NAIST-IS-DD0761016

Doctoral Dissertation

High Speed and Efficient Path Planning for Mobile Robot Systems

Yumiko Suzuki

March 24, 2010

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Yumiko Suzuki

Thesis Committee:

Professor Tsukasa Ogasawara	(Supervisor)
Professor Masatsugu Kidode	(Co-supervisor)
Associate Professor Jun Takamatsu	(Co-supervisor)
Professor Takeo Kanade	(Carnegie Mellon University/AIST)
Professor, Satoshi Kagami	(AIST)

High Speed and Efficient Path Planning for Mobile Robot Systems*

Yumiko Suzuki

Abstract

This thesis explores appropriate approaches to planning strategies that can quickly create paths. Real indoor environments are considered to define a metric called the "obstacle rates" that is used as a threshold to select the best planning strategy in a given search space.

First, two types of obstacle rates, grid cell and edge obstacle rates, are defined for evaluating the planned path. Obstacle rates express the map dispersion. In the past, paths have been typically evaluated by path length and planning time. However, in this thesis the planned paths are evaluated based on their dispersion. It was determined that dispersion is useful to set a threshold to select the best planners in a given search space.

A new path planning method, called steering sets, is described, which is used to generate trajectories with smooth changes in direction using predesigned steering sets including three trajectory types. This method was implemented on real car-like robots that performed experiments in an office environment. Assuming knowledge about the predicted trajectories of moving obstacles, such as walking humans, the method of smooth path planning was extended to include moving obstacles. The steering set planner generates smooth paths that are suitable for a wheel-robot running without the need for postplanning and smoothing.

Historically, forward planners iteratively grow the search trees by requiring collision checks at each branching node. A precomputed search tree does not need to check all obstacles in the search space. Instead, it checks them just along

*Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761016, March 24, 2010.

the nodes of paths that reach the goal. Using a precomputed search tree reduces the number of nodes checked for collisions with obstacles. The precomputed search trees are one method of reducing planning time. However, there is a time-memory trade-off. A pruning method is developed that generates an efficient precomputed search tree and allows the creation of a heuristic that effectively tracks back paths. Furthermore, the precomputed search tree quickly plans the optimal path in indoor environment maps. The effectiveness and the efficiency of the precomputation planning based on the experimental results comparing to forward planning are presented.

The results of this research suggest that measures to evaluate path planning depend on map dispersion. The proposed measure, the obstacle rates, provides an efficient guideline in evaluating path strategies and selecting the best plan in the environments.

Keywords:

path planning, speed-up, map dispersion, mobile robot, steering set

ロボットシステムのための経路探索高速化の研究*

鈴木　夢見子

内容梗概

移動ロボットの自律の三要素には認識・計画・制御があり，認識と制御については研究が盛んだが，特に国内では，計画は従来からあるA*のようなForwardに場を探索する最短経路探索を用いた既存の手法が流用されている．Forwardな探索は，解が存在する場合には必ず解が得られるという利点の一方で，探索領域の大きさに応じてチェックするノード数が増大するという欠点があり，探索時間増大の原因となっている．移動ロボットの安定した動作，安全のためのすばやい反応を実現するためには，計画の性能向上が必要であり，経路探索の高速化によって，ロボットシステムに負荷の少ない計画の実現が可能となる．本研究では高速な経路探索手法を構築することを目的とし，環境の分類法を新たに定義し，最適な経路探索手法の選択基準を提案する．

はじめに環境の評価基準を定義する．最適な経路探索手法は環境という場が決まって初めて決定する．しかし二次元地図の最適な空間分割分類法は確立されておらず，経路探索は探索速度または経路長のみで評価されることが一般的であった．本研究では環境の評価基準を定義し，実際の室内環境の地図を用いて算出した障害物の密度およびエッジ長によって地図を分類し，環境の混雑度を評価した．この密度の評価基準は障害物の大きさに依存しないことを実験により明らかにした．またこれらの評価基準によって環境と経路探索の関係を示し，Forwardな探索とPre-Computed Search Tree (PCS)を用いたBackwardの探索手法使用境界を示す．実験によってForwardとBackwardの探索手法切り替えは，探索対象の地図で決定されることを明らかにした．

本研究では二輪駆動ロボットに焦点をあて，二輪駆動ロボットの動きが安全でロボットにとって効率の良い滑らかな経路を生成した．探索時に経路の滑らかさを生み出すために，二輪駆動ロボットの挙動を模した滑らかなカーブを固定

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DD0761016, 2010年3月24日.

形状として持つ探索木をステアリングセットと定義し，この離散的な枝を用いるForward な探索手法を提案し，実際の二輪駆動ロボットに実装しその挙動を検証した．従来から用いられているグリッドベース上の最短経路探索と比較し，提案手法は滑らかで効率のよい走行結果が得られた．また人の動きをとらえ人の歩行を追跡する研究の成果を利用し，人の移動情報から移動障害物の動きが予測可能であると仮定し，動的に変化する環境下におけるステアリングセットを用いた時空間探索手法を提案した．

一方で，Forward な探索では経路として使用しない領域に対しても枝と障害物との衝突チェックを行うため，探索時間が地図の大きさに依存して増大することが知られている．この問題を事前に探索木を計算しておくことにより高速化できることが知られているが，この事前探索木の手法は使用メモリ量の増大するというトレードオフな問題を抱えている．本研究ではメモリ量を増大させることなく事前探索木を生成し，かつ探索の高速化を同時に実現する事前探索木生成手法，および事前探索木による探索法に適した経路取得高速化法を提案する．A*の枝を用いた汎用的な事前探索木を用いて実験を行い，その使用限界を明らかにした．

本研究の探索の高速化によって，制御ループと等速な高速なロボットシステムのループが実現可能となり，新たに定義した評価指標によって，距離だけでない別のコストを元にした経路探索を実現することが可能となる．以上の結果は，経路探索を評価するための有効な基準であり，さまざまな環境に対して最適な経路探索手法を選択する際の有効な指針となる．

キーワード

経路探索，高速化，混雑度，移動ロボット，ステアリングセット

Contents

List of Figures

List of Tables

1. Introduction

With recent advances in robotics, the operational domain of mobile robots has extended into domestic human environments. However, it is difficult to use mobile robots that offer services in human domestic environments because the mobile robots have difficulty with their behavior and travelling speed. Thus, motion planning is a basic and important function for mobile robots that provide a plan of action for the autonomous elements of mobile robots. At present, the techniques for the shortest path search, such as the grid-based A* search, are used for determining the path for mobile robots. To improve performance of the plan is important to achieve stability and fast response of the mobile robot. High speed planning is necessary to reduce the system load aspect of the plan. Therefore, this research purposes to generate a high speed planning method for mobile robots and to define an appropriate metric for path evaluation and map dispersion.

1.1 Problem Description

Forward planning, for example A* and RRT, are used for motion planning. Forward planning obtains a solution when a solution exists but the number of nodes checked for collision with obstacles increases with the size of the search space. One approach to minimize the number of nodes checked for collision with obstacles is to use a “ precomputed search tree. ”Precomputed search trees have been used for speeding animation and CG motion. A forward planner knows the search space for the plan, but, the precomputation planner does not know the search space during the growth of branches. A forward planner checks collision with all the neighbors ’ nodes on the map, while the precomputation planner checks obstacles just on the path. Using a precomputed search tree reduces the number of nodes checked for collision with obstacles. Precomputation planning is faster planning, but there is a time-memory trade-off. For evaluating and using the high-speed performance of a precomputation planner, it necessary to define the evaluation index for planning and to reduce memory usage.

It is proposed that these are the requirements for using a precomputation planner. Thus, it is first necessary to define the evaluation index for planning. Secondly, an effective pruning method that generates a lightweight and better

performance tree is required to reduce the memory usage of precomputation planning. Finally, an efficient backtracking method is developed to reduce repeatedly returning to the starting point.

One of the goals is to find a boundary between forward planning and precomputation planning. To clearly give the limits for forward and precomputation planning, the research is conducted using a computational perspective. The computational results can be applied to engineered certain problems.

The obstacle rate is defined as the dispersion, which expresses some of the difficulties in the planning problem. The difficulties in search spaces consist of dispersion and spatial configuration. The spatial configuration issue revolves around finding narrow spaces. However, it will not be considered in this thesis because it is another area of active research [1–23].

1.2 Related Work

1.2.1 Smooth Path Planning

Using mobile robots in a human living environment requires that robots move smoothly and avoid collisions, particularly with pedestrians.

The A* grid-based search is widely used in path planning to find optimal paths through a search space. The A* search algorithm concept is from artificial intelligence [24] and is typically applied to grid-based map path planning problems. Such path planning approaches using A* or Dijkstra expand the search tree by branching from the current node to all adjacent grid cells, to calculate the absolutely optimal trajectories. However, this approach cannot produce smooth paths because of limitations in the grid-based search [24, 25].

Grid-based search trajectories combine straight lines, whereas the natural trajectory for car-like robots must be smooth and continuous. Different approaches have been tried to derive such trajectories. In one proposal, a navigation path obtained using the A* or Dijkstra algorithm is converted to a path with a cubic curvature polynomial trajectory and then to a fourth-order polynomial if there are obstacles. This approach will yield a smoothly curved trajectory executed under hardware control [26]. However, initial path planning remains in a two-dimensional grid-based space, and the trajectory is not guaranteed to be optimal

and complete. The planning algorithm Field D* [27], where D* represents the Dynamic A* search algorithm, arose from the grid-based planning limitations. Field D* checks all adjacent cells of a current grid cell as in an A* grid-based search, trying to produce smoother discontinuities, although paths are still combinations of straight lines. The fuzzy method [28, 29] is also known to produce paths for car-like motion. The main point of a fuzzy approach producing paths for car-like movement is to classify car trajectories and to use these classes to plan paths, but separate trajectory connections are not guaranteed to be smooth. While searching paths on the map environment, a smooth trajectory can be created, thereby improving planned trajectory quality [30].

1.2.2 Dynamic Path Planning

Dynamic environment path planning, which includes D* [31–33] by Stentz and D*Lite [34] by Koenig *etal.*, which are extensions of A* for changing environments, are suitable for changing environments, but they do not guarantee that moving obstacles are avoided, since the main objective is to speed up planning and not predict the anticipated dynamic obstacle 's movement. Fig. 1 shows the genealogy of path planning in dynamic environments.

Fig. 2 shows the path planning of typical algorithms and the planning requirements.

Pedestrians can be tracked and their position predicted [35] even from a mobile robot [36]. Tsubouchi et al. calculated robot paths for using a space-time search space and predicting multiple obstacle movement [37]. One path plan method created to avoid moving obstacles used a six-dimensional function to smooth the path [36]. By adding a conversion step to the initial planning path, such as smoothing or interpolating, it is possible that the new path may hit moving obstacles. The planner itself must generate a smooth path when planning in changing environments; predicting robot and obstacle movement based on the sensor information requires less reactive replanning and less time for calculating avoidance paths.

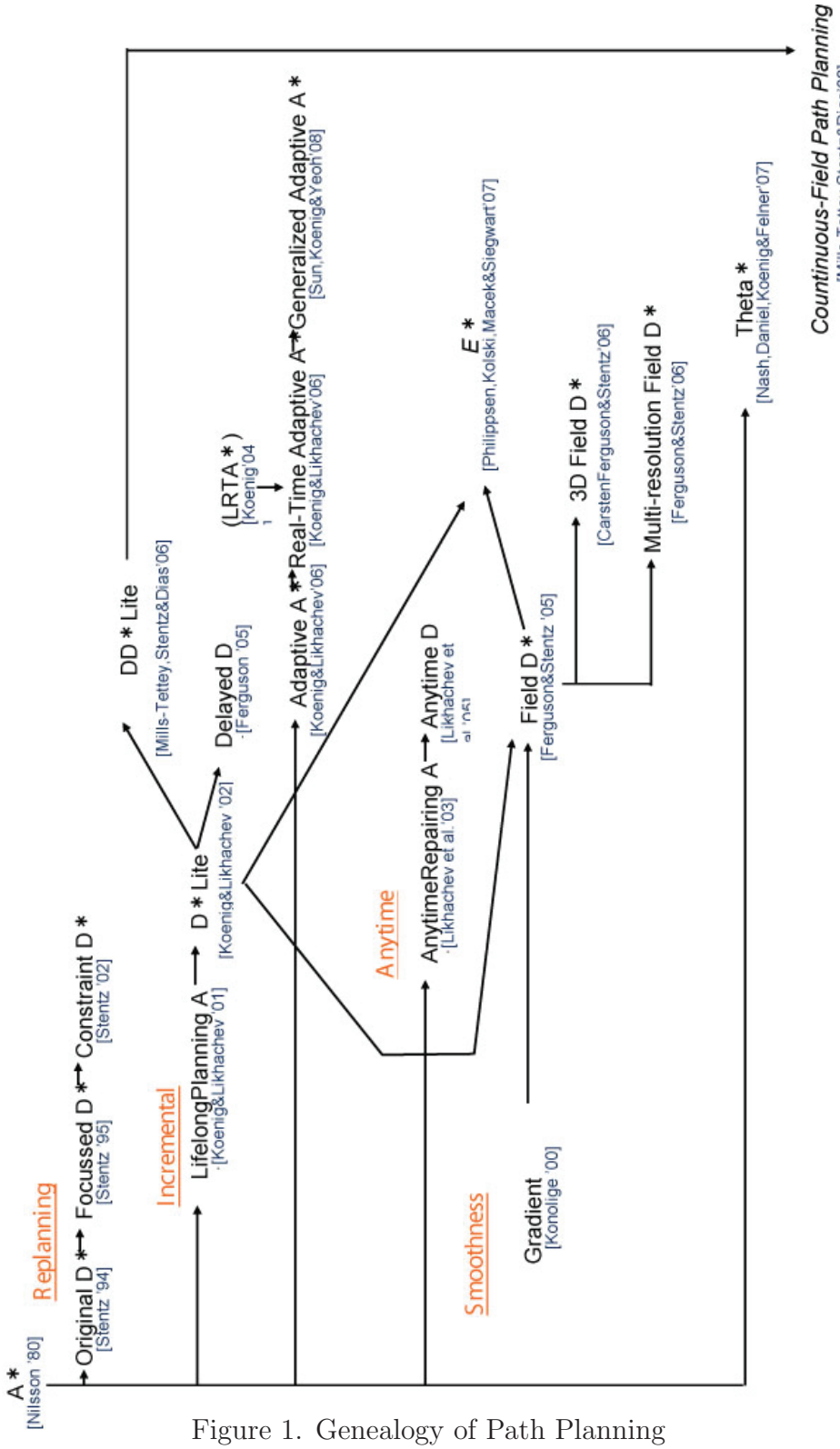


Figure 1. Genealogy of Path Planning

	Anytime	Optimal	Obstacle		Start		Goal		Smooth
			Static	Dynamic	Static	Dynamic	Static	Dynamic	
A*		○	○		○		○		
D*/D* Lite		○		○		○	○		
ARA*	○	(○)	○		○		○		
AnytimeD*	○	(○)		○		○	○		
Field D*		○		○		○	○		○
Requirements				○	○		○		○

Figure 2. Requirements of Path Planning in the Presence of Moving Obstacles

1.2.3 Precomputation Search

Various path planners, such as A* [24], RRT [38], and D* [31], have been used to create robotic path planning that can find the shortest and optimal path through a search space. Forward planners search from a root to leaves and check collisions between obstacles in the search space and current neighbor nodes. Forward planning obtains a solution when a solution exists but the number of nodes checked for collisions with obstacles increases with the size of the search space because the forward planners have to check not only the area near the path that the planner finally obtains, but also all of the search space where the path does not pass through. It is important to reduce this redundancy for improving the path planner.

The preprocessing of the configuration space is used for path planning for robots with many degrees of freedom and for articulated robots moving in static environments, for example [39, 40]. This method uses preprocessing to speed up planning by paying a preprocessing cost. The key idea is that, to precompute a search tree, one can reduce planning time [41, 42]. This was implemented in motion generation for animation [43]. One property of precomputed search trees is that collision checking is not required while growing the nodes.

Lau et al. [43] have an FSM of behavior, jump, crawl, and jog, which is used to precompute the tree (Fig. 3). When a tree is growing, the precomputation planner does not know the map, the start, or the goal. During the find path

phase, the precomputation planner takes the search space, places a root on start, and puts a goal on its own leaves.

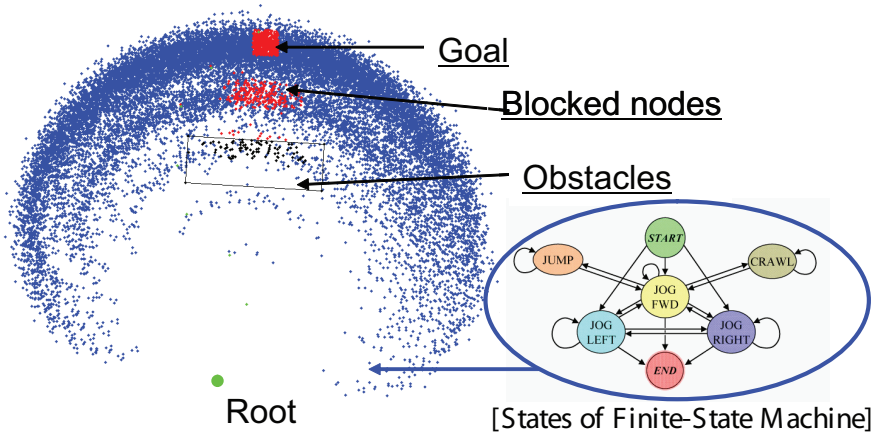


Figure 3. Image of Precomputed Search Tree
Dots are nodes which are grown from a root to leaves

On these leaves' position, many nodes are stacked. In a precomputed search tree, the nodes are sorted in priority queue on each grid cell, and each node has the path from the root. Nodes are sorted in order by shortest path (Fig. 4). Since each node has only one parent, the planner can trace back the nodes to the root to find the node that can reach that point. Hence, each node also represents the path up to and including that point. If a node hits an obstacle, the planner sets a blocked variable, which means that this node and any descendant nodes cannot reach the root node (Fig. 5 (1),(2)). However, the path from the root to the parent node of that point may still be reachable (Fig. 5 (3)). A forward planner checks collisions with all neighbors ' nodes on the map, while the precomputed search tree does not need to check all obstacles in the search space; it checks them just around the path. Using the precomputed search tree reduces the number of nodes checked for collision with obstacles. The precomputed search trees are one of the ways to reduce planning time. However, this method has a time-memory trade-off.

