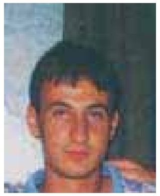
Mobile Robot Navigation using a Sensor Network

by Maxim A. Batalin, Gaurav S. Sukhatme, Myron Hattig



Min-Uk Kim
Theory of Computation Laboratory
Konkuk University
http://tc.konkuk.ac.kr/
minuk@konkuk.ac.kr

2007.12.12

Agenda

- Introduction
- Previous Work
- Probabilistic navigation
- Experiments
- Conclusion
- References
- Appendices

Introduction

Introduction

- Navigation is a fundamental problem in mobile robotics.

- Navigation problems
  - local – deals with navigation on the scale of **a few meters**
  - global – deals with navigation on a larger scale in which the robot **cannot observe the goal state** from its initial position

- There are many solutions.

- But their approaches assume that a map is **given a priori**.

Introduction

- Our approach's properties
  - The sensor network is **pre-deployed** into the environment
  - The network are synchronized in time.
  - The robot does not have a pre-decided environment map or access to GPS, IMU or a compass.
  - The environment is **not required to be static**.
  - The robot does not perform localization or mapping.
  - The robot does not have to be sophisticated.
    - The primary computation is **performed distributively** in the sensor network, the only sensor required is for obstacle avoidance.

Previous Work

- Coverage, Exploration and Deployment by a Mobile Robot and Communication Network(2004)
  - Maxim A. Batalin , Gaurav S. Sukhatme
  - Deployment of network
  - Probabilistic navigation
  - Navigation field

- This paper is just **experimental report** of previous work.

Probabilistic navigation

Probabilistic navigation

- In order for the robot to be able to navigate through the environment from A to B, the robot should choose an action that **maximizes its chances** of getting to its goal.

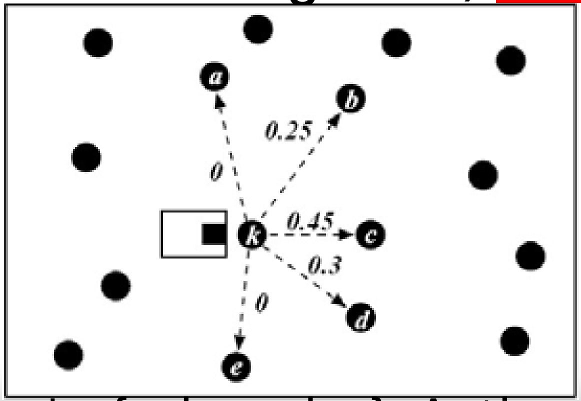
- Our approach relies on a **pre-deployed sensor network** with **determined transition probabilities**.

Probabilistic navigation

- Transition probabilities are probability of arriving **at s'** given that the robot started **at s** and commanded **action a**
  - Actions = {East, West, South, North}
  - s is start point(node)
  - s' is next point(node)

$$P(s'|s,a)$$

- If we want that every nodes have the transition probabilities about their neighbor, **Robot should explore whole network**.



< s = {k}, s' = {a, b, c, d, e}, Action = East >

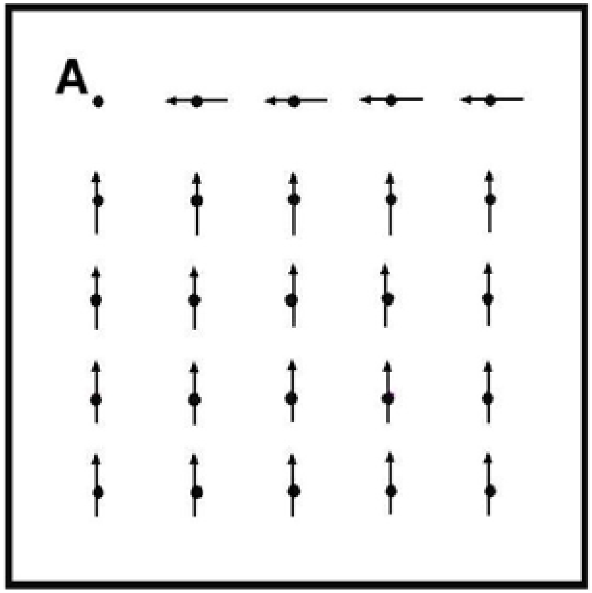
Probabilistic navigation

Planning

Probabilistic navigation

- When the navigation goal is specified, the node that is closest to the goal triggers **the navigation field** computation.

- During this computation, every node probabilistically determines **the optimal direction** in which the robot should move, when in its vicinity.



