Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Knowledge-Based Robot Searching for an Unpredictable Goal under Unknown Environment ⋆

Huifang Wang * **Yangzhou Chen** *

* *Beijing University of Technology, Chaoyang district, Beijing 100022, China (Tel: 86-10-67396189-602; e-mail: Elizabethhw@gmail.com).*

Abstract: This paper describes a knowledge-based robot that explores an unknown environment for an unpredictable goal. Distinguishing characteristics of such environments for robotic navigation are that the goal's position is unpredictable and some obstacles cannot be sensed directly. Considering such features we propose a search algorithm for finding the goal and a simplified DPLL to allow robotic reasoning. Moreover we demonstrate the completeness and the execution cost of the search algorithm and also support the completeness and soundness of the simplified DPLL. The set memory rules allow for computer processing limitations. In addition, the simulation results of randomly produced environments demonstrate the completeness, soundness and effectiveness of method.

1. INTRODUCTION

This paper presents a knowledge-based planning method for a mobile robot which travels in an unknown environment looking for an unpredictable goal and navigates in a safe route with the help of reasoning based on its knowledge. The unpredictable goal means that the goal position is unknown until the robot reaches or near it. Such environments have two distinct characteristics which need to be considered. The first is that the overall picture of its environment is unavailable, so that the robot only has the information around its current position and does not know the exact position of the goal until it actually reaches there. This could be regarded as an agent-centered search problem in unknown domains for mapping build. The second distinct characteristic of our environments is that the robot must be able to judge the safety of its next step from its knowledge base and inference rules.

Robot path planning algorithms in unknown terrain or partial known environment have attracted high research interests since the early 1990's from Korf [1990] and these algorithms provide the theoretical search methods for sensor-based robots. Koenig [2003] categorizes these algorithms into agent-centered search and assumption-based planning. Assumption-based planning makes the path planning between the current location and the goal location with the assumption that all unknown terrains are travelable. Dynamic A* (D*) in Nilsson [1971] Stentz [1995], D* Lite in Koenig [2005] and LPA* in Koenig [2002]) moves a robot to a given goal location in unknown terrain with such assumption. In this paper, these algorithms can be used to plan a path back to the entrance after a robot achieving a goal location. An agent-centered search in Koenig [2001], however, restricts planning to the part of the domain around the current state of the agents. Greedy Mapping in Koenig [2001], Node Counting in Pirzadeh [1990], Learning Real-Time A* (LRTA*) in Bulitko [2006] can be categorized as Agent-centered search. Our proposed algorithms in section 3 also can be considered as an Agent-centered search since it limits planning only on the known locations. Moreover sharing ideas with above mentioned algorithms it gives high priority to safe unvisited locations. However in this paper we focus our attention on the completeness of the planning algorithm after introducing locations whose safety needed be referenced.

Another kind of related algorithms is coverage path planning which guides a robot to pass all points in a given environment. These coverage algorithms focus on their completeness. The coverage path planning first being developed for known space gives a good survey in Choset [2001]. Afterward the coverage algorithms for unknown space are introduced in Acar [2003] and Conner [2005]. These methods rely on finding the critical points of a function to guarantee the completeness of the algorithms. Its critical points have the similar role as $MPoint$ in our refined algorithm.

The second distinct characteristic of our environments motivates a robot that develops its ability to judge the safety of its adjacent locations by inference. The inference ability is a useful addition for the widely-used behavioral-based robots in Arkin [1998]. In this paper we use propositional logic to produce and increase the knowledge base and Davis-Putman-Logemann-Loveland (DPLL) algorithm in Russell [2003] for inference about the safety of surrounding environments. Due to the unique properties of this navigation application we simplified the procedure and still maintained its completeness and soundness. From repeated simulation experiments we recognize the importance of choosing the content of the knowledge, i.e. of remembering essential information and forgetting trivial items. See Section.4.

