Mobile Robot Localization Using Active Sensing Based on Bayesian Network Inference

Hongjun Zhou, Shigeyuki Sakane

Department of Industrial and Systems Engineering
Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112 Japan
{zhou,sakane}@indsys.chuo-u.ac.jp

Abstract

In this paper we propose a novel method of sensor planning for a mobile robot localization problem. We represent the conditional dependence relation between local sensing results, actions, and belief of the global localization using a Bayesian network. Initially, the structure of the Bayesian network is learned from the complete data of the environment using the $K2$ algorithm combined with a genetic algorithm (GA). In the execution phase, when the robot is kidnapped to some place, it plans an optimal sensing action by taking into account the trade-off between the sensing cost and the global localization belief, which is obtained by inference in the Bayesian network. We have validated the learning and planning algorithm by simulation experiments in an office environment.

Key words:
Bayesian network, Structure learning, Sensor planning, Mobile robot, Localization

1 Introduction

The navigation and localization of a mobile robot is a traditional, fascinating research area. Many studies to date have focused on how to obtain an accurate map, and then how to match the sensing information of the robot to the map for localization. However, in robot navigation, the robot cannot always determine its unique pose only by local sensing information since the sensor is prone to errors and a slight change of the robot's pose deteriorates the sensing results. Therefore, many probabilistic approaches have been proposed to cope with uncertainties and to improve the robustness of the localization (1) (2).

The localization of a mobile robot involves estimating the pose of the robot in an environment. There are two classes of localization problem, *position tracking* and *global localization*. In the *position tracking*, a robot knows its initial position (3) and only needs to reduce uncertainty in the odometer reading. If the initial position is not known or the robot is kidnapped to somewhere, the problem is *global localization*, i.e., the mobile robot has to estimate its global position through a sequence of sensing actions(4).

In this paper, we deal with the *global localization* problem. The mobile robot estimates its pose in an environment where locally similar sensing patterns prevent the robot from simple identification of the global position.

There are two key features of our system. Firstly, we use a Bayesian network (BN) to represent the conditional dependence relation among the sensing evidences, the actions performed at intersections of corridors, and the global localization belief. Initially, the robot navigates in every corridor at least once and records the sensing data of the environment. The system learns the BN structure from the complete environment data. Secondly, in the execution phase (global localization phase), the robot plans an optimal sensing action by taking into account the trade-off between the sensing cost and the global localization belief which is obtained by inference in the BN. We have validated the learning and the sensor planning algorithm by simulation experiments in an office environment which has locally similar sensing patterns.

The paper is organized as follows. In Section 2, we review previous studies on sensor planning of mobile robots. We describe the representation of the environment using BN, and BN structure learning in Section3. In Section 5, we introduce the calculation of the localization belief using BN, and planning of the optimal sensing actions for the localization by taking into account the balance of localization belief and the sensing cost. The experiments are described in Section 6. In the conclusions (Section 7), we summarize our work and discuss some future plans.

2 Previous works

2.1 Probabilistic Approaches for Global Localization

Some Bayesian approaches to mobile robot navigation and localization have been proposed. Thrun et al. (1) (4)proposed localization of a mobile robot using a particle filter. The particle filter resamples and updates the belief of localization, and estimates the maximal posterior probability density for the localization. However, since the robot moves randomly without sensor planning, the system does not always assure efficient convergence of the particles if the environment has locally similar sensing patterns.

Some studies used Bayesian networks for modeling the mobile robot navigation environment. Asoh et al. (5) developed a system to combine local information for localization using a Bayesian network. They employed speech interaction to obtain evidences in the Bayesian network to decrease uncertainty of the localization. However, the system could not actively plan how the mobile robot should gather sensor information, and the Bayesian network structure is manually designed. Basye et al. (7) built planning and control systems that integrated sensor fusion, prediction, and sequential decision-making using a temporal belief network (dynamic Bayesian network).

2.2 Sensor Planning of Mobile Robot

Sensor planning of mobile robots is also an area of active research. Fox et al. (8) proposed an *Active Markov Localization* method to improve the efficiency of localization. The sensing actions were planned based on the expected entropy of the localization probability. However, since their system was based on the first-order Markov process, it cannot represent complex relations between actions, local information, and the global localization. A multiple hypothesis tracking approach has been used in active global localization (9). However, the approach was based on Kalman filtering which assumes a model of linear dynamics with Gaussian noise.

Kristensen (10) proposed a mobile robot sensor planning method based on a top-down decision tree algorithm. However, the utility-based Bayesian decision tree theory is too simple to evaluate the conditional dependence relations between local sensor information and the global localization. Miura et al. (11) also defined an utility value based on sensing cost and sensing uncertainty to plan the sensor action for navigation, however, the system used a simple Bayesian rule.

2.3 Main Features of the Research

We have proposed an algorithm to reconstruct a BN and use it to plan efficient sensing actions for mobile robot localization (12). The probability of localization will be improved when the robot obtained a sensor information, and the obtained sensor information is added as a node to BN. The structure of the BN is a naive Bayes, there are not any dependent relation between the sensor information nodes. In this paper, the BN is learned from the environment data, the learned structure of BN captures partial conditional dependence relation between sensor information nodes, action nodes and label of intersections.

Additionally, the research of citation (12) only represented partial environment, it can not copy with a large environment for learning and localization. Conversely, the method of this papers learns the BN from complete environment, it generalize the previous version. Sensor information of this paper is categorized into four types of nodes. Instead of the naive Bayes style, we build some conditional dependence relations between them.

In the field of Bayesian networks, how to obtain the best structure of Bayesian network from statistical data is an interesting issue which has been pursued recently in theoretical analysis (13) (14) (15) as well as its application to real problems (16).

We propose a sensor planning method for mobile robot localization. Initially, we represent conditional dependence relations between local sensing results, actions, and beliefs of the global localization in a Bayesian network (BN) structure. The BN structure, as well as the parameters, is learned automatically from the environment data using the $K2$ algorithm combined with a genetic algorithm (GA). In the execution phase, when the robot is kidnapped to some place, it plans an optimal sensing action by taking into account the trade-off between the sensing cost and the global localization belief, which is obtained by inference in the BN (17) (18) (19).

3 Environment Information Gathering and BN Configuration

3.1 Path for Environment Information Gathering

We performed the simulation experiments in an office environment (Fig. 2). Initially, to obtain complete environment information, the robot must navigate in all of the corridors and intersections. We employ a framework of the *Chinese postman problem* (20). The *Chinese postman problem* requires finding the shortest tour in a graph which visits every edge at least once. As shown in Fig. 1, we represent the topology of the environment as a graph and search a path from A to A using the *next node algorithm* (21). Then the robot navigates in all corridors and intersections along the path and gathers the environment information to be used for localization tasks. The motion of the mobile robot is shown in Fig. 2.

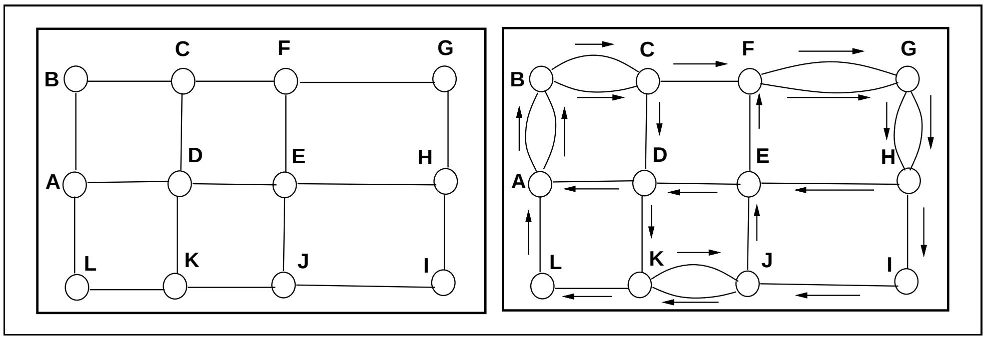


Fig. 1. (left) A graph which represents the topology of the environment. (right) A path (from A to A) obtained as a solution of the Chinese postman problem.

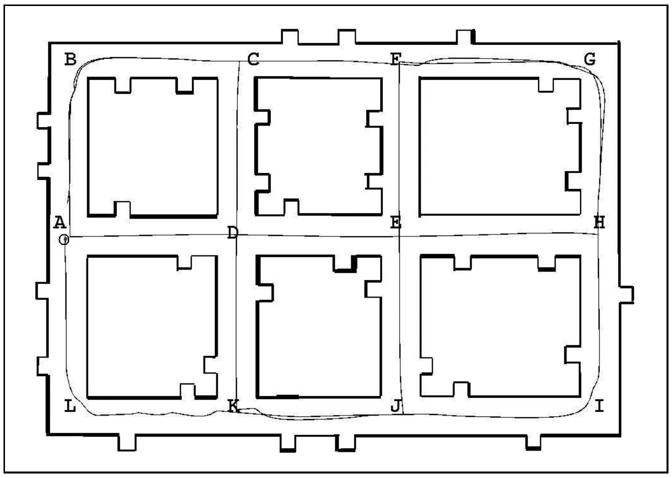


Fig. 2. The mobile robot gathers the environment information along the path obtained from Fig. 1.

