2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Distributed Roadmaps for Robot Navigation in Sensor Networks

Zhenwang Yao and Kamal Gupta
Robotic Algorithms and Motion Planning (RAMP) Lab,
School of Engineering Science, Simon Fraser University
Email: {zyao,kamal}@sfu.ca

Abstract— **This paper studies a *distributed path planning* problem: how can a sensor network help to navigate a robot to its desired goal in a distributed manner. We consider the case where each sensor node is equipped with sophisticated sensors capable of giving a map for its sensing region. We propose a distributed sampling based planning framework (*Distributed PRM*), where every sensor node creates a local roadmap in its locally-sensed environment; these local roadmaps are "stitched" together by passing messages among nodes and form a larger implicit roadmap without having a global representation. Based on the implicit roadmap, a feasible path is computed in a distributed manner, and the robot moves along the path by interacting with sensor nodes, each of which gives a portion of the path within the local environment of the node. Preliminary simulations show the proposed framework is able to solve path planning problem with low communication overhead.**

I. Introduction

Robot navigation is a fundamental problem in mobile robot research, and the navigation problem in the context of sensor networks has been studied in several recent works [1], [2], [3], [4], [5], [6], [7], [8], [9]. While traditional navigation relies on sensors on-board the robot to sense the environment, sensor networks greatly extend the sensing capability of the mobile robot, and make it possible for the robot to move beyond its current sensing range, and respond to distant events. This is particularly useful in navigating a robot (or humans) across a hazardous environment [10]. In such applications, sensor nodes can sense "dangers" (e.g., spots with excessive heat or radiation), and model danger areas as obstacles (in classic path planning sense). However, all existing works assume only simple sensors, such as temperature sensors, are available, and hence the entire sensing region of a sensor is either free or in danger (so we call such sensing model as "binary" model). Moreover, they either assume light-of-sight communication [5], which can be blocked by physical obstacles (e.g. walls), or they do not take into account physical obstacles in the planning phase, and leave physical obstacle avoidance to the execution phase and require replanning when the robot detects an obstacle in the way [8]. This may result in quite lengthy and costly paths.

Clearly, such a sensing and navigation scheme is too simple for more sophisticated applications. For example in emergency rescue applications [11], a pre-deployed static sensor network, such as overhead cameras, or a mobile robotic sensor network, where each robot is equipped with laser scanners, can be used to detect victims and guide fire-fighters and rescue robots. These sensors provide much richer information, e.g., a spatial map rather than a single reading. At the same time, the navigation tasks in this case are usually more constrained. For instance, collapsed walls may block some areas, and a feasible path may run through a narrow passage of some sort lying across sensed regions of multiple sensors. In order to effectively navigate a rescue robot or a fire-fighter through the rubble, these multiple sensors need to cooperate with each other, take into account obstacles when planning paths, and thereby plan a feasible and more efficient path.

Such cooperation can be costly (in term of communication) due to the distributed nature of sensor networks. More generally, building a centralized and global representation of the environment can be time- and bandwidth-consuming, therefore distributed approaches are intrinsically desirable for better scalability and efficiency. This is true even in systems with simple sensing, and our previous work [9] and many others [4], [6], [7], [8] have studied how to reduce communication costs therein. In our case, each node perceives much more information, i.e., a local map around it, and collecting such maps to build a centralized representation would have huge communication burden.

In this paper, we propose a distributed planning framework, *Distributed PRM* (D-PRM), to systematically incorporate a general spatial sensing model for each sensor. It takes into account "obstacles" in determining feasible paths. Each sensor creates a local roadmap (a patch) similar to the classic sampling based techniques, such as PRM [12] and RRT [13], but only in its locally-sensed environment[1]. Two different patches of roadmap are "stitched" together with a set of *relay points*, lying in the common region shared by the two patches. Sensor nodes mutually negotiate the connectivity of their patches by sending messages regarding the status of their respective relay points. When two adjacent nodes see a relay point free, it becomes a connecting point for the two patches. To find the shortest path on distributed roadmaps, a distributed (discrete) navigation field is created across the sensor network, which maps each sample in local roadmaps into distance to the desired goal, and the best path is computed by gradient descent.

To the best of our knowledge, this is the first work to study the distributed path planning in sensor networks with such complex spatial sensing capability. It is worth pointing out that there is another thread of research studying distributed motion planning from the perspective of parallelism [15], [16], [17], [18], which focuses on distributing computation into different processors. All these works, however, assume processors have access to either a shared or a private copy of a global C-space representation, and therefore address a different problem.

[1]We have also implemented a distributed RRT (D-RRT), however, due to space limitations we report only D-PRM. D-RRT implementation is rather similar. Please see [14] for details.