⋆ This work was supported by Doctoral Fund of Ministry of Education of China, 20060005014 and by the National Natural Sciences Foundation of China 60774037

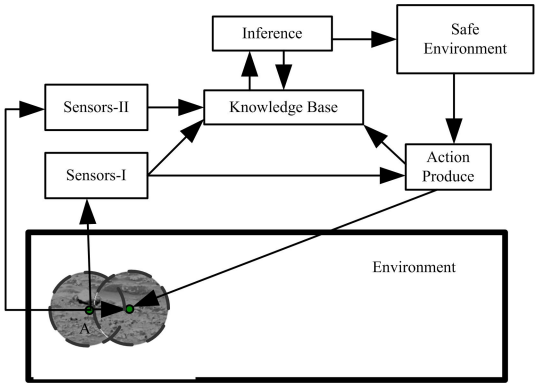

Fig. 1. A robot works with its environment

2. THE ROBOT NAVIGATION SYSTEM

In this section we first introduce how this knowledge-based robot works in a given environment. Then we explain the whole procedure of the robot for exploring this unknown environment to achieve its unpredictable goal.

Assume that the mobile robot is omni-directional, point-sized, equipped with two types of sensors. The first type of sensors (denoted by S_I) can directly perceive the obstacles, which are called O_I. Whereas the second type of sensors (S_{II}) only gets some information O_{II}, which is located in adjacent positions but its exact locations only can be obtained by inference. In addition, the robot does not know the position of the goal until it nears it. The robot only senses the around environment so that it interleave planning in deterministic domains with execution.

Fig.1 illustrates how a robot works in its environment which is initially unknown for it. When the robot is at location A, from S_I the robot directly perceives the obstacles in the surroundings and produces an avoidance action from a candidate list. If from S_{II} the robot senses some obstacles in the adjacent positions, however it can not know their exact location. Data from both groups of sensors is added to the knowledge base so that the robot is able to judge which adjacent position is safe according to the inference rules. Finally using the route search algorithm the robot determines a safe and complete action to take and adds its action to its knowledge base.

Procedure below describes how the robot determines its action according to its inputs from sensors. Two distinct procedures shown in two blocks produce actions due to the different objectives of the robot. In the first process the main objective of the robot is to find its goal and gain information about its environment. It is similar to a map building problem but it has an unpredictable goal position. Line 4 means that S_I get positions of obstacles and then the robot remembers their positions. Line 6 to 10 describe that if S_{II} do not get any information about O_{II}, depending on its knowledge base and its previous routes the robot determines its next action. From Line 11 to 14, if S_{II} percepts the signals then it needs to logically judge the surrounding environment and determines the candidates for action depending on the adjacent safe positions. There are two important functions to be designed. For function inference(), we use a DPLL algorithm to deduce the safe positions around the current location, which we will describe in Section 4. For chooseAction(), we propose a search algorithm to make safe and complete action decisions when the environment is unknown and location of the goal is unpredictable in Section 3.

In the second process the objective of the robot is to go back the entrance after it achieves the goal. This procedure can use any classical search algorithms such as A* graph search algorithm (cf. Nilsson [1971]; Russell [2003]) and dynamic A* (D*) (cf. Stentz [1995]), D* Lite (cf. Koenig [2005]).

Procedure Robot ($Sense$) returns $Action$

1. **while** (! Find($Goal$))
2. Update x, y; //the robot's position
3. KB←add($Sense$) ; //update new knowledge Base
4. **if** (S_I in points $(x+J, y+J)$) then $j = J$;
5. **if** ($!S_{II}$) $SafePoints \leftarrow$ put$(x+i, y+i)$ $i = 0, 1, i! = j$;
6. $CandidateAction \leftarrow$ relation$((x+i, y+i)$,$(x, y))$;
7. $Action \leftarrow$ **chooseAction**(KB,$Route(x, y$,$Action)$, $SafePoints$);
8. update.$Route$ $(x, y$,$Action)$;
9. **return** $Action$;
10. **if** (isSafe$((x+i, y+i)$ $i = 0, 1, i! = j$) ← **inference**(KB,$(x+i, y+i)$))
11. $SafePoints \leftarrow$ put$(x+i, y+i)$;
12. $CandidateAction \leftarrow$ relation$((x+i, y+i)$,(x, y);
13. $Action \leftarrow$ **chooseAction**$((x, y)$,$Route$,$SafePoints)$;
14. update.$Route$ $(x, y$,$Action)$;
15. **return** $Action$;
16. $ReturnRoute \leftarrow$ A*-**Graph-Search** ($SafePoints$, $Goal$,$Start$);

