Robot Navigation in Corridor Environments using a Sketch Floor Map

Vachirasuk Setalaphruk Atsushi Ueno Izuru Kume Yasuyuki Kono
Masatsugu Kidode
{setala-v,ueno,kume,kono,kidode}@is.aist-nara.ac.jp
School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, 630-0192, Japan

Abstract

This paper presents a new robot navigation system that can operate on a sketch floor map provided by a user. This sketch map is similar to floor plans as shown at the entrance of buildings, which does not contain accurate metric information and details such as obstacles. The system enables a user to give navigational instructions to a robot by interactively providing a floor map and pointing out goal positions on the map. Since metric information is unavailable, navigation is done using an augmented topological map which described the structure of the corridors extracted from a given floor map. Multiple hypotheses of the robot's location are maintained and updated during navigation in order to cope with sensor aliasing and landmark-matching failures due to factors such as unknown obstacles inside the corridors.

1 Introduction

In recent years, personal robots have received a growing interest from the public, and many robot platforms like PaPeRo[1] and Robovie[2] are becoming available. These robots are expected to interact with users and carry out specified tasks in unmodified environments. Therefore, it is important that they provide intuitive interface to the users, and can process information the users give them.

Navigation from one place to another is one of the most basic tasks users require of their robots. To accomplish this, a robot will have to know how to get to the goal from its current position. If the robot have a prior knowledge about its environment, in other words, if it has a map of the environment, it can try to figure out how to get to the goal given its current position and goal position. Maps can be provided by the users; however, human is not very good at producing accurate and detailed maps. Therefore, the robots will have to be able to deal with inaccuracies and lack of details that comes with the maps provided by users.

In this paper, we propose a robot navigation system that operates in corridor environments using an inaccurate sketch floor map provided by human. This map is similar to two-dimension floor plans usually found in buildings. It does not require accurate metric and geometric information, and details about obstacles inside the corridor can be omitted. This map is very easy to created, for example, by using simple drawing programs. With this system, the user can initiate a navigation task intuitively by drawing a map and interactively pointing out the robot's current position and goal position on the map.

One of the characteristics of the user-provided map is its lack of accurate metric and geometric information, thus, the navigation has to be done without them. In our system, only the structure of the corridors is used for navigation. An augmented topological map describing the structure of the corridors is extracted using a Voronoi diagram of the sketch map. The robot then uses the topological map during subsequent navigation.

Another characteristics of the user-provided map is the omission of certain obstacles, such as pillars or furniture, inside the corridors. This can cause failures when the robot try to match features extracted from sensor readings with features from the topological map. To cope with this problem, multiple hypotheses of a robot position are maintained during navigation, and their associated belief values are updated whenever a new feature is found. This also provides a good solution to sensor aliasing and wrong observation of the environment that can occur during navigation.

The paper is organized as followed. Related works regarding robot navigation is discussed in Section 2. After an overview of the sketch map provided by the user and its characteristics Section 3, an augmented topolocigal map used to represent the environment and a method for generating it from the sketch map is described in Section 4. In Section 5, a navigation system architecture with localization method based on Multiple Hypothesis Tracking is described. Results of navigation experiments in simulated environments are presented in Section 6. Finally, a conclusion is given in Section 7.

2 Related Works

An example of systems that rely on user-provided map for navigation is one proposed by Terabayashi, et al.[8]. The system takes a scanned image of a hand-written map, which is a labeled graph representing the topology of the corridor, as input. During navigation, localization is carried out by matching nodes in the graph to Voronoi vertices extracted from sensor readings, which represents the topological structure of the environment, using straightforward algorithm. Their localization process is not designed to handle ambiguous situations that are caused by the difference between the map and the real environment. Furthermore, using a graph as input severely limits the expressiveness of the map, making it difficult to understand and limiting the possibility of future extension.

Many modern systems use a landmark-based topological map[11] for navigation. Topological maps are more efficient for path planning and do not depend on

geometrical accuracy. These systems usually rely on accurate topological maps that are either learned directly from sensor readings during exploration[13] or by decomposition of previously learned grid-based map[14]. Many researches augment topological map with approximate metric information or Markov models, which is primarily used to resolve topological ambiguity. In our case, this information is not available in the map provided by the user, so we have to rely on another mean to resolve the ambiguity.