KU KONKUK UNIVERSITY

Probabilistic navigation

- When the navigation goal is specified, the node that is closest to the goal send **Start Computation packet**.
- Nodes that receive the Stat Computation packet initialize **utility**.
  - The utility of the goal node is set to **1** and of the other nodes to **0**.
- Every node in the network **updates** its utility and **computes** the optimal navigation action of its own.

$$U_{t+1}(s) = C(s,a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times U_t(s')$$

$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times U(s')$$

Probabilistic navigation

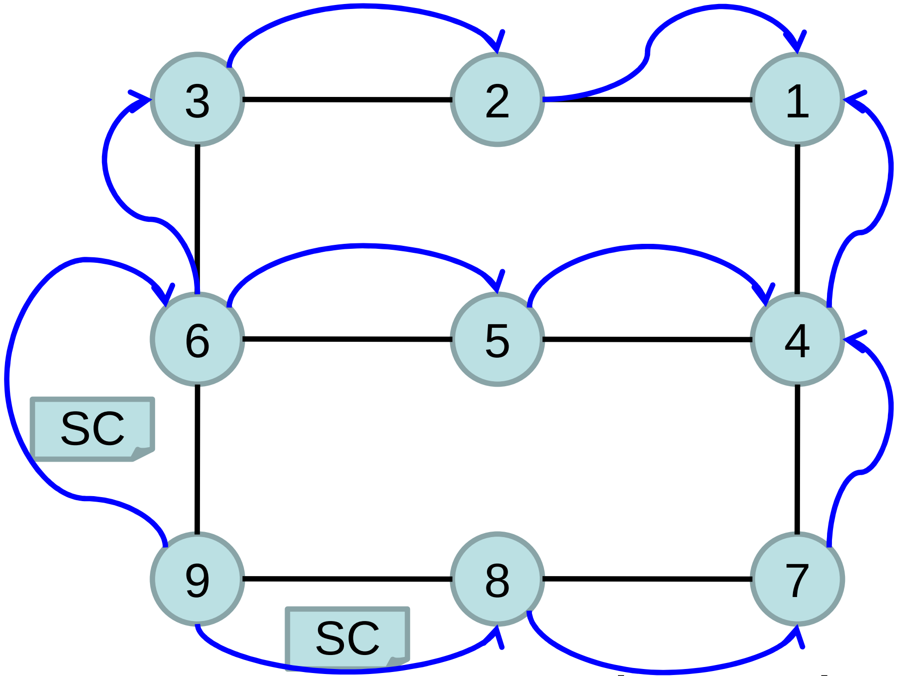
$$U_{t+1}(s) = C(s,a) + \max_{a \in A(s)} \sum_{s' \in S-s} P(s'|s,a) \times U_t(s')$$

- where C(s , a) is **the cost** associated with moving to the next node. Usually **< -1/k** where k is the number of nodes.

- This is an **iteration model**.
  - The utility update equations have to be executed until **the desired accuracy is achieved**. For practical reasons, the accuracy in our algorithm is set to **10^{-3}**, which requires a reasonable number of executions of the utility update equation per node(**Approx. 20**).

Probabilistic navigation

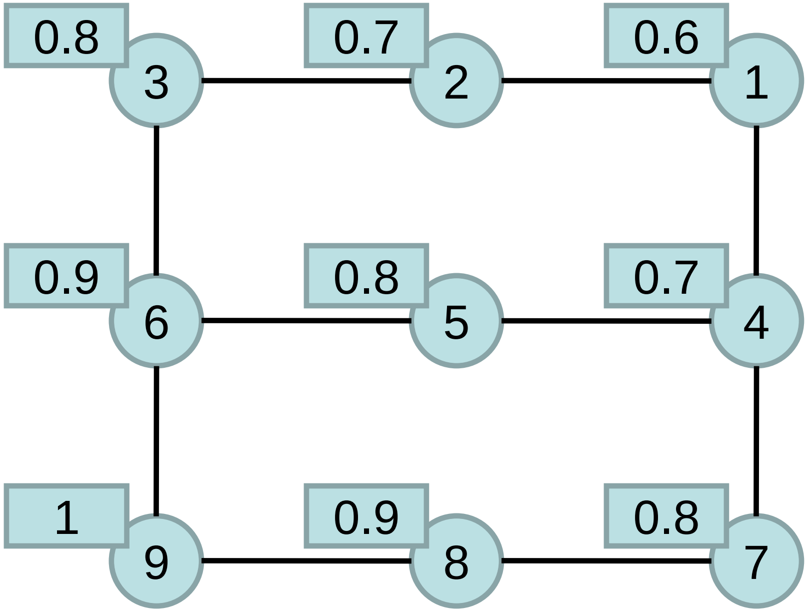
- Example: When the navigation goal is specified to 9



- 1. Start Computation packet is sent by 9 to its neighbors

Probabilistic navigation

- Example: When the navigation goal is specified to 9



$$C(s,a) = -\frac{1}{10} < -\frac{1}{k} = -\frac{1}{9}$$

where k is the number of nodes

- 2. Node that receive SC packet starts **updating its utility**.
  – If neighbors of all nodes are known exactly, then **P(s'|s,a) = 1**
  – In this paper, every node is **preprogrammed** with information about its neighbors.