3.2 Assumption of System

To simplify the planning and localization processes, the current system uses the following assumption:

(1) The robot is guided by commands at each intersection along the path obtained by the *next node algorithm* (21). The intersection's labels $A, B, ..., L$ in Fig. 9 are given by a human at every intersection.
(2) The landmarks are hollows which appear on both sides of every corridor and geometric feature of the intersections. Since the landmarks are not unique in the environment, the robot localization often falls into difficulty by observing only a few of them. In the simulation experiments, we assume a sensor, such as a laser range finder, to find the hollows. However, our localization and planning algorithm is not restricted to using a specific sensor and allows other sensors to perceive the landmarks.
(3) We deal with uncertainty in the global localization when similar patterns of the sensing data inevitably appear at different locations. However,

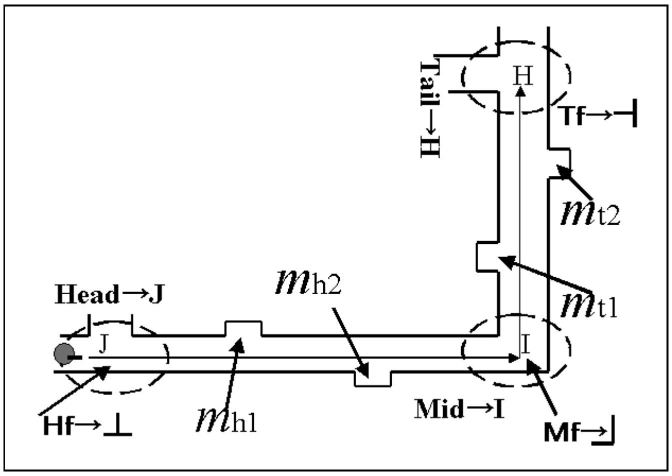


Fig. 3. Mapping the environment information of two neighboring corridors into nodes of BN.

uncertainties in the sensor data caused by the device itself or by the dynamic environment are not considered.

3.3 Environment Representation and BN Configuration

We define a *segment* (Sg) as the environment information of a corridor between two neighboring intersections. One *segment* involves four kinds of information as follows:

(1) Two intersection labels,
(2) Landmarks on both sides of the corridor between two intersections,
(3) Geometric features of the intersections sensed when the robot enters the intersections,
(4) Action taken by the robot when it enters the corridor.

In our system, we call the environment information of two neighboring corridors an *environment information set*. As shown in the following sections, we use a path, a solution of the "Chinese postman problem", to guide the robot to gather the environment information. Since there are corridors on the path that have been sensed in one way only, we add complementary data, i.e., to be obtained when the robot senses from the reverse direction, to the environment information data set.

The information of every *environment information set* (for example, label of an intersection, geometrical feature of an intersection, etc.) corresponds to a value of nodes in BN.

We define sensing information as *observable variables*, and labels of intersections as *hypothesis variables* of a BN. We put together all of the environment information of two neighboring corridors and save them into a training database. The database is used to learn the parameters and structure of BN. For example, the training database obtained from a data gathering tour in Fig. 2 has 138 data cases. The BN (Fig. 5) is learned from the training database. In this case, the BN has 13 probabilistic variables (nodes). As shown in Fig. 3, the nodes, *Head, Mid, Tail*, are defined by labels of the entrance intersection, middle intersection, and exit intersection of two neighboring corridors, respectively. In the experiments, the nodes *Head, Mid, Tail* have twelve possible values $(A, B, ..., L)$. The nodes *Action1* and *Action2* denote the actions which the robot takes when it enters *head* and *middle* intersections, respectively. The action nodes have three possible values: *go forward, turn left, turn right.* The nodes *Hf, Mf, Tf* correspond to geometric features (such as a range pattern) recognized by the robot when it enters the entrance, middle, and exit intersections, respectively. As shown in Fig. 3, these nodes have six possible values: $+, \top, \dashv, \vdash, \neg$ and $\perp$. In Fig. 3, there are four possible landmarks (hollows) in two neighboring corridors, represented by the nodes $m_{h1}, m_{h2}, m_{t1}, m_{t2}$. In the experiments, we assume that two hollows can appear on a side of a corridor, and the hollows are used as landmarks. We define the landmark in a list (geometric feature, *local distance* [1]). The landmark nodes have four possible values: "1$\sim$ 4" which denotes four layout types of the landmark. In addition, we define a *mediating variable* (17), **Cn**, by label of every data set. The variable of Cn represents the label of data set. In our experiment, we can obtain 138 sets of environment data for BN learning, so the node Cn has 138 values.

Note that the BN used for inference plays a different role from that of a topological network. The topological network represents only geometric relations of the landmarks and the intersections in a map, so the size of a topological network grows with the size of the environment. In contrast, the BN represents the conditional dependence relations between observed landmarks and beliefs of the intersections. The number of nodes in the BN's is independent of the size of the environment, since the *environment information set* and the conditional dependence relations among the nodes are represented in the node's values and the BN structure. The size of the environment is reflected in the number of values which the probabilistic variables (BN's nodes) can take. We can normally use the same BN structure even when the size of the environment becomes larger, and so can utilize BN structure learning as described in the following sections.

In our system, we map the *environment information set* (= the environment information of two neighboring corridors) into BN node's values, and represent the conditional dependence relation of environment information in BN structure. Therefore, the robot can only predict the next corridor's informa-

[1] The distance between an intersection and its neighboring landmark, or two neighboring landmarks

tion and action selection is limited to decide which corridor the robot should go to next and how far the robot should go in the next corridor to obtain enough information for the localization. However, when the belief of global localization is still low even if the robot sensed two neighboring corridors, the robot has to plan to select the next corridor based on the already sensed information. In this case, a framework of sequential decision-making based on one "segment" information will be necessary for the sensor planning; we leave this to a future study.

4 Learning BN Structure from Data

BN is a directed acyclic graph that represents dependencies between probabilistic variables. An arc between two nodes of BN represents the conditional dependence relation between the nodes. However, it is often difficult to determine the conditional dependence relation among nodes. In our localization tasks, we usually do not know which landmark has dependency with the other nodes, so we take a BN structure learning approach instead of designing the network structure manually.

4.1 $K2$ Algorithm Combined with GA

We apply a structure search method based on Bayesian score, named the $K2$ algorithm (14), to learn the conditional dependence relation between *local environment information*, *robot action*, and *global localization*. The Bayesian score is a joint probability $P(B_s, D)$ between BN structure (B_s) and database (D). The $K2$ algorithm is a greedy search algorithm. Initially each node has no parents, then the algorithm incrementally adds its parent which most increases the score of the resulting structure. When the addition of no single parent can increase the score, it stops adding parent nodes to the current node. Ref(14) describes that the search space is too huge to evaluate all of the possible structures. For example, when node number $n = 5$, the number of possible structures is $29,000$. To reduce the search space, the $K2$ algorithm uses a constraint of ordering of nodes (i.e., the conditional dependence attributes of a node should appear earlier in the order). However, it is often difficult to determine the order.

In our system, we employ a *genetic algorithm*(GA) to search the best ordering as described in Ref. (15). Using this ordering, $K2$ learns the best BN structure from the data. Then the Bayesian score of $K2$ gives a *fitness* value to GA. The combination of GA and $K2$ iterates until the average fitness is improved no further.

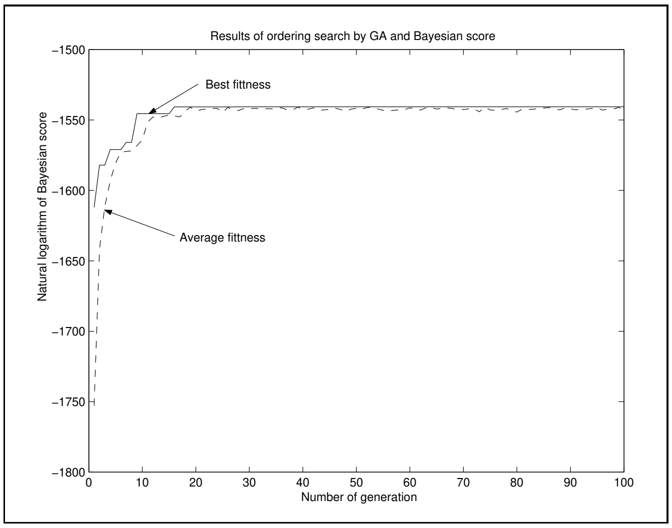


Fig. 4. The results of ordering searching by GA and Bayesian score

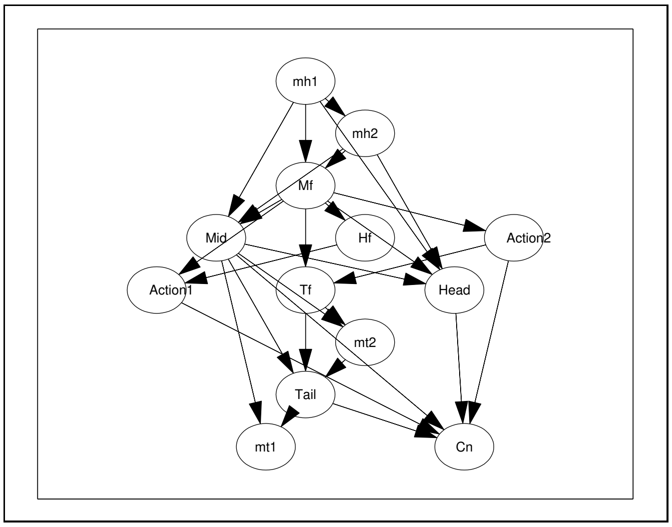


Fig. 5. Learned BN's structure by $K2$ and GA

4.2 Example of BN Structure Learning

Using the training database from Fig. 2, we attempt to learn a structure of BN. The population size of the GA is 80 and the algorithm uses *crossover* and *mutation* operations. Figure 4 shows the convergence of fitness value with 100 generations. The dashed line and solid line in the figure show the average and the best fitness scores of each generation, respectively. By combining the $K2$ algorithm with the GA search, we can obtain a *suboptimal* ordering of the nodes and a *semi-optimal* BN structure as shown in Fig. 5.