We considered using a technique based on Multiple Hypothesis Tracking(MHT) to solve the ambiguity that can arise during navigation. MHT is widely used to solve global localization problems[9][10], in which a robot has no knowledge of its initial position and has to determine its current position based on past observations of the environment. In our case, instead of starting with an empty hypothesis, we start with a highly reliable hypothesis of robot starting point, which is provided by user. On the other hand, the map involved in the process is now ambiguous.

3 Sketch Floor Map

In this section, we describe a sketch floor map that is used as input for the navigation system, and a method to extract topological map from it for use in navigation.

3.1 Information on Sketch Floor Map

Our aim is to create a robot navigation system for indoor corridor environments, therefore we define a sketch floor map as an abstract map of the environment that shows structural information of corridors in a building. It is given to a robot as a bitmap image, with white pixels denote passable terrain, and black pixels denote wall or other obstacles that robot cannot pass through. The map should models the existence and the connectivity of corridors correctly, but accurate measurement and geometric information, such as length of corridors or exact shape of walls, are not necessary. Additionally, the sketch map may contain information about rooms that are attached to the corridors, in order to help user visualize the building, but it is not strictly necessary. This kind of map can be created easily with a simple drawing tool. Figure 1 shows a sample map of first floor of School of Information Science building at NAIST.

3.2 Characteristics of Sketch Floor Map

Although a sketch floor map is given as a bitmap image, it cannot be used directly as a grid-based or metric map for navigation because of the following restrictions.

- **Scale is not uniform across the map.** The map can be partially deformed, so scale of the map computed at runtime on one part of the map may not apply to other parts. This makes navigation with dead reckoning difficult.
- **Geometrical details are not available.** Details such as exact shapes of the walls can be omitted and shapes of intersections are usually inaccurate, making it hard to match raw sensor readings with the map.
- **Obstacles are left out of the map.** When drawing a sketch map, users tend to omit details about obstacles, such as pillars and furnitures, presented in the corridors. This not only makes raw sensor matching difficult, extra landmarks not expected by robots can make landmark-based matching difficult too.

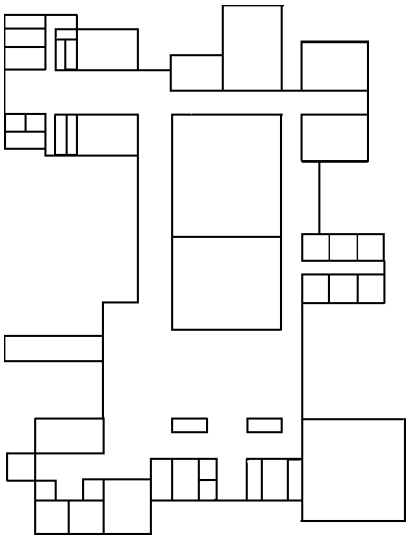


Figure 1: An Example of a Sketch Floor Map

We address these problems by creating an augmented topological map that represent the structure of the corridor as shown in the floor map and use it for navigation instead. During navigation, A localization method based on Multiple Hypothesis Tracking (MHT) is used to deal with ambiguity in landmark matching. We will descibe the augmented topological map and its extraction process in more details in Section 4, while MHT-based localization will be described in Section 5.

4 Augmented Topological Map

In this section, first, we describe a Voronoi diagram, which is used as a tool to extracted the structure of the corridor, and an augmented topological map, which is used to represent the environment, then we outline a process of generating the topological map.

4.1 Definition of Voronoi Diagram

Let $P = p_1, p_2, \ldots, p_n$ be a set of points in the two-dimensional Euclidean plane. P is called the generators. Partition the plane by assigning every point in the plane to its nearest point $p \in P$. All those points assigned to p_i form the *Voronoi region* $V(p_i)$, that is,

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x| \ \forall j \neq i\}. \qquad (1)$$

Note that some points do not have a unique nearest neighbor. The set of all points that have more than one nearest neighbor forms the Voronoi diagram $\mathcal{V}(P)$[4]. A Voronoi vertex is a point $p \in \mathcal{V}(P)$ that has more than 2 nearest neighbors and a Voronoi edge is a set of points that forms a boundary between Voronoi regions.