At present, there has not been an investigation of the limitation of the pre-computation planner. For evaluating and using the high-speed performance of the

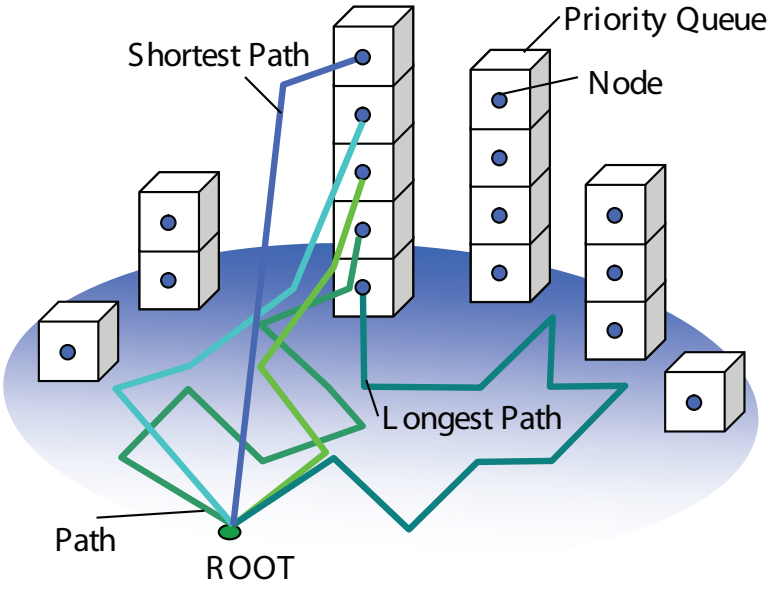


Figure 4. Inside a Precomputation Tree

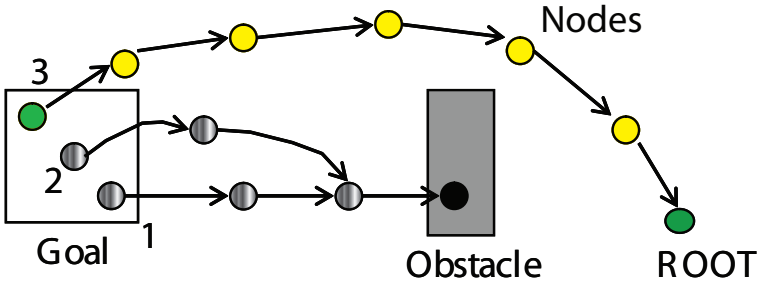


Figure 5. Backtracking to Find Path

precomputation planner, it is necessary to define an evaluation index for planning and to reduce memory usage.

1.3 Thesis Overview

The contributions associated with each of the chapters can be summarized as follows:

Chapter 1: Introduction describes the related works and the approach uses in this thesis.

Chapter 2: Dispersion in Real Indoor Environments describes the obstacle rates as a scale of dispersion and investigates an obstacle rate in real indoor environments.

Chapter 3: Path Planning with Precomputed Search Tree describes a pruning method to generate an efficient precomputed search tree, and proposes a heuristic to effectively trackback a path.

Chapter 4: Generalized Precomputed Search Tree clarifies the generalized precomputed search tree based on accuracy, speed, and optimality.

Chapter 5: Smooth Path Planning With Pedestrian Avoidance for a Wheeled Robot proposes smooth path planning with pedestrian avoidance for a wheeled robot using the steering set method, and demonstrates it experimentally.

Chapter 6: Precomputed Planning with Steering Sets explains the performance of the precomputed planning with steering sets using experiments.

Chapter 7: Conclusions concludes this doctoral dissertation.

2. Dispersion in Real Indoor Environments

On a grid-based map, the grid cell and edge obstacle rates were created for planning evaluation and investigating indoor environmental dispersions.

2.1 Obstacle Rate

Grid cell Obstacle Rate (GR) The grid cell obstacle rate (GR) is determined by the percentage of the obstacle cell number divided by the total cell count for the map, which can be computed as follows:

$$Gridcell\ Obstacle\ Rate[\%] = \frac{Number\ of\ Obstacle\ Cell}{Total\ Cell\ Count\ in\ Map} \times 100 \tag{1}$$

where GR means the obstacle quantity, with a higher GR map containing more obstacles.

Edge Obstacle Rate (ER) The edge obstacle rate (ER) is determined by the percentage of the obstacle edge number divided by the total edge count for the map, which can be computed as follows:

$$Edge\ Obstacle\ Rate[\%] = \frac{Number\ of\ Obstacle\ Edge}{Total\ Edge\ Count\ in\ Map} \times 100 \tag{2}$$

where ER indicates the ratio of connected obstacles, and a smaller ER map contains more linked obstacles (Fig. 6).

2.2 Dispersion in Real Indoor Environments

Fig. 7 shows the grid-based map generated by mapping data using the laser range finder sensor, located on the robot shown in Fig. 8 on one floor of a building. The grid resolution is 10 cm, and the floor map size is 1,310 cells wide by 393 cells long. An office room is enclosed by an orange rectangle, the experimental area (Fig. 9) by a red rectangle, and the living room (Fig. 10) by a green rectangle in Fig. 7.

Table 1 lists the obstacle rates in the maps are calculated by Eq. 1 and 2. The obstacle rate of the living room is the highest in these maps. Thus, the obstacle rates of real indoor environments are less than GR 5% and ER 4.3%.

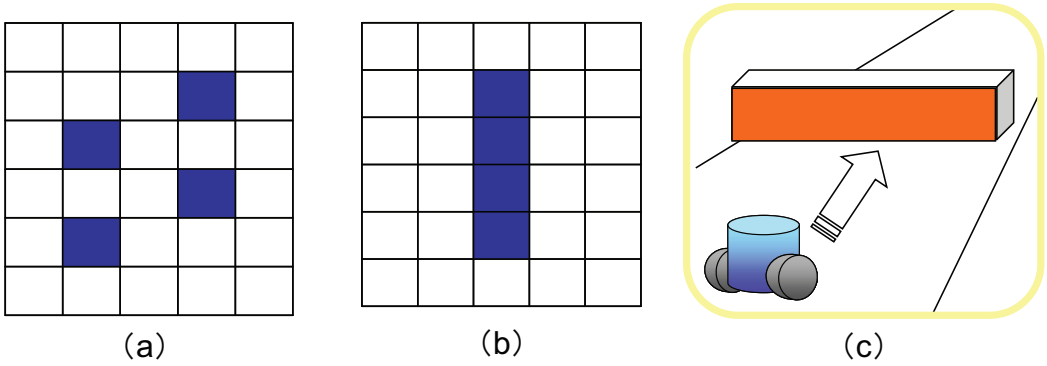


Figure 6. Edge Counts
(a) Not-Linked Obstacles Edge Counts :16 (b) Linked Obstacle Edge Counts: 10. (c) The spaces are divided with linked obstacles, and the possibility of the movement falls.

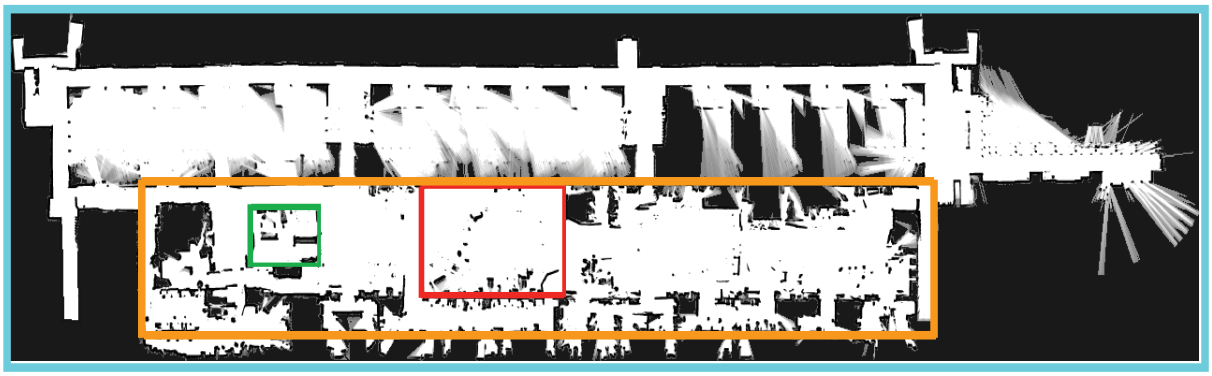


Figure 7. Floor Map
generated by mapping data using a sensor on the robot

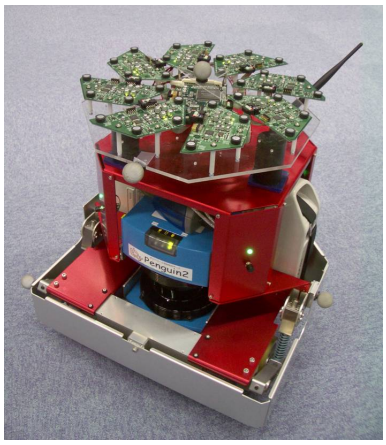


Figure 8. Pen2, wheeled robot with a laser range finder

Table 1. Obstacle Rates in the Indoor Environments

Area	GR [%]	ER [%]	Map Size [cell]
Floor Map	4	3.4	1,310 × 393
Office Room	4.5	3.9	849 × 136
Experimental Area	1.8	1.6	261 × 101
Living Room	5.1	4.3	69 × 49

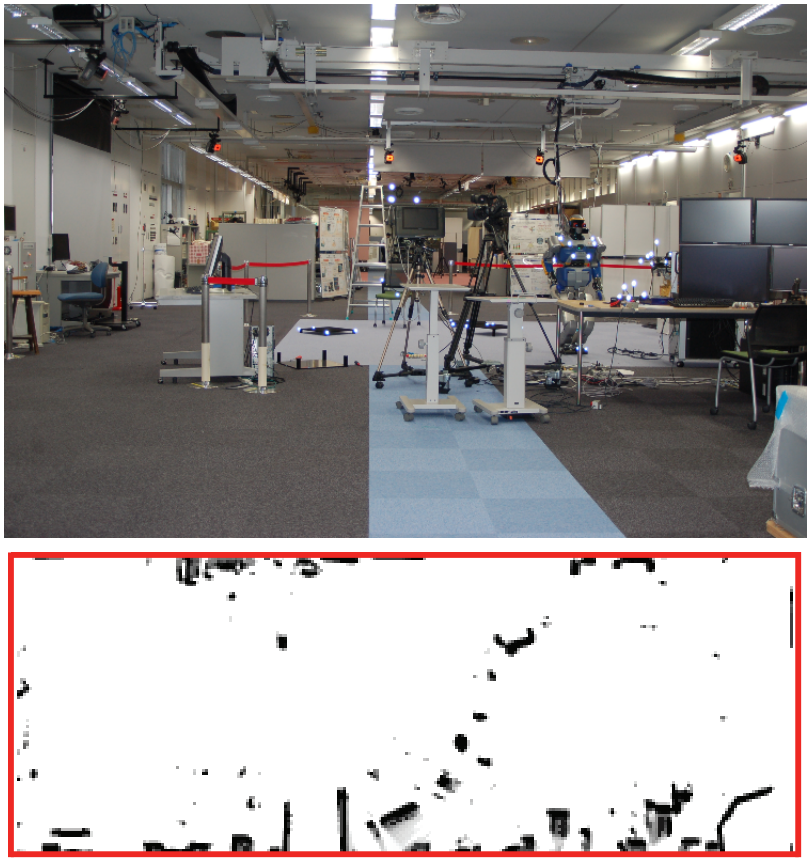


Figure 9. Experimental Area and Its Grid-Based Map

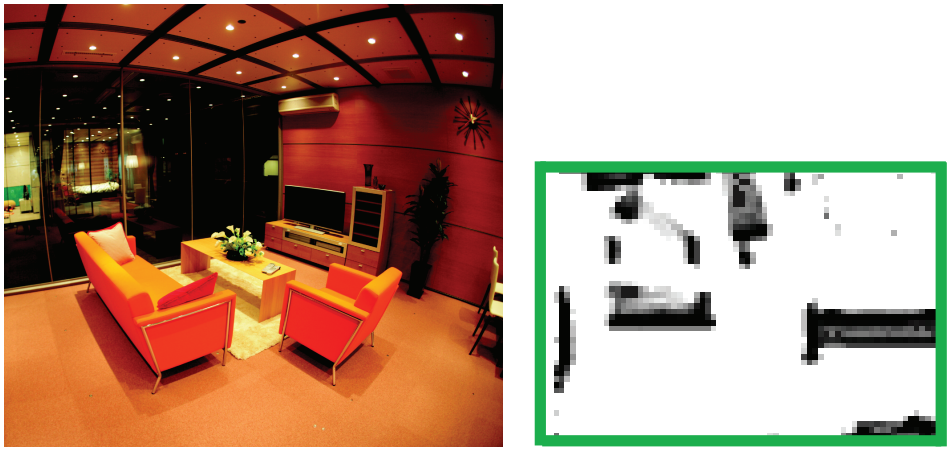


Figure 10. Living Room and Its Grid-Based Map

2.3 Summary

This chapter has presented two obstacle rates for grid-based maps: the grid-cell obstacle rate (GR) and the edge obstacle rate (ER). The GR represents the number of obstacles, while the ER represents the concentration ratio of obstacles. For an indoor environment, the obstacle rates are less than GR 5% and ER 4.3%. The use of these metrics for path planning evaluation is discussed in Section 4.

3. Path Planning with Precomputed Search Tree

This chapter describes a high-speed planning method with compact precomputed search trees using a new pruning method. As well, the effectiveness and the efficiency of the proposed precomputation planning algorithm is considered. The algorithm 's speed is faster than an A* planner in maps in which the obstacle rate is the same as in indoor environments. The precomputed search trees are one way of reducing planning time; however there is a time-memory trade off. The precomputed search tree (PCS) is built with pruning based on constant memory, using the maximum size pruning method (MSP), with a preset pruning ratio. Using MSP, a large precomputed search tree is obtained that has a reasonable size. As well, the node selection strategy (NSS) is applied to MSP to extend the outer edge of the tree and enhance path reachability. Alternate branch backtracking (ABBT) enhances the success rate in crowded environments.

3.1 Pre-Computed Search Tree (PCS)

As was described in section 1.2.3, one property of the precomputed search trees is that collision checking is not required during node growth. Although forward planners check collisions with all neighbor nodes on the map, a precomputed search tree does not need to check all obstacles in the search space. Only those obstacles located around the path need to be checked, which reduces the number of nodes that need to be checked. Precomputed search trees are one of the ideas for reducing planning time; however it has a time-memory trade-off.

An effective pruning method was developed. The precomputed search tree is automatically built using the pruning based on the rule of constant memory. A compact precomputed search tree of reasonable size was obtained using this method. Furthermore, the precomputed search tree quickly plans the optimal path for indoor environment maps. A new strategy was applied to the pruning method to extend the outer edge of the tree and enhance path reachability.

3.2 Maximum Size Pruning Method (MSP)

The new pruning method is called "*maximum size pruning (MSP),*" which is a preset pruning ratio. MSP flow and its image are shown in Fig. 11. Before growing branches, the maximum depth and maximum memory size of each depth are defined. The search tree grows in depth in the first order, and this method uses the memory usage ratio for each depth to control the growing node count at each depth. Random numbers were used to decide which node to prune with this presetting. Reducing the total memory usage, the ratio is set for pruning more accumulated points where the deep depth node counts increase exponentially. Using this preset ratio, it is not necessary to manually reduce redundant nodes after growing the complete branches. With MSP, compact PCS pruning while growing can be achieved.

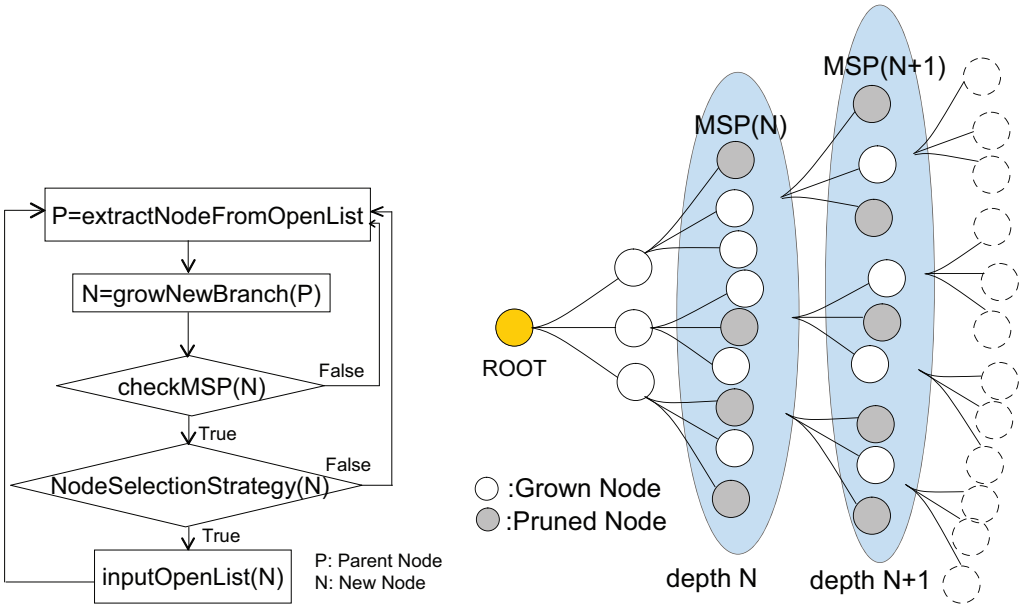


Figure 11. MSP Flow and Image

Random numbers are used to determine which node to prune with presetting. The Mersenne Twister (MT) ([44, 45]) is used as the random number generator in the pruning method. The same seed is given to the MT, a pruned tree is consistently obtained at the same rate and geometry.