Probabilistic navigation

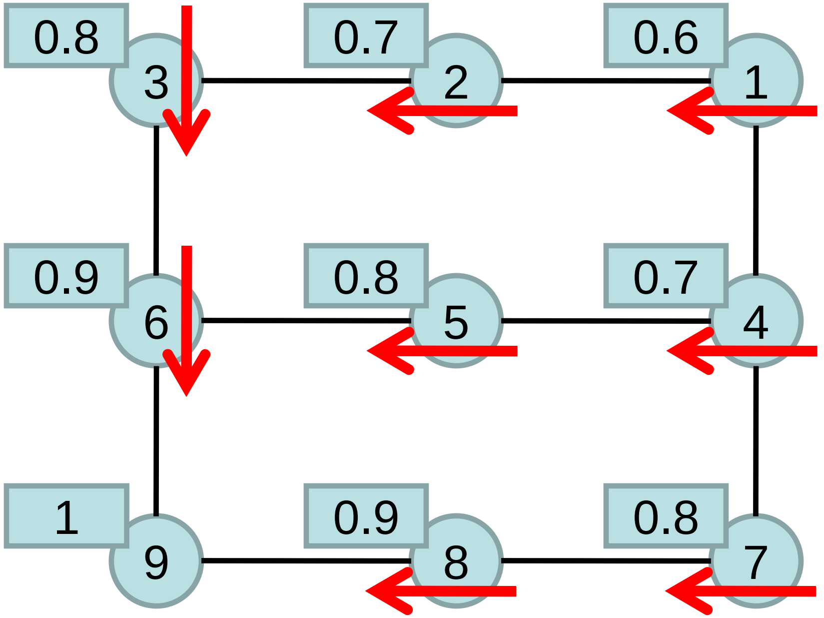
$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S - s} P(s'|s,a) \times U(s')$$

- After the utilities are computed, every node computes an **optimal policy** for itself according to this equation.

- The computed optimal action is **stored** at each node and is **emitted** as part of a **suggestion packet** that the robot would receive if in the vicinity of the node.

Probabilistic navigation

- Example: Compute optimal action at each nodes



- 3. At node 8, I will **<u>choose West</u>** direction because 9's utility is bigger than 7's utility.

Probabilistic navigation

Navigation

Probabilistic navigation

- Note that deployed sensor network **discretizes** the environment.

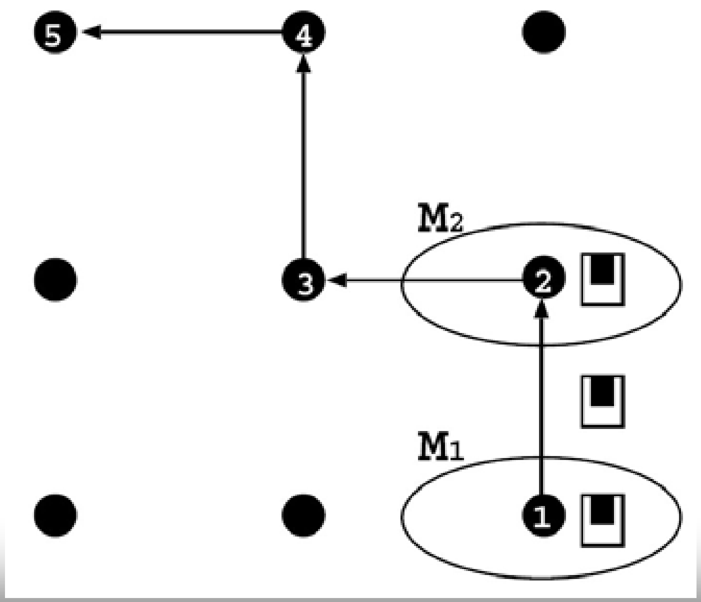
- Now we should navigate the robot.

- There are 3 phases.
  - 1. Robot **accepts** command which is given by current node.
  - 2. Robot **moves** 'forward' using the VFH algorithm for local navigation and obstacle avoidance.
  - 3. During this phase **the current node switches** and the navigation algorithm start from phase 1 again.

- In this manner, the robot can navigate to goal wherever it locates. This is the **node-wise approach**.