The concept of Voronoi Diagram is utilized in many robot navigation systems, in many different applications, such as path planning[5], environmental modeling[6], exploration[7] and localization[8].

Voronoi diagram generated from the floor map represent its structure in the sense that Voronoi vertices are formed in the middle of the corridors, and vertices with

more than 2 neighbors indicate intersections and corners. This facilitates the generation of topological map that use intersections and corners as landmarks. The type of structural features suitable as a navigation landmarks are in fact largely depended upon the available sensors. Our robot is equipped with a range sensor so it is relatively easy to detect intersections and corners.

An example of the Voronoi diagram of a simple floor map is shown in Figure 2.

4.2 Augmented Topological Map

A traditional topological map is a graph that represents connectivity between landmarks, without containing any metric or geometrical information. In this work, the map is represented as a topological map, with nodes correspond to such landmarks as intersections, corners or a dead-ends in the corridor, and arcs correspond to adjacency between them. Each node in the map is augmented with a circular queue that stores a label of each node adjacent to it in counterclockwise order. In order to increase recognition capability, each node contains an attribute about its geometric property as follows:

- **+-intersection** is a node connected with 4 arcs that extends approximately perpendicularly to each other.
- **T-intersection** is a node that is connected with 3 arcs, with two of them perpendicular to the other.
- **Endpoint** is a node that where the Voronoi diagram ends at a wall.
- **Dead-end** is a node that has 3 neighbors, and two of them are Endpoints.
- **Corner** is a node that has 3 neighbors, and one of them is an Endpoint.
- **Generic node** is a node that is not any of the above

+-intersection and T-intersection contain additional attributes about which neighbors are on arcs that are perpendicular. Corners and Dead-ends contain additional attributes about which neighbors of them is on the walls.

Creating Augmented Topological Map

First, a set of points G are generate from a given sketch map by taking a sample of black pixels in the map at a regular interval. These sample points are used as generators for Voronoi diagram. In our implementation, the Voronoi diagram is generated by creating *Delaunay triangulation* of the generators. A set of 3 points $P = \{p_1, p_2, p_3\} \subset G$ form a Delaunay triangle if and only if there is a point c in the plane equidistant to each point in P and no other points in G is nearer to c. In other words, c is the center of the circumcircle that pass through p_1, p_2, and p_3 and has no other points in G inside it. Therefore, c is on a Voronoi edge of Voronoi regions generated by P. We consider two Delaunay triangles that share a side and have no points in G between them *adjacent*. By connecting centers of each circumcircle of all adjacent Delaunay triangles, a Voronoi diagram is created.

For each Voronoi vertex that has more than two neighbors, a topological node is created with a label. The vertices that has only two members becomes arcs between nodes. The labels of nodes that are connected to it by Voronoi diagram are add to the queue attached, in counterclockwise order. Average direction of each arc is calculated. Nodes connected by a very short arc are combined together. Lastly, each topological node is then categorized as stated above, using information about direction of arcs and neighbor nodes connected to it.

A topological map generated from Voronoi diagram in Figure 2 is shown in Figure 3(a) and its graphical representation is shown in Figure 3(2). Note that node kl is a combined node and is corresponding to two Voronoi vertices generated from the cross intersection on the map.

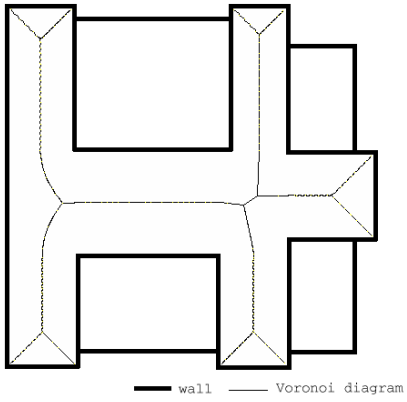


Figure 2: A Voronoi diagram from a floor map

5 Navigation System Architecture

In this section, we first describe the overview of the navigation system. Then, the localization process, which is designed to deal with ambiguous situation using Multiple Hypothesis Tracking, is discussed in more detail.

5.1 Overview

Our navigation system is developed for a wheel robot equipped with a laser range finder. The overall system architecture consists of four main components as shown in Figure 4.