BN is represented in two factors (17), e.g. graph structure and CPTs. Graph structure of BN is to represent the casual relations between nodes. The graph structure of our BN has been learned by $K2$ algorithm combined with GA. CPTs is conditional probability tables of nodes, it can be learned from case data directly. Since the environment information data is complete, we use $Maximum\ Likelihood\ Estimation$(13) method to learn the CPTs. For example, The conditional probability of node X's value x_i is $P(X = x_i|Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)$, and $Z_1, Z_2, ..., Z_k$ is K parents nodes of node X. $z_1, z_2, ..., z_k$ are denoted by the values of nodes $Z_1, Z_2, ..., Z_k$. So the conditional probability $P(X = x_i|Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)$ should be calculated as the following:

$$
\begin{aligned}
& P(X = x_i|Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k) \\
= & \frac{P(X = x_i, Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)}{P(Z_1 = z_1, Z_2 = z_2, ..., Z_k)} \\
\simeq & \frac{N(X = x_i, Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)}{N(Z_1 = z_1, Z_2 = z_2, ..., Z_k)}
\end{aligned}
$$

$N(X = x_i, Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)$ denotes the number of training data cases which has $X = x_i, Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k$. $N(Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k)$ denotes number of the data case which has $Z_1 = z_1, Z_2 = z_2, ..., Z_k = z_k$ in the total environment data.

The conditional probability tables (CPTs) of the BN is learned by *Maximum Likelihood Estimation*. Since we can obtain complete environment data using *Chinese postman solution* (see section 3.1), the learned CPTs should be a exact estimation based on the gathered complete environment data.

In the learned BN, some nodes with large number of values, but it does not effect the inference accuracy and speed. In our case, we use the toolbox of *Maximum Likelihood Estimation* which is implemented in MATLAB. The learning speed is about 2 seconds. We also test the BN using training data, it shows a very accurate result (see the next section).

4.4 Test the Learned BN

To validate the performance of the learned Bayesian network, we have test the Bayesian network in the training data[2]. We set the evidence of $Head$, Hf, Mh_1, *action*2, and Mh_2, then estimate the probability of Mf, Mid, Mt_1, Mt_2, Tf, and $Tail$. The estimated probability is the joint probability of the intersection labels (Mid, *tail*) and sensor information of the second corridor given $Head$ intersection label, sensor information of the first corridor and robot action. The test experiments have been conducted using 138 sets of environment information data (complete environment data), inference results show the correct rate is 100%. So the learned structure is sufficient for our prediction and planning experiments, even if the learned BN structure without a global maximum Bayesian score.

5 Sensor Planning for Localization

5.1 Summary of the Sensor Planning System

The execution phase of the planning system consists of the following three steps:

(1) **Inference for localization:** Initially, a mobile robot starts navigation from an unknown position. While the robot is sensing in a corridor, the BN is used to infer the *global localization belief* whenever the robot obtains new sensing information.
(2) **Prediction for sensor planning:** If the sensing information of this corridor is insufficient for localization, the system predicts possible actions and sensing information to be obtained by the actions. The sensor planner runs at the exit intersection of the sensed corridor.
(3) **Sensor planning for localization:** Then the sensor planner uses the predicted information to select an optimal sensing action to perform active sensing by taking into account of the global localization belief and the sensing cost.

[2] Since we have used the complete environment data to learn the Bayesian network, the training data and test data are identical.

5.2 Inference for Localization

The robot starts navigation from an *unknown* position without sensor planning. The navigation basically uses a potential method in a corridor. The robot gathers sensing information events, including landmarks and geometric features of intersections, in the current corridor. Then the information events are given to the BN as evidences to infer global localization, i.e., which corridor the robot has sensed. The probability of the corridor's label is calculated as $P(Head, Mid, Tail|obtained\ sensing\ event)$ using the BN.

We define belief of the global localization ($TolBef$) as follows:

$$TolBef = (1/2)(max(P(Head)) + max(P(Mid))) \tag{1}$$

where $max(P(Head))$ and $max(P(Mid))$ are the maximum values of the probability of node $Head$ and Mid, respectively. $P(Head)$ and $P(Mid)$ are calculated by the BN inference. Since the conditional dependence relation between $Head$ and Mid is reflected in BN, it is not necessary to construct their joint probability, so we calculate the belief of the global localization ($TolBef$) as the average of probabilities.

If $TolBef \geq thd1$(thd1 is a threshold), the system terminates the localization process. Because in this case the robot can estimate the labels of the corridors only by using the current environment information, there is no need to perform sensor planning. Otherwise ($TolBef < thd1$), the robot has to move to the next corridor to perform active sensing. Therefore, the sensor planner selects an optimal sensing action for the localization.

Since the BN of our system is not a tree structure but has loops as shown in Fig. 5, we use the *Junction tree algorithm* (17) to infer probabilities of the nodes.

5.3 Prediction for Sensor Planning

The sensor planner consists of two processes: (1) prediction and (2) planning. The prediction process predicts some possible actions and sensing information expected to be obtained by these actions. The prediction algorithm has the following two steps:

(1) The first step is to search data cases, i.e., values of the node **Cn**, in the database, whose probabilities are not zeros based on the sensing event

obtained from the just-sensed corridor [3]. That is, the system stores the node **Cn**'s values, which satisfy the following condition, in a list $\mathbf{cnl} = (cnl_1, cnl_2, ...)$.

$$P(\mathbf{Cn}|obtained\ sensing\ information) \neq 0;$$

Based on the results, we can estimate which data case in the recorded sensing information database is closer to the obtained sensing information.

(2) The second step is to predict possible actions (PA) and sensor information (SI) [4] based on the obtained sensing information (OSI) and the estimated **Cn** values. The prediction is performed using the following probabilities:

$$P(PA|\mathbf{cnl}, OSI) > thd2\ (a)$$
$$P(SI|PA, \mathbf{cnl}, OSI) > thd2\ (b)$$

If the values of (a) and (b) exceed a certain threshold $thd2$, we save the possible actions in a list **actlist**, and save the predicted sensor information in a matrix $\mathbf{M_{sen}}$.

5.4 Sensor Planning Procedure

Through the prediction step, the system obtains **actlist**, a list of possible actions and also a matrix of predicted sensing information $\mathbf{M_{sen}} = (sn_1, sn_2, ..., sn_n)^T$, Each element of $\mathbf{M_{sen}}$ represents a predicted sensor information list to be obtained by a possible action (the element of **actlist**). Each predicted sensor information list of $\mathbf{M_{sen}}$ is sorted in the order of sensing cost, i.e. the distance from the current intersection to the location of the sensor information.

Consequently, the sensor planning process selects an optimal action from **actlist** which allows the robot to acquire enough sensing events to decrease ambiguity of the global localization belief by taking into account the trade-off between the sensing cost and the global localization belief.

For example, an **actlist** and a $\mathbf{M_{sen}}$ are shown in Fig. 6. The possible actions are "*action1, action2, action3*", and the integers on the right side of Fig. 6 represent expected sensor information to be obtained by the actions. In the Fig. 6, each row of the $\mathbf{M_{sen}}$ is one set of the predicted sensor information when taking the action on the left side. Every row of the $\mathbf{M_{sen}}$ is sorted in

[3] The prediction and planning processes are performed when the robot is in the middle intersection.