Probabilistic navigation

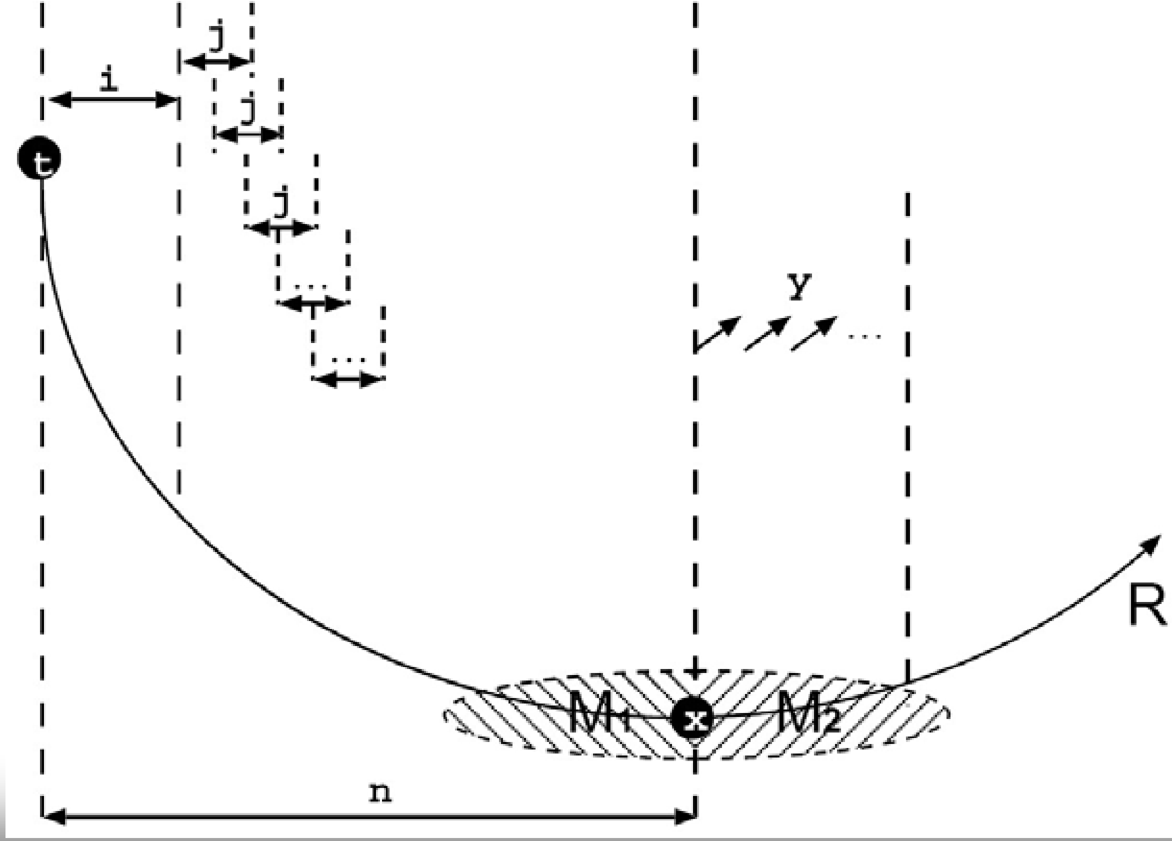
- Suppose initially current node is set to node 1.
- Node 1 suggests the robot to **go forward** direction.
- In M_2 area, current node will be changed to node 2.
- And node 2 will suggest the robot to **go left** direction.
- Then, How can we switch **current node** to **node 2**?



KU KONKUK UNIVERSITY

Probabilistic navigation

- This is the solution about switching current node based on processing **signal strength values**.
  - Adaptive Delta Percent



Probabilistic navigation

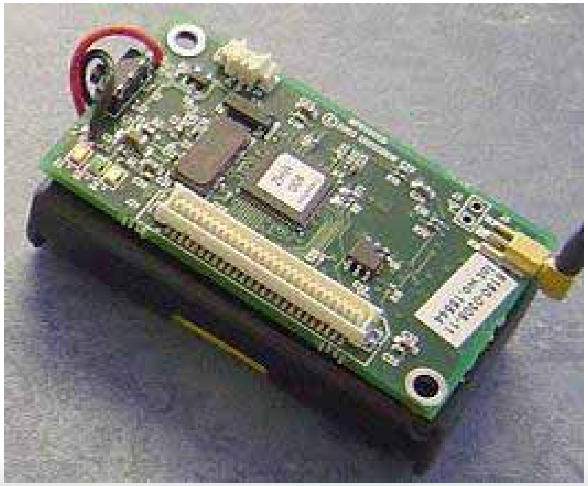
- This algorithm has 4 phases.
  - 1. Compute an initial maximum average A_{im} – an average of the first i samples.
  - 2. Compute a running average A_r which is an average of j consecutive samples.
  - 3. If $R = A_r / A_{im} < M$, where M is **the threshold value**, then return from the algorithm. Put R into list L_R.
  - 4. If y consecutive elements of **L_R are in nondecreasing order**, then **return y** and quit the algorithm, else repeat 2~4.