3. SEARCH ALGORITHMS IN UNKNOWN ENVIRONMENT

In this section we propose a search algorithm and its refined version to navigate a mobile robot in unknown terrain where the goal location is unpredictable and the locations of some obstacles are determined by inference. Soundness and completeness are main concerns for designing a search algorithm. Soundness means all actions determined by this algorithm are safe, whereas the completeness means that algorithm would find an accessible goal that is located anywhere.

3.1 The Proposed Search Algorithm

To clearly explain the proposed algorithm we impose the grids on the terrain, however the algorithm also can be used to any other geometrical graphs such as the triangles and Voronoi diagrams in Choset [2005]. Fig. 2 illustrates a simple grid world for the terrain and its corresponding graph, with the assumption that the robot can move to

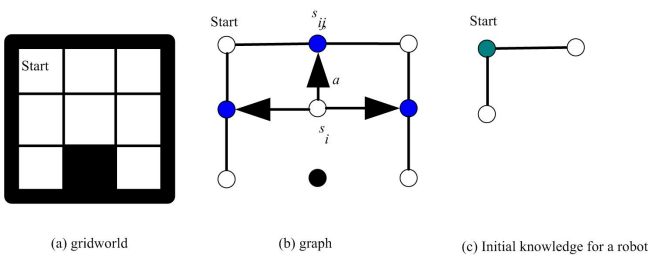

Fig. 2. Illustration of grid world and graph for the search algorithm

any safe one of four adjacent grids with the same cost and it only can sense the information for these four grids.

The algorithm 1 shows the procedure that a robot chooses the action at location s_i depending on its knowledge on terrain. $SafePoint$ is the set of all the known locations that are traversable. First we put the action-location pair $< a_{i,j}, s_{i,j} >$ to the set $Fringe$, where $s_{i,j}$ is accessible by the robot takes $a_{i,j}$ from s_i shown in Fig. 2 (b). In order to guarantee that the algorithm is complete in lower execution cost we put the action-location pair in the fixed order rather than a random order. For instance, in our experiments, we used the action order as "up-left-right-down". Line 2-line 7 show that the robot determines the acceptable action in a given order if the action corresponds to an adjacent safe unvisited location. Line 8 and 9 show that if there is no unvisited safe location for the robot, it will choose to track back to previous route until it finds the new location. $s_{i-(c+1)}$ is the location that the robot moves back and begins to explore the new location where c remembers how far it goes backward.

Algorithm 1. **chooseAction** (s_i, $Route$, $SafePoint$)

1.$Fringe \leftarrow$ put$< a_{i,j}, s_{i,j} >$;

2.while($\neg$ $Fringe$.isEmpty())

3. $< a_{i,j}, s_{i,j} > \leftarrow Fringe.pop()$;

4. if ($s_{i,j} \notin Visited$)

5. $c = 0$;

6. $Route \leftarrow < s_i, a_{i,j} >$;

7. return $Action = a_{i,j}$;

8. $Route \leftarrow < s_{i-(c+1)}, a_{i-(c+1)} >$;

9. return $Action = a_{i-(c+1)}$

3.2 The Completeness of the Proposed Search Algorithm

Theorem 1. The proposed search algorithm is complete; that is, it is guaranteed to achieve any accessible goal.