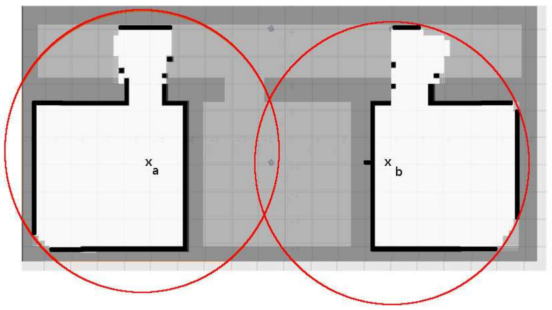


Fig. 1. Sensor model of nodes (a) and (b). Black represents *Blocked*, white represents *Free*, and Gray represents *Unknown*. The circles represent the respective sensor ranges.

The remainder of the paper is organized as follows. The problem formulation is given in Section II, overview of the proposed solution are given in Section III, details of proposed algorithms are given in Section IV, and simulation results are given in Section VI, followed by conclusions in Section VII.

II. Problem formulation

Consider a system with a sensor network and a robot. The sensor network consists of a set of sensor nodes, $\mathcal{S} = \{s_1, \cdots, s_n\}$. Sensor nodes know their own positions, $\{x_1, \cdots, x_n\}$, and two nodes can communicate if they are within distance, d_c, the communication range. The network formed by sensor nodes is modeled as a proximity graph, $G(V, E)$, whose vertices, $V = \{1, \cdots, i, \cdots, n\}$, represent sensor nodes, and an edge (i, j) represents the communication link between nodes i and j. The set of neighbors of node i is denoted by $N(i) = \{j \big| \|x_j - x_i\| \le d_c\}$;

A sensor node can sense obstacles and other potential dangers within its sensing range, d_s, and creates a map, $\mathcal{H}_i$, for its local physical environment, as shown in Fig.1. We assume $d_s = d_c$ for simplicity but can be easily extended (see Section V for discussion). A point on the map is in one of three different states: *Free*, *Blocked*, or *Unknown*, and we denote the free portion of $\mathcal{H}_i$ as $\mathcal{H}_i^{free}$. The distributed representation of the environment is $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_n\}$. Certainly two maps may overlap, and we call the overlapped region of two maps as *relay zone*, $RZ(i, j) = \mathcal{H}_i \bigcap \mathcal{H}_j$. Note that it is possible the same point in the overlap region has different states in two maps, for example due to occlusion, a point may be *Free* in $\mathcal{H}_i$, but *Unknown* in $\mathcal{H}_j$.

The robot, $\mathcal{A}$, is assumed to be a point (circular) robot, for which the workspace and the configuration space (C-space) are identical, although our approach can be easily extended to the more general C-space. The robot is mounted with sensor and wireless communication devices that can communicate with sensor nodes within d_c. For simplicity, we assume the robot knows its own location, x_A, this can be realized either by mounting the robot with devices such as GPS, or by localization using the sensor network [19].

Thus, the distributed path planning problem is to find a path, $\Pi = \{\Pi_1, \Pi_2, \cdots, \Pi_m\}$, for the robot to go from the current position, x_s, to a desired goal, x_g, such that (i) Π is a continuous path, i.e., $\Pi_k(1) = \Pi_{k+1}(0)$, and (ii) each path segment, Π_k, is collision free inside a local environment sensed by a sensor node.

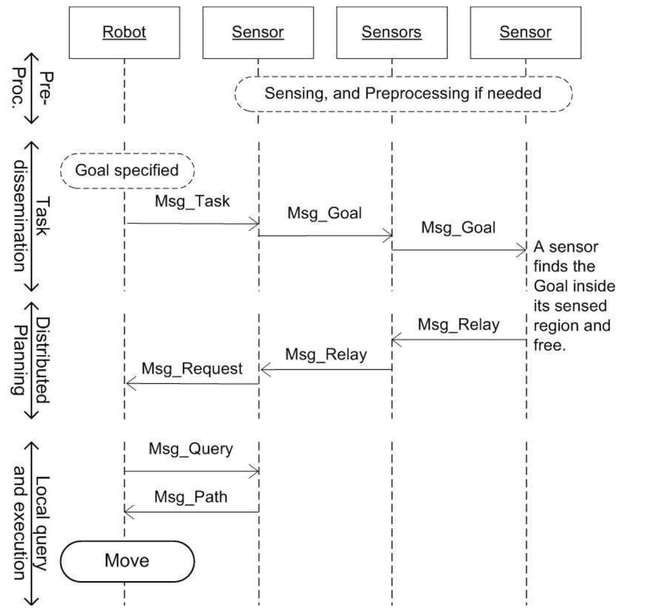


Fig. 2. Messaging in distributed path planning.