- In case, several nodes returned from the algorithm, pick the node with **the smallest ratio** and switch to it.
- Experimentally we determined threshold **M = 0.65**.

Experiments

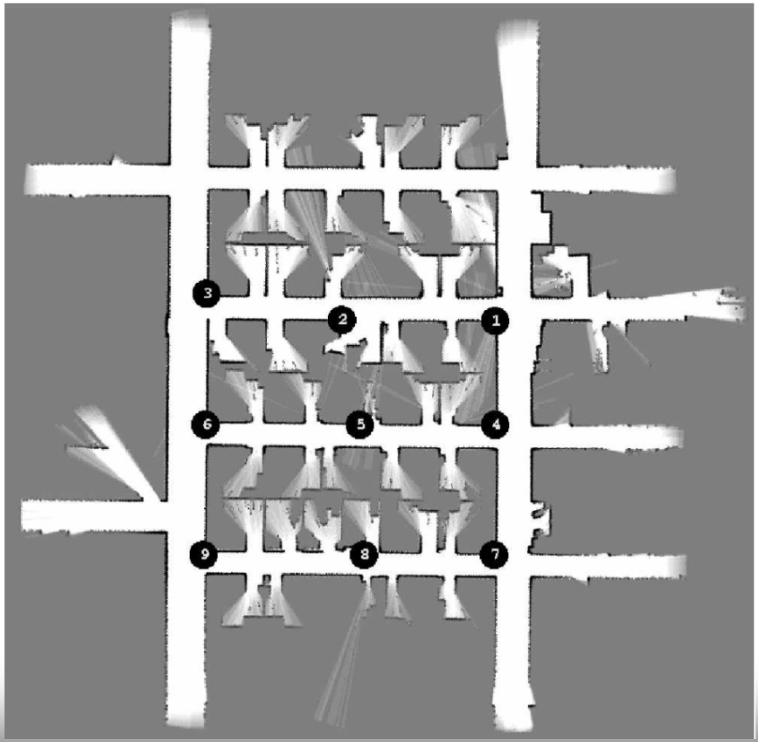
Experiments

- We conducted experiments at **Intel Research facilities**.
- We used
  - a **Pioneer 2DX** mobile robot.
    - with 180° laser range finder used for obstacle avoidance
  - a **base station**(Mica 2 mote)
    - for communication with the sensor network



Experiments

- The sensor network of 9 nodes was **pre-deployed**.
- Every node is **preprogrammed** with information about its neighbors.



Experiments

- The environment itself resembles a **regular cubicleoffice-like environment** with narrow corridors(about 1m), **changing topology**, **crowded with people and obstacles**.

- Figure 5 shows the **mobile robot** and one of the deployed **nodes** in the experimental environment.

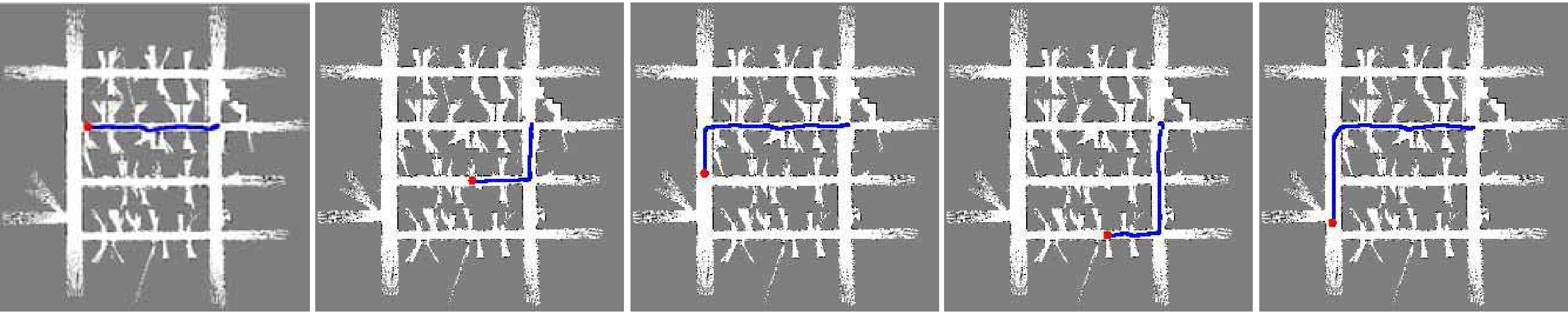
< Fig. 5 >

Experiments

- The experimental scenario that we consider for navigation is **alarm handling**.
- An alarm occurs when a certain node detects an event.
- The task of the robot is to navigate **from the 'home base'(around node 1) towards the triggered alarm**.
- Requirements for the successful experiment
  - navigation field should yield shortest paths
  - robot should stop within 3m of the goal node