Map Interpreter

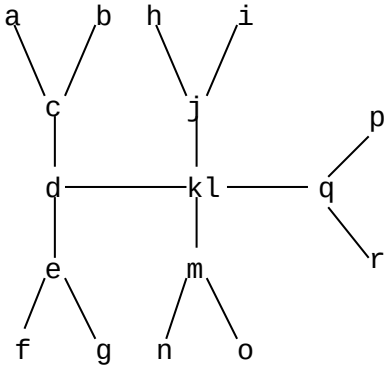
The Map Interpreter creates a topological map from a floor map provided by user, as described in Section 4. It receives a starting location and orientation of the robot, as well as a goal location relative to the floor map and associates them to arcs on the generated topological map.

Movement Controller

While navigating, the Movement Controller performs local obstacle avoidance while trying to move to a given goal position. To accomplish this, it uses a potential field approach, in which obstacles radiate repulsive forces, whose strength is inversely proportional to distance to the robot, and the goal is represented as an attractive force. The robot sums the force vectors and moves in that direction, while controlling its speed according to distance to the nearest obstacle in front of it to avoid collision.

Label	Neighbors	Attributes
a	c	Endpoint
b	c	Endpoint
c	a, d, b	Dead-end, Endpoint=a, b
d	c, e, kl	T-intersection
		$c \perp kl, e \perp kl$
e	d, f, g	Dead-end, Endpoint=f, g
f	e	Endpoint
g	e	Endpoint
h	j	Endpoint
i	j	Endpoint
j	h, kl, i	Dead-end, Endpoint=h, i
kl	j, d, m, q	+-intersection
		$j \perp d, j \perp q, d \perp m, m \perp q$
m	kl, n, o	Dead-end, Endpoint=n, o
n	m	Endpoint
o	m	Endpoint
p	q	Endpoint
q	p, kl, r	Dead-end, Endpoint=p, r
r	q	Endpoint

(a) Topological Map



(b) Graphical Representation

Figure 3: The topological map of Figure 2

Cartographer

The Cartographer component integrates readings from two-dimension laser range finder (LRF) into a robot-centric grid-based map, which is used for landmark recognition and obstacle avoidance. The map covers area of five meters around the robot, and has one centimeter resolution.

We assume that a corridor environment consists mostly of straight walls, with little irregularities. Under this assumption, at each update cycle, new readings are integrated into the map by matching straight lines extracted from the readings to those in the map. To do this, Hough transform is applied to the readings from the laser range finder and the robot-centric map. Each straight line is projected to a point in Hough plane. Under an assumption that the change of robot's pose is small between updates, the correspondences between straight lines in new sensor readings and those in the robot-centric map can be determined by simply searching for the two nearest point in the Hough plane, one from the map and the other from the new readings. Then, translation and rotation between update is calculated from the difference of Hough parameters between the matching straight lines. Odometry is used instead when the matching is too ambiguous or straight lines cannot be extracted.

A local topological map is generated from the robot-centric map, using the method described in the previous section. Each node is categorized and attributes are assigned to it. The exceptions are nodes that is not the nearest to the robot, and is not an Endpoint or a Corner. These nodes cannot be categorized reliably, due to the fact that the robot can observe the environment only partially. Therefore we categorize them as Generic nodes instead. This module then pass the local topological map to the Thinker for use in localization.

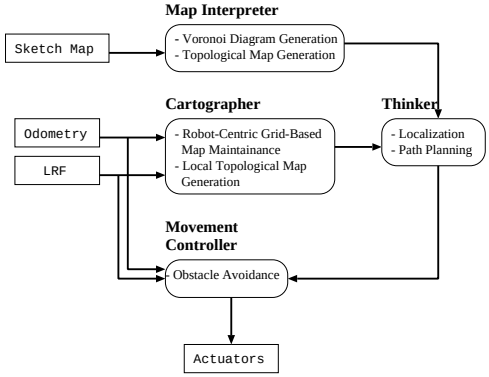


Figure 4: Navigation System Architecture

Thinker

Localization is performed using a technique based on Multiple Hypothesis Tracking. The localization technique will be discussed in the next subsection. The Thinker also acts as a path planner, giving a goal position in the current robot-centric map to the Movement Controller. The goal position is choosen from points on a Voronoi edge on which, according to localization results, the robot should move in order to (hopefully) reach the next node on the global path.

