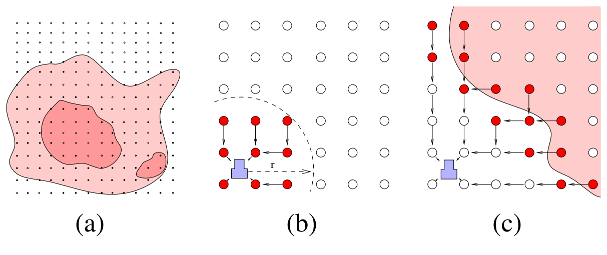


Fig. 2. Our sensor network query strategies. (a) Sensor network with different danger levels (darker regions). (b) Spatiotemporal query gets response from the nodes in the vicinity. (c) Border-response query returns the answer from the nodes in the border of danger. Red (darker) circles denote the motes that generate data messages.

Node failures. There are two cases when a node fails, either it stops sending messages or it sends wrong information. We believe we are handling both cases. In the first case, a mote stops sending the messages when either its destroyed by the danger or it has a mechanical failure. As we will see in the next section, in order to increase the safety, the robot pessimistically assumes a region is dangerous if it does not receive any danger information from the motes in that region. This way if a mote does not respond the region covered by that mote is avoided. If the failed mote is a parent mote, than the messages from its children will also fail to reach the robot so the children will also be assumed in danger. However, at the next query, alternative routes to children will be established and the status of their regions will be updated. In the second case where the mote sends the wrong danger information, the mote will either send high danger when there is no danger or low danger when there is one. In the first scenario, the robot would just avoid the region with the wrong information. Hence there is no significant effect of this kind of error. In the second scenario, while evaluating the possible paths, since the robot incorporates the danger information from nearby motes, the impact of the wrong information will be minimal.

5. NAVIGATION STRATEGY

5.1. Roadmap-Based Path Planning with PRMs

Given a description of the environment and a movable object (the 'robot'), the motion planning problem is to find a feasible path that takes the movable object from a given start to a given goal configuration [3]. Since there is strong evidence that any complete planner (one that is guaranteed to find a solution, or determine that none exists) requires time exponential in the number of degrees of freedom (DOF) of the movable object [3], attention has focused on randomized or probabilistic methods.

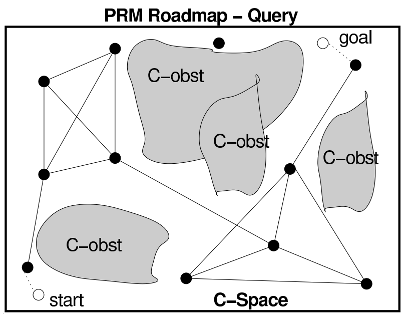


Fig. 3. Querying a PRM roadmap (C-space).

As mentioned in Section 1, our approach utilizes a roadmap encoding representative feasible paths in the environment. While noting that our techniques could use any roadmap, our current implementation is based on the probabilistic road-map (PRM) approach to motion planning [4]. Briefly, PRMs work by sampling points 'randomly' from the robot's configuration space (C-space), and retaining those that satisfy certain feasibility requirements (e.g., they must correspond to collision-free configurations of the movable object). Then, these points are connected to form a graph, or roadmap, using some simple planning method to connect 'nearby' points. During query processing, the start and goal are connected to the roadmap and a path connecting their connection points is extracted from the roadmap using standard graph search techniques (see Figure 3).