III. Overall Solution

The proposed distributed sampling based framework consists of four different phases outlined as follows. Detailed algorithms are given in Section IV. For conceptual simplicity we present these phases as distinct and in a sequential order. In practice, these phases may overlap or interleave, especially when there are dynamic changes in the environment.

1) Perception and pre-processing. In this phase, the sensor network gains knowledge of the system connectivity and the environment. Sensor nodes broadcast their basic information (e.g., positions), establish connections with their neighbors, and perceive their local environment by using sensors (e.g., cameras, laser, etc.). Pre-processing of planning is also done in this phase. For example, D-PRM creates a C-space roadmap, $\mathcal{R}_i = (\mathcal{V}_i, \mathcal{E}_i)$, for the locally-sensed environment, where $\mathcal{V}_i$ is the set of landmarks (random samples), and $\mathcal{E}_i$ is the set of connections among landmarks. The set of neighbors of a landmark, v, is denoted by $\mathcal{N}_i(v) = \{u \big| (u, v) \in \mathcal{E}_i\}$. Note that each roadmap is local, there is no sharing of roadmaps, and hence there is no communication involved, as far as roadmap building is concerned.
2) Task dissemination. When the robot wants to go somewhere, it sends out a request (in *MSG_TASK* message) to the sensor network asking for direction. A sensor node that has the goal inside its sensed region then initiates distributed planning, in the next phase.
3) Distributed planning. This is the main phase. Here, local roadmaps in different sensor nodes are "stitched" together by sensor nodes sending *MSG_RELAY* messages, thereby creating a distributed navigation field throughout the roadmap, based on a cost function, $C(v)$. With the navigation field, each sensor node knows the local segment of the path (to the goal) from any point inside its local environment via gradient descent.
4) Query and execution. The robot queries for the path

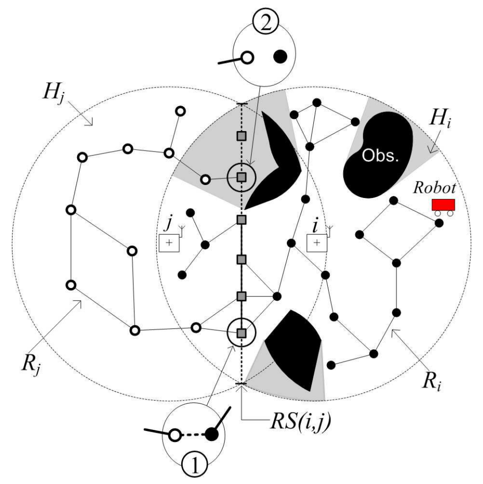


Fig. 3. Distributed Roadmaps. Dark dots represent landmarks for node i, white dots represent landmarks for node j, and gray squares represent relay points. As shown in the insets, each relay point comprises two relay landmarks, one each from the neighboring roadmaps.

from the sensor network (with *MSG_QUERY* message), and moves from one sensor node to another. A sensor node receiving the query message returns the local segment of the path to goal (in *MSG_PATH* message) based on its local navigation field. The robot moves along the returned segment of the path, reaching next sensor node and repeats the query/execution procedure again until the desired goal is reached.

A. Notation Summary

- $G = (V, E)$: proximity graph of the system;
- $N(i)$: neighboring sensor nodes of a node i;
- $\mathcal{H}_i$: local physical environment perceived by node i;
- $\mathcal{R}_i = (\mathcal{V}_i, \mathcal{E}_i)$: C-space roadmap for environment $\mathcal{H}_i$;
- $\mathcal{N}_i(v)$: neighboring landmarks of v on $\mathcal{R}_i$;
- $C_i(u)$: cost to goal from a local landmark $u \in \mathcal{V}_i$;
- $\pi_i(u)$: next landmark in the path from u to goal.
- $RZ(i,j) = \mathcal{H}_i \cap \mathcal{H}_j$: relay zone (in physical space) between node i and j;
- $RS(i,j) = \mathcal{R}_i \cap \mathcal{R}_j$: relay set (in C-space) between node i and j;

We use i, j, k to refer to sensor nodes in the sensor network, and u, v, w for landmarks in a roadmap.