5.2 Localization Based on Multiple Hypotheses

The Thinker performs localization using the local topological map created by the Cartographer and the global topological as inputs. Before discussing the localization method, the following terms that will be used in this subsection is defined.

Match between nodes

Two nodes *strictly match* if they have the same number of neighbors and have the same attributes, without considering difference of labels. For example, a Corner node v that has three neighbors v_1, v_2, v_3 with v_2 as the Endpoint matches with a Corner node w that has three neighbors w_1, w_2, w_3 with w_1 as the Endpoint. Additionally, two nodes *match* if they are strictly match, or at least one of them is a generic node and they have the same number of neighbors. In other words, a generic node match everything that has the same number of neighbors as itself.

Robot Pose

A pose of a robot is a directed arc $\vec{ij}$ in a topological map that the robot is moving on. In other words, robot is somewhere between node i and j, and is heading to node j.

Pose Hypothesis

A hypothesis of the current robot pose $h = v_i\vec{v}_j$ is an the arc robot is presumed to be moving on. A list of nodes that the robot detected before getting to the current pose and a belief value (ranging from 0 to 1) are attached to each hypothesis. A distance between two hypothesis $h = h_1\vec{h}_2$ and $k = k_1\vec{k}_2$ is defined as number of nodes in the shortest path between h_2 and k_1 that do not pass through $h_2\vec{h}_1$ or $k_2\vec{k}_1$. If such path is impossible, then we define the distance as infinity.

Current Pose Candidate

Let the topological node the robot is approaching be n_{curr}, its neighbors be $n_1, n_2, ..., n_k$, in counterclockwise order, and $n_1\vec{n_{curr}}$ be the arc the robot approached n_{curr}. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be nodes in the topological map that match n_{curr}. A directed arc $m_i\vec{c}_j$ is a candidate of the current pose when m_x match n_y, where $x = i \bmod k+1, (i+1) \bmod k+1, \ldots, (i+k-1) \bmod k+1$, $y = 2, 3, \ldots, k$, and $m_1, \ldots, m_k$ are neighbors of c_j in counterclockwise order.

Localization and Local Path Planning

The Thinker maintains a set of hypotheses of the current robot pose $\mathcal{H} = \{h_1, \ldots, h_N\}$ that starts out with one initial hypothesis created with the pose of the robot as given to the Map Interpreter by the user, and belief value of 1. $\mathcal{H}$ is updated when the robot gets near a topological node detected by the Cartographer, denoted here as n_{curr}. The update algorithm is as followed:

Let set $\mathcal{C}_{curr} = \{c_1, \ldots, c_M\}$ be a set of current pose candidates generated by n_{curr}. If $\mathcal{C}_{curr} = \phi$, then it is assumed that this n_{curr} is conflicting with the map, and is ignored and the algorithm ends here. Otherwise, for each hypothesis $h_i = v_j\vec{v}_k \in \mathcal{H} \cup \mathcal{C}_{curr}$

- If $h_i \in \mathcal{H}$ and $h_i \in \mathcal{C}_{curr}$: The robot has found the destination node, v_k, as expected.
  1. Create hypothesis $h_{N+1} = h_i$ and add it to $\mathcal{H}$, N=N+1.
  2. If n_{curr} strictly match with v_k, increase the belief value of h_i by 100%, else increase it by 50%.
  3. If v_k is the goal, and h_i has the highest belief value, then assume that the robot has reached its goal and stop.
  4. Find the shortest path from node v_k to the goal, using Dijkstra's shortest path algorithm. Let $v_k\vec{v}_l$ be a directed arc included in the path.
  5. Add v_k to the detected node list associated with it and let $h_i = v_k\vec{v}_l$.
- else if $h_i \in \mathcal{H}$ and $h_i \notin \mathcal{C}_{curr}$: The robot has arrived at an unexpected node. It might not be where it though it was, or may be this is due to noise in the environment.
  1. Decrease belief value of h_i by 25%.
- else if $h_i \notin \mathcal{H}$ and $h_i \in \mathcal{C}_{curr}$: The robot has arrived at an unexpected node. If this h_i is not too far from those in the hypotheses list, chances are the robot has passed through the nodes between them without noticing. Otherwise, this node is probably generated from noise in the environment.
  1. Find k, a hypothesis in $\mathcal{H}$ that is nearest to h_i.
  2. If the distance d between k and h_i is more than 1, discard h_i, else create a new hypothesis $h_{N+1} = h_i$ with belief value equal to $1/(d+1)$ times the belief value of k.
  3. Find the shortest path from node v_k to the goal, using Dijkstra's shortest path algorithm. Let $v_k\vec{v}_l$ be a directed arc included in the path.
  4. Add v_k to the detected node list associated with it, and let $h_i = v_k\vec{v}_l$