Proof. The algorithm is complete if the robot can search all accessible locations. Because there are two types of obstacles: O_I and O_{II}, all possible environments can be categorized into two situations. Fig. 3 (a) illustrates the route of the robot in the first situation, where the convex O_{II} (such as B11 and D4) can be avoided and no unnecessary repeat route, whereas for concave O_I (such as $(F4, F5, G5)$) the robot would enter the caves of obstacles and then return along the same way. But when the S_{II}

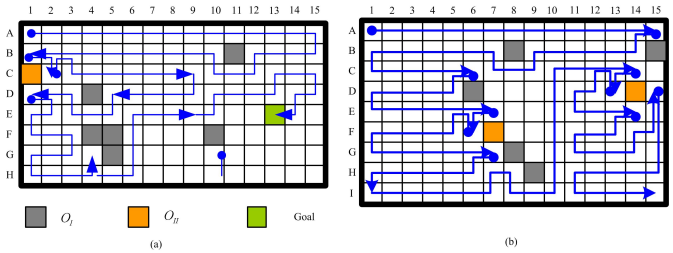

Fig. 3. (a)Routes of the robot without the missed area; (b)Routes of the robot with the missed area in the first round

receives the signals at the current location (such as B1), the O_{II} (C1) may be located at any adjacent position, so the robot returns along the way that it comes. Once it comes back to the safe place (B2), then it explores the certain safe places (C2). These situations are very amenable for the search algorithm, which is complete and effective for finding the goal.

However, Fig. 3 (b) shows the situation in which some safe locations are missed when the robot first passed by them, which are called the missed area. It happens because of the existence of O_{II}. When the robot is located at E7 it senses O_{II} near to it. Because it does not make sure the exact positions of the obstacles so it returns to E6. If the goal is located at this area, the robot has to return back after it finishes the first round for searching all unvisited certain safe terrain. So the algorithm is complete. □

It is necessary to note that in some scenarios there are some locations which the robot would never achieve at, such as C15 in this example. But C15 is not accessible because the robot cannot judge whether C15 is safe. It is caused by the limitation of the sensors but unrelated to the completeness of the algorithm.

3.3 Execution Convergence Cost of the Search Algorithm

We introduce the notion of execution convergence cost to measure the performance of our algorithm in the worst case. Suppose there are n safe locations in a given terrain and the number of O_{II} is m. The execution convergence cost is the sum of execution costs of the actions taken by the robot during the convergence process. The convergence process requires that the robot searches all safe positions.

Theorem 2. The execution convergence cost of the proposed search algorithm is $O(2^m n)$.

Proof. From the proof of theorem 1 we can see that the execution cost of the algorithm increases when the robot has searched most positions in the given environment but the accessible goal is located at the missed area. From Fig. 3 (b) and also refer to the below simulation results we can see that the missed area exists because the robot has no chance to approach all adjacent positions of O_{II} in the first round. This round means the robot travels from an end of the terrain to the other end. In this example two ends are A1 and I15. Moreover each O_{II} corresponds to at most one missed area. No matter how large of its area, once the robot comes back to it will cover all missed safe area. So in the worst case, the goal is located at the m^{th} missed area that the robot searches the area

at last time. In first round the execution cost c_0 is less than n. Because the algorithm drives the robot along the coming way until it finds the unvisited position, in order to visit the first missed area, the number of all searched nodes $c_1 < 2c_0$. Then we get $c_i = 2c_{i-1}, i = \{1, \cdots, m\}$. Therefore the execution convergence cost of the proposed search algorithm is $O(2^m n)$. □

3.4 Refinement of the Search Algorithm

Although the search algorithm is complete, its execution convergence cost is high. Suppose its planning cost is much cheaper than its execution cost and m is large in the environments, so we can refine it. Algorithm 2 describes the procedure of the chooseAction() for this refined algorithm. We set up a data structure MP to store the missed positions whose safety is known by the robot, and $Visited$ stores all visited points. Note that $MP \cap Visited = \emptyset$ and $MP \cup Visited = SafePoint$. After the robot determines $MPoint$, the classical search algorithms such as A* and D* Lite algorithms can be used to navigate the robot from the current position to a missed area or an $MPoint$.