In the discussion, the MSP is defined as beginning pruning at the 4th generation, and 20 KB of memory usage at each generation is the ratio of the maximum memory size. One node of this tree occupies 20 bytes of memory, and the tree grows branches for a depth of 40. The length of one branch is about 1 grid cell. One grid cell is 10 cm in a real environment. Fig. 12 (a) shows the configuration of a PCS expanded on a 2-dimensional space, where the z-axis refers to the number of nodes that reach each position of the grid. The shape of the branch is the extended A* [46], which has five branches in one step. The total memory usage of this tree is about 15 MB.

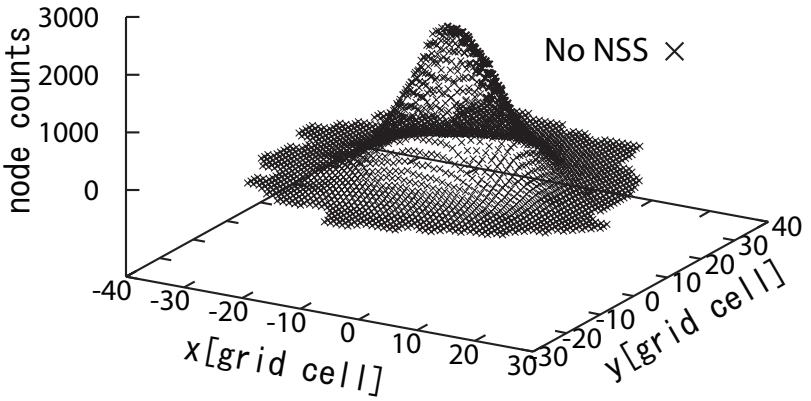
If the tree grows all the branches without pruning, the memory usage for retaining all the node data is 3.6×10^{20} GB. Using MSP, a lightweight tree was successfully obtained that had a size 4.1×10^{-24} times the size of the fully expanded tree.

3.3 Node Selection Strategy (NSS)

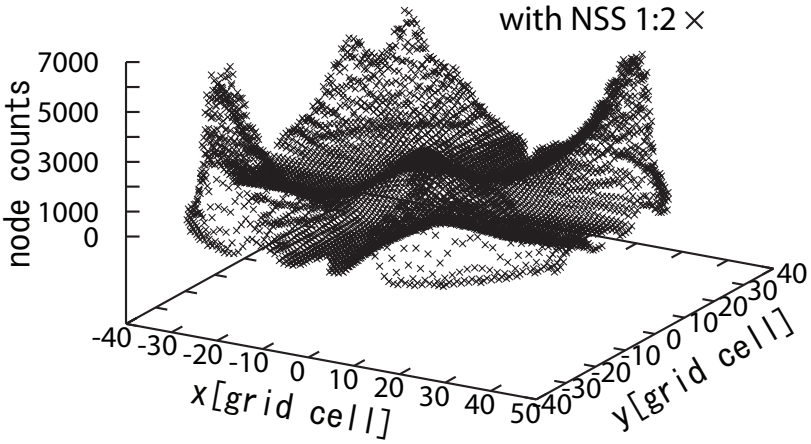
NSS is the node distribution rule based on the distance from the root.

The search branch spreads radially in five directions. Several nodes grown from different parent nodes reach the same position of the grid (Fig. 13). Therefore, the grown tree acquires a heaped central shape (Fig. 12(a)), and the tree starts to extend a branch on the center of the precomputed search tree. Even though it uses the other branches in the grid-based planning, the search branches extend in 4 or 8 directions on the grid map, and the shape of the precomputed search tree will resemble a mountain as shown in Fig. 12(a). Decreasing these turn-around expanding nodes and spreading the amount of nodes on positions far from the root node, The NSS will create a more efficient precomputed search tree.

To improve this wasteful heaping, the Node Selection Strategy (NSS) is introduced, which is a node selection algorithm for putting a node in the open list for distributing turning-around expanding nodes over the outer side of the tree using MSP. An image of NSS is shown in Fig. 14. The NSS's threshold is based on the distance from the root node. The node exchange ratio between pruning and not pruning is preset stochastically in each node generation. The pseudo-code for NSS is shown as Algorithm 1.



(a) using only MSP



(b)using NSS applied to MSP

Figure 12. Precomputed Search Trees with growth 40 deep

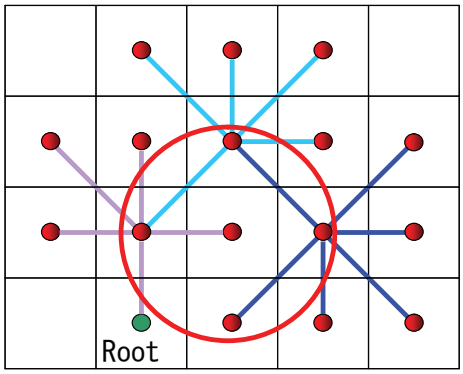


Figure 13. Image of turning-around

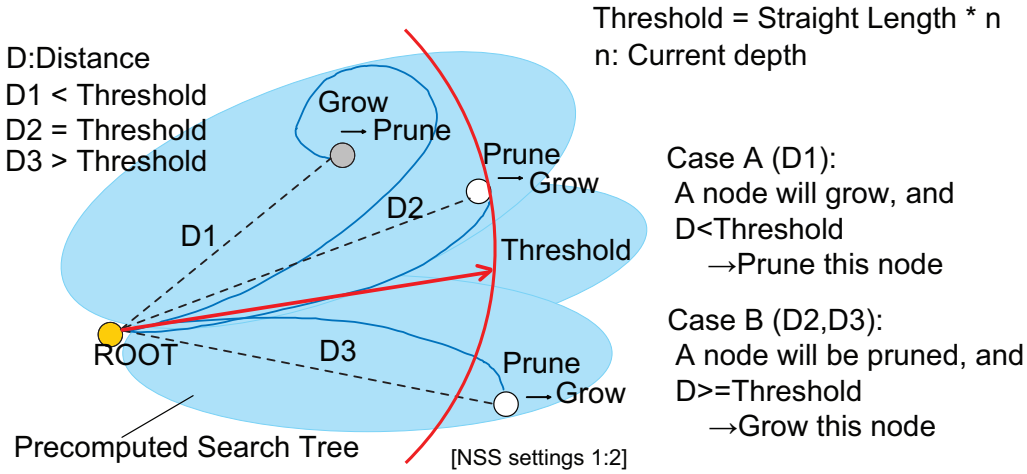


Figure 14. Image of NSS

Algorithm 1: Node Selection Strategy
Performs the Precomputed Search Tree with MSP

Data: n: the nodes of the extending branch
R: the exchange ratio
Result: a Boolean value indicating pruning

Distance = GetDistanceFromRoot(n)
Threshold = n.Generation ∗ n.StraightLength

if StatisticalPruning (n) **then**
  if *Distance* ≥ *Threshold* **then**
    if IsExchange (R) **then**
      else return false;
  else return true;
  end
else
  if *Distance* < *Threshold* **then**
    if IsExchange (R) **then**
      else return true;
  else return false;
  end
end

The node exchange ratio that is used by R in the pseudo-code above can be modified before growing the precomputation tree. An exchange ratio, for example 1:2, means that one node that will turn around is pruned, while two nodes will be grown at the far end from the root node. *Distance* represents the distance from the root node to a node, and *Threshold* is calculated as the straight branch length multiplied by the current generation. In the case when the current node is pruned, if *Distance* is longer than *Threshold* and *IsExchange()* is true, then this node is not pruned. In the case when the current node is not pruned, if Distance is shorter than Threshold and *IsExchange()* is true, the current node is pruned (Fig. 14).

Fig. 12(a) shows the PCS with MSP "not" using NSS, and Fig. 12(b) shows the PCS with MSP using the NSS, with an exchange ratio of 1:2. The effect of

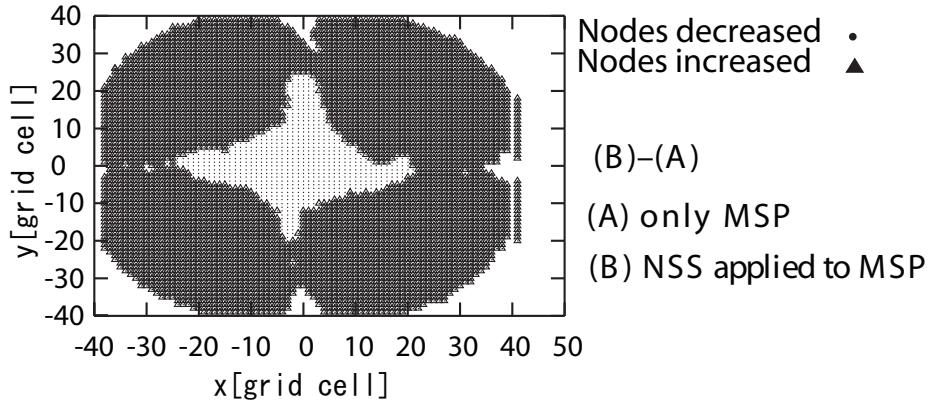


Figure 15. X-Y Plane View Obtained from the Difference in not using NSS and using NSS

NSS changes the form of two parts of the trees. First, in Fig. 12(b), the heap in the center of the PCS using the NSS decreases compared with not using one (Fig. 12(a)), and its maximum height becomes lower when not using the NSS. Second, the tree edge using NSS expands farther than when not using NSS.

Fig. 15 shows an X-Y plane view obtained from the difference in PCS not using NSS (Fig. 12(a)) and PCS using NSS (Fig. 12(b)). Dark colors show an area with more nodes compared to not using NSS, and bright colors show an area with fewer nodes. When NSS is applied to MSP, it cuts down the node count in the center, but enlarges the tree on the outside and outer nodes. With the even exchange ratio used by NSS and the 5 factors of one branch, memory usage increases, in the case of Fig. 12(b), where its size is 117 MB. Using NSS, the number of the nodes in the central heap decreases by about 35,000 nodes, and the distance from the root node reaches about 19 grid cells at the maximum. The maximum increase in nodes at the edge of the tree is about 5,000.

3.4 Alternate Branch Backtracking (ABBT)

In a precomputed search tree, many nodes are stacked on these leaves' position, and nodes are sorted in a priority queue for each grid cell, and each node has the path from the root (Fig. 4). To find paths, the precomputation planner traces back a goal node on these leaves to the root.

In the paper [43], the path finding algorithm takes a goal position as input, and returns the tree node that represents the solution path. As they trace back towards the root (Fig. 16 Top), they mark each node with the blocked flag, if it is not already blocked or not obstructed by an obstacle (the gray circle in Fig. 16 Top). One position on the tree has the list of sorted nodes towards the root node. If the node hits an obstacle (1 in Fig. 16 Top), they set the blocked flag to all tracked back nodes and select the next node to track in the list of the goal (2 and 3 in Fig. 16 Top). However, this procedure is very time consuming, if the tree has a massive number of sorted nodes for each node position.

The "*Alternate Branch Backtracking (ABBT)*" is an effective heuristic in the environment of few local minima. This heuristic reduces the repetition of restarting from the goal nodes and resolves the time consuming problem. ABBT uses a traced path that is cut down by an obstacle. If the node hits an obstacle (1 in Fig. 16 Bottom), the planner gets the next tracking node from the list of neighbor nodes, which are predecessors of the obstructed node (2 in Fig. 16 Bottom). ABBT picks up a node from a queue on the traced path, and it cuts down on the repetition of returning to the list of the goal for selecting a next node to start again.

3.5 Summary

A pruning method was presented to generate the compact precomputed search trees. The precomputed search tree is built with pruning based on the rule of constant memory and the maximum size pruning method (MSP) with a preset pruning ratio. Using MSP, a large precomputed search tree with a reasonable memory size is obtained. Furthermore, by applying the node selection strategy (NSS) to MSP, the outer edge of the tree is extended, leading to enhanced path reachability. Alternate Branch Backtracking (ABBT) enhances the success rate in crowded environments.

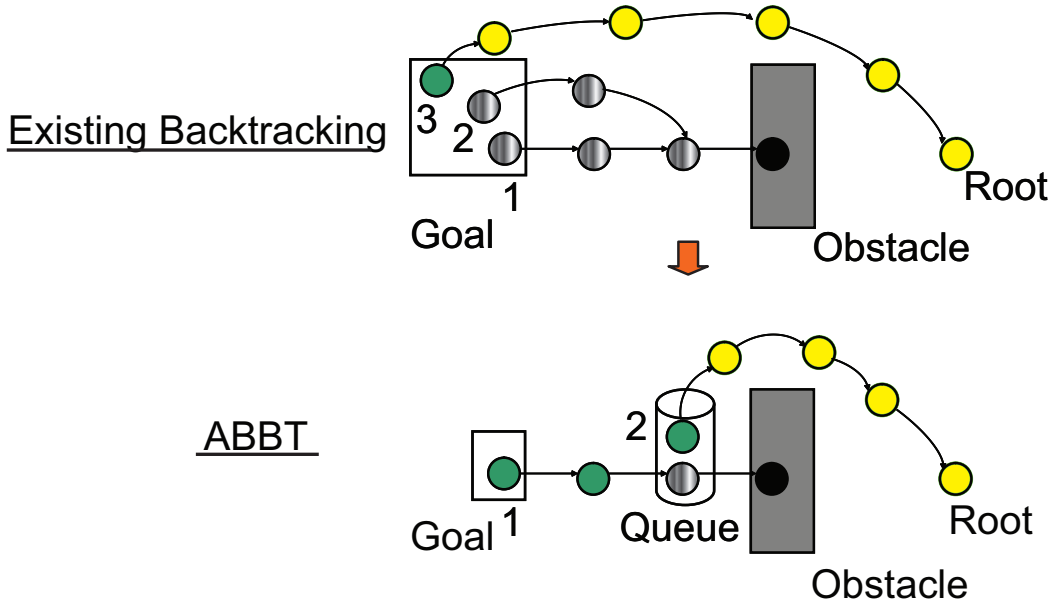


Figure 16. Alternate Branch Backtracking

In the backtracking method (Bottom), the planner selects the next node from the list of neighboring nodes that are predecessors of the obstructed node of an obstacle when starting again.

4. Generalized Precomputed Search Tree

This chapter explains the generalized precomputed search tree based on accuracy, speed, and optimality.

4.1 Precomputed A*

This section describes the branches of the extended A* planner that have grown to 5 neighboring grid cells to create the precomputed search tree. A* grid-based search is a commonly used algorithm for path planning that can find the optimal path through a search space.

The exhaustive growing of branch nodes will consume time exponentially if collision checking is simultaneously performed and will use a large amount of memory to store this gigantic tree. The proposed algorithm will reduce the required time for collision checking by growing branches using a precomputed search tree.

A precomputed search tree using path planning with an efficient version of extended A* grid-based search [46] is used. The extended A*, which is a simple technique, is used to speed up optimal path planning on Euclidean-cost grids and lattices. The technique applies to grids and lattices with edges between diagonally-adjacent grid points whose relative costs obey the triangle inequality. Then, the 5 neighbors branch (Fig. 17) based on the extended A* planner can be used to create the precomputed search tree.

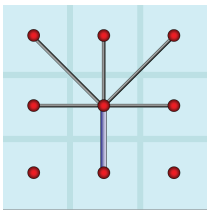


Figure 17. Branch Set of the Extended A*

These 5 neighboring branches based on the extended A* planner are used to create the precomputed search tree

A tree of the A* search is built up to a certain depth. If the complete tree

is built, its size increases exponentially, and its memory usage exceeds the upper limit of the physical memory. The total number of tree nodes is expressed by Equation (3) with depth level d.

$$The\ total\ number\ of\ nodes = \sum_{k=0}^{d} BranchFactorCount^{k} \tag{3}$$

Memory usage is given by multiplying the total number of nodes by node size. This can be used to prune nodes. Further information on the pruning method is described in Chapter 3. After building the tree, the start position is set in the environment map on the root node of the tree, and the goal position is set on a branch node that is in a geometrical relation from the root node. Each node of the precomputed search tree has a position, link to a parent node, state flag, and cost for path length up to that node starting from the root node. Each node has only one parent, so that the precomputed planner can trace back the path to the root node. The tree map manages the precomputed trees and the nodes that came from every direction. When backtracking the nodes, the planner checks for collision between a node and the obstacles on the environmental map.

4.2 Experimental Settings

This section describes the experimental results to verify the accuracy of the A* planner using PCS and the effect of MSP, NSS, and ABBT. As well, the speed-up and path optimality of the precomputation planning is compared with an A* planner.

For the experimental simulation, a PCS is created with 40 generations. The root node is placed in the center of the map, and the tree grows from A*, which has five branches. The length of the one branch is about 1 gridcell. PCS applied to prune processing is based on the predefined tree limit of 20 KB for each generation after the fourth generation. The experimental map is a square map of 55 grids on one side that reflects a real living room in size (see, Fig. 18). The map includes static grid obstacles. The number of obstacles on the map was based on the predefined ratio to the environmental map size. Experimental maps with obstacle ratios from 0% to 31% were prepared based on the dispersion ratios determined in

Section 2. The obstacles were randomly scattered on the map. The start position for planning was specified as the origin point of the experimental map, and 700 goal positions were randomly selected on the map.

The CPU used for these experiments was the Intel(R) Core2 X6800 2.93 GHz with 4 GB of memory running Linux. All of the experiments below were executed with the same hardware and software.

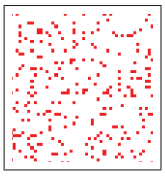


Figure 18. Random Number Map with GR 10%
a square map with 55 grids on one side

4.3 Accuracy Limit of Precomputed Search Tree

Figures 19, 20, 21, and 22 show the results of path planning using PCS as a function of the percentage of goals reached of the randomly assigned goals. This ratio is called the "*success rate*". In these figures, the vertical axis is the obstacle rate (GR), and the horizontal axis is the distance from the start node. The success rate is shown by a change in color. Bright yellow denotes a success rate of 100%. Darker shades of yellow indicate a smaller success rate. The light blue vertical line drawn perpendicular to the horizontal axis indicates the position of the start of pruning, which is based on the MSP settings.