IV. Algorithms

A. Preprocessing

a) Local environment perception: A sensor node senses the environment with its own sensor. In our simulation, each sensor node is assumed equipped with a laser range-finder to sense the environment, and uses grid map as local world representation, however other choices of sensors (e.g. cameras) and world representation (e.g., continuous function) are also possible, and our algorithms easily extend to these cases as well.

b) Relay sets - the stitches: $\mathcal{R}_i$ defines local C-space connectivity corresponding to $\mathcal{H}_i$, and connectivity between $\mathcal{R}_i$ and each of its neighbors, say $\mathcal{R}_j$, is defined by a *relay set*, $RS(i,j)$, within $RZ(i,j)$, the *relay zone*. As shown in Fig.3, a relay set consists of relay points, and each relay point comprises two relay landmarks, one each from the two neighboring roadmaps, R_i and R_j. The two landmarks are at the same position but their status may differ in two roadmaps. When the two landmarks are both *Free*, they are implicitly "stitched" together (as in inset 1), and the relay point becomes a connecting point between $\mathcal{R}_i$ and $\mathcal{R}_j$; Otherwise, they are disconnected, as shown in inset 2, where the dark dot is occluded by obstacles. The connectivity of the relay set is determined by *MSG_RELAY* messages in the planning phase (detailed in the next section). To simplify negotiation between neighbors, and hence reduce the communication overhead, we choose relay sets to be deterministic and symmetric ordered sets, i.e., $RS(i,j) = RS(j,i)$. More specifically, we choose $RS(i,j)$ to be the set of points evenly distributed on the line segment connecting the two intersecting points of communication boundary circles, distance between two consecutive points is a predefined resolution, d_r. See Section V for further discussion of the choice for relay sets.

Algorithm 1: Local PRM Planning of Sensor Node i

```
1  begin
2      if $q_g \in \mathcal{H}_i$ then                          /* sensed the goal? */
3          Add $q_g$ into roadmap $\mathcal{R}_i$;
4          $v_g \leftarrow (q_g)$; $C_i(v_g) \leftarrow 0$;
5          $U \leftarrow \{v_g\}$;
6          Update_Field($U$);                           /* see sub-routine */
7      endif
8      if MSG_RELAY:$(j, \{C_j(v), v \in RS(i,j)\})$ received then
9          foreach $v \in RS(i,j)$, s.t. $C_j(v) < C_i(v)$ do
10             $C_i(v) \leftarrow C_j(v)$;
11             $U \leftarrow U \bigcup \{v\}$;
12         endfch
13         Update_Field($U$);
14     endif
15 end
16 Sub-Routine: Update_Field($U$)
17 begin
18     foreach $u \in U$ do                             /* Loop until $U = \emptyset$ */
19         foreach $v \in \mathcal{N}_i(u)$ and $C_i(v) < C_i(u) + \|v - u\|$ do
20             $C_i(v) \leftarrow C_i(u) + \|v - u\|$;
21             $\pi_i(v) \leftarrow u$;
22             $U \leftarrow U \bigcup \{v\}$;
23         endfch
24         $U \leftarrow U \backslash \{u\}$;
25     endfch
26     for $k \in N_i$ do
27         $V_u \leftarrow \{v | v \in RS(i,k)$, and $C_i(v)$ is improved$\}$;
28         Send $MSG\_RELAY$:$(k, \{C_k(v), v \in V_u\})$ to $k$;
29     endfor
30 end
```

B. Planning

In planning phase, a distributed navigation field is computed over the local roadmaps. For each landmark u in $\mathcal{R}_i$, we define two functions: $C_i(u)$ and $\pi_i(u)$, where $C_i(u)$ is the distance from u to the desired goal, and $\pi_i(u)$ is the next landmark in the shortest path from u to the desired goal. Note that (u, v) points in the negated gradient direction of $C_i(u)$,

$$\pi_i(u) = \arg\min_{v \in \mathcal{N}_i(u)} \big(C_i(v) < C_i(u) + \|v - u\|\big).$$

The procedure of updating the navigation field is given in sub-routine, *Update_Field*() (Line 16-25, Algorithm 1), which is essentially a distributed Dijkstra algorithm. It maintains U as a list of landmarks with the smallest cost so far

Algorithm 2: Robot Query for Paths

```
1  begin
2      $x \leftarrow$ Robot current position;
3      $path \leftarrow \emptyset$, $c \leftarrow \infty$;
4      Send MSG_QUERY message with $x$;
5      repeat
6          Receive $MSG\_PATH$:$(p, l)$, where $p$ is the found path
           segment and $l$ is the cost defined;
7          if $l < c$ then
8              $path \leftarrow p$, $c \leftarrow l$;
9          endif
10     until timeout ;
11     Robot execute path $p$;
12     Repeat from Line 3, until $p(1) = x_g$;
13 end
```