Experiments

- We conducted 10 experiments for **5 different goal nodes**.
- This is **representative trajectories** that the robot took on its route from the start.



< Goal 3 > < Goal 5 > < Goal 6 > < Goal 8 > < Goal 9 >

Experiments

- Table I shows the **final distances** from the robot to the goal nodes after the robot has signaled that it had completed navigation.

- The robot was able to navigate to the correct goal node in all cases.

TABLE I

EXPERIMENTAL DATA (DISTANCE TO GOAL AT FINISH, IN METERS). FIVE GOALS, TEN EXPERIMENTS PER GOAL.

Trial	Goal 3	Goal 5	Goal 6	Goal 8	Goal 9
1	0.7	1.4	0.78	2.9	0.96
2	0.82	1.26	0.86	1.6	0.96
3	0.94	1.45	0.72	1.62	1.35
4	0.91	1.41	0.91	2.4	1.26
5	0.85	1.4	0.87	1.4	1.21
6	0.97	1.39	1.3	2.1	1.24
7	0.85	1.01	0.85	1.7	0.95
8	0.98	1.55	0.88	2.8	1.51
9	0.89	1.5	0.55	1.79	1.4
10	0.66	1.04	1.02	2.1	0.92
Average	0.86	1.34	0.87	2.04	1.17

Conclusion

- We have presented an algorithm that allows the robot to navigate **using a deployed sensor network**.
  - without a map, GPS, IMU, compass
- The navigation occurs through **node-wise motion** from node to node on the path.

References

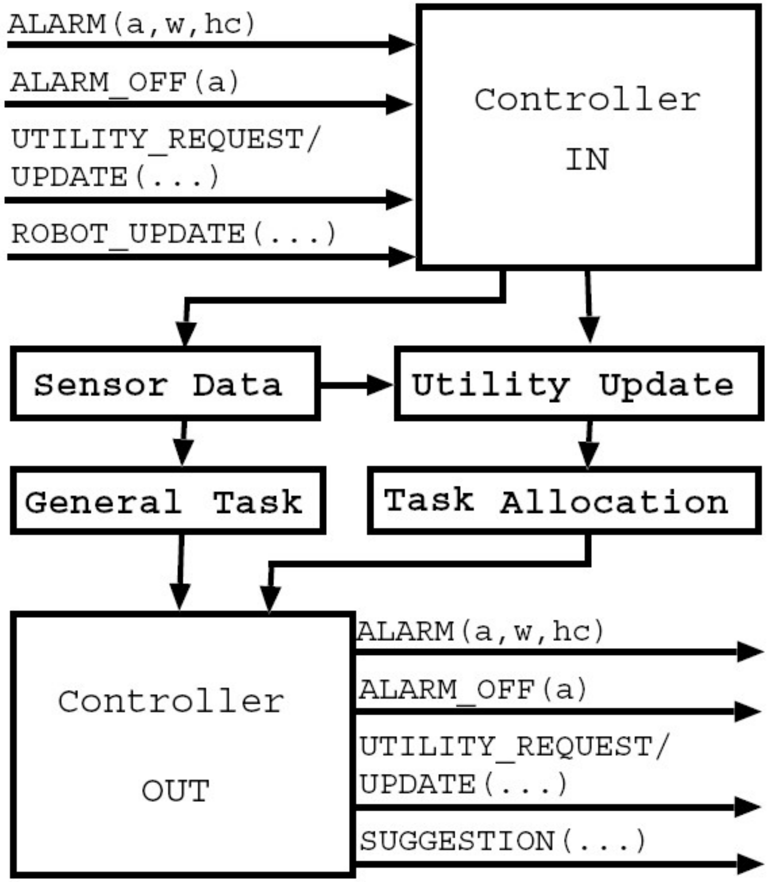
- M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," in *The 2nd Intl. Workshop on Information Processing in Sensor Networks(IPSN '03),* Palo Alto, 2003, pp.376-391
- M. A. Batalin and G. S. Sukhatme, "Sensor Network-based Multi-Robot Task Allocation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems(IROS '03),* Las Vegas, 2003, pp.1939-1944
- I. Ulrich and J. Borenstein, "VFH*:Local Obstacle Avoidance with Look-Ahead Verification," in *IEEE int. Conf. on Robotics and Automation*, 2000, pp.2505-2511