4.3.1 Success Rate using MSP

Figure 19 shows the execution of a PCS that was pruned using MSP. Even though the precomputed tree is incomplete after a depth of 4, the planner reliably generates a path, but the success rate drastically decreases in the case of a high obstacle rate. The inefficient spread of nodes caused by pruning using MSP decreases the number of nodes far from the root node. Collisions with obstacles increase when

the obstacle rate is high. Thus, the success rate of the PCS using only MSP is affected by the increase in the obstacle rate.

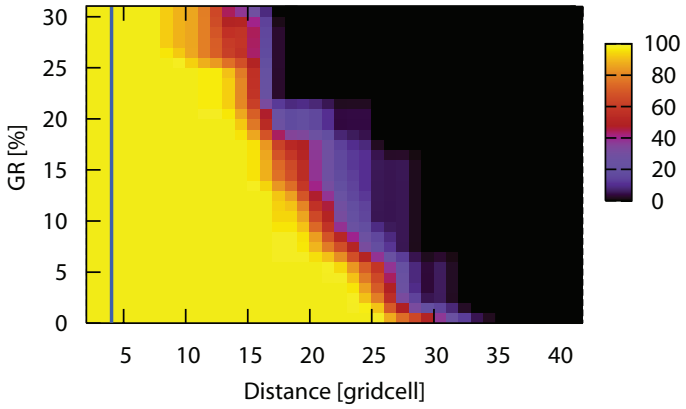


Figure 19. Success Rate of the Precomputation Planning using MSP

4.3.2 Improved Success Rate: Effect of ABBT

Figure 20 shows the execution of a PCS that was pruned using MSP and used the proposed tracking method ABBT. Comparing the success rate to the case of using only MSP (Fig. 19), the new method is improved, especially for the case where the distance is 27 or less grids from the root position and the obstacle rate is high.

Thus, it can be concluded that the ABBT enhances the success rate in crowded environments. As Fig. 20 shows, the success rate decreases in the case of a large distance. For distances greater than 35 grids, the planner could not find a path because of the inefficient spread of nodes.

4.3.3 Improved Success Rate: Effect of NSS

Figure 21 shows the execution of a PCS that was pruned using NSS applied to MSP. In this experiment, the exchange ratio of NSS was 1:2. The other settings, such as goal counts and maps, are the same as above. Comparing the success

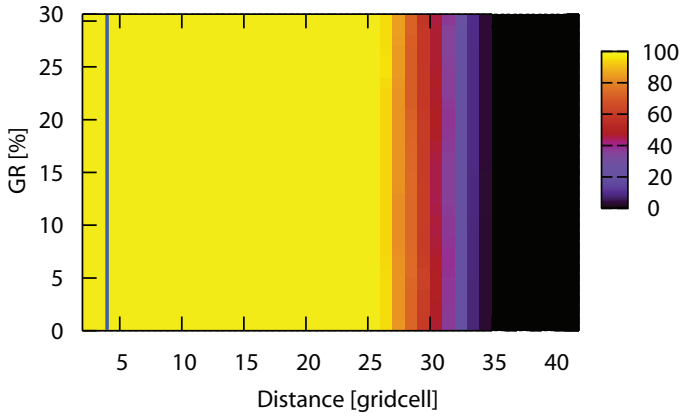


Figure 20. Success Rate of the Precomputation Planning using MSP with ABBT

rate using NSS (Fig. 21) with the rate when not using NSS (Fig. 19), the success rate with NSS is close to 100% for GR less than 12% of the obstacles rate. When GR is greater than 13%, the success rate with NSS is much better than for the case when NSS is not used and the success rate is close to 0Therefore, applying NSS to MSP can significantly improve the performance of the precomputation planning in regions far from the root node.

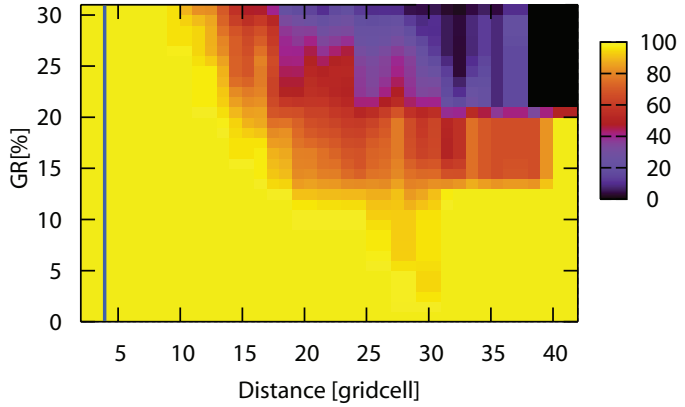


Figure 21. Success Rate of the Precomputation Planning using NSS Applied to MSP

4.3.4 Improved Success Rate: Effect of NSS and ABBT

Figure 22 shows the execution of a PCS that was pruned using NSS and ABBT applied to MSP. Comparing the success rate using NSS and ABBT applied to MSP (Fig. 22) with the rate when neither NSS nor ABBT was used (Fig. 19), at distances greater than 27 grids in the Fig. 22, the color of the success rate is higher than when not using NSS. Using NSS improves the success rate, especially when the obstacle rate is high and the start is far from the goal. The success rate using NSS is 100% for GR from 0% to 31% of obstacles coverage at a distance of more than 27 grids, whereas the corresponding success rate is 0% when NSS is not used.

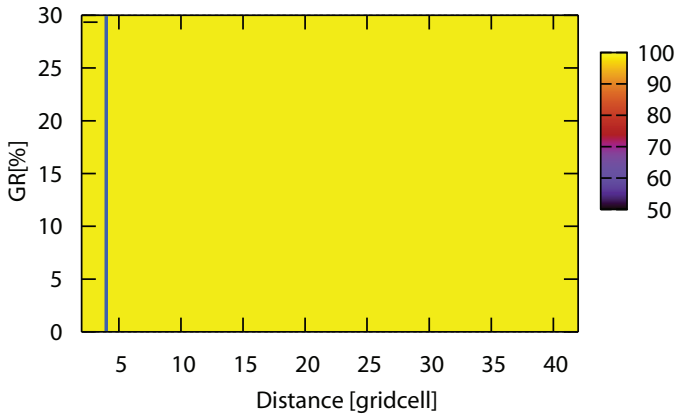


Figure 22. Success Rate of the Precomputation Planning using NSS applied to MSP and ABBT

4.3.5 Large Map Situations

Figure 23 shows the success rate of PCS applied to a larger map than the tree size. The experimental map is a square map of 65 grids on one side. The experimental settings are the same as in Section 4.2, and PCS was pruned using NSS and ABBT applied to MSP. The precomputation planner found paths in the search area that were covered with its own tree, while the planner could not find paths when GR was greater than 36% of the obstacle rate.

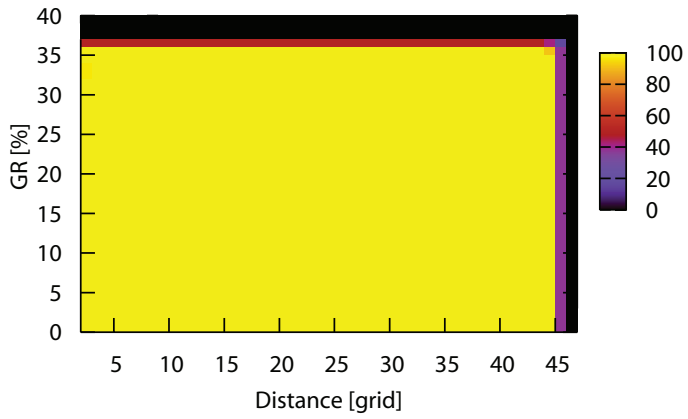


Figure 23. Success Rate of PCS applied to a Large Map

4.4 Sped-Up Search

This section considers techniques to reduce the search time for precomputation planning. Figure 24 shows the extracted results of the planning time from the experiment described in Section 6.4 and compares then to the runtime of the path planning without any precomputational techniques. The graph is a semilog plot of the planner runtime with the vertical axis being the planning time in milliseconds and the horizontal axis being the goal distance. The red and pink lines show the runtime of the A* planning when GR is between 0% and 31% of the obstacles rate in a map. The other lines show the precomputed planner runtime when GR is 0%, 5%, 10%, 20%, 30%, and 31% of the obstacle rate. The runtime of the precomputed planners for GR less than or equal to 30% is constantly below the value of A* planning. In these cases, the precomputation planner runtime can increase with distance and obstacle rate. Thus, it was found that the planning time using the proposed searching method is reduced considerably compared to the simple A* planning that does not use precomputed trees, as shown in Fig. 24. For maps with GR less than 20% of the obstacle rate, the runtime of the precomputation planning is more than one order of magnitude faster than an A* planning method without precomputation.

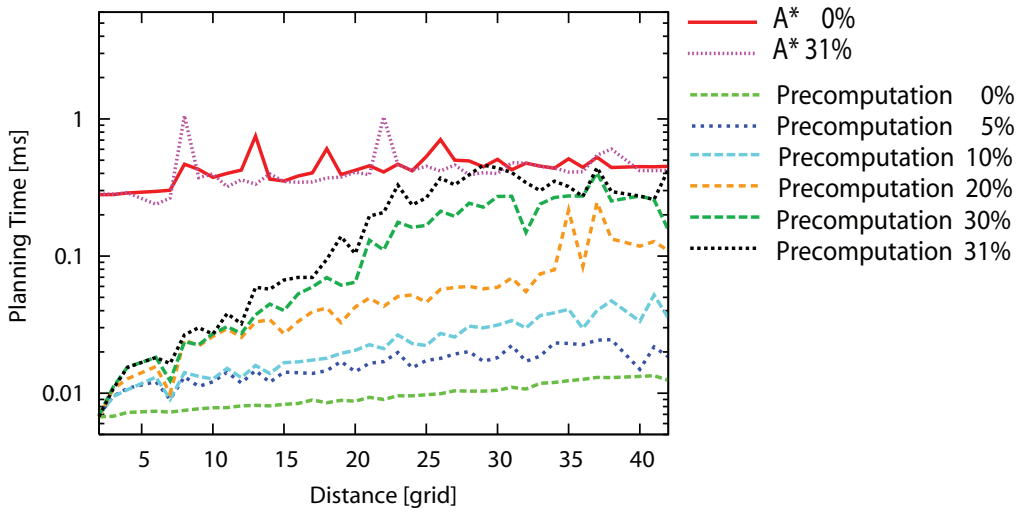


Figure 24. Planning Runtime

4.5 Path Optimality

This section describes an evaluation of optimality of the paths that are provided by precomputation planning. Optimality is defined by the percentage of the path length of the precomputation planning as compared to the path length of the existing planning methods. The extended A* planning [46] produces the shortest path, and, hence, this method is used as the reference value when the optimality values are calculated. The experimental setting is the same as in Section 4.2 with 700 random goals. Figure 25 shows the path optimality for every obstacle rate from GR 0% to 30% on the experimental maps. The horizontal axis is the obstacle rate, and the vertical axis is path optimality. The blue dot is the average path optimality, and the dotted line is the standard deviation. The red dot is the mode value.

The average optimality shown in Fig. 25 shows that the precomputation planning finds a path that is the same in length as the shortest path on the maps with GR less than 25% of the obstacle rate. On maps with a GR from 26% to less than GR 29% of the obstacle rate, the proposed finds a path that is on average less than 115% of the length of the shortest path.

The mode value is always 100% of the length of the shortest path for all trials.

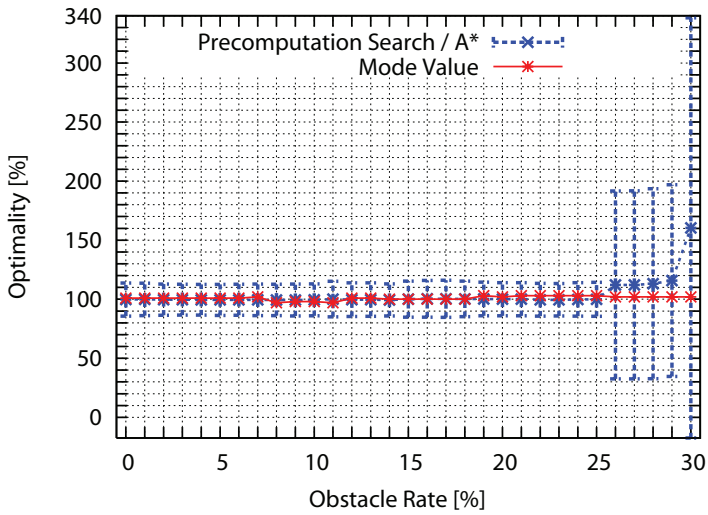


Figure 25. Path Optimality for the Precomputation Planning with A* Planner

4.6 Obstacle Size Differences

Obstacle size effects are validated with precomputation planning. The experimental maps with length 55 grids include square obstacles, whose lengths ranged from 1, 5, 10, 15, and 20 grids. The map's obstacle rate varied from GR 0% to 30%. PCS without NSS plans paths from a map center start to random 700 goals.

Success Rate Figure 27 shows the success rate for different-sized square obstacles. The success rates for (b), (c), (d), and (e) are the same as for a 1-grid square obstacle's planning result, as shown in Fig. 26 (a). Thus, PCS's accuracy is only slightly affected by differences in the square obstacles ' sizes.

Search Speed Figure 29 shows the planning time results of experiments for different-sized square obstacles. Planning times for (a), (b), (c), (d), and (e) in Fig. 28 and 29 are faster than the simple A* planning results. PCS's planning time is more than one order of magnitude faster than A* planning plans with different-sized square obstacle maps.

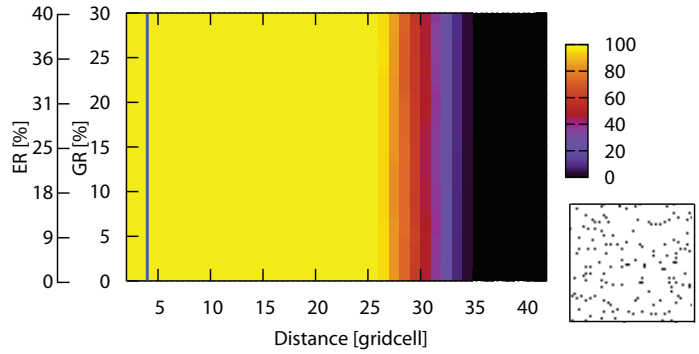


Figure 26. Success Rates for Different Square Obstacle Sizes

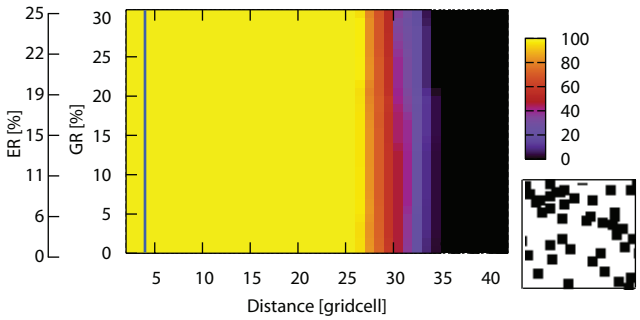
(a) 1-grid square obstacles map using PCS without NSS, the GR 30% maps are shown in the bottom right.

4.7 Obstacle Shape Differences

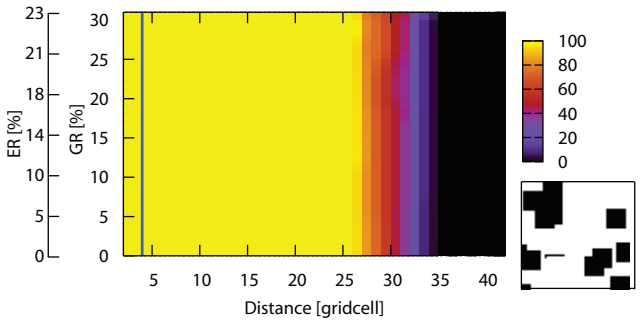
The effect of differences in the shape of an obstacle are considered. Square experimental maps with length 55 grids with linear obstacles and square shapes with lengths of 1, 5, 10, 15, and 20 girds strewn longitudinally crosswise and obliquely on the grids were used. The map's obstacle rate varied from GR 0% to 30%. PCS without NSS plans paths from a map center start to 700 random goals.

Success Rate Figure 30 shows the success rates of the experimental results with linear obstacles. The success rates for (b), (c), and (d) are lower than for a 1-by-5 grid linear obstacle. Longer obstacles cut more PCS branches in one sitting, and thus PCS's accuracy is affected more severely by differences in the linear obstacles. PCS finds paths in an area covered with its own tree and a GR of 5%, with a success rate of 100%.

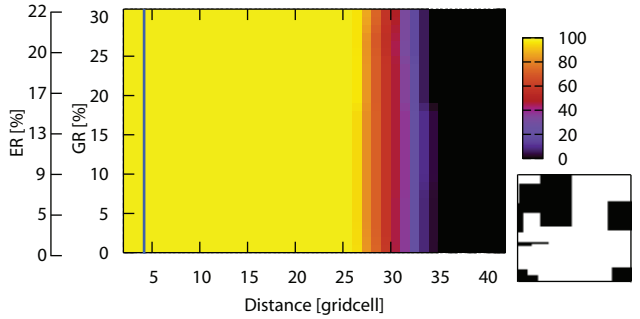
Search Speed Figure 31 shows the planning time results of experiments with differences in linear obstacles. Paths with an ER of 5% of the planning time in (a), (b), (c), and (d) are more than one order of magnitude faster than A* planning plans with square obstacles with different sizes. The planning time increases for longer obstacles. The separated PCS branch connection with longer obstacles



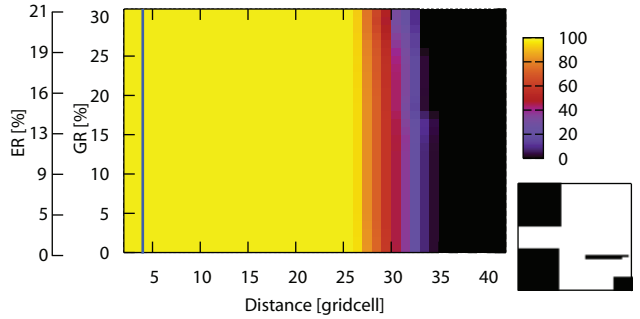
(b) 5-grid square obstacles



(c) 10-grid square obstacles



(d) 15-grid square obstacles



(e) 20-grid square obstacles

Figure 27. Success Rates for Different Square Obstacle Sizes using PCS without NSS, the GR 30% maps are shown in the bottom right.