Algorithm 2. **chooseAction** (s_i, $Route$, $SafePoint$)

1. if ($\neg MRoute$.isEmpty())
2. {$Action \leftarrow MRoute.getFirst()$;
3. $MRoute.removeFirst()$;}
4. Fringe $\leftarrow$ put$< a_{i,j}, s_{i,j} >$;
5. $MP.remove(s_i)$;
6. while ($\neg$ Fringe.isEmpty())
7. { $< a_{i,j}, s_{i,j} > \leftarrow Fringe.getFirst()$;
8. $Fringe.remove()$
9. if ($s_{i,j} \notin Visited$)
10. {$Route \leftarrow < s_i, a_{i,j} >$;
11. if ($s_{i,j} \in Fringe \wedge s_{i,j} \notin Visited()$
12. $MP.add(s_{i,j})$
13. return $Action = a_{i,j}$}}
14. if ($\neg MP.isEmpty()$)
15. { $Mpoint \leftarrow$ determp(s_i, MP)
16. $MRoute \leftarrow searchM(recentPoint, Visited, Mpoint)$}

Note that our first proposed search algorithm is the real time planning with the minimal local search algorithm, whereas the refined search algorithm is the planning with the maximal local search algorithm when it returns back to the missed areas.

Theorem 3. The execution convergence cost of the refined search algorithm(Algorithm 2) is $O(n)$ in the worst case.

Algorithm 2 makes the robot effectively find $MPoint$ such that it can return to the missed area with the minimal execution cost. Even in the worst case the execution convergence cost of the refined search algorithm is less than $2n$. However, the data structure MP increases the planning cost that includes the manage cost of data structure MP and local search cost when the robot has to return.

4. LOGICAL REASONING UNDER CONSTRAINT

We use propositional clauses and literals as the knowledge base representation language. For example $S_{x,y}$ denotes that the robot perceives signals at (x, y), then we can reason that there is a kind of O_{II} (denoted by P) in the adjacent positions: i.e. $S_{x,y} \Leftrightarrow P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y}$. If the robot perceives $S_{x,y}$, it needs to judge whether $(x, y+1)$ is safe. This can be regarded as a satisfiability problem (SAT). We take the knowledge base as a formula ($\mathcal{F}$) and put the clause $\neg P_{x,y+1}$(there is no P at location $(x, y+1)$) to the formula. So if $\mathcal{F}$is satisfiable then $\neg P_{x,y+1}$ is true. DPLL algorithm Ouyang [1999] is a complete, highly efficient procedure for solving the SAT problem. But the complete DPLL procedure is too complex for a real time navigation of a robot. Then we introduce the simplified DPLL algorithm for complex environment and also ensure the completeness of DPLL.

Algorithm 3. **Simplified DPLL algorithm**

DPLL($\mathcal{F}$)

if ($\mathcal{F}$includes unit clause $\{\upsilon\}$)

$\mathcal{F} = \mathcal{F}|\upsilon$;

if $\mathcal{F}$ has an empty clause return UNSATISFIABLE;

return SATISFIABLE;

The simplified DPLL algorithm above allows the robot to judge the safety of its surroundings. In this procedure $\mathcal{F}$ includes the knowledge base and the states of surrounding points which the robot deduces. Let υ is a unit clause in$\mathcal{F}$. $\mathcal{F}|\upsilon$ denotes the operation that removes all the clauses that contain υ , deletes $\neg\upsilon$ from all the clauses that contain $\neg\upsilon$ and removes both υ and $\neg\upsilon$ from the list of literals. The empty clause {} is achieved by $\upsilon \wedge \neg\upsilon$, that implies $\mathcal{F}$ is unsatisfiable, otherwise it returns satisfiable.