[4] It includes landmarks and intersection's geometric features expected to be perceived when the robot takes the actions

ascending order of the sensing cost, i.e., the sensing cost of the right entry is larger than that of the left. In the evaluation process, we use the elements of the $\mathbf{M_{sen}}$ and the possible actions to estimate the probabilities of the labels of the intersections, and calculate the sensing cost. Since the robot uses sensing information of a set of two neighboring corridors, the *TolBef* should be defined as the average of the maximum probabilities of the three intersection labels.

$$TolBef = (1/3)(max(P(Head)) + max(P(Mid)) + max(P(Tail))) \tag{2}$$

Using the above possible actions and predicted sensor information, the system performs the sensor planning which has the following three steps:

(1) The first step is to use the already-obtained sensing information, the possible action, and sensing information to be obtained by the action, to infer (*TolBef*), the belief of the global localization. In this step, we must evaluate every action and every set of sensor information (every row of $\mathbf{M_{sen}}$ in Fig. 6). For example, when we evaluate "action1" of Fig. 6 and the corresponding sensor information of the three rows, the procedures are as follows ((a)∼(d)):

(a) The system creates an empty list (**SenEvn**), and pushes the left-most element of the first row's sensor information into **SenEvn**.

(b) Using the **SenEvn**, "action1" and obtained sensor information to estimate the $TolBef$ based on Eq.2 and BN.

(c)
```
IF TolBef > thd3 (thd3 is a threshold)
         OR
   all of the elements in this row
   have been pushed into SenEvn,
     THEN evaluation of the first
          row's sensor information
          is finished.
ELSE
     THEN the next element of the
          first row's sensor inf-
          ormation is pushed into
          SenEvn.
          GOTO (b)
END IF
```

(d) Using the procedure (a)∼(c), the system also evaluates the other two row's sensor information corresponding to "action1". After the above procedures are finished, the number of sensor information sets (**count**), which have beliefs of the localization $TolBef > thd3$ [5] will be recorded. (**count**)

[5] Since $TolBef > thd3$ means the robot can uniquely determine three intersections' labels (entrance, middle, and exit intersection), in other words, the robot can determine its global location by "action1" and the first row's sensor information.

represents to what degree the robot could determine the *location* when it takes that action.

(2) The system sums up the sensing cost of every row's sensor information, which is used in the first step (**Cost**), and also sums up the **Cost** of each row (sensing information sets) which satisfy $TolBef > thd3$.

(3) Selects an optimal action by an *efficiency criterion*, i.e., an action which has the largest **count** and lower sensing cost.

For example, in the Fig. 6, each **count** of "action1" and "action3" is "3", and the **count** of "action2" is "1", so the optimal action can be selected from "action1" and "action3". Since the sensing cost of "action3" is lower than that of "action1", in this case, the optimal action should be "action3".

5.5 Speedup of the Sensor Planning

If the algorithm enumerates and checks all possible cases, enormous computation will be required in step (1) of the sensor planning procedure. To reduce the computational cost, instead of simply running the BN inference engine to evaluate the action and sensor information, we compare the sensor information features (include *local distance* and *geometric feature*) and check whether they can uniquely determine a *location* by the same action.

We will explain the method using the case of Fig. 6.

(i) We compare the sensing information of the $\mathbf{M_{sen}}$ that has the same action from the left side to right side. For example, the *action1* corresponds to three sets of sensing information. In the first row of the sensing information sets (the row with $R1$ of Fig. 6), since the first element of rows $R2$ and $R3$ are $''2''$, if we get only the first sensing information $''1''$ of row $R1$, the system can distinguish the sensor information (row $R1$) from the other two sets of sensor information (rows $R2$ and $R3$) which have the same action (action1). Of course, we can also use more sensing information of row $R1$, but taking into account the sensing cost, using only the first sensing information, $''1''$, is more efficient.

(ii) We must test the *TolBef* (by Eq.2) using the selected sensing information ($''1''$) and its action, "action1". If $TolBef > thd3$, we consider that the first sensing information ($''1''$) of row $R1$ expected by the "action1" is sufficient to uniquely determine a *location*. Otherwise, we must extend sensing information from its right side [6] and test the *TolBef* until the condition $TolBef > thd3$ is satisfied.

[6] For example, in row $R1$, if $''1''$ is not sufficient, we must add the right side of the elements, $''2''$ or $''2,\ 2''$.

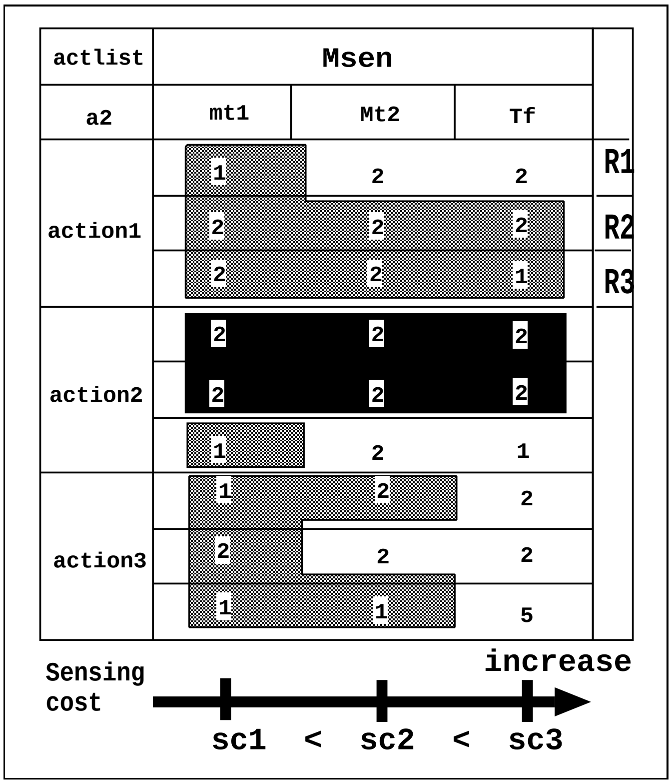


Fig. 6. An optimal action is determined by comparing the sensing information's *local distance* and *geometric feature*. The integer represents the sensing information. The numbers from left to right are values of m_{t1}, m_{t2}, and Tf, respectively. $Action1, action2, action3$ are values of a_2. The gray region indicates the *narrowest* sensor range to identify the sensor information sets which correspond to each action. The black region indicates the sensor information sets are identical, the system cannot distinguish each other. The area of gray region illustrates the sensing cost which is used at *sensor planning for localization* step to evaluate each possible action.

(iii) Using steps (i) and (ii), we obtain the *narrowest* sensing range to distinguish the other sensing information sets which is shown in the gray region (before testing the $TolBef$) with the same action. If there are some sensing information sets, corresponding to the same action, which are identical, we cannot distinguish the information sets and cannot determine a *location* uniquely as shown in Fig. 6 by the black region.

Since the majority computational cost of sensor planning is depended on the joint probability calculation of labels nodes (*Head, Mid, Tail*), given obtained and predicted sensor information set. We would evaluate the computing time of the sensor planning process using the frequency of Bayesian network engine running for joint probability estimation at the remained paragraph.

We define the N is the frequency of the Bayesian network engine running during the sensor planning process. Since the planning algorithm include three steps: *inference for localization, prediction for sensor planning* and *sensor* *planning for localization*, the frequency of the Bayesian network engineer running should be:

$$N = N_1 + N_2 + N_3$$

N_1 denotes the frequency of Bayesian network running during *inference for localization*, N_2 denotes the frequency of Bayesian network running during *prediction for sensor planning*, and N_3 denotes the frequency of Bayesian network running during *sensor planning for localization*. Since the N_1 and N_2 hold very small part of N, we only take into account the effect of N_3 at the remained computational complex discussion.

In the original algorithm, the N_3 of the above formulation is defined as follows:

$$N_3 = \sum_{a_i \in N_{PA}} \sum_{s_j \in N_{SI_{(a_i)}}} \left(SenNum_{(a_i,s_j)} \right) \tag{3}$$

N_{PA} denotes the number of possible actions, $N_{SI_{(a_i)}}$ is the the number of the predicted sensor information sets which correspond to sensor action a_i. And $SenNum_{(a_i,s_j)}$ denotes the number of sensor information which is in the *narrowest* sensing range. Intuitively, N_3 is the sensor information number of the gray and black region of Fig. 6.

In order to shorten the computing time of the step III, i.e., *sensor planning for localization*, we proposed a *speedup* algorithm to reduce the frequency of the Bayesian network running. Instead of evaluating joint probability of nodes (*Head, Mid, Tail*) given all of the sensor information and its combination [7], the speedup algorithm compares the sensor information sets and checks whether they can uniquely determine a location by the same sensing action. Since the speedup algorithm has to evaluate the joint probability of intersection labels given the compared results, the frequency of the Bayesian network running in the speedup algorithm should be defined as follows:

$$N_3^{speedup} = \sum_{a_i \in N_{PA}} N_{SI_{(a_i)}} \tag{4}$$

The parameter $N_3^{speedup}$ denotes the frequency of the Bayesian network inferring using the speedup algorithm. Intuitively, the $N_3^{speedup}$ of Eq. 4 is the number of rows. Comparing the Eq. 4 and Eq. 3, we can obtain a result:

$$N_3 > N_3^{speedup}$$

[7] The number of combination is $SenNum_{(a_i,s_j)}$

The above analysis results indicate that the speedup algorithm can effectively reduce the computational cost of the sensor planning process.