5.2. PRM Navigation in Dynamic Sensor Networks

The original PRM algorithm is developed to avoid obstacles. Two nodes are connected if a robot can reach from one configuration to another configuration using a simple planning algorithm. We need to modify it so that the robot will follow not only a collision free but also a safe path. In our implementation, the robot does not need to know the positions of the sensors. It is only aware of its position, and if there are obstacles in the environment, their locations. Our navigation algorithm first builds a roadmap of the environment. As discussed before, a roadmap is basically a weighted undirected graph. A path is a sequence of edges, first connecting robot's current configuration to the roadmap, following the roadmap edges, and connecting the roadmap to the goal configuration. Since there may be more than one path reaching the goal, the robot should select the most cost efficient path, i.e., a path that has the lowest weight among other sequences. If the edge weights are known, this path can be found using Dijkstra's shortest path algorithm [21]. We defined the weight ($weight_{e_i}$)of each edge (e_i) connecting two configurations c_k and c_l as $weight_{e_i} = w_{dist} \times |c_l - c_k| + w_{temp} \times e_{i temp}$, i.e., weighted sum of the length of the edge and the temperature of the edge. If w_{temp} is 0, the robot will take the shortest path to the destination, if w_{dist} is 0, the robot will take the safest path to the destination. This formulation of an edge weight requires finding the temperature of an edge. We defined a temperature of an edge as the maximum temperature the robot would face if it would have taken that edge. This temperature is found by first discretizing the edge to a constant number of points (100 in our experiments) that are equally spaced on the edge and then finding the the temperature on each point. Note that we do not have a temperature reading at the exact location of the point unless we have a sensor there. On the other hand, several sensors may cover the same point because their sensing ranges may intersect. So we interpolate the temperature by averaging the temperature readings from the nearby sensors (i.e., the sensors which cover the point). As the new sensor readings are obtained, the edge weights of the roadmap are updated, hence modifying the path to avoid the spreading fire. Please note that by using a probabilistic roadmap algorithm, we gain an advantage over other motion planning techniques in terms of efficiency in the computation while maximizing our objective (i.e., staying away from the danger).

6. EXPERIMENTS

In our experiments we would like to answer following questions: (i) how successful our algorithms are preventing the robot moving into the danger, (i) how well different strategies work, and, (iii) how the algorithm performs with a real robot.

In order to answer those questions, we have run our experiments both in the simulated sensor network and a real robot with MICA2 motes.

6.1. Sensor Network Simulator

In order to validate our approach we developed a sensor network simulator which is available at:

http://www.cse.wustl.edu/∼ bayazit/software.

In the simulation, the robot controller and the sensor network controller run synchronized in real-time. The mote on the robot and the sensor network is simulated at message-passing level, including packet loss probability, radio and processing delays.

Since we are interested in robot navigation in the case of dynamically changing dangers (i.e., spreading fire), we need a realistic representation of the danger. We have selected NIST Fire Dynamics Simulator (FDS) [6] to simulate a fire. During the simulation, fire may separate to several branches, some branches may continue to spread while some get extinguished. FDS runs at small time steps and stores the temperature information of selected locations. One drawback of FDS is that a realistic simulation would require hours to run. In contrast, our sensor network simulator is in real-time. Our solution for this difference in running time is based on the assumption that the movement of the robot does not have a significant effect in the fire. This way, we can run FDS without a moving robot. Later, our sensor network simulation reads FDS temperature files at specific time steps and respond the queries. This simulates a realistic distribution of the fire. A typical temperature readings by our sensor network simulator after FDS simulates the fire distribution can be seen in Figure 4.

While FDS gives us a realistic fire simulation, we also need our sensor network to be able to simulate realistic working conditions of real sensors nodes. For this purpose we have utilized several parameters: (i) *communication distance*, the distance that two nodes can exchange messages, (ii) *loss ratio*, the probability that a packet can be lost, (iii) *radio delay*, the delay introduced by radio transmission, (iv) *processing delay*, the delay introduce by processing in motes, and, (v) *sensor distribution*, position and number of motes in an environment. In order to increase the realism of the simulator, we have used the parameters close to those obtained from the experiments with real motes [14, 22]. In the simulation, when a node receives a message and finds out that it needs to respond, the node prepares the message and sends it after some simulated processing delay. The receiving node will see the message ready after the sum of radio delay and processing delay. When a query is disseminated by the robot, the answer will be ready at the query center after all delays are included in the total time. We assume links are symmetric so that each node will send messages to its parents.

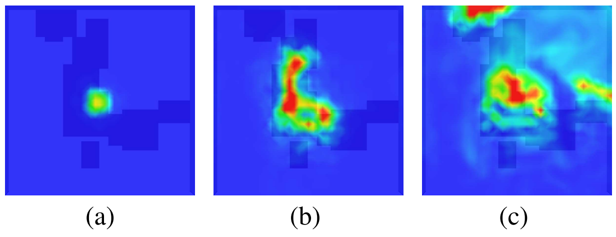


Fig. 4. Example simulated sensor readings by our sensor network simulator for a spreading fire. The light regions are high temperature areas.