Algorithm 3: Sensor Respond to Query with Distributed PRM

```
1  begin
2      Receive $MSG\_QUERY$:$(x)$ from the robot;
3      $u_s \leftarrow \{x\}$;
4      $U \leftarrow \{v \in \mathcal{V}_i \big| \|u_s - v\| < r, (u_s, v)$ collision free$\}$;
5      $C_i(u_s) \leftarrow \min_{v \in U} \left(C_i(v) + \|u_s - v\|\right)$;
6      $\pi_i(u_s) \leftarrow \arg\min_{v \in U} \left(C_i(v) + \|u_s - v\|\right)$;
7      $l \leftarrow C_i(u_s)$, $\Pi_i \leftarrow \{u_s\}$;
8      while $\pi_i(u_s) \neq$ nil do
9          $\Pi_i \leftarrow \Pi_i \bigcup \{\pi_i(u_s)\}$;
10         $u_s \leftarrow \pi_i(u_s)$;
11     endw
12     Send $MSG\_PATH$:$(\Pi_i, l)$;
13 end
```

(often called open list in the literature), and examines all adjacent landmarks of every $u \in U$. If the condition

$$v \in \mathcal{N}_i(u) \text{ and } \left(C_i(v) < C_i(u) + \|v - u\|\right)$$

in Line 19 holds, it means a shorter path has been found from v to the goal, via u. Therefore, v is added into U. There are two ways to trigger the navigation field update:

- A sensor node that has the desired goal within its sensed region initiates the planning by adding the desired goal as a landmark in the roadmap, and sets its distance to goal as 0, then updates its navigation field (Line 2-7, Algorithm 1).
- When a sensor node i receives an *MSG_RELAY* message from one of its neighbor, say j, it compares its own relay set to those in the message, if some in the message give any smaller cost (better path) for any relay points in the set, these relay points are updated to the smaller cost, and these newly-improved relay points are added into U for update of the navigation field (Line 8-15).

After updating the navigation field, if the cost for any relay point in a relay set, $R(i, k), k \in N(i)$, has been improved, it means a shorter path is found for the point, and a *MSG_RELAY* message is sent to notify node k (Line 26-29). The message contains a list of relay points and their newly-improved costs.

C. Path Query

When the navigation field has been computed as in previous section, an *MSG_REQUEST* message is sent to the robot, indicating that the sensor network is ready to help navigate it. To move toward the desired goal, the robot constantly interacts with sensor nodes of the network, as shown in Algorithms 2 and 3:

- The robot sends an *MSG_QUERY* message to sensor nodes around, and waits (Line 3-5 in Algorithm 2).
- A sensor node receives the message, and looks up its navigation field, computes the segment of the best (shortest) path, and sends the path segment back to the robot in an *MSG_PATH* message (Algorithm 3).
- The robot may receive multiple *MSG_PATH* messages from different sensor nodes. It decodes the segment in each message, picks the best path, and moves along the chosen path segment (Line 6-12 in Algorithm 2). When it reaches the end of the segment, it approaches another sensor node, and it sends out another query message. The robot repeats such query-and-move procedure until it reaches the desired goal.