Each time a pose candidate is used to update a pose hypothesis, there is a risk of association error. This risk grows with the degree of discrepancy between the map and the real environment, and the uncertainty of the hypothesis. Furthermore, a complex environment tends to create many "ghost" nodes that greatly increase chance of association error. In order not to loose the true pose hypothesis by updating it incorrectly, in the algorithm above, each hypothesis is split into two identical copies and the update is applied to only one of them.

To keep number of hypotheses low, when a hypothesis is added to the set $\mathcal{H}$, it is check if there is already the same hypothesis existed. If the data associated with the hypothesis are different, then the one with higher belief value is kept and the other discarded. Lastly, after finished updating $\mathcal{H}$, belief value for each hypothesis is normalized so that the sum of belief values of all hypotheses equals to one. Hypotheses with belief value below a threshold are discarded. In our case, the threshold is half the average of belief value of hypotheses.

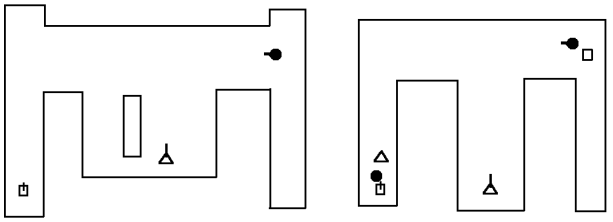
6 Experimental Results

In this section, we report on experiments in simulated corridor environments. Three navigation experiments were performed with a robot simulator using maps shown in Figure 5. On the left side is the map of environments that the robot navigated in, and on the right side is the sketch map of the environment given to the robot. In each environment, three navigation tasks are given, denoted by triangles, rectangles, and circle in the map, which are call task 1, task 2 and task 3, respectively. Starting poses of the robot are shown as the marks with arrow, and goal locations are the marks hatt have no arrow.

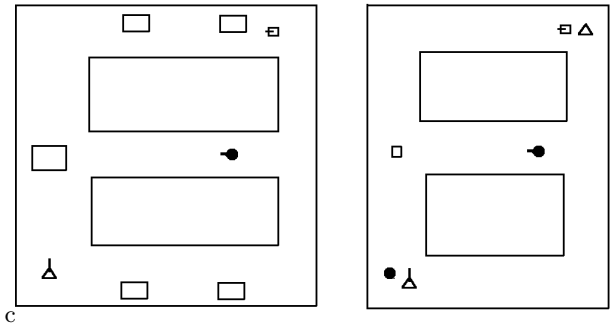
If the robot comes to stop around the goal position, we considered the task successful. On the other hand, if the robot stop elsewhere or shows no sign of coming to a stop in a reasonable time, we consider the task as a failure.

The results are shown in Table 1. Out of 9 tasks, 7 are performed successfully. In the successful runs, the number of hypotheses are low and the most likely pose and the second most likely one have a distinct different in belief values. This shows that the robot can handle the unexpected obstacles and other irregularities well enough in those situation.

However, the robot failed to come to a stop in experiment 2 during the green and blue navigation task. The number of hypotheses and the belief values are the values at the time we decided to stop the robot. In these experiments, the paths to the goal locations lead the robot through many unknown obstacles. In these circumstances, many hypotheses are created and there is not enough known landmarks along the way to justify them.