Theorem 4. Simplified DPLL algorithm is complete and sound.

Proof. : From Ouyang [1999] we know that $\mathcal{F}$ is satisfiable if and only if $\mathcal{F}|\upsilon$ is satisfiable. Suppose $\{\upsilon\} = \{P_{x+1,y}\}$, then only useful knowledge is a set that includes all clauses that contains υ and $\neg\upsilon$, such as $\{\{P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y}\}, \{\neg P_{x+1,y}\}, \{\upsilon\}\}$. If $\upsilon \wedge \neg P_{x+1,y} = \{\}$, then $\mathcal{F}$ is unsatisfiable and the position $(x+1, y)$ is safe. If there is no {} after the operation of $\mathcal{F}|\upsilon$, the robot cannot tell whether $(x+1, y)$ is safe. In order to ensure the safety, we return satisfiable and pass around the location $(x+1, y)$. Therefore this simplified DPLL algorithm is complete. □

Next we describe the memory rules for speeding up planning and reasoning. If the robot remembers all things it used and all the produced knowledge it will become very slow in its thinking and response. Checking the knowledge base of the robot there are two kinds of clauses, the uncertain information such as $\{P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y}\}$

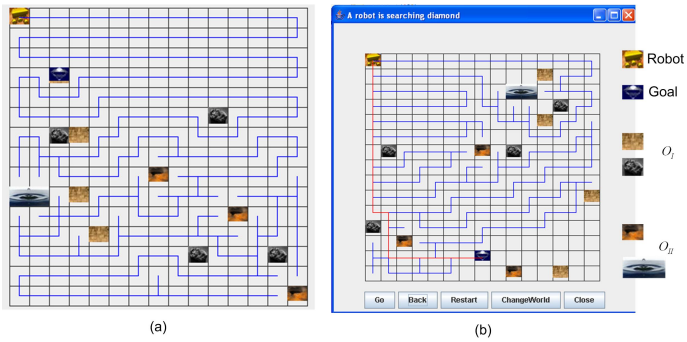

Fig. 4. (a)The completeness of the search algorithm; (b)The navigation of the robot with two types of sensors searching for an unpredictable goal

which means there are at least one P in these four locations and certain information $\{\neg P_{x+1,y}\}$ which means no P at $(x+1, y)$. Using simplified DPLL the uncertain information is only useful in its own inference round. But the robot remembers all certain information but forgets all uncertain information.

5. A SIMULATION EXPERIMENT

We design a computer simulation experiment to show the effectiveness of the action control presented in section 2 and completeness of the search algorithms presented in section 3. And at the same time the simplification of DPLL is verified in a real-time run of program. We suppose a robot is searching a diamond in a cave where rocks and walls belong to the O_I and pits and water areas belong to the O_{II}

5.1 Simulation with proposed algorithm in under built platform

We build the platform in Java shown in Fig. 4 (b) where the environment is produced randomly, it can form different environments using the "Change Environment" button. The "Go" button is pressed to navigate the robot to search the goal, its track shown in blue lines records its actions and how it judges the environment. And back routes showed in red line are designed by A* algorithm to search the home way after the robot finds the goal.

More than a hundred experiments for random produced environment show the completeness of proposed algorithms, i.e. if there is a way to be directed to the goal then it is certain that the robot can find this way. Fig. 4 (a) shows when the goal is inaccessible then the search algorithm moves the robot to all traversable positions.

Fig. 5 demonstration the simulation results to compare the performance the Algorithm 1 and Algorithm 2. The black grids denote the first kind of the obstacles O_I and orange ones are for the second kind of the obstacles O_{II}. The light gray area is the visited area by two algorithms. Their features of two algorithms are summarized as following: Algorithm 2 tends to search more positions in the first round, because it remembered all the known safe position. The Algorithm 2 has good robustness for all kinds of environment. However the Algorithm 1 is relatively competitive when the size of environment is small or the execution cost is not relatively much high to the planning cost.