V. Discussion

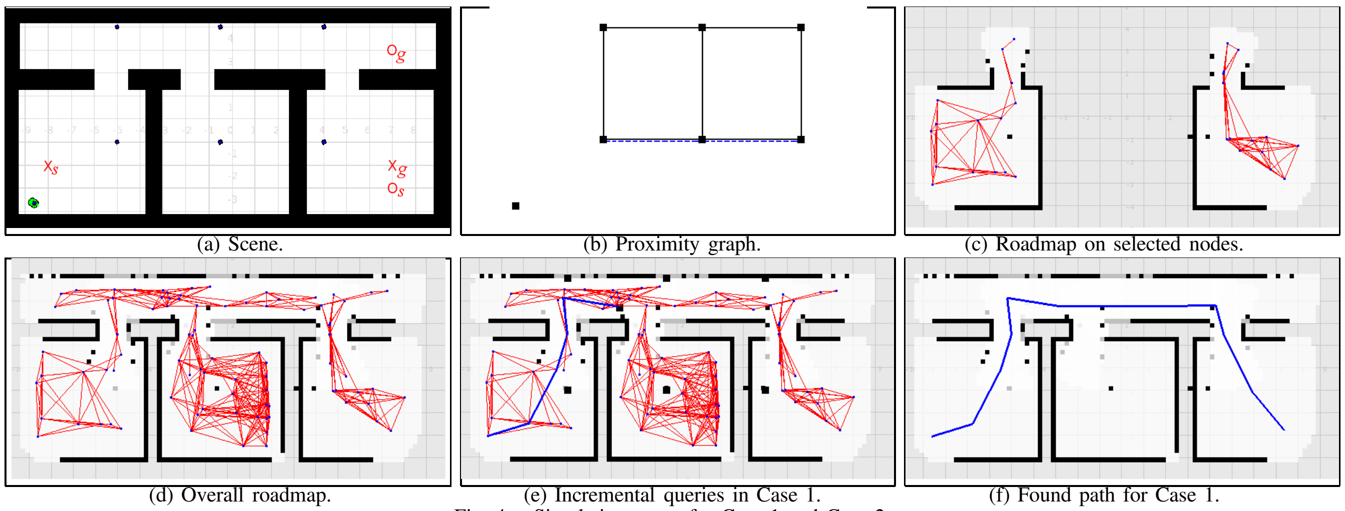
A. Sensing and communication

In our implementation, we assume the sensing region of a sensor node is simply a disk (e.g., a laser scanner at a fixed position), and the sensing range is equal to communication range, i.e., $d_s = d_c$. In practice, this is not necessarily the case, but other cases essentially reduce to this assumption, mainly because we would like to ensure that the robot is able to communicate with at least one sensor node at any time along the path. When $d_s > d_c$, the actual sensing region extends beyond the communication range. We ignore the sensed region beyond the communication range, and treat it as unknown (equivalent to setting d_s to d_c), just to make sure the robot does not go beyond d_c. When $d_s < d_c$, we can simply increase the map size up to communication range, and mark the margin between the communication range and the sensing range as *Unknown*.

The proposed *D-PRM* is intrinsically robust to communication failure. If a node fails during the planning phase, it is automatically excluded from consideration, since there is no message in and out of the failed node; if a node fails during execution phase, it will not respond to robot enquiry, and the robot will choose amongst the paths sent by other nodes, and continue with the best alternative path available.

B. Relay sets choices

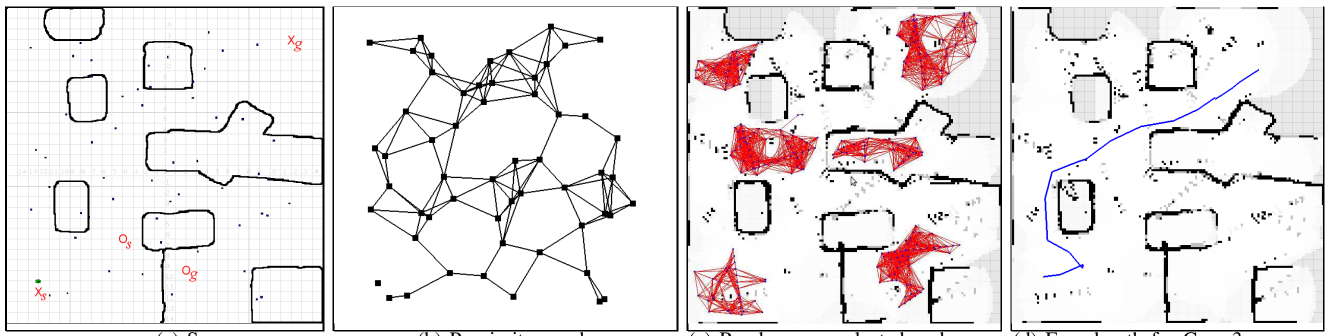
There are two main ways of choosing relay sets: in a deterministic manner, or in a probabilistic manner. For instance, one choice would be to lay a grid of uniform resolution d_r in $RZ(i, j)$ and the relay points correspond to the center of grid cells. Another choice would be random samples in $RZ(i, j)$. The latter is more in line with the probabilistic philosophy of sampling based framework, however, they carry higher communication overhead, since messages will contain position information of each sample, which is 8 bytes for (x,y) compared to 2 bytes for an index for the deterministic case. We chose to use the deterministic manner, and further restricted the set of relay points to be on a line segment, as shown in 3, to reduce the communication overhead. However, when the environment gets more complicated, particularly with narrow passages, a full-blown fine-resolution grid might be needed in order to find the solution, or for the probabilistic case, more sophisticated sampling techniques, such as bridge test sampling in [20], can be used to reduce the number of samples in relay zones while maintaining good connectivity of the roadmap.



(a) Scene. (b) Proximity graph. (c) Roadmap on selected nodes.

(d) Overall roadmap. (e) Incremental queries in Case 1. (f) Found path for Case 1.

Fig. 4. Simulation scene for Case 1 and Case 2.



(a) Scene. (b) Proximity graph. (c) Roadmap on selected nodes. (d) Found path for Case 3.

Fig. 5. Simulation scene for Case 3 and Case 4.