6.2. Experiments with the Simulators

In these experiments we would like to learn how the different parameters in the network can effect our algorithm. We are interested in the safe passage of the robot to a goal, so we compare the success rate of the travel with different parameters. In all our experiments we compare the performance of our algorithm with different query response range, r. We have compared the success of the algorithm when either robot is getting its data from on-board sensors or getting its data from the sensor network with a query range of 15, 20, 30, 40, 50, 60, 70 meters or all sensors (global). We have also compared spatiotemporal query to border-response. The robot is simulated using Player/Stage [7].

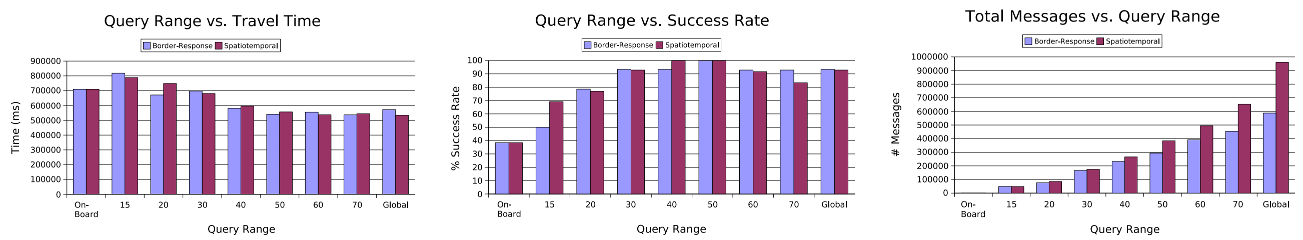
Environment. All our experiments are run on a 100x100 meters environments. We have used 10 different environments. The robot always starts at the position (10,10) and tries to reach the position (90,90). Each environment has 10 rectangular burning material randomly placed. The dimensions of burning materials are probabilistically generated using a normal distribution of ($\mu = 20, \sigma^2 = 5$). Fire always starts at the center of the environment. We have run NIST Fire Dynamics Simulator in each environment for a simulation of 50 minutes. An example distribution of the fire can be seen in Figure 4. We assume, the robot starts after 400 seconds of burning to let the fire spread the environment. In the sensor network simulator, there are 11x11 motes distributed around a uniform grid. Each mote has a radio delay of 0.2 seconds, processing delay of 0.2 seconds and a message loss rate of 20% (ARQ is not used in our simulations). The radio communication range for the motes are 15 meters. The simulated robot is a Pioneer-III DX [23] (similar to our robot) with a maximum speed of 20 meters per minutes and 16 on-board temperature sensors distributed uniformly on a circle around the robot with 1 meter radius.

Experiments are run on a Pentium-IV 3 Ghz Linux machine with 2GB memory. The movies of the experiments can be found at *http://www.cse.wustl.edu/∼bayazit/sn*.

Results. In our experiments we have first compared the effect of query strategy to travel time of robot to reach the goal. Figure 5(a) shows our result. In the figure x-axis represent different query ranges, including only on-board temperature reading and y-axis is the time it took robot to reach the goal. The dark colored bar is the results for spatiotemporal query for given range, and the light colored bar is the results for border-response. Both type of strategies run 10 different environment twice (with a different roadmap at each run). The roadmap size is 100 nodes, and we try to connect every node to all the remaining nodes. The results show the average of a total of 20 runs on each query strategy. From the results it is clear that as the range of query increases, the time to reach the goal decreases in both spatiotemporal and border-response strategies. This shows that, the better the robot knows how the fire is distributed, the faster it can reach the goal. If the robot knows only a small area, a path could take the robot closer to fire, so robot spends extra time to avoid such local encounters. There were no significant differences in the running times of both strategies. One interesting observation was that if the robot uses on-board sensors, it reaches the goal fast. This can be explained by the fact that since the robot's knowledge about the temperature data is limited, the distance plays an important role in the edge weight computations (see Section 5.2). Also, after the query range of 50 meters (half of the environment), there is no significant decrease in the travel time. That shows that after some query range is reached, more data no longer becomes an advantage.

A fast travel time does not guarantee a safe travel. We accept a path safe if the robot always stays in low temperature (less than 70 degrees Celsius) areas. Figure 5(b) shows the success rate (robot reaches the goal while following a safe path) for a query strategy over 20 runs. The success rate of our query strategies increase until we have a query range of 50 meters (similar to travel time change after 50 meters). Surprisingly if the query range further increases the success rate starts to decrease. We believe this phenomena is result of increased data flood to the robot controller. As the number of messages to the controller increases, the processing time in the controller increases as well. We have a very fast spreading fire, and as the number of messages increase, the robot will spend more time on the processing and fire will catch up with the robot.