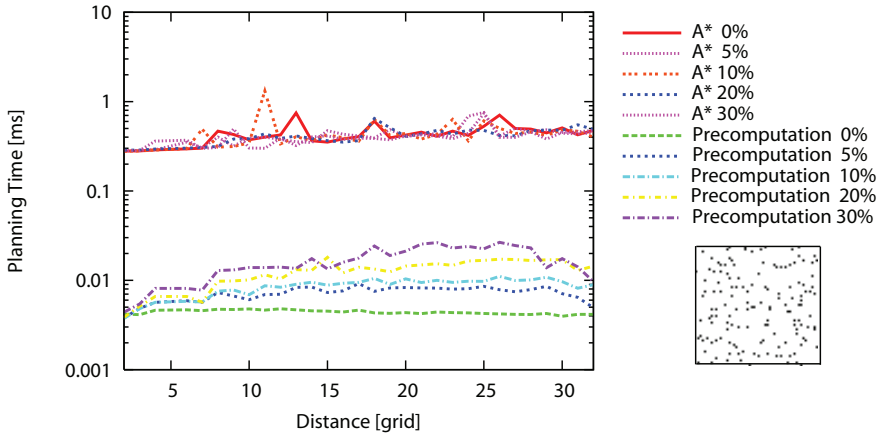


Figure 28. Planning Times for Different-Sized Obstacles

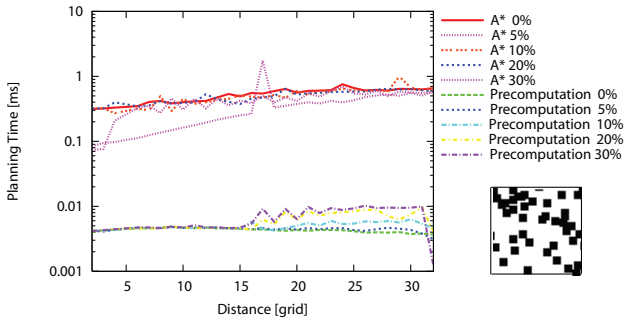
(a) 1-grid Square Obstacles using PCS without NSS, GR 30% maps are shown in the bottom right.

causes reselecting node counts to increase. PCS planning time is affected by differences in linear obstacles.

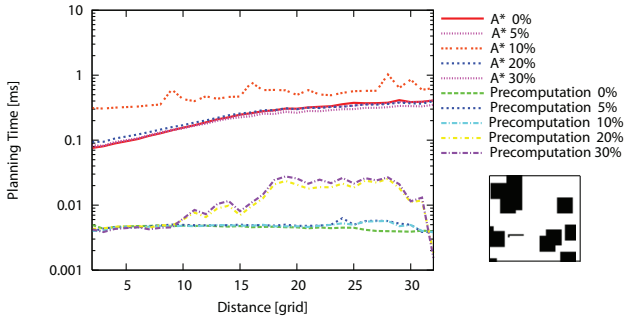
4.8 Edge - Gridcell Obstacle Rates and Real Indoor Environments

Figure 32 shows the ER and GR obstacle rates for each map in Section 4.6 and 4.7. Increasing obstacle concentration in the maps causes the ER - GR gradient to descend more quickly, and, hence, the lower the ER in the same GR values includes more linked obstacles.

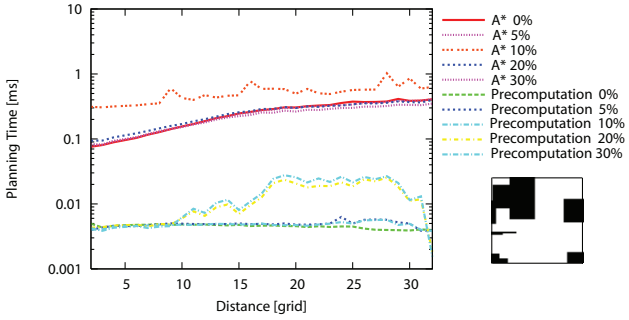
In Section 2, GR and ER in real indoor environments were shown in Fig. 32. The dispersion in real indoor environments approaches that of the linear obstacles. According to the results shown in Fig. 31, the PCS speed limitation in this experiment occurs when ER equals 14 %. However, an ER of 14% is much higher than for most indoor environments, as can be seen from Fig. 32. Thus, the PCS is fast enough for real indoor environments.



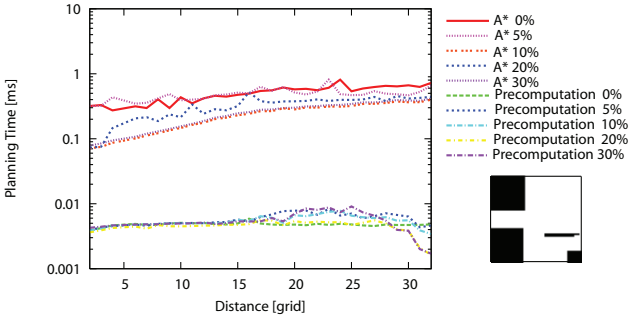
(b) 5-grid square obstacles



(c) 10-grid square obstacles

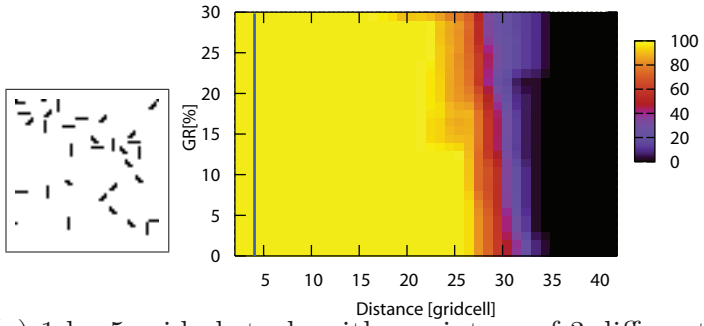


(d) 15-grid square obstacles

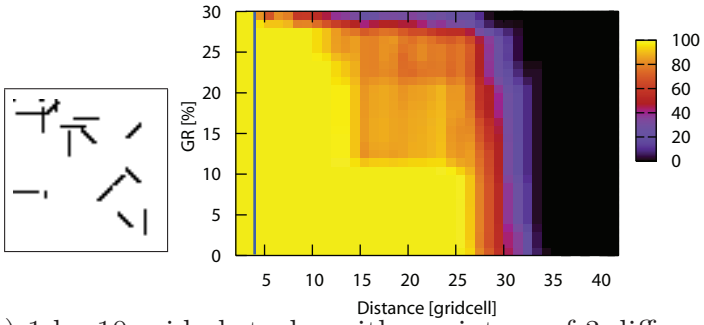


(e) 20-grid square obstacles

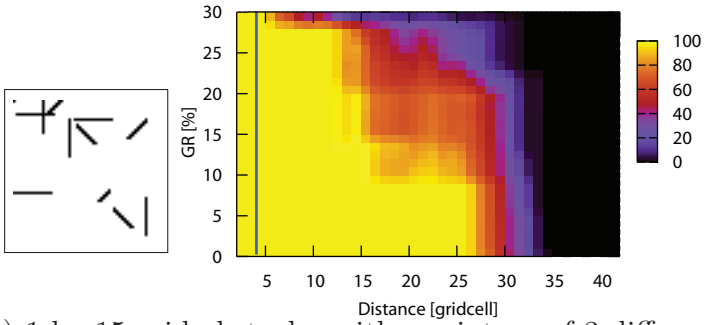
Figure 29. Planning Times for Different-Sized Obstacles using PCS without NSS, GR 30% maps are shown in the bottom right.



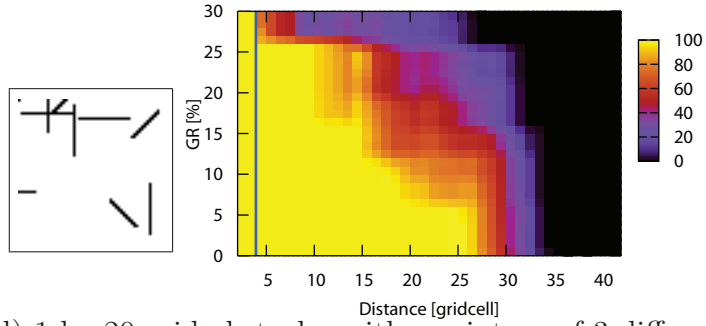
(a) 1-by-5 grid obstacle with a mixture of 3 different types



(b) 1-by-10 grid obstacles with a mixture of 3 different types



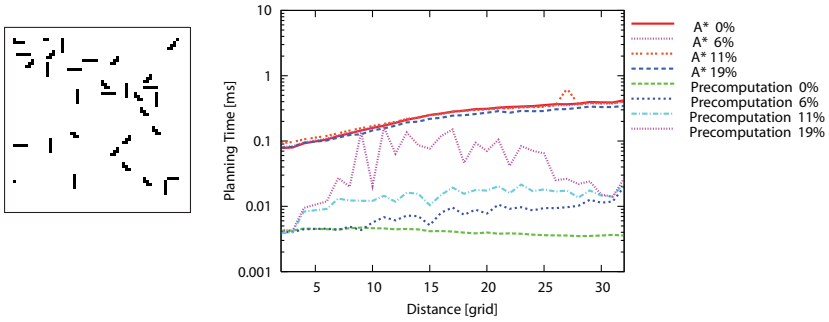
(c) 1-by-15 grid obstacles with a mixture of 3 different types



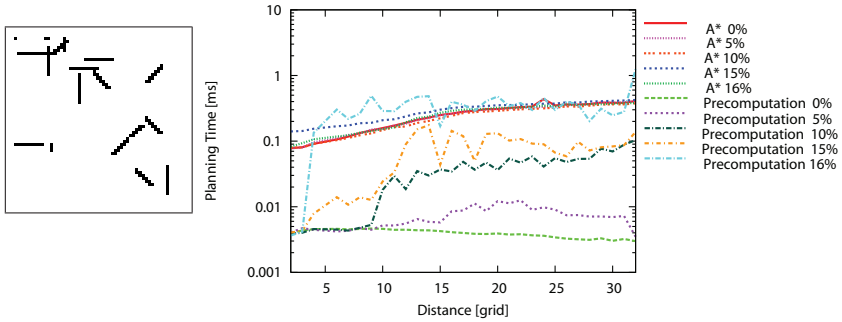
(d) 1-by-20 grid obstacles with a mixture of 3 different types

Figure 30. Success Rates for Different Obstacle Shapes

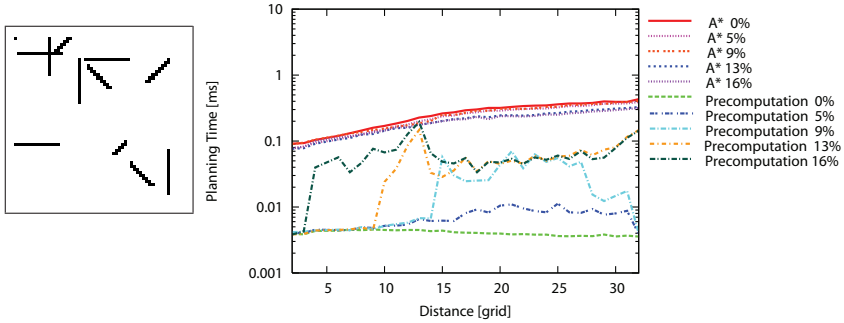
using PCS without NSS, the GR 5% maps are shown in the bottom left, with the vertical axis being the GR.



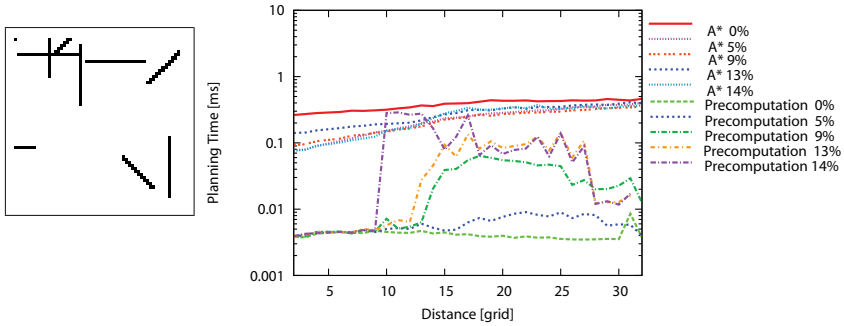
(a) 1-by-5 grids obstacles with a mixture of 3 different types



(b) 1-by-10 grids obstacles with a mixture of 3 different types



(c) 1-by-15 grids obstacles with a mixture of 3 different types



(d) 1-by-20 grids obstacles with a mixture of 3 different types

Figure 31. Planning Time for Different Obstacle Shapes

using PCS without NSS plotted with ER, the GR 5% maps as shown in bottom left

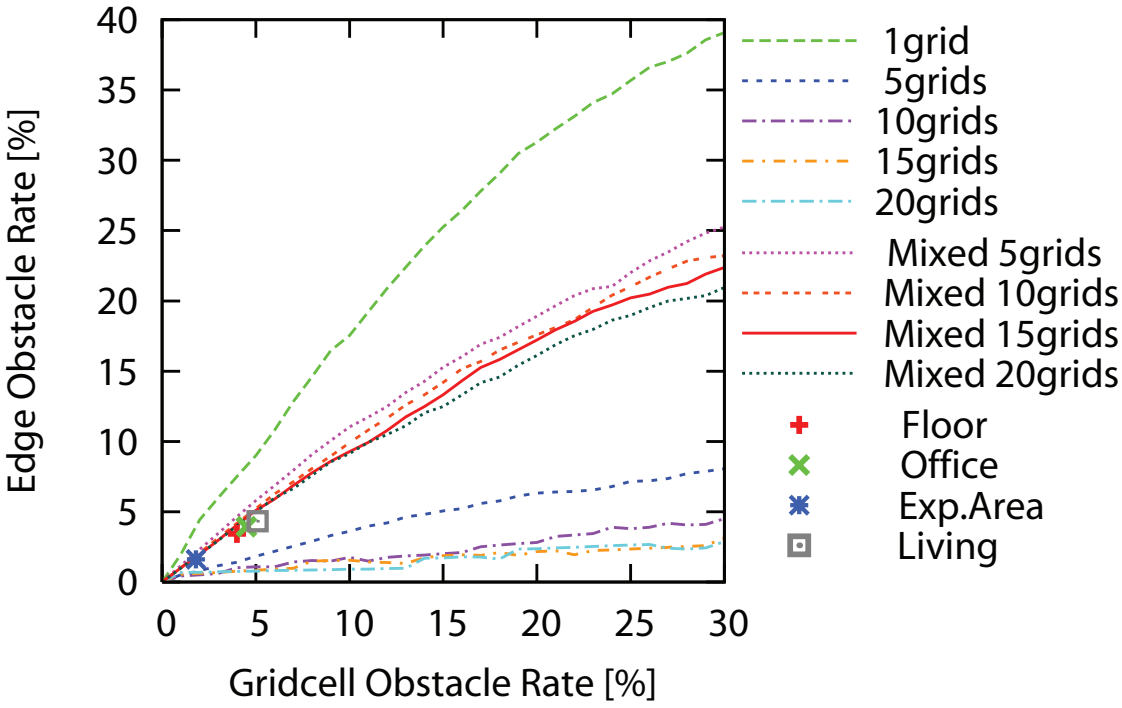


Figure 32. Edge - Gridcell Obstacle Rates

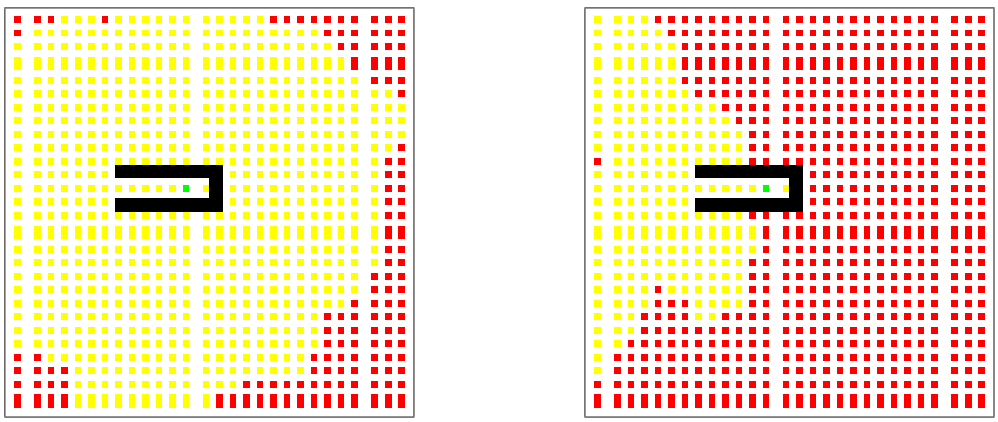
4.9 Concave and Convex Obstacles

Concave and convex obstacles are validated for precomputation planning. Square experimental maps of length 55 grids were strewn with concave and convex obstacles. The PCS plans paths from the map center start to all over the map.

4.9.1 Concave Obstacles

Inside Start The success map is shown in Fig. 33, where subfigure (a) represents the results of a 40-deep PCS search without NSS, and subfigure (b) represents the same results, but with NSS. Root is set for concave obstacles inside. The black area is shown as obstacles, and the highlighted area is shown as the region where the proposed planner obtains the correct path. PCS without NSS successfully obtains the paths in an area that covers its own tree, and finds paths outside of the concave obstacle. The pruned internal branches with NSS for expanding nodes outside of PCS have a lower accuracy than for PCS without NSS.