C. Communication Complexity

We focus on communication complexity in the distributed planning phase, since this is the most time-consuming phase in the proposed framework, and most of time is spent in message exchange, particularly for *MSG_RELAY* messages. *D-PRM* planning (Algorithm 1) is based on the distributed Dijkstra's algorithm, whose communication complexity is known to be $O(|V|^2)$ [21].

VI. Simulation Results

We now present simulation results to show the effectiveness of the proposed distributed sampling based planning method. Our simulation is based on Player/Stage [22]. We adapted the *wsn* (wireless sensor network) model that comes with Player to transmit user data of different sizes. We simulate one robot and multiple sensor nodes that form a sensor network in different scenarios. Sensor nodes are equipped with a laser scanner which can detect obstacles (in surveillance application, sensors can be overhead cameras). A brief video can be found as an attachment in the proceedings.

In the first scenario, we have a relatively small office-like environment with three rooms, as shown in Fig. 4(a). Six sensor nodes with the associated proximity graph forming a grid (Fig. 4(b)) help to navigate robot between rooms. Since every room has only one door facing the corridor, the only way to go from one room to another is through the corridor. The coordinates of the lower-left and upper-right corners are $(-9,-4)$ and $(9,4)$, respectively. We have 2 cases in this scenario, each corresponding to a different start/goal. In Case 1, the robot moves from $X_s : (-7,-2)$ to $X_g : (7,-2)$; and in Case 2, the robot moves from $O_s : (7,-3)$ to $O_g : (7,3)$, as shown in Fig. 4(a).

In Case 1, the robot tried to go from the leftmost room to the rightmost one. Most existing distributed planning works, in this case, will give the shortest path in the sensor network (i.e., the dotted line in Fig. 4(b)), and replan the path when the robot moves closer and detects the blocking walls, because they do not take physical obstacles into account during planning. With our proposed method, each node senses its local environment within the sensing region, then creates a local roadmap. Fig 4(c) shows the sensing regions for sensors 1 and 5, and roadmaps therein. We emphasize that each node maintains only its own roadmap. These local roadmaps are "stitched" together and form an implicit roadmap, as in Fig 4(d). In order to compute a feasible path for the robot, a navigation field is propagated over the implicit roadmap, with the goal as the unique global minimum. Then, the robot queries the sensor network for the path and moves along the planned path (Fig 4(e) for Case 1), and the final path for Case 1 is shown in Fig 4(f).

In the second scenario, we simulated a larger outdoor environment, where polygonal shapes simulate high grounds, or dangers (e.g., minefield), and the sensor network consists of 55 sensor nodes randomly dropped (Fig. 5(a)). The resulting proximity graph is shown in Fig.5(b). The coordinates of the lower-left and upper-right corners are $(-15,-15)$ and $(15,15)$, respectively. As indicated in Fig. 5(a), the robot moves from $X_s : (-12, -12)$ to $X_g : (12, 12)$ in Case 3, and from $O_s : (-4, -6)$ to $O_g : (3, -10)$ in Case 4. Fig.5(c) shows only selected local roadmaps for clarity. The final path executed by the robot for Case 3 is shown in Fig.5(d).

Table I gives a summary of some important metrics and parameters in the simulations, including size of the communication network ($|V|$ and $|E|$), size of roadmap ($\mathbf{N}_{lm}$, number of landmarks), total number of messages ($\mathbf{N}_{msg}$), total number of collision checks ($\mathbf{N}_{cc}$), planning time (T_{plan}, duration between MSG_TASK sent, and the last $MSG\_REQUEST$ received), and the traveling distance of the robot along the path (L_{path}). The results are averaged over 12 runs, with the roadmap being regenerated in every run.

Note that in Distributed PRM, planning involves all sensor nodes in the network, so the size the roadmap and the total number of messages are similar in the same environment (e.g., Cases 1 and 2, or Cases 3 and 4). However, the planning time depends on the distance between start and goal. As message passing being a major part of the planning, a shorter path indicates less intermediate sensor nodes need to be involved in finding the path, and less message passing before the MSG_RELAY reaches the robot, hence a shorter planning time. This means even when the robot starts moving toward its goal, some sensor nodes might be still planning the path. In our implementation, $d_r = d_c/10$, therefore there are at most 21 relay points in a relay set. The average number of messages is much smaller than the worst case bound given in the Section V.