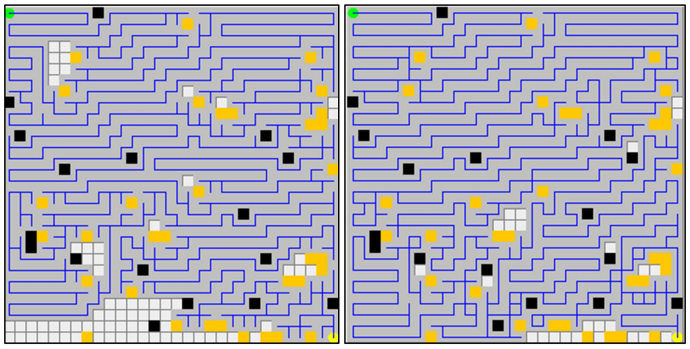

Fig. 5. Simulation results of two algorithms in the same environment. (a)Result from the first version of the proposed algorithm; (b)Result from the refined version of the proposed algorithm

Table 1. A comparison between the features of two algorithms

Algorithms		1st algorithm			Refined algorithm		
		Exe steps	Total time (ms)	Visited Pos	Exe steps	Total time (ms)	Visited Pos
Small Terrain: 15×15 $11 : O_I$; $10 : O_{II}$	High EC	299	6597.3	204	246	5583.3	210
	Marginal EC	310	2103.3	205	232.5	1729.8	209
	Low EC	261	754.7	209	239	1127.7	212
Large Terrain: 30×30 MEC; $30 : O_I$; $40 : O_{II}$	Amenable 65.3%	1067	11579.7	789	973	14656.3	828
	Worse 32.6%	1804	16928	822	982	13546.8	834
	Worst 2.1%	18834	181188	822	930	11719	806
EC: Execution Cost; M:Marginal							

5.2 Comparison of two proposed algorithms

When the size of unknown terrain is large or the sense range of sensors is relatively small to the environment, that is to say that the number of positions becomes large, we compare the Algorithm 1 and Algorithm 2. Fig. 5 shows the demonstration of the simulations where the black grids denote the first kind of the obstacles and orange ones are for the second kind of the obstacles. The light gray area is the visited area by two algorithms. Here we assume the environment is static so we use A* algorithm for function $searchM()$ in the Algorithm 2.

Table I gives the execution steps, total times and visited positions for two algorithms in random produced environments where each value is the average value for more than ten times experiments. The first block of experiments works in a relatively small terrain that is divided by 225 grids,11 O_I and 10 O_{II}. Three rows are for different weights of execution cost relative to the planning cost. The second row is for marginal execution cost which means that in this situation the elapsed times of two algorithms are closed. The second block shows in a lager terrain that

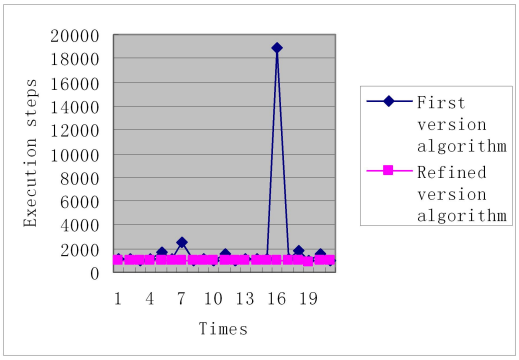

Fig. 6. The execution steps for two algorithms in 21 different large environments

is divided by 900 grids and under the marginal execution cost. The three different kinds of environments are categorized by the performance of the Algorithm 1. The percentage below each environment means the proportion of produced environment. Their features of two algorithms are summarized as following:

A. In general compared with the Algorithm 1, the Algorithm 2 tends to search more positions in the first round, because it remembered all the known safe position. It is shown in Fig. 5 and the column 3 and 6 in Table I.