5.6 Theoretical Analysis of Sensor planning for Localization

In *sensor planning for localization* process, we have defined three steps, *inference for localization*, *prediction for sensor planning*, and *sensor for localization.* The purpose of the three steps is to find an action which can balance the probability of the nodes ($Head$, Mid, and $Tail$) and sensing cost. The action ($action^{\star}$) will let the integrated utility to be maximal:

$$action^{\star} = \arg \max_{action \in PA} \left(\theta \times TolBef + \frac{1-\theta}{Cost} \right) \tag{5}$$

θ denotes a parameter to balance the localization probability and sensing cost. $Cost$ indicates the sensing cost for localization. $TolBef$ is an average value of intersection labels probability, and is calculated based on the probability of $P(LABEL|PA, OSI, SI, Cn)$. $LABEL$ denotes the intersection label nodes ($Head$, Mid, and $Tail$). $P(LABEL|PA, OSI, SI, Cn)$ should be calculated as follows:

$$\begin{aligned}
& P(LABEL|PA, OSI, SI, Cn) && (6)\\
= & \frac{P(LABEL) \times P(PA, OSI, SI, Cn|LABEL)}{P(PA, OSI, SI, Cn)} \\
\propto & P(LABEL) \times P(PA, OSI, SI, Cn|LABEL) \\
& \textbf{In this system, we assume P(LABEL) is a constant} \\
\propto & P(SI|PA, OSI, Cn, LABEL) \times P(PA, OSI, Cn|LABEL) \\
\propto & P(SI|PA, OSI, Cn, LABEL) \times P(PA|OSI, Cn, LABEL) \\
& \times P(OSI, Cn|LABEL) \\
\propto & P(SI|PA, OSI, Cn, LABEL) \times P(PA|OSI, Cn, LABEL) \\
& \times P(Cn|OSI, LABEL) \times P(OSI|LABEL) && (7)
\end{aligned}$$

We can rewrite $P(OSI|LABEL)$ of the above formulation as follows:

$$P(OSI|LABEL) \propto P(LABEL|OSI) \times P(OSI) \tag{8}$$

Eq. 8 is estimated at step I (*inference for localization*). The robot estimates the probability of node $Head$ and Mid using the obtained sensor information (OSI)) from the moved corridor. If the localization belief satisfies the condition which is shown in Eq.1, the robot will stop active sensing for localization. Otherwise, it has to select an action to move to the next corridor.

The probability $P(Cn|OSI, LABEL)$ of Eq. 7 allows the robot to estimate which data case of the environment is closer to the obtained sensor information. In Eq. 7, the probability $P(PA|OSI, Cn, LABEL)$ predicts the possible actions, and $P(SI|PA, OSI, Cn, LABEL)$ predicts the possible sensor information (SI) which correspond to each possible action based on Cn and OSI. Cn is estimated based on $P(Cn|OSI, LABEL) \neq 0$ (see section 5.3). The calculation of $P(Cn|OSI, LABEL)$, $P(PA|OSI, Cn, LABEL)$ and $P(SI|PA, OSI, Cn)$ are conducted at the process of *prediction for sensor planning*.

In our algorithm, the calculation of $P(Cn|OSI, LABEL)$ and $P(PA|OSI, Cn, LABEL)$ are used to predict the possible actions, and the possible actions are determined based on an threshold ($th2$). After we determined the predicted actions, we can assume that the value of Eq. 6 is directly proportional to $P(SI|PA, OSI, Cn, LABEL)$. $P(SI|PA, OSI, Cn, LABEL)$ is probability of the predicted sensor information correspond to possible action PA and obtained sensor information OSI.

The calculation of $P(SI|PA, OSI, Cn, LABEL)$ allows the robot to predict the sensor information of the next corridor based on an threshold, $thd2$ (see section 5.3). Since the higher value of $P(SI|PA, OSI, Cn, LABEL)$ can guarantee $P(LABEL|PA, OSI, Cn, SI)$ to hold higher score, we can use the predicted sensor information (SI), action (PA), and Cn to calculate $P(LABEL|PA, OSI, Cn, SI)$, and evaluate every possible action. The evaluation of actions and predicted sensor information is performed in step III, i.e., *sensor planning for localization*.

In the section 5.4, we illustrate a concept, **count**, which represents to what degree the robot could determined the *location* when it takes an action. The determination of the *location* is calculated in $P(LABEL|PA, OSI, Cn, SI)$ using Bayesian network inference. $P(LABEL|PA, OSI, Cn, SI)$ denotes the probability of nodes $Head$, Mid and $Tail$. And $TolBef$ is an average value of the $P(LABEL|PA, OSI, Cn, SI)$ (see Eq. 1 and 2 of paper). Based on the above calculation and the thresholds ($th1, th2$), we can obtain some candidates of optimal actions. Taking into account the localization belief and sensing cost, we can find out an optimal action from the action candidates.

If we select an action for localization randomly, the robot can not guarantee to obtain a high score of $P(SI|PA, OSI, Cn, LABEL)$. Since we assume the value of Eq. 6 is directly proportional to $P(SI|PA, OSI, Cn, LABEL)$, the robot can not guarantee to localize itself by the randomly determined action, or the the randomly determined action cannot guarantee the robot to obtain a best balance point between $P(LABEL|PA, OSI, Cn, SI)$ and sensing cost.

6 Experiments

Using the above learning and planning algorithm, we performed simulation experiments in an office environment (Fig. 7). We implemented the BN learning and inference in a MATLAB BN Toolbox(22). Note that we assume that the length of corridors $F \rightarrow G$,$E \rightarrow H$ and $J \rightarrow I$ is longer than the other corridors. In Fig. 7,10 and 9, the real numbers in parentheses, the numbers with black squares, and the numbers with hatched squares represent the probabilities of the nodes *Tail, Mid, Head*, respectively.

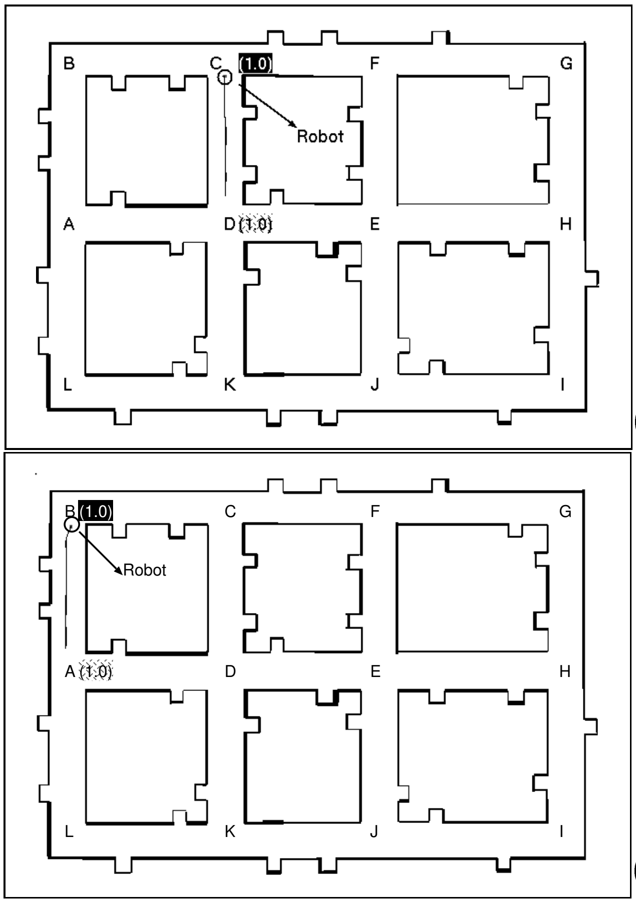
The thresholds *thd1* and *thd3* are convergence conditions of global localization. In step I of sensor planning process, i.e., *inference for localization*, we define the average probability of the nodes ($Head$, Mid) to be $TolBef$. If the robot can obtain $TolBef \geq th1$, the active sensing action should be stopped. The convergence condition is determined by practical wisdom. In our experiments, we define the *thd1* and *thd3* is 0.9, that means if we obtain average probability of node $Head$ Mid is lager than 0.9, the pose of robot is determined.

The threshold *thd2* serves the determination of sensor information prediction. The purpose of the *thd2* is to reduce the sensor planning computational cost, so the value of *thd2* don't effect the result of sensor planning, it just effects the computational cost of sensor planing. If we set a very low *thd2*, we have to evaluate a lot of predicted sensor information using Bayesian network. In our experiment, we set the threshold to be 0.9.

6.1 Inference for Localization

Initially, the robot starts from an unknown position of the environment. As shown in Fig. 7(a), without loss of generality, we assume the robot starts from an intersection D. After finishing sensing of the corridors, the robot's *global localization beliefs* are calculated by Eq.1. The probabilities of $Head$ and Mid are inferred using the learned BN and the sensor information of landmarks (m_{h1}, m_{h2}) and geometric feature Mf of the intersection. The sensor information which the robot obtained from corridors $D \rightarrow C$ is information of two landmarks, m_{h1}, m_{h2}, and the geometrical feature of intersection C. For example, information of two landmarks is denoted by number "2", and the geometrical feature is denoted by "⊤". Hence, the conditional probability of the node "Head", "Mid" is calculated as follows:

$$P(Head, Mid|m_{h1} = 2, m_{h2} = 2, Mf ='' \top'')$$



(a)
(b)

Fig. 7. Global localization using BN inference

The results of the above conditional probability are shown in Table 1. Among the values (i.e., labels) of the nodes "Head" and "Mid", D and C take the maximum probabilities. Based on Eq.1, the global localization belief is calculated as $TolBef = (1/2) \times (1.0 + 1.0)$= 1.0. Since $TolBef > thd1$ (thd1=0,9), the start point and current position are determined as "D" and "C", respectively, and the global localization is determined. The experiment shows if the sensor information is sufficient, the robot can localize itself using the BN inference by the sensor information of only one corridor, and so sensor planning is not necessary.