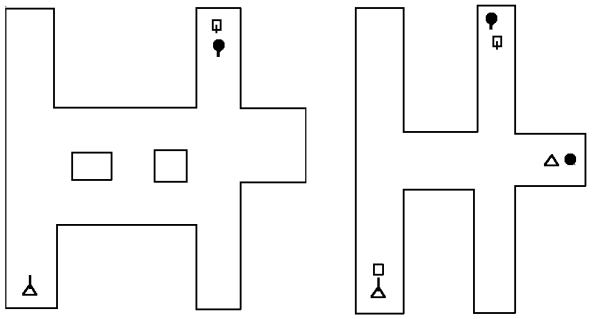


(a) Experiment 1



c

(b) Experiment 2



(c) Experiment 3

Figure 5: Maps used in navigaiton experiments

7 Future Works and Conclusions

We have described a robot navigation system designed to operate in indoor corridor environments, using a sketch map provided by a user. The system enable users to easily supply the robot with information about the environment, thus eliminating the need for exploration. This work consists of two main aspects: one is to make a robot understand a sketch map given to it, and the other is to use the map effectively in navigation. In order to interpreted the sketch map, a Voronoi diagram is extracted from the sketch map and then interpret in into a topological map. To make use of the topological map, which is an incomplete description of the environment, we deploy a technique based on Multiple Hypothesis Tracking in localization. The basic idea is tested in a simulated environment with satisfactory results. We plan to test this system further in real environment, where path will be much longer than and more complex than in the simulation.

The system can still be improved in many ways. A proper planning and action selection algorithm is needed to improve chance of successful navigation in confusing situations. Additionally, we would like to incorporate a probabilistic framework, such as Baysian evidence fusion, into the hypothesis update method in the localization process, where we are relying on trials-and-errors to determine various parameters for updating belief values of the hypotheses. Another problem is that, navigation using Voronoi vertices as landmarks is not feasible in some situation, such as when robot is moving in a large hall, where farther side of the hall is not in robot's sensor range, thus making it difficult to create accurate Voronoi diagram of the hall. Further work will investigate the strategies for navigation in large space to address this problem.

Table 1: Experimental results

Exp#	Task	Result	# of Hypo.	# 1 belief	# 2 belief
1	1	success	2	0.78	0.22
	2	success	2	0.80	0.20
	3	success	2	0.84	0.16
2	1	success	2	0.66	0.33
	2	failed	8	0.37	0.22
	3	failed	7	0.38	0.20
3	1	success	2	0.82	0.18
	2	success	3	0.59	0.32
	3	success	4	0.58	0.20

References

[1] `http://www.incx.nec.co.jp/robot/PaPeRo/english/p_index.html`

[2] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda and R. Nakatsu. Robovie: A robot generates episode chains in our daily life. In *Proc. Int. Symposium Robotics*, pp.1365–1361, 2001.

[4] J. O'Rourke. *Computational Geometry in C second edition*. Cambridge University Press, pp.155–192, 1997

[5] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.1024–1031, 1999.

[6] S. Thrun. Learning maps for indoor mobile robot navigation. *Artificial Intelligence*, vol. 99, pp. 21–71, 1998.

[7] H. Choset, S Walker, K. Eiamsa-Ard, and J. Burdick. Sensor-Based Exploration: Incremental Construction of the Hierarchical Generalized Voronoi Graph. In *Int. Journal of Robotics Research 19*, no. 2, pp.126–148, 2000.

[8] K. Terabayashi, T. Emaru, K. Oikawa, and T. Tsuchiya. Navigation of Autonomous Robot with Handwritten Map. In *Proc. of The Eight Robotics Symposia*, pp.361–366, 2003.

[9] P. Jensfelt and S. Kristensen. Active Global Localization for a Mobile Robot Using Multiple Hypothesis Tracking. In *IEEE Trans. on Robotics and Automation*, vol. 17, no. 5, pp.748–760, 2001

[10] M. Tomono. and S. Yuta. Mobile Robot Localization on a Map with Large Inaccuracy. In *Journal of Robotics Society of Japan*, vol. 20, no. 4, pp.425–436, 2002

[11] B. J. Kuipers and Y. T. Byun. A Robust, Qualitative Method for Robot Spatial Learning. In *Proc. of the AAAI*, pp.774–779, 1988.

[13] H. Shatkay and L. P. Kaelbling. Learning Topological Maps with Weak Local Odometric Information. In *Proc. 15th Int. Joint Conf. on AI*, pp.920–927, 1997

[14] S. Thrun and A. Bücken. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In *Proc. of the AAAI*, 1996.