Appendices

Appendix A: Node Architecture

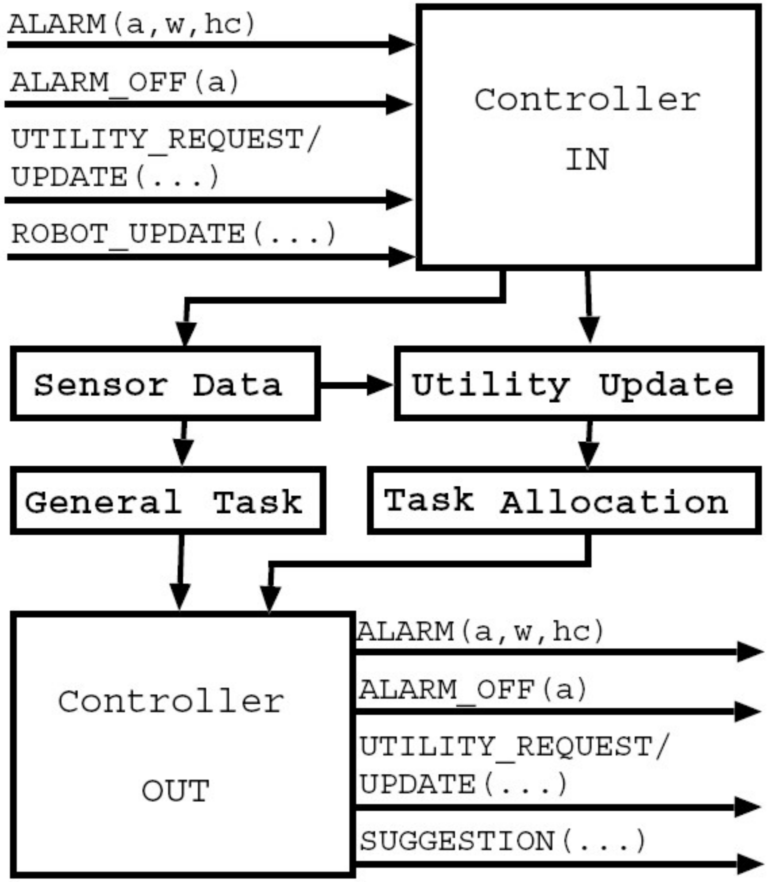
- It is not exactly the same architecture used in this paper.
- But it helpful to understand.

- This architecture is used when an alarm event may be occurred different nodes **concurrently**.
- It assume that there are several robots.
- Robot has a compass.



Appendix A: Node Architecture

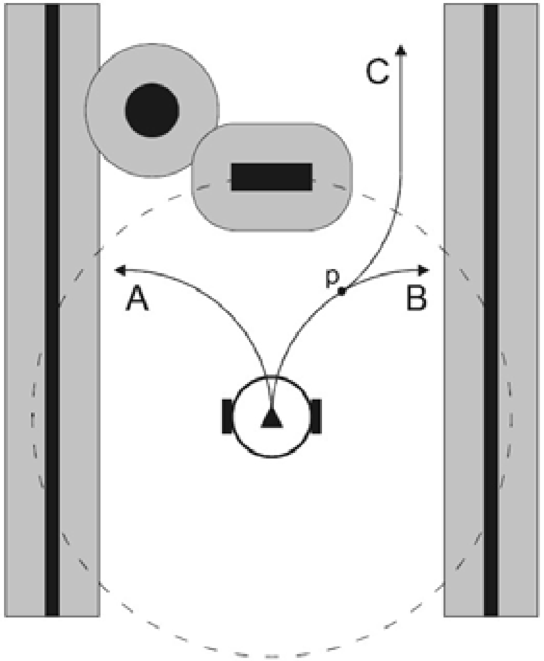
- ALARM(a,w,hc)
  - Id of the node that detected the alarm
  - Weight
  - Hop count
- If a node receives an ALARM msg, the alarm is placed on the **list**
- Every node maintains a **current alarm** variable, which is the element of L with largest utility.



Appendix B: VFH*

- This is an enhancement version of earlier developed **Vector Field Histogram**(VFH) method for mobile robot obstacle avoidance.

- Earlier version VFH+ sometimes fails.
- Figure 1 shows a situation where a mobile robot travels down a corridor and encounters **two obstacles** in its path.
- Obstacles are shown in **black**, while the configuration space is **gray**.