Fig. 7(b) shows another example of BN inference for global localization. The robot can localize itself using the sensor information of only corridors $A \rightarrow B$.

6.2 Prediction for Sensor Planning

However, if the sensor information obtained from the just sensed corridor is insufficient, the robot has to perform active sensing to gather more sensor information to localize itself. In case of Fig. 9, the robot starts from intersection

nodes	probability of the intersection's labels											
	A	B	C	D	E	F	G	H	I	J	K	L
Head	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mid	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 1
The probabilities of the nodes *Head* and *Mid* which inferred by the BN in Fig.7(a).

nodes	probability of the intersection's labels											
	A	B	C	D	E	F	G	H	I	J	K	L
Head	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mid	0.5714	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4286	0.0

Table 2
The probabilities of the node *Head* and *Mid* which inferred by the BN in Fig. 9(a).

D (of course, the robot initially does not know its global position) and moves to intersection K, and the obtained sensing information is landmarks m_{h1}, m_{h2} and geometric feature Mf of the intersection. We use "1" and "2" to denote the landmarks m_{h1} and m_{h2}, respectively, and we denote geometric feature Mf of the intersection by "⊤". Using this sensor information, the system calculates the following conditional probability using the BN. The results of the conditional probability are shown in Table 2.

$$P(Head, Mid|m_{h1} = 1, m_{h2} = 2, Tf ='' \top'')$$

The $TolBel$ is calculated based on Eq.1, $TolBef = (1/2) \times (1.0 + 0.5714)=$ 0.787. Since $TolBel < thd1$(thd1=0.9), the robot has two candidate locations indicated by a dot circle and a solid circle as shown in Fig. 9(a). The robot must perform sensor planning to decrease this uncertainty.

Using the *prediction algorithm* (described in Sec. 5.3), the robot predicts possible actions and sensing information expected by taking the actions, based on obtained sensing information (m_{h1}, m_{h2}, Mf). Using the *prediction algorithm*, the robot obtained **actlist** and $\mathbf{M_{sen}}$ as shown in Fig. 8. The predicted possible actions are "turn left" and "turn right", and the sensor information predicted by the possible actions has two rows, respectively.

In contrast to the predicted results (Fig. 8) and the simulation environment information (Fig. 9(a)), there are two possible positions of the robot (A and K). At the intersection A (or K), the robot can only perform two possible actions, "turn left" and "turn right", corresponding to the predicted results (**actlist** of Fig. 8). If the robot performs the action "turn left" at the intersection K (or A), it can obtain two sets of sensor information ($K \to J$ or $A \to L$), and if the robot performs the action "turn right", it can obtain the

actlist	M_{sen}		
a2	mt1	mt2	Tf
turn left	2	2	3
	2	2	6
turn right	2	1	5
	1	1	5

Fig. 8. Predicted possible actions (**actlist**) and the sensing information predicted by the actions ($\mathbf{M_{sen}}$) based on sensing information of the corridor ($D \to K$). (The integers of the table represent the predicted sensor information, i.e., instantiations of the probabilistic variables (m_{h1}, m_{h2}, Tf) in the BN)

sensor information from $K \to L$ or $A \to B$. The integers of Fig. 8 represent the predicted sensor information, they also correspond to the environment information. The experimental results show that the prediction algorithm works effectively.

6.3 Sensor Planning for Localization

Using the sensor planning procedure (described in Sec.5.4) and the speedup method (described in Sec.5.5), the robot can determine its location based on the sensing information which is shown by the grey background in Fig. 8. The experimental results show that we can obtain the same sensing range (marked in grey) using the *sensor planning procedure* as well as the *speedup method*. In Fig. 9, either the "turn left" or "turn right" action can determine two possible robot *locations*, but the sensing cost of "turn right" is lower than that of "turn left" (the area of grey region expected by taking the action "turn right" is smaller than that of "turn left"). Take into account the trade-off between the global localization belief and sensing cost, we can determine that the action "turn right" should be optimal action. As shown in Fig. 9 and Fig. 10, according to the sensor planning results, if the robot takes the "turn left" action, it cannot localize itself until it goes to intersection J. If the robot performs the action "turn right", it need not go to the next intersection for the global localization (Fig. 9(b)).

Figure 11 shows another localization experiment. The robot starts from an unknown position (K), and navigates to intersection K. Using the obtained sensor information, the estimated current positions are indicated as dot circles and solid circles. Using the prediction procedure (see Sec.5.3), we can obtain some possible actions and predicted sensor information lists (Fig. 12). Using

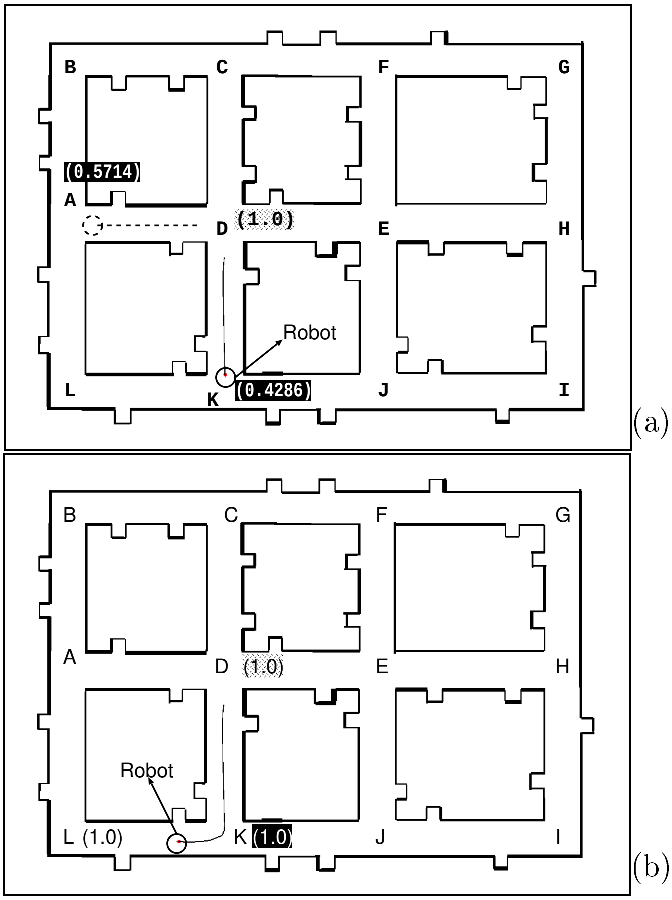


Fig. 9. Example of sensor planning experiments for robot localization. (In the figure, the real numbers in ( ), ( ) with black squares and ( ) with hatched squares represent the probability of node *Tail, Mid, Head*, respectively. If the intersection is the instantiation of node *Tail, Mid, Head*, the probability is shown at the intersection.

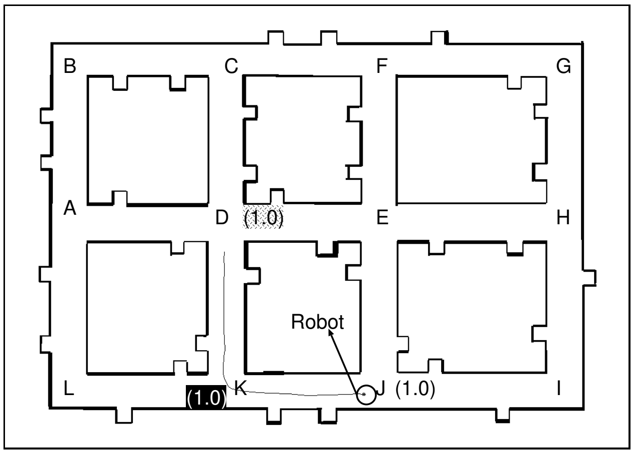


Fig. 10. The robot cannot obtain sufficient sensor information for localization until it goes to intersection J.

the sensor planning algorithm (see Sec.5.4 and 5.5), the sensing range for the global localization can be determined. The ranges are marked by the gray and