TABLE I

SIMULATION RESULTS

	Case 1	Case 2	Case 3	Case 4
$\|V\|$	6	6	55	55
$\|E\|$	7	7	129	129
$\mathbf{N}_{lm}$	149	139	996	1010
$\mathbf{N}_{msg}$	22	22	333	362
$\mathbf{N}_{cc}$	13706	12283	207579	226095
T_{plan}	5.0	1.5	16.6	2.2
L_{path}	24.5	9.6	43.0	16.7

VII. CONCLUSIONS

In this paper, we proposed a distributed sampling based planning framework for robot navigation in sensor networks. It is particularly applicable to networks where each sensor node is equipped with sophisticated sensors capable of giving a map for its sensing region. To keep communication cost low, there is no global representation of C-space in our framework. Instead, each sensor node creates a local roadmap within its locally-sensed environment, and these local roadmaps are "stitched" by a set of relay points, which lie in the overlapping sensing regions of sensor nodes. When the desired goal of the robot is specified, a navigation field is propagated over the implicit roadmap, and gives directions to the desired goal. The robot moves toward its goal by constantly querying the sensor network for the directions.

Even though we focus on static sensor networks in this paper, our navigation framework can be extended to mobile sensor networks, where mobile nodes can sense the environment while moving, and carry larger maps accumulated along their trajectories. Another direction for future work is to extend the framework to deal with a general robot with non-trivial shape, the corresponding C-space being three-dimensional, since the orientation of the robot matters. Current one-dimensional choice of relay set is not sufficient, and new mechanisms of choosing relay sets will need to be explored. Besides, a more fundamental problem in this case is collision checking. While collision checking for a point robot can be simply done by one sensor node within its local map, robots with shape may span multiple maps of adjacent sensor nodes, and will requires cooperative collision checking.

REFERENCES

[1] M. A. Batalin, G. S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *Proceedings of ICRA*, volume 1, pages 636–641, 2004.

[2] Q. Li and D. Rus. Navigation protocols in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):3–35, August 2005.

[3] A. Verma, H. Sawant, and J. Tan. Selection and navigation of mobile sensor nodes using a sensor network. In *Proceedings of PerCom*, pages 41–50, 2005.

[4] G. Alankus, N. Atay, Chenyang Lu, and O. B. Bayazit. Spatiotemporal query strategies for navigation in dynamic sensor network environments. In *Proceedings of IROS*, pages 3718–3725, 2005.

[5] K. J. O'Hara, V. L. Bigio, E. R. Dodson, and A. J. Irani. Physical path planning using the gnats. In *Proceedings of ICRA*, pages 709–714, 2005.

[6] S. Bhattacharya, N. Atay, G. Alankus, C. Lu, O. Bayazit, and G.-C. Roman. *Distributed Computing in Sensor Systems*, chapter Roadmap Query for Sensor Network Assisted Navigation in Dynamic Environments, pages 17–36. 2006.

[7] C. Buragohain, D. Agrawal, and S. Suri. Distributed navigation algorithms for sensor networks. In *Proceedings of InfoCom*, pages 1–10, 2006.

[8] G. Alankus, N. Atay, Chenyang Lu, and O. B. Bayazit. Adaptive embedded roadmaps for sensor networks. In *Proceedings of ICRA*, pages 3645–3652, 2007.

[9] Z. Yao and K. Gupta. Backbone-based roadmaps for robot navigation in sensor networks. In *Proceedings of ICRA*, pages 1023–1029, 2008.

[10] P. Corke, R. Peterson, and D. Rus. Localization and navigation assisted by networked cooperating sensors and robots. *International Journal of Robotics Research*, 24(9):771–786, 2005.

[11] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, 3(4):24–33, 2004.

[12] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566, 1996.

[13] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University, 1998.

[14] Zhenwang Yao. *Distributed Motion Planning in Robotic Sensor Networks*. PhD thesis, School of Engineering Science, Simon Fraser University, 2010.

[15] D. Henrich. Fast motion planning by parallel processing – a review. *Journal of Intelligent and Robotic Systems*, 20(1):45–69, 1997.

[16] N. M. Amato and L. K. Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proceedings of ICRA*, volume 1, pages 688–694, 1999.

[17] E. Plaku and L. E. Kavraki. Distributed sampling-based roadmap of trees for large-scale motion planning. In *Proceedings of ICRA*, pages 3868–3873, 2005.

[18] B. Taati, M. Greenspan, and K. Gupta. A dynamic load-balancing parallel search for enumerative robot path planning. *Journal of Intelligent and Robotic Systems*, 47(1):55–85, 09/23 2006.

[19] A. Brooks, S. Williams, and A. Makarenko. Automatic online localization of nodes in an active sensor network. In *Proceedings of ICRA*, volume 5, pages 4821–4826 Vol.5, 2004.

[20] David Hsu. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of ICRA*, pages 4420–4426, 2003.

[21] D. U. Friedman. Communication complexity of distributed shortest path algorithms. Technical report, MIT, Cambridge MA., 1986.

[22] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of ICAR*, pages 317–323, June 2003.