For 770,000 nodes in a 40-deep PCS without NSS, a full-grown tree generates just a depth of 8 with this node count (Fig. 34). A full-grown tree's branches reach only a narrow area, and many nodes stack around the root.



(a) without NSS (b) with NSS

Figure 33. Success Map with Concave Obstacle Start Inside, PCS is grown at the 40^{th} generation

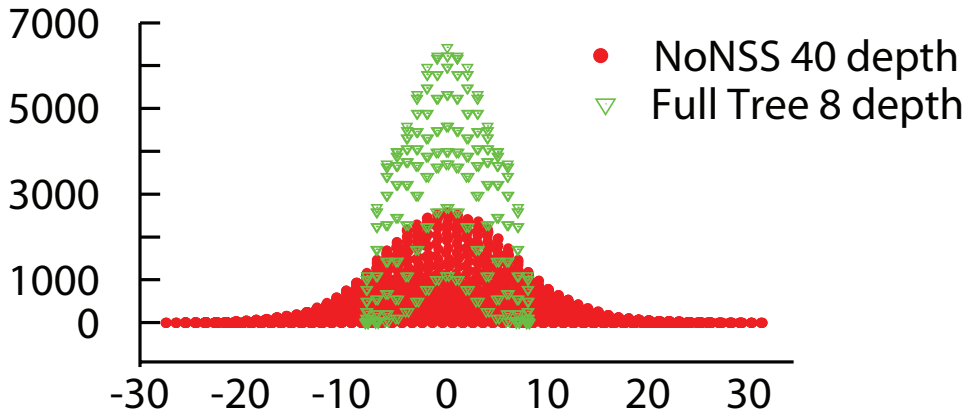
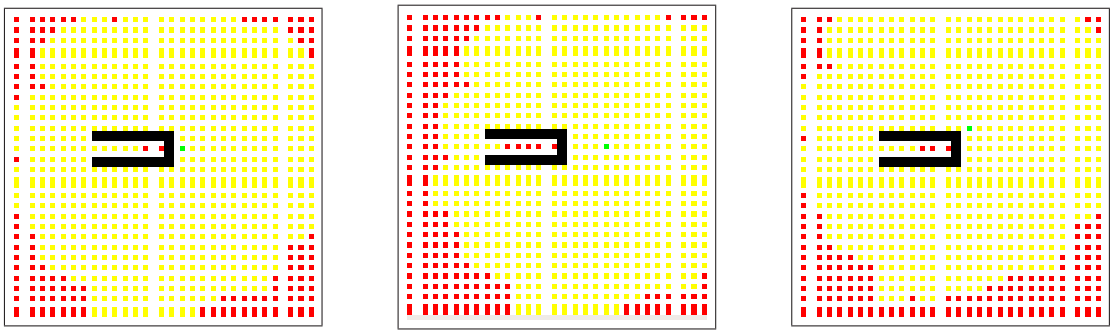


Figure 34. PCS full grown tree (Green) and without the NSS 40 deep tree (Red) in cross-section

Outside Start The success maps are shown in Fig. 35 using a 40-deep PCS without NSS. The root is set for concave obstacles located outside the main area. This method could not find paths whose goal was inside the concave obstacle because, in this area, the planner has to pass through a narrow area and to turn into the deep back side of wall.

4.9.2 Convex Obstacles

Convex versus PCS The success maps are shown in Fig. 36 using a 40-deep PCS without NSS The root is set along the convex obstacles. PCS finds the paths into the deep back side of a convex obstacle in the case where the root is placed near the boundary of the convex obstacle, as shown in Fig. 36 (a). If the root is set further from the boundary of the convex obstacle, the convex obstacle cuts the branches in PCS, and a path cannot be found (see, Fig. 36 (b), (c)). Using NSS and setting the root in the middle of the convex obstacle increases the accuracy, but the PCS branches cannot reach the deep back side of the convex obstacle as shown in Fig. 37.



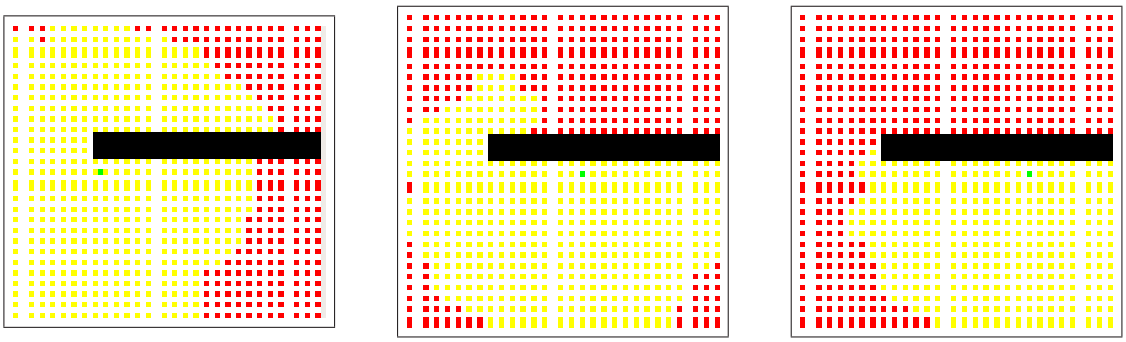
(a) (b) (c)

Figure 35. Success Map with a Concave Obstacle and the Initial Point is Outside the Obstacle
using PCS without NSS, (a) the root close to the outside boundary of the obstacle (b) the root is far from the outside boundary of the obstacle (c) the root is close to the upper-outside boundary of the obstacle, green: root (start), yellow: success, red: error

4.10 Summary

This chapter presented generalized, precomputed search trees that were shown to be effective in delivering high-speed performance during precomputation planning. The speed of the generalized search trees is faster than an A* planner for maps where the obstacle rate is the same as for indoor environments. The precomputed search tree is built with pruning based on the rule of constant memory and the maximum size pruning method (MSP) with a preset pruning ratio. Using MSP, a large precomputed search tree is obtained that has a reasonable memory size. Furthermore, by applying the node selection strategy (NSS) to MSP, the outer edge of the tree is extended to give better path reachability. The alternate Branch BackTracking (ABBT) enhances the success rate in crowded environments with a few local minima.

Experiments were performed, and the success rate was analyzed. It was determined that an improvement in the speed of path searching and path optimality was obtained with the new algorithm. As well, the experiments showed that the precomputed search tree finds the paths for maps with an obstacle rate of 31%. Even if the goal position is set on an incomplete tree that is pruned, precomputation planning can successfully find a path. The speed and path optimality were



(a) (b) (c)

Figure 36. Success Map with a Convex Obstacle
using PCS without NSS, (a) the root is placed near the outside boundary of obstacle, (b) the root is placed in the middle of obstacle, and (c)the root is placed far from the outside boundary of the obstacle.

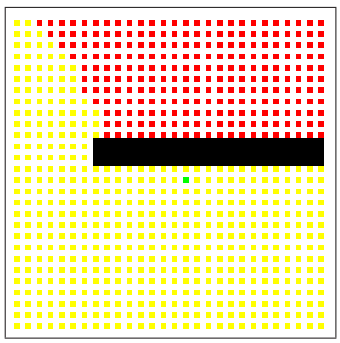


Figure 37. Success Map with a Convex Obstacle using PCS with NSS and the root is placed in the middle of the obstacle, green: root (start), yellow: success, and red: error

evaluated by comparing the result of the A* planning to that of the precomputed searching. The runtime of the precomputed planners for obstacle rates less than or equal to 30% is constantly less than the value of the A* planning. In maps with less than a 20% obstacle rate, the runtime of precomputation planning is more than one order of magnitude faster than planning without a precomputed search tree. It was found that the same optimal paths, those with the same shortest path, were found on the maps with less than an obstacle rate of 25%. Every mode value is 100% of the length of the shortest path in all the trials.

PCS planning is primarily affected by obstacle shape, such as a longer one that cuts more PCS branches in one sitting. For root placements that are on the backside of convex obstacles, PCS planning could not find any paths.

Since the obstacle rate of real indoor environments is less than GR 5% and ER 14%, the precomputation planning quickly determines a path in the real world with a path optimality of 100%. Thus, the proposed precomputation planning will rapidly produce the optimal path in real indoor environments.

5. Smooth Path Planning With Pedestrian Avoidance for Wheeled Robot

Path planning with a steering set was used to generate trajectories with smooth directional changes. To put path planning to practical use in dynamic environments, robots must be more quickly motion and efficiently, without, for example, endangering pedestrians. Assuming that the trajectories of moving obstacles are predictable, smooth path planning worked in the presence of moving obstacles. We defined new path evaluation method suitable for wheeled robots, evaluating our planner experimentally in an office, confirmed the efficiency of our planning.

5.1 Smooth Path using Steering Set

5.1.1 Steering Set

Our smooth path planning algorithm is named "Steering Set Planner", uses behavior steering sets resembling natural car movement. Steering sets consist of at least three behaviors – "straight line," "right curve" and "left curve," referencing of "straight-ahead," "turn-right-ahead" and "turn-left-ahead" car movement as shown in Fig. 38. Numbers of branches, turning angles, and curve lengths are adjustable. We designed a new planner using these steering sets to build a discrete search tree.

Tree branch shown at right in Fig. 38 was based on the length of one forward search movement. Branch shapes are geometrically fixed and precalculated. Planning with a fixed branch shape has two problems at these geometric constraints. First, sampling-based planning may cause discontinuity between branches at connection points [47] that, without post-processing such as smoothing, allows movement for changing robots direction. This problem can cause a sudden decrease in velocity during robot motion. Second, in exhaustive growing of branch nodes for an arc angle variable consume collision checking time exponentially with branch growth and memory to store the exponentially grown tree [30]. We solve these problems by ensuring a tangential connection between nodes.

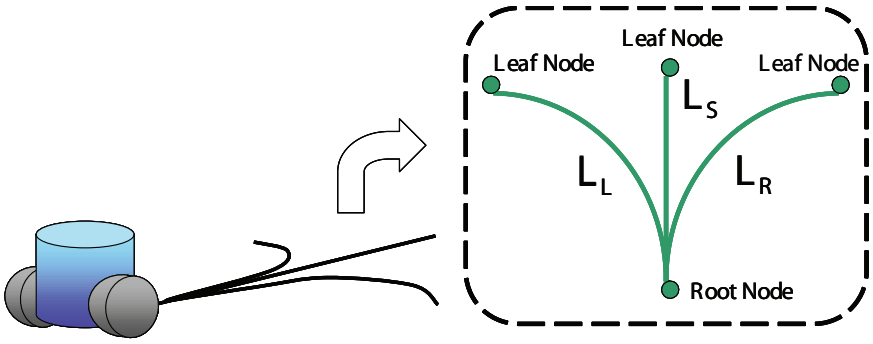


Figure 38. Image of Steering Sets

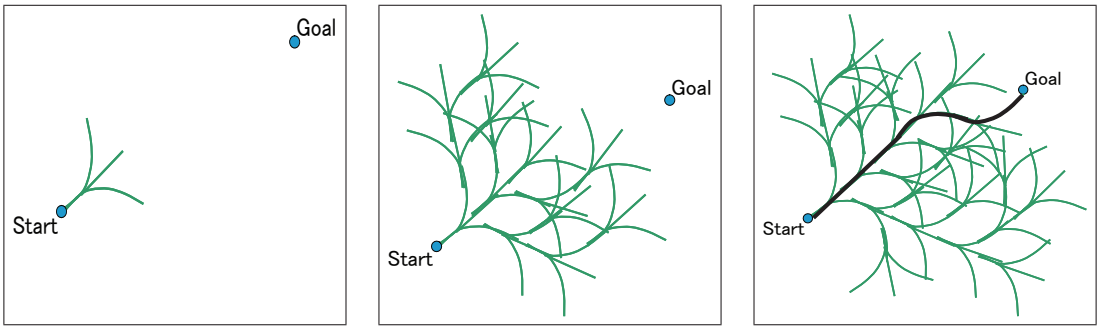
5.1.2 Steering Set Search

The steering set planner grows repeatedly by adding branches duplicating car behavior from the starting point on the static map, as shown in Fig. 39 (a) to the goal shown in Fig. 39 (b). Once a branch reaches the goal, the planner backtracks on nodes from the goal node to the root node. The resulting path is smooth free of discontinuous points along the map grid and suitable for wheeled robots, as shown in Fig. 39 (c). Green lines in the figure show the grown tree and black line expresses the path for a wheeled robots.

To confirm that our new planner creates a smooth path, we compared our planned path to the shortest path search using a non-informed A* algorithm, Dijkstra algorithm [48], with efficient optimal search [46] on the grid map as shown in Figs. 40 and 41. We applied an octant arc to the curve in the steering set, using a tangent to ensure the smooth connection between two successive branches. This tangential continuity yielded a smooth connection (Fig. 41).

5.2 Path Evaluation Method

Path planning time and path length are widely used to evaluate paths. Path length assumes that shorter is better, but is not an optimal evaluation of paths for wheeled robots because it does not consider wheel movement in actual path following. We defined two measures to evaluate paths, one is an expressing path smoothness and the other is an optimal evaluation of paths length for wheeled robots.



(a) (b) (c)

Figure 39. Steering Set Search

(a) start search, (b) grow search trees when reaching the goal, (c) get a smooth path backtracking from the goal

Figure 40. Shortest Path

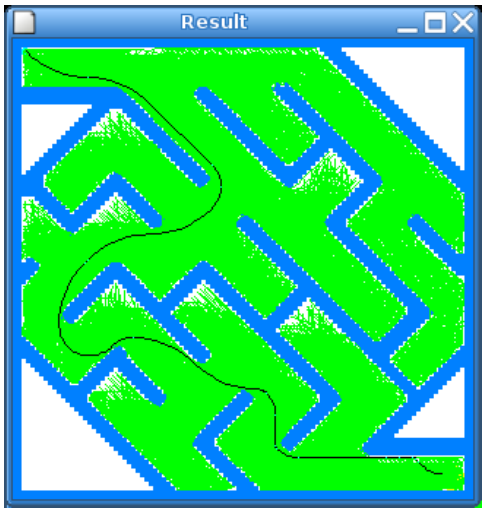


Figure 41. Steering Set Path

5.2.1 Smoothness Evaluation

To evaluate path smoothness, we defined the Average degree of Discontinuous Connection Points (ADCP), calculated by the summing discontinuous angles divided by the number of discontinuous events – the smoothest path scores zero in this measure. The formula for "ADCP" is as follows:

$$Smoothness = \frac{\sum_{Start}^{Goal} (DiscontinuityAngle)}{Number\ of\ Discontinuities} \tag{4}$$

Definition of Transit Length We define path length evaluation that includes wheel movement "Transit Length" (Fig. 42).

The widely used conventional path length expresses only a path's absolute length, without including a consideration of how real wheeled robots actually move and thus not evaluating actual movement efficiency. A path may indeed be the shortest but its actual movement may not be optimal and its travel time may be long. Typically, optimal paths based on path length have a problem with discontinuous path points. To implement this path in the real world, wheeled robots change direction by turning on the spot at these discontinuous points. The transit length we propose calculates total trajectories for robot's two wheels. Using the robot movement speed, transit length estimates time effectiveness of the planned path.

At left in Fig. 42 is the front of the two-wheeled robot shown at right. Two square areas at left of Fig. 42 are the robot cross-section and the black solid line at the center is the planned path. Dotted lines express transit length, which we calculate as the sum of both transit lengths of the two-wheeled robot when the planning path is centered at the robot midpoint.