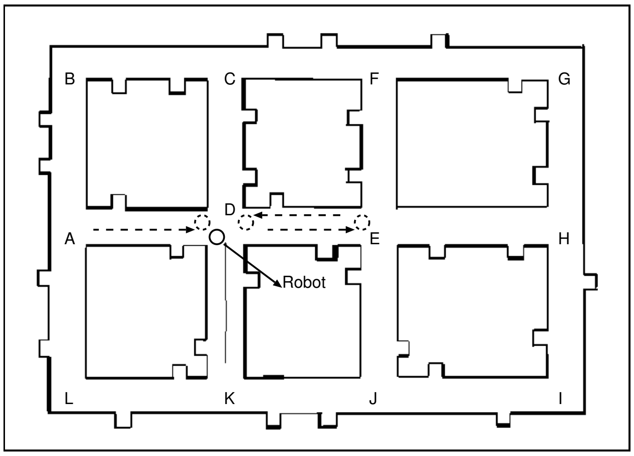


Fig. 11. The robot cannot localize itself only by the sensor information of the corridor ($K \rightarrow D$). The solid circles and the dot circles mean the possible positions inferred by BN based on the sensor information of the corridor ($K \rightarrow D$).

black background. Since the actions “go forward” and “turn right” allow the robot to determine four “location”s while the action “turn left” can determine two possible “location”s. As shown in Fig.12, if the robot acts “turn left”, it can obtain four sets of possible sensor information, corresponding to the simulation environment. The four sets of possible sensor information should be $D \rightarrow A$, $D \rightarrow C$, $D \rightarrow K$ and $E \rightarrow F$ (Fig.11), respectively. Fig.12 also indicates that if the robot acts “turn left”, it should obtain two sets of identical sensor information (be marked in black background), corresponding to the simulation environment, the two sets of identical sensor information should be $D \rightarrow A$ and $D \rightarrow K$. That means, in Fig. 11, if the sensed corridor is $K \rightarrow D$ or $E \rightarrow D$, even if it acts “turn left”, the corridors ($K \rightarrow D \rightarrow A$ and $E \rightarrow D \rightarrow K$) have identical sensor information, so in this case, the robot can not localize itself by action “turn left”. And if the robot acts “go forward” (or “turn right”), it should not obtain two sets of identical sensor information. The system eliminates the “turn left” action from the candidates.

In addition, the sensing cost of the action “go forward” is lower than that of the action “turn right” (the area of grey region expected by taking the action “go forward” is smaller than that of “turn right”). Therefore, the optimal sensing action should be “go forward”. As shown in Fig. 12, if the robot takes the action “turn right”, it cannot localize itself until it goes to the next intersection E(Fig. 14), however, the robot need not go to the next intersection C for global localization if it performs the action “go forward” (Fig. 13). If the robot takes the action “turn left”, it cannot localize itself even if it has sensed two corridors (Fig. 15).

Based on the experimental results, we verified that the proposed learning and planning algorithms are effective for global localization of a mobile robot.

actlist	$\mathbf{M}_{sen}$		
a2	**mt1**	**mt2**	**Tf**
go forward	1	2	2
	2	2	2
	4	4	2
	1	2	1
turn left	1	2	2
	1	2	2
	1	1	2
	2	2	2
turn right	2	2	2
	1	2	1
	2	1	2
	1	2	2

Fig. 12. Predicted possible actions (**actlist**) and sensing information to be obtained from the actions ($\mathbf{M_{sen}}$) based on sensing information of the corridor ($K \to D$). (The integers in the table represent values of the probabilistic variables (m_{h1}, m_{h2}, Tf) of the BN)

6.4 Real Robot Experiment

We conduct a real robot experiment in a practical environment which is similar to the above simulation environment. Initially, the robot starts from the intersection D, and moves to the intersection K (Fig. 16(a)). After sensing all of the sensor information of the corridor $D \to K$, the probability of nodes ($Head$ and Mid) are shown in Table. 1. Since the sensor pattern of the corridor $D \to K$ and $D \to A$ are identical, the robot cannot localize itself. Currently, the robot has two possible poses, i.e., the intersection K and A. The localization probability of K and A are 0.4286 and 0.5714, respectively (Fig. 16(b)). Using our sensor planning algorithm, the robot obtains an optimal action for localization is *turn right* (see section 6.3 for detail). Fig 16(c) shows that the robot localizes itself in a lowest sensing cost. The probability of $Head$, Mid and $Tail$ are 1.0. That means the robot pose is determined.

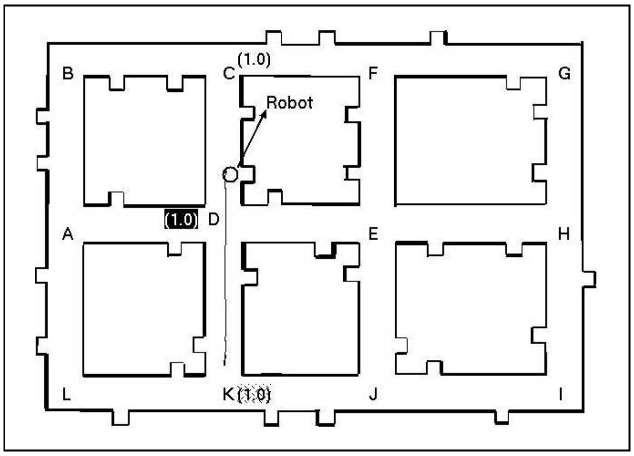


Fig. 13. The robot selects an optimal sensing action (go forward) using the sensor planner, so it can localize itself by using only the sensor information of corridor ($K \to D$) and a part of the sensor information of the corridor ($D \to C$).

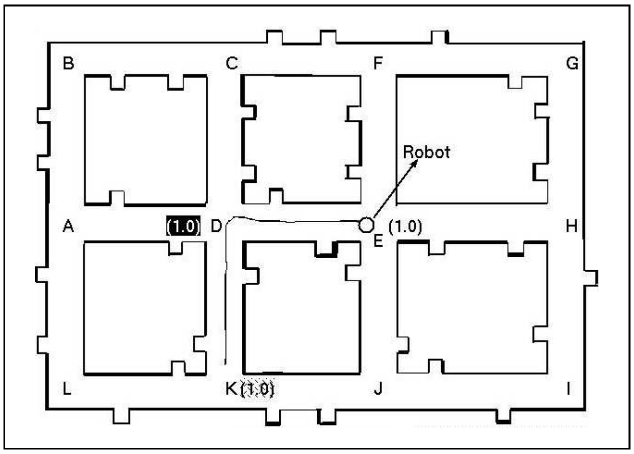


Fig. 14. The robot cannot localize itself until it moves to intersection E.

6.5 Comparison Between Sensor Planning and Randomly Action Selection

In the case of section 6.4, our speed up sensor planning method can be executed in about 0.35 seconds on a Pentium Mobile CPU (1.40GHz) at the intersection K. If the robot evaluates all of the sensor actions and the possible sensor information combination, the computational time should be 0.5 seconds. In the case of Fig.11-15, the proposed speed up sensor planning algorithm need 1.23 seconds for sensor action selection at the intersection D, then if the robot evaluates all of the sensor actions and sensor information combination, the computational time should be 2.72 seconds. So our proposed speed up sensor planning algorithm is effectiveness in the sensor action selection process.

In the case of section 6.4, the robot can localize itself based on the planned sensor action (*turn right*) at the first landmark of the corridor ($K \to L$), the moving time from intersection K to the the first landmark is about 20

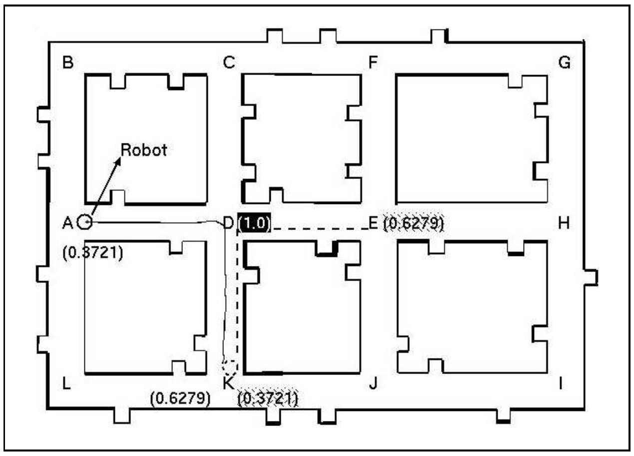


Fig. 15. If the robot performs the *"turn left"* action, it cannot localize even if it has sensed two corridors.

seconds (the speed of the mobile robot is 10cm/s). Then if the robot take *turn left* action for localization, the robot can not localize itself until it goes to the intersection J, the robot moving from intersection K to J need about 55 seconds. So the proposed algorithm can reduce moving cost for the localization.

7 Discussion and Conclusion

The space of Bayesian networks is a combinatorial space, consisting of a super-exponential number of structures $\longrightarrow$ $2^{O(n^2 \log n)}$. Therefor, finding the highest-scoring network from Bayesian network space is intractable. The K2 algorithm combined with GA allows us to estimate a semi-optimal structure. This heuristic search method cannot guarantee the learned structure has a best structure for the data sets fitting. However, through the learning of K2 with GA, we can obtain a structure with local maximum Bayesian score. This structure reflects partial conditional dependence relations between the variables, but cannot capture all of the conditional dependence relations between the nodes. We have test the learned Bayesian network using the training data, we found the test result is very good, even if the structure is not a global optimal. The experiment results validate the learned Bayesian network is sufficient for sensor planning and prediction process. However, we also found some inference bias errors during the prediction and planning process. For example, the experiment of Fig. 9 (a), the robot sensed the sensor information from the corridor (D$\rightarrow$ C). Since the sensor patterns of D$\rightarrow$ A and D$\rightarrow$ C are identical, the probability of the robot pose at intersection A and K should be 0.5, but the inferred results based on our learned Bayesian network are 0.5714 and 0.4286, respectively. We think the cause of the inference bias errors maybe the learned structure is not a global optimal.