< Fig. 1 >

Appendix B: VFH*

- The large circle drawn in a dashed line shows the approximate distance at which an obstacle triggers an **avoidance maneuver**.

- At the position shown in the example, VFH+ detects **2 openings**.

- Unfortunately, both trajectories A and B appear **equally** appropriate to VFH+.

- In problematic situations like this, VFH+ would thus select the appropriate direction on average only **50%** of the time.

Appendix B: VFH*

- VFH* algorithm **overcomes** problematic situations like this one most of the time by combining VFH+ **with the A* search algorithm**.
- Figure 5 shows the trajectories of VFH* with 4 different goal depth values.

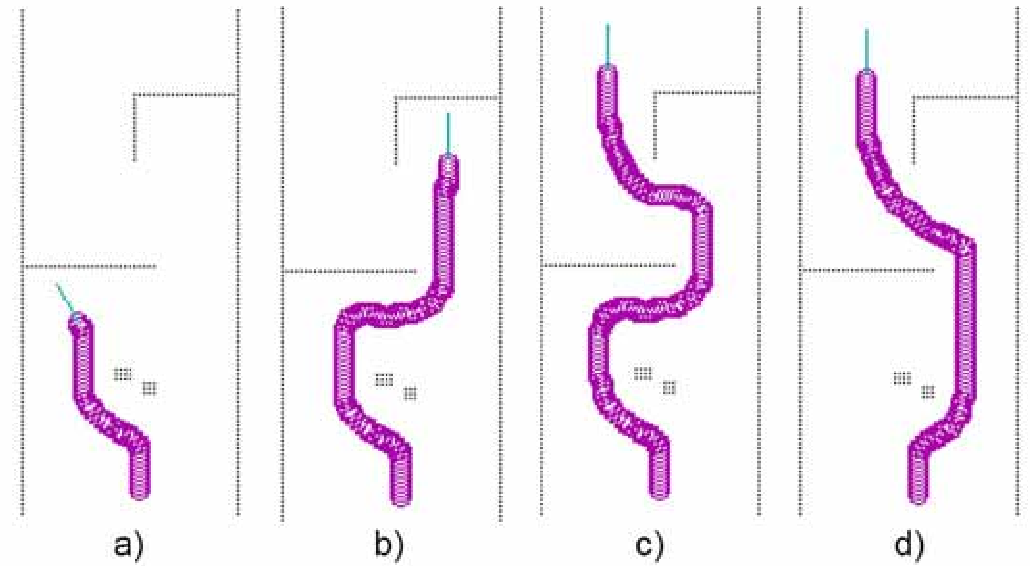


Figure 5: VFH* trajectory with: a) $n_g = 1$, b) $n_g = 2$, c) $n_g = 5$, and d) $n_g = 10$.

Appendix B: VFH*

- Figure 5 shows that the higher n_g is selected, the better VFH* performs.
- However, this improvement is at the expense of **computational time**.
- Table 1 shows an execution time comparison based on the GuideCane's embedded computer, a PC 486 running at 66 MHz.

n_g	$T_{average}$	$T_{maximum}$
1	3 ms	6 ms
2	5 ms	11 ms
3	8 ms	22 ms
4	10 ms	39 ms
5	12 ms	82 ms
10	30 ms	242 ms

Table 1: VFH* execution time.

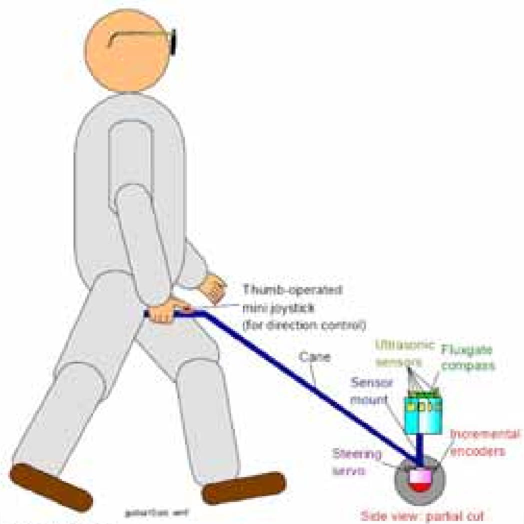


Figure 2:
A blind person walks with the GuideCane.

Appendix B: VFH*

- VFH* which is a local obstacle avoidance algorithm that uses **look-ahead verification** can consider more than the robot's immediate surroundings.

- While VFH* has the same obstacle avoidance performance as VFH+ for regular obstacles, **VFH* is also capable of dealing with problematic situations** that would require the robot to substantially slow down or even stop.