B. When the size of environment is small and the execution cost is not relatively much high to the planning cost, the Algorithm 1 is relatively competitive than the Algorithm 2 . It is shown by the third row in Table I and the second assumption is reasonable when the computation cost is also expensive for some robots for search mission.

C. The Algorithm 2 is more effective than the first version when execution cost is relatively high to the planning cost. It is shown by first row in Table I.

D. The Algorithm 2 has good robustness for all kinds of environment. In second block data of Table I for three kinds of environment the performance of the Algorithm 2 does not change, whereas the Algorithm 1 has different performance in different environment. It is clearly shown by Fig. 6 in 21 different environments execution steps has small change for the Algorithm 2, whereas the execution steps change largely for the Algorithm 1, even there is one time in the worst situation for the Algorithm 1 the execution steps is very high.

6. CONCLUSION

This paper describes a system for a mobile robot that would be able to explore unknown environments for unpredictable goal with its reasoning. Considering the characteristics of an unpredictable goal position, we have proposed a search algorithm. This search algorithm gives the unvisited position high priority and is complete in an unknown terrain. Moreover we have presented a refined algorithm to decrease the execution cost from $O(2^m n)$ to $O(n)$.

On the other hand, we have applied simplified DPLL algorithm to deduce the environmental information. We have used propositional clauses and literals as the knowledge-base representation language and set the memory rules in order to avoid storage space limitations and to increase the inference speed. We have proven the simplified DPLL algorithm is complete and sound.

7. ACKNOWLEDGE

The authors would like to thank Drs. Rhoda E, and Edmund F. Perozzi for extensive editing and English language assistance.

REFERENCES

R. Korf. Real-time heuristic search. *A*rtif Intell, 42:189-211, 1990

S. Koenig, C.Tovey,Y.Smirnov. Performance bounds for planning in unknown terrain. *A*rtif. Intell, 147:253-279, 2003.

N. Nilsson. Problem-Solving Methods in Artificial Intelligence. McGraw-Hill. 1971

A. Stentz. The focussed D algorithm for real-time replanning. In Proc. Int. Joint Conf. Artificial Intell. 1652-1659.1995

S.Koenig,M.Likhachev. Fast replanning for navigation in unknown terrain. IEEE T Robot [see also IEEE T Robotic Autom]. 21:354- 363.2005

S.Koenig,M.Likhachev. Incremental A*. In: Dietterich, T., Becker, S., Ghahramani, Z. (ed) Advances in Neural Information Processing Systems 14. Cambridge, MA: MIT Press. 2002

S.Koenig. Agent-Centered Search. AI.MAG. 22:109-131.2001

A.Pirzadeh,W.Snyder. A unified solution to coverage and search in explored and unexplored terrains using indirect control. In Proceedings of the International Conference on Robotics and Automation. 2113-2119. 1990

V.Bulitko,G.Lee. Learning in Real-Time Search: A Unifying Framework. J. Artif. Intell. Res. 25:119-157. 2006

H.Choset. Coverage for robotics - A survey of recent results. Ann Math Artif Intel. 31:113-126. 2001

E.Acar,H.Choset,Y.Zhang,M.Schervish Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods. Int. J. Rob. Res. 22:441-466. 2003

D.Conner,A.Greenfield,P.Atkar,A.Rizzi and H.Choset. Paint Deposition Modeling for Trajectory Planning on Automotive Surfaces. IEEE Trans. Autom. Sci. Eng, 2: 381-392. 2005

R.Arkin. Behavior-based robotics. The MIT Press, Cambridge, MA. 1998

S.Russell,P.Norvig. Artificial Intelligence: A Modern Approach. 2nd Edition. Prentice Hall. 2003

H.Choset,K.Lynch,S.Hutchinson,G.Kantor,W.Burgard, L.Kavraki and S.Thurn. Principles of Robot Motion-Theory: Algorithms, Implementation. The MIT Press. 2005

M.Ouyang. Implementations of the DPLL algorithm. Ph.D. Dissertation. Rutgers University. 1999