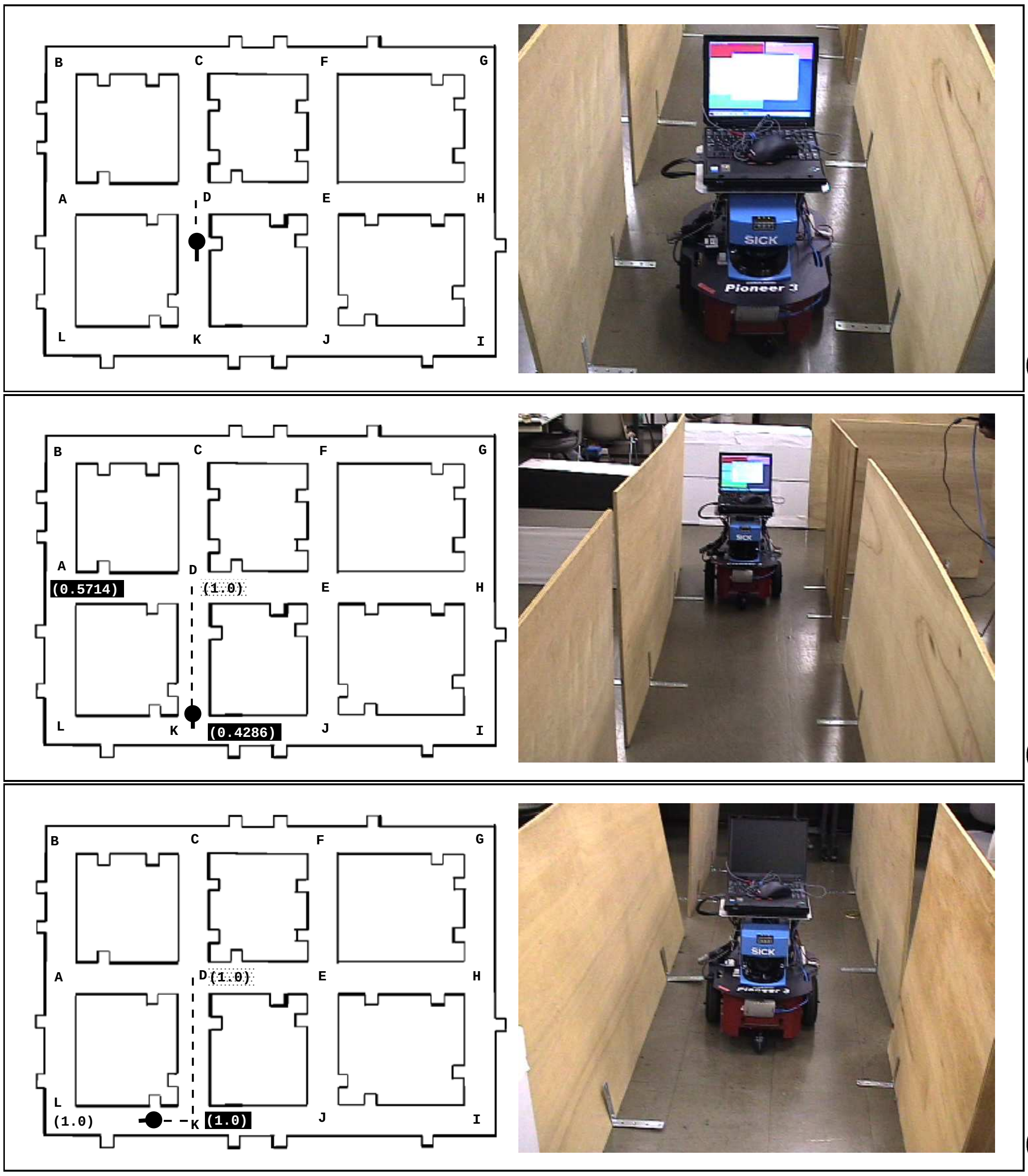


Fig. 16. A real robot experiment of sensor planning for mobile robot Localization using BN inference

In this paper, we proposed a novel sensor planning method for mobile robot localization using a Bayesian network. The BN structure is learned from environment data based on the $K2$ algorithm combined with GA. In the execution phase, the sensor planner predicts possible actions and sensing information to be obtained from these actions, and selects an optimal plan by taking into account the trade-off between the global localization belief and the sensing cost. The BN structure learning algorithm and the sensor planning algorithm are validated by simulation experiments. The following research issues should be considered in future work:

(1) As the landmarks of each *segment* are designed by a human, the system

cannot understand which landmarks must be focused on. We will investigate learning of the landmarks from the environment for a Bayesian network in a future study.

(2) We will integrate multiple sensor information for BN learning and sensor planning.

References

[1] S. Thrun, "Probabilistic Algorithms in Robotics," *AI Magazine*, 21(4):93-109, 2000.

[2] S.Thrun, "Bayesian Landmark Learning for Mobile Robot Localization," *Machine Learning* 33, pp.41-76, 1998.

[3] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation: Robot navigation under uncertainty in dynamic environments," *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1999.

[4] S.Thrun, D.Fox, W.Burgard, and F.Dellaert, "Robust Monte Carlo Localization for Mobile Robots," *Artificial Intelligence (AI)*, 2001.

[5] H. Asoh, Y. Motomura, I. Hara, S. Akaho, S. Hayamizu, and T. Matsui, "Combining Probabilistic Map and Dialog for Robust Life-long Office Navigation," *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS'96)*, pp.880-885, 1996.

[6] T. Inamura, M. Inaba, H. Inoue. "PEXIS : Probabilistic Experience Representation Based Adaptive Interaction System for Personal Robots," Systems and Computers in Japan, Vol.35, No.6, pp.98–109, 2004.

[7] K. Basye, T. Dean, J. Kirman, and M. Lejter, "A Decision-Theoretic Approach to Planning, Perception, and Control," *IEEE Expert*, Vol.7, No.4, pp.58-65, 1992.

[8] D. Fox, W. Burgard, and S. Thrun, "Active Markov Localization for Mobile Robots," *Robotics and Autonomous Systems*, Vol.25, pp.195-207, 1998.

[9] P. Jensfelt and S. Kristensen, "Active Global Localization for a Mobile Robot Using Multiple Hypothesis Tracking," *Proc. of Workshop on Reasoning with Uncertainty in Robot Navigation (by IJCAI'99)*, pp.13-22, 1999.

[10] S. Kristensen, "Sensor Planning with Bayesian Decision Analysis," *PhD thesis*, Faculty of Technology and Science, Aalborg University, Aalborg, Denmark, 1996.

[11] J. Miura and Y. Shirai, "Vision-Motion Planning for a Mobile Robot considering Vision Uncertainty and Planning Cost," *Proc. of 15th Int. Joint Conf. on Artificial Intelligence*, pp.1194-1200, 1997.

[12] H. Zhou and S. Sakane, "Sensor Planning for Mobile Robot Localization using Bayesian Network Representation and Inference," *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp.440-446,

2002.
[13] D. Heckerman, "A Bayesian approach to learning causal networks," *Technical Report MSR-TR-95-04, Microsoft Research*, March, 1995.
[14] G. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, 9:309-347, 1992.
[15] P. Larranaga, C. Kuijpers, R. Murga, and Y. Yurramendi, "Learning Bayesian network structures by searching for the best ordering with genetic algorithms," *IEEE Trans. on System, Man and Cybernetics*. Vol.26, No.4, pp.487-493, 1996.
[16] D. Corney, "Designing Food with Bayesian Belief Networks," *Fourth Int. Conf. on Adaptive Computing in Design and Manufacture*, 2000.
[17] F.V. Jensen, "Bayesian networks and Decision Graphs," *Springer*, 2001.
[18] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter, "Probabilistic networks and expert systems," *Springer*, 1999.
[19] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Morgan Kaufmann*, 1988.
[20] M. Kwan, "Graphic Programming using ODD or EVEN Points," *Chinese Math*, 1, pp.273-277, 1996.
[21] J. Edmonds and E.L. Johnson, "Matching Euler Tours and the Chinese Postman," *Mathematical Programming*, 5, pp.88-124, 1973.
[22] Kevin Murphy, "The Bayes Net Toolbox for Matlab," *Computing Science and Statistics*, Vol.33, 2001. Software is available at http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html.
[23] C. Houck, J. Joines, and M. Kay, "A Genetic Algorithm for Function Optimization: A Matlab Implementation," Technical Report NCSU-IE 95-09, 1995. Software is available at http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/.
[24] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector Machines," *Cambridge University Press*, 2000.