It is clear from Figures 5(a) and (b) that sensor networks gives significant advantage to robot navigation over traditional approaches based on-board sensors. Using both query techniques were safer than on-board sensor navigation. For larger query ranges, usage of the sensor network also reduced the travel time. On the other hand, there are no significant performance penalties for using border-response instead of spatiotemporal query. Next, we investigate the power consumption. For this purpose, we need a metric that approximates the power consumption. Hence, we have selected to count the number of messages passed in the network. Figure 5(c) shows the message count for each strategy. As expected, the number of messages are significantly less in border-response when query range is large.



(a) (b) (c)

Fig. 5. Experimental results for simulated sensor network using different query strategies. (a) Travel time of the robot. (b) Success rate of the robot reaching the goal. (c) Number of messages sent.

Based on these figures, we found out that a query range of half of the environment (in our case 50 meters) with border-response strategy gives best safety while introduction minimum sensor network resource overhead.

6.3. Experiments with the Real Robot

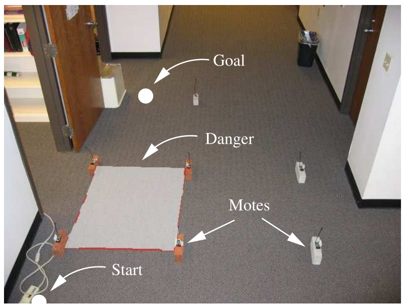


Fig. 6. Test arena for the real robot.

We have also tested our algorithm in a real robot. We have evaluated the different query strategies as well as validated our navigation strategy in the simulation results. In this experiment, we want to show that it is feasible to use a real sensor network with PRMs. The robot we have used is a Pioneer-3 DX by ActiveMedia [23]. Our test arena is 2x3m, robot starts in $(0, 0)$ and tries to reach $(0, 3)$. There is a fire in the 1x1 meter square centered at $(0.5, 1.5)$. Figure 6 shows our test arena. Our sensor network in this environment is made up of 7 MICA2 motes with temperature sensors. The robot controller has also one mote to communicate with the sensor network. 4 motes are deployed on the corners of the square in fire. Due to safety concerns, instead of a real fire, we have setup those motes to broadcast high temperature. We first tested our robot to find its path without any help from the sensor network. As before, the robot controller used the PRM algorithm to find a path. Due to the fact that we have a small sensor network, we have implemented a global spatiotemporal query strategy. Since the shortest path is through the fire region, robot went through danger area (see Figure 7). Next, we have used sensor network to supply temperature information of the arena. Using these information, robot was able to avoid the fire region and safely reached the goal (Figure 8). Since we have a limited number of motes, we only tested spatiotemporal query strategy (all motes were in the border of danger). However, as we have showed in the previous section, border-response strategy performs very similar to spatiotemporal query strategy. Since the arena is small, we have used 25 nodes to generate the roadmap. Since the passage through fire was shorter, travel time for that route was slightly faster than travel time for the safe passage (39 seconds vs. 52 seconds).

7. CONCLUSION

In this paper we have presented how sensor networks may assist probabilistic roadmap methods in robot navigation. We have showed that sensor network increases the performance of a robot controller. We have proposed a border-response query strategy that successfully minimizes the trade-off between robot safety and sensor network resource consumption. Our future work includes experimenting with a larger sensor network with the real motes and coordinating multiple robots over the sensor network. We are also working on issues related to the sensor networks, such as network congestion, noise, and better border detection etc.

8. REFERENCES

[1] Q. Li, M. De Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. 2003, pp. 313–325, ACM Press.

[2] K. J. O'Hara and T. J. Balch, "Distributed path planning for robots in dynamic environments using a pervasive embedded network," in *AAMAS*, July 2004, pp. 1538–1539.

[3] J. C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[4] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[5] O. B. Bayazit, J.-M. Lien, and Nancy M. Amato, "Better flocking behaviors using rule-based roadmaps," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Dec 2002.