Transit Length Formula The transit length formula for planning with the shortest path on grid-based map and for planning with the steering set is as follows:

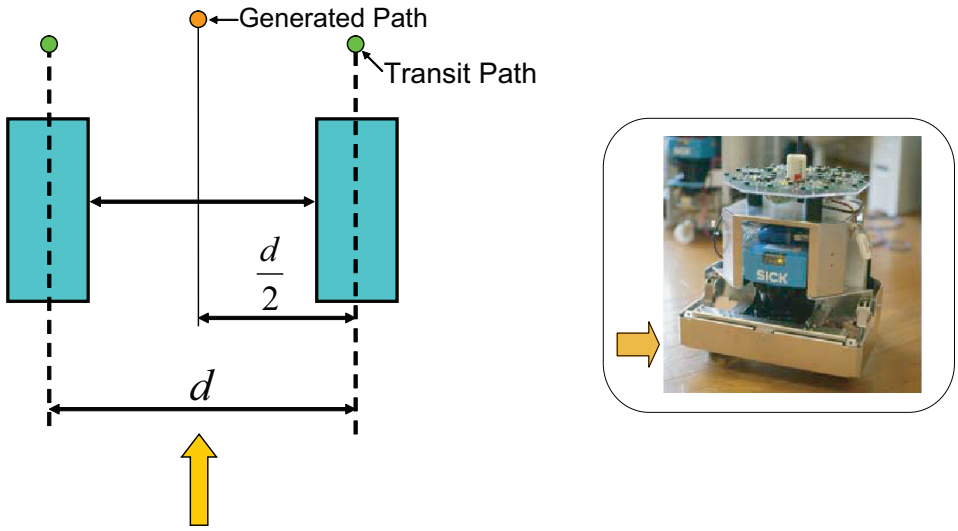


Figure 42. Transit Length Definition

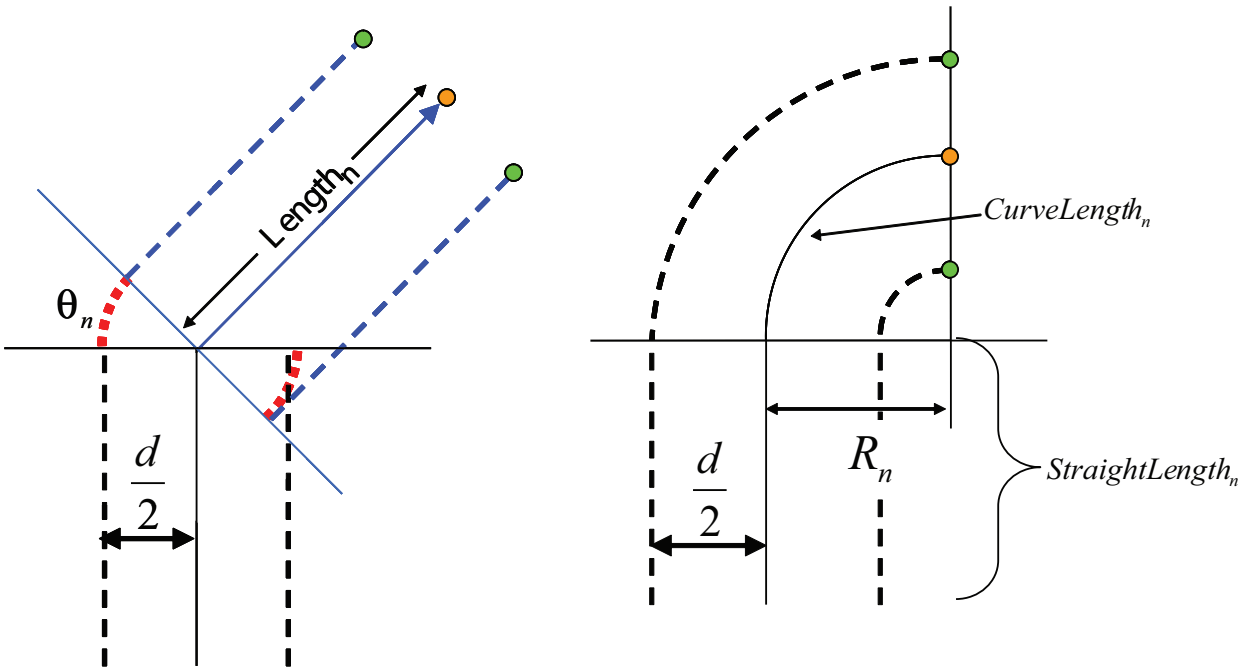
$$TransitLength_{ShortestPath} = 2 \times \sum_{n=Start}^{Goal} Length_n + \sum_{n=Discon.Points} 2\pi \frac{d}{2} \left(\frac{2\theta_n}{360} \right) \quad (5)$$

$$TransitLength_{SteeringSets} = 2 \times \sum_{n=Type} StraightLength_n + 2 \times \sum_{n=Type} \{CurveCounts_n \times CurveLength_n\} \quad (6)$$

Here d is the axle length. The transit length of the grid-based shortest path search method is calculated by summing the dotted line as shown in Fig. 43 (a) and in Eq. 5 above. The steering set transit length is calculated by summing the dotted line as shown in Fig. 43 (b), where R_n is the quadrant arc radius.

5.3 Pedestrian Avoidance Approach

In the real world, parts of the environment change dynamically during planning. Regardless, however, of how quickly the environment changes, the wheeled robots path must be smooth. Grid-based shortest path search requires recalculation



a) Shortest Path Search b) Steering Sets

Figure 43. Transit Length

process for smooth by a high-dimensional function, and the trajectory is not guaranteed to be either optimal or completeness. Change must be detected as it occurs and used to predict the environment in the future. We expand steering sets to include planning in dynamic environments and a pedestrian tracking system as detailed in the sections that follow.

5.3.1 Path Planning with Moving Obstacles

Search space in a changing environment including moving obstacles is expressed as a three-dimensional space with a time axis in addition to the spatial x and y axes. Static obstacles in this search space are expressed as solid rectangles vertically elongated along the time axis. Moving obstacles in this search space are expressed as columns elongated obliquely along the time axis. The planning for that three-dimensional space then becomes a geometric spatial search as shown in Fig. 44. We assume that the planner obtains the position and speed data on the moving obstacles from external devices and uses this data to predict the future positions of moving obstacles. Based on the these predicted position of moving obstacles, this three-dimensional search space is recognized as two-dimensional search space sliced by a time period [37] as shown in Fig. 45. We assume that search space including all static obstacles is known. The planner receives the obstacle prediction path for time $t+n$ at time t in each planning cycle so that paths can be planned to avoid moving obstacles by checking intersections between growing branches and predicted obstacle paths.

5.3.2 Pedestrian Tracking System

Horiuchi et al. [49] developed pedestrian tracking system using a laser range finder (LRF) on a mobile robot and an environment map. Fig. 46 shows a visualization of the trajectory of a walking human using the tracking system [49]. Squares denote objects recognized as pedestrians and circles show the robot location. Black solids are static obstacles in the environments, pale green shows LRF information and red line is a pedestrian trajectory. The tracking data from this system includes the location and speed of moving obstacles. The predicted position of moving obstacles such as walking humans can be made by these data.

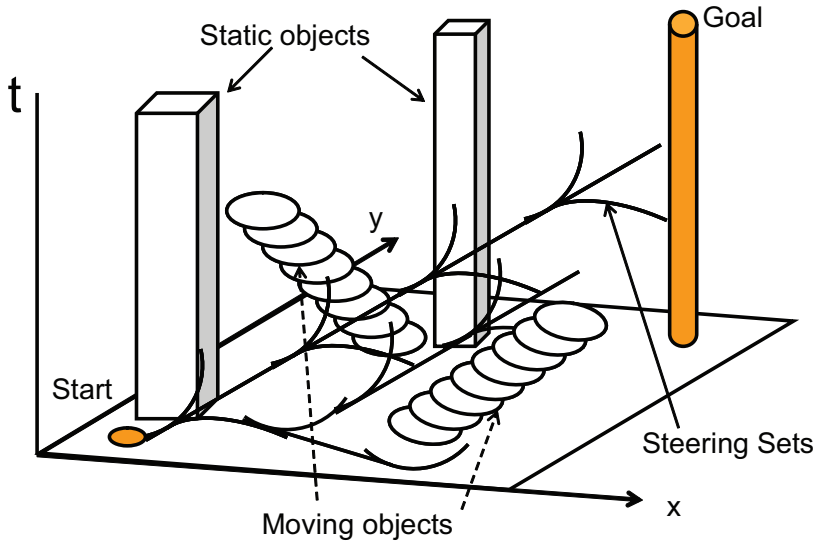


Figure 44. Search Space with Time Dimension

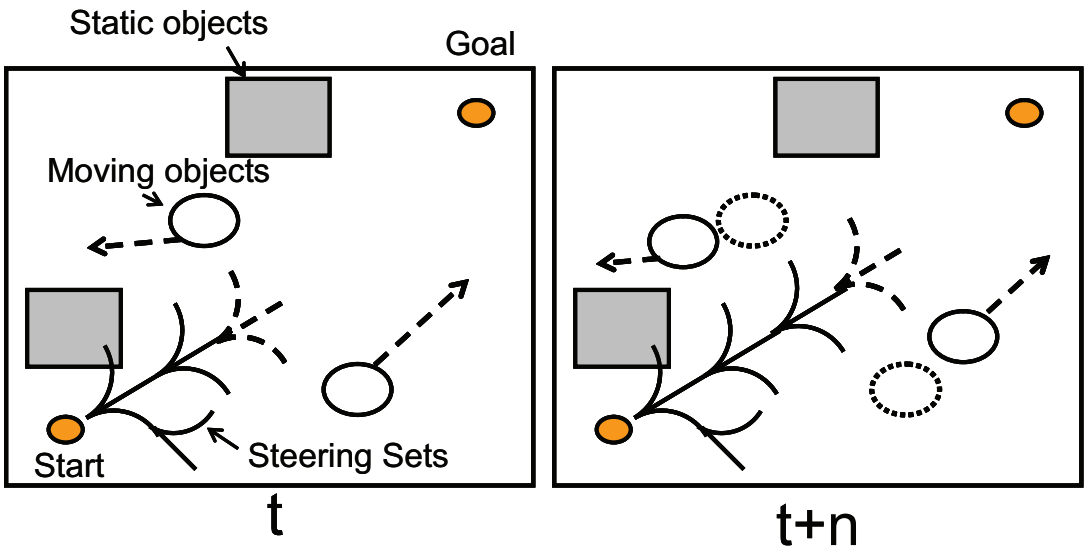


Figure 45. Search Space with Time Slice

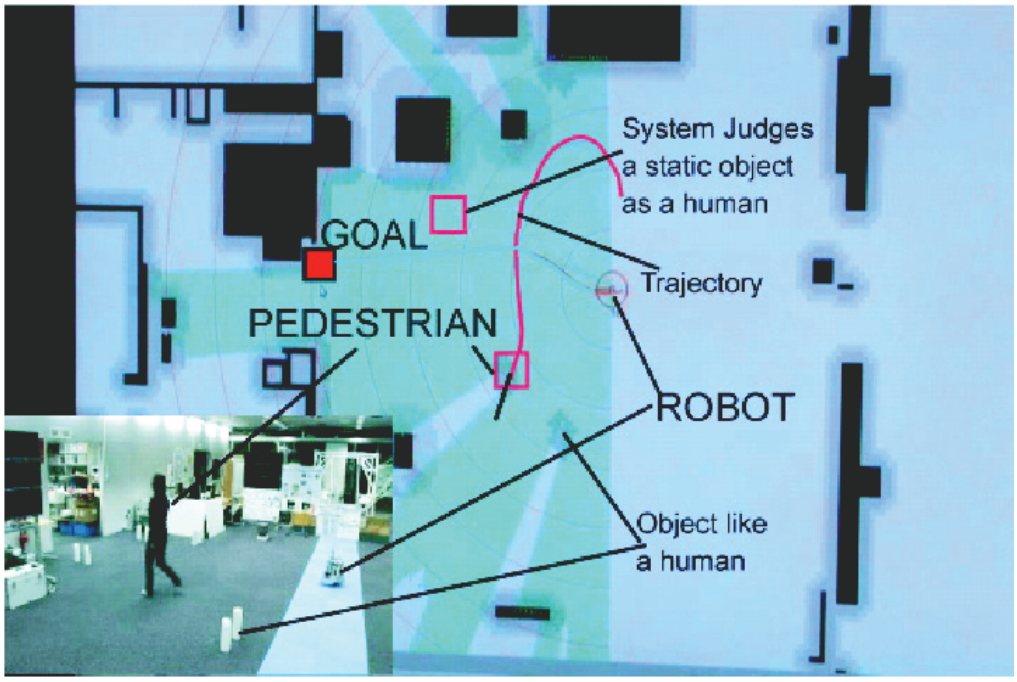


Figure 46. Pedestrian Tracking System

5.4 Comparative Experiments with Steering Set Planning and Grid-Based Shortest Path Search

5.4.1 Experimental Setup

Mobile Robot "Penguin II" Our "Penguin II", mobile robot introduced in Fig. 42 is 40 cm long, 45 cm wide, and 42 cm high not including the wireless antenna. A quadrangular prism cut off at the edges, Penguin II has two front drive wheels and two back caster wheels, and can turn on the spot within a 50 cm radius. The robot is equipped with a Pentium-M 2.0 *GHz*, FSB400 *MHz* processor, 1 *GB* of memory, RT-Linux for OS and a 1 [*ms*] PD controller for the motor.

Experimental Overview In the experimental layout in Fig. 47, two 1.2 meter long partition boards are placed 2 meters apart in an 8 meter square area. The starting point is 2 meters from one board and the planning task involves passing between the two boards. For comparison with our planner, we implemented the shortest path search using non-informed A* [48] with an efficient optimal search [46] on an real wheeled robot for an experiments.

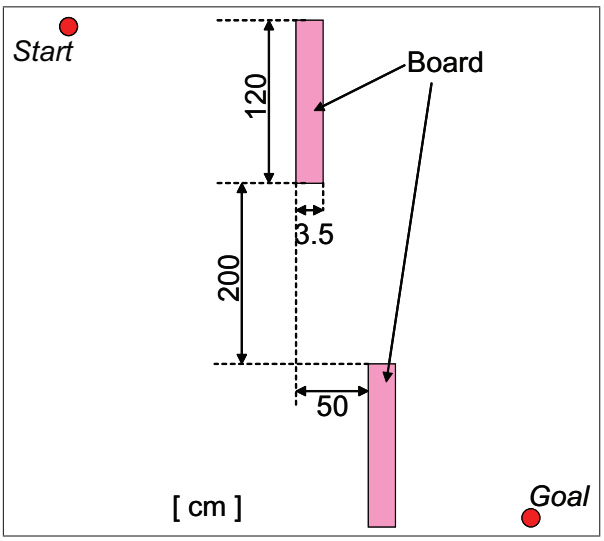


Figure 47. Experimental Environment

5.4.2 Results of Transit Length Simulation and Analysis

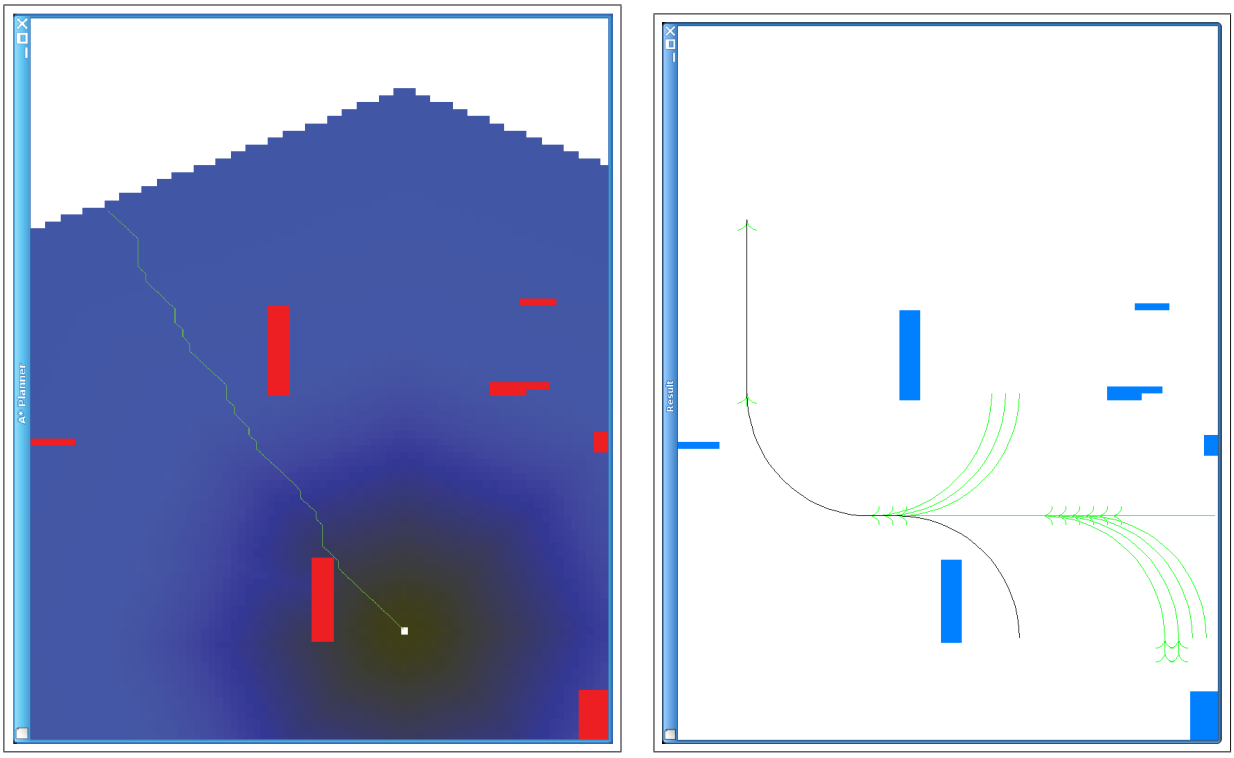
Fig. 48 (a) outputs shortest path planner and Fig. 48 (b) output our planner in a simulation for the same map configuration. Table 2 gives resulting transit lengths and estimated robot running time assuming a forward speed of 40 [cm/s] , 90 [deg/s] rotation, and a 10 [cm] grid resolution for the environmental map. Table 2 shows that the our planner path length is longer than that of the shortest path planner, but, our planner's transit length is by 400 cm shorter. Our calculated running time is also shorter than that of the shortest path planner. Smoothness calculated by "ADCP," gave our planner a score of zero, i.e., perfect smoothness. Total rotation time calculated by "ADCP," based on the above results, showed that our planner completed movement faster than the shortest path planner on a real wheeled robot.