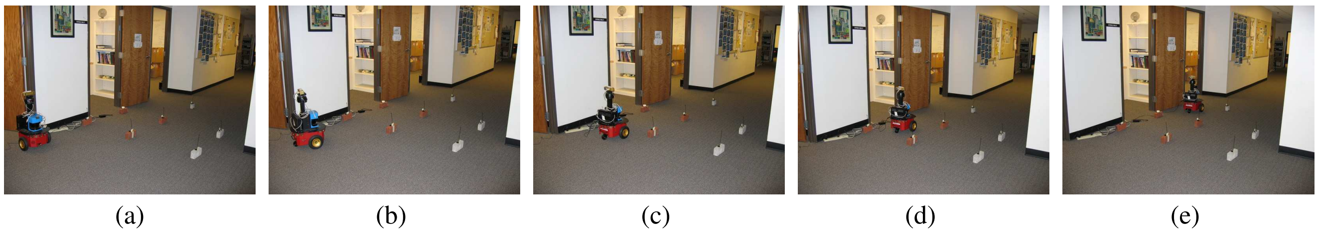


Fig. 7. Navigation without a sensor network. Robot goes through the simulated danger region.

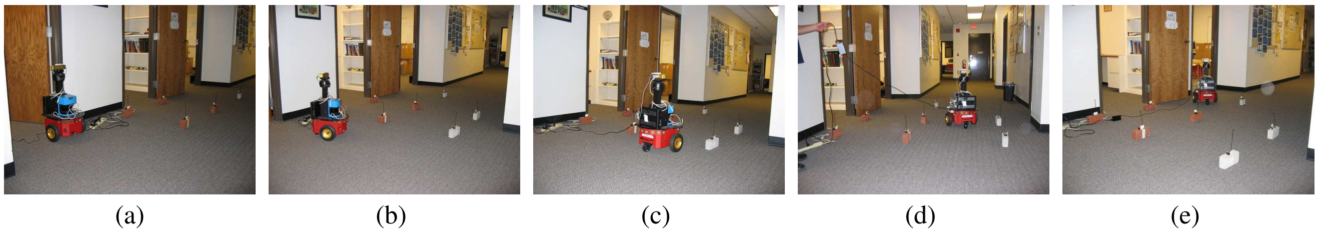


Fig. 8. Navigation with spatiotemporal information from a sensor network. Robot avoids the fire and takes a safe route.

[6] K. McGrattan, *Fire dynamics simulator (version 4) technical reference guide*, National Institute of Standards and Technology, 2004.

[7] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, , and M. J. Mataric, "Most valuable player: A robot device server for distributed control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1226–1231.

[8] P. Corke, R. Peterson, and D. Rus, "Coordinating aerial robots and sensor networks for localization and navigation," in *Proceedings of the Seventh International Symposium on Distributed Autonomous Robotic Systems*. June 2004, Distributed Autonomous Robotic Systems 6, Springer-Verlag.

[9] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 181–196, August 2004.

[10] M. A. Batalin and G. S. Sukhatme, "Sensor network-based multi-robot task allocation," in *Proceedings of IEEE/RSJ International Conference On Intelligent Robots and Systems*, October 2003.

[11] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Deployment and connectivity repair of a sensor net with a flying robot," in *Proceedings of the Ninth International Symposium on Experimental Robotics*. June 2004, Sage Publications.

[12] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. June 2004, pp. 111–124, ACM Press.

[13] D. B Johnson and D. A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Imielinski and Korth, Eds., vol. 353. Kluwer Academic Publishers, 1996.

[14] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A spatiotemporal query service for mobile users in sensor networks," in *International Conference on Distributed Computing Systems (ICDCS'05)*, 2005.

[15] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. 2003, pp. 81–95, ACM Press.

[16] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, August 2001.

[17] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio iregularity on wireless sensor networks," in *International conference on Mobile systems, applications, and services*. 2004, pp. 270–283, ACM Press.

[18] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Mobile Computing and Networking*, 2001, pp. 85–96.

[19] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. 2003, pp. 28–39, ACM Press.

[20] R. Nowak, U. Mitra, and R. Barniuk, "Estimating inhomogeneous fields using wireless sensor networks," *JSAC*, vol. 22, no. 6, pp. 999–1006, August 2004.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms, second edition*, MIT Press and McGraw-Hill Book Company, 2001.

[22] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. 2003, pp. 1–13, ACM Press.

[23] "Activmedia," http://www.activmedia.com/.