Table 2. Results of Simulator and Transit Length

	Shortest Path Search	Steering Set
Path Length [cm]	767	840
Transit Length [cm]	1685	1280
Smoothness [deg]	45	0
Total Rotation Time [sec]	28	0
Calculated Total Movement Time [sec]	33	21

5.4.3 Results of Real Mobile Robot

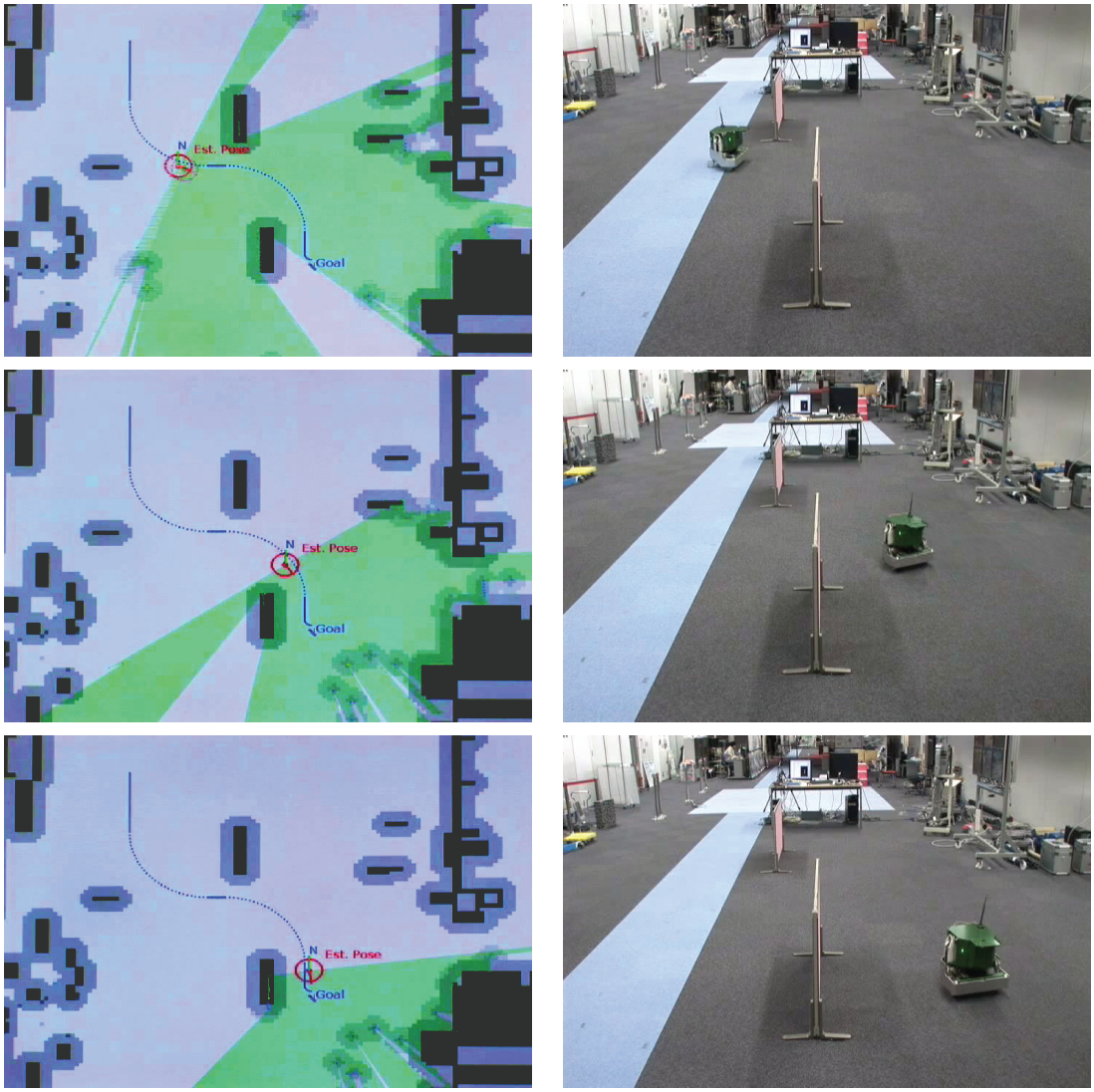
Fig. 49 (a) and 50 (a) show visualizations of the running experiment for the real wheeled robot and Fig. 49 (b) and Fig. 50 (b) show video images of the movement using the mobile robot from Section 5.4. Fig. 49 (a) shows that our planner created a smooth path avoiding static obstacles without replanning the path from start to goal. Wheel-running behavior for our planner was smooth, the robot did not have to stop to change direction, and it reached the goal. The experiment using the shortest path planner required stopping and turning, making the robot have to turn on the spot to change orientation and pose, making its travel movement rough. Our planner took 18 seconds to reach the goal, versus



a) Shortest Path Search b) Steering Sets

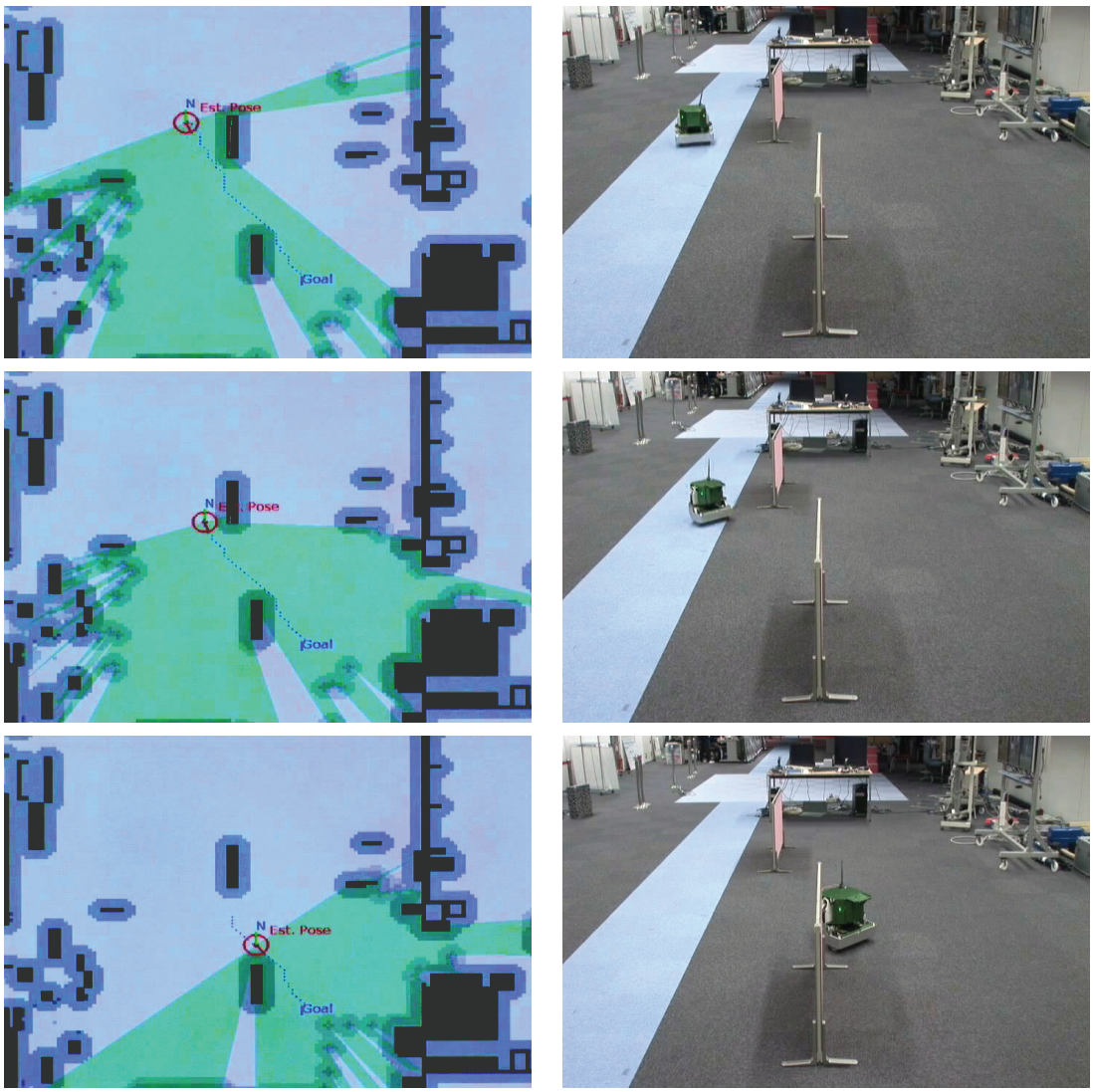
Figure 48. Board Array Map Results

28 seconds for the shortest path planner, making the difference between predicted and the actual travel time up to 5 seconds. The transit length presented as path evaluation method in Section 5.2.1 thus effectively evaluates the path length for wheeled robots. Using the path of the shortest path planner made the robot pass very close to static obstacles, meaning that a safe range must be initially calculated and included in planning.



(a) Visualization (b) Travel Motion on Mobile Robot

Figure 49. Result of SteeringSets Planning Implementation



(a) Visualization (b) Travel Motion on Mobile Robot

Figure 50. Result of Shortest Path Planning Implementation

5.5 Experiment on Pedestrian Avoidance

5.5.1 Experimental Setup

Assuming that the planner gets prediction data about moving obstacles in each system cycle, we planned simulation using an octant steering set in an environ-

ment including moving obstacles. Pedestrian data is taken from a high-resolution laser range finder with the human tracking system.

5.5.2 Results of Avoiding Moving Obstacles

Fig. 51 shows the results of path planning with moving obstacles with obstacle position predicted at each planning cycle. Small dots are pedestrian data from the pedestrian tracking system and crosses are the predicted position and trajectories. The dashed line is the planned path using our planner, which plotted a straight path without considering moving obstacles. Using collision checking between growing branches and the predicted moving obstacles positions, the planner plotted a path avoiding moving obstacles. If the planner could not predict moving obstacles, the planner would have to calculate a path including a large safety margin to account for the size of moving obstacles uncertainty. The planner, however, achieved avoidance of moving obstacles using collision checking between growing branches and predicted obstacle positions.

5.6 Summary

A path planner using steering sets has duplicated car-like behavior and a path evaluation method has been proposed for wheeled robots. Transit length evaluation estimates the execution time of paths implemented on a real wheeled robot. The feasibility of path evaluation was confirmed in simulation and in real world experiment with a robot. The experiment showed that although paths produced by our planner are not optimally short in length, they result in shorter transit length and time for a wheeled robot.

We assumed that the planner gets data on moving obstacles using external devices and uses predicted future positions of moving obstacles as simulated with steering set planning using pedestrian movement data. A smooth path avoiding moving obstacles was then generated the planner to reliably predict obstacle position certainly.

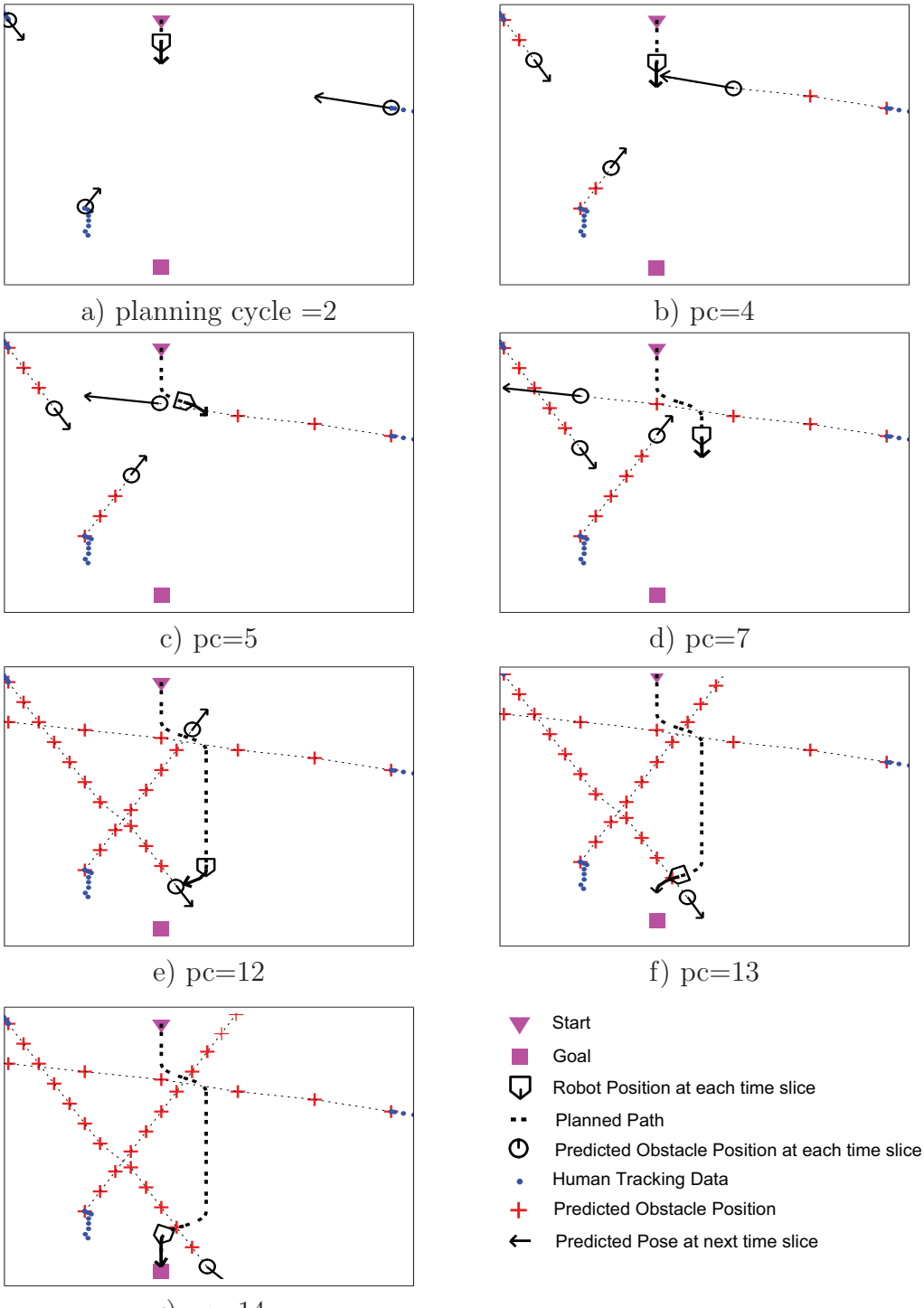


Figure 51. Avoiding Moving Obstacles

ts: time slice

6. Pre-Computed Planning with Steering Sets

This chapter describes a new pruning method for compact precomputed search trees with steering sets branches and evaluates its effectiveness and the efficiency.

6.1 Precomputed Search Tree with Steering Sets

A steering set for the tree is built up to a certain depth. If the complete tree is built, its size increases exponentially, and its memory usage exceeds the upper limit of the physical memory size. The total number of tree nodes is expressed by Equation (7) with depth level d. The Branch Factor is 3 in the steering set (Fig. 52).

$$The\ total\ number\ of\ nodes = \sum_{k=0}^{d} BranchFactorCount^{k} \tag{7}$$

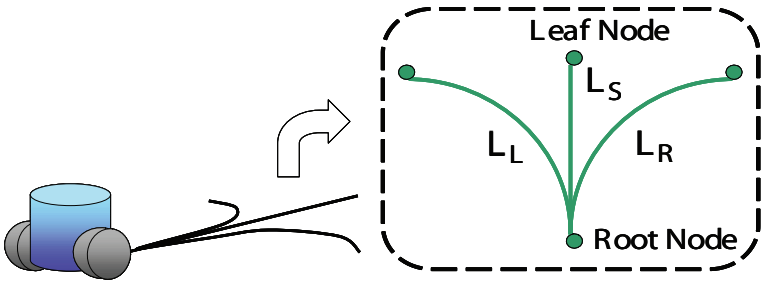
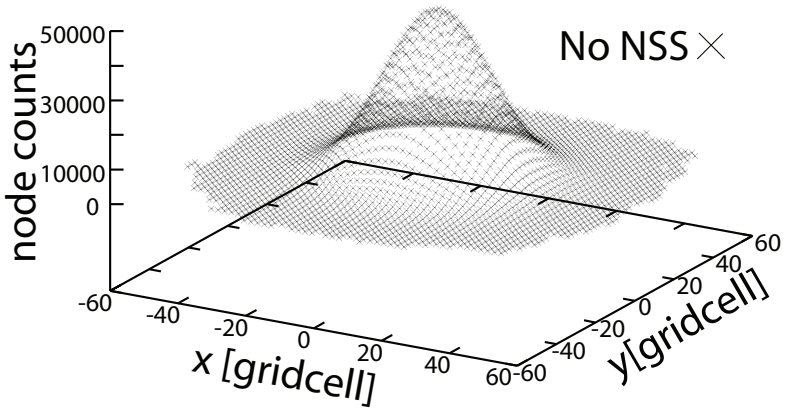


Figure 52. Image of the Steering Sets

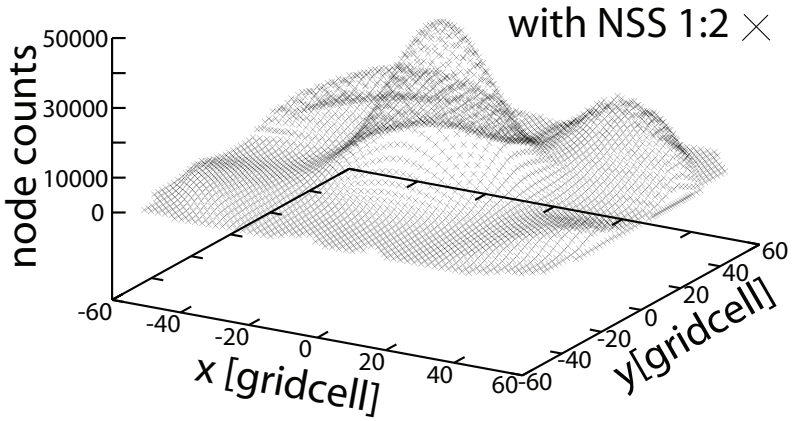
The tree is automatically pruned at the same time as the branches are grown. There is no need for user intervention to reduce redundant nodes. The proposed pruning method is described in Chapter 3.

The present pruning start value is defined to occur at the 12^{th} generation or when more than 20 MB of memory are used for each generation. Furthermore, one node of this tree occupies 32 bytes of memory, and the tree grows branches for a depth of 30. The length of one branch is about 2 gridcells. One grid cell is 10 cm in a real environment. Figure 53 (a) shows the configuration of precomputed search trees (PCS) expanded on a 2-dimensional space, where z-axis refers to the number of nodes that reach each position of the grid. The shape of the branch is

the steering set that has three fixed branches including quadrant curves. The total memory usage of this tree is just 760 MB. If the tree grows all branches without pruning, the memory usage for retaining all the node data is 6.6 PB. Using MSP, a lightweight tree that is 1.2×10^{-7} times the size of the fully expanded tree was obtained.



(a)using only MSP with Steering Sets



(b)using NSS applied to MSP with Steering Sets

Figure 53. Precomputed Search Trees with a depth of 30, grown with Steering Sets

Figure 53 (a) shows the PCS with MSP not using the NSS, and Fig. 53 (b) shows the PCS with MSP using the NSS, with an exchange ratio of 1:2. The

effect of the NSS changes the form of two parts of the tree. First, in Fig. 53(b), the heap in the center of the PCS using the NSS decreases compared with not using NSS (Fig. 53(a)), while its maximum height decreases when using NSS. Second, the verge of the tree using NSS expands farther than when not using NSS. Figure 54 shows the x-y plane view obtained from the difference in PCS when not using NSS (Fig. 53(a)) and PCS using NSS (Fig. 53(b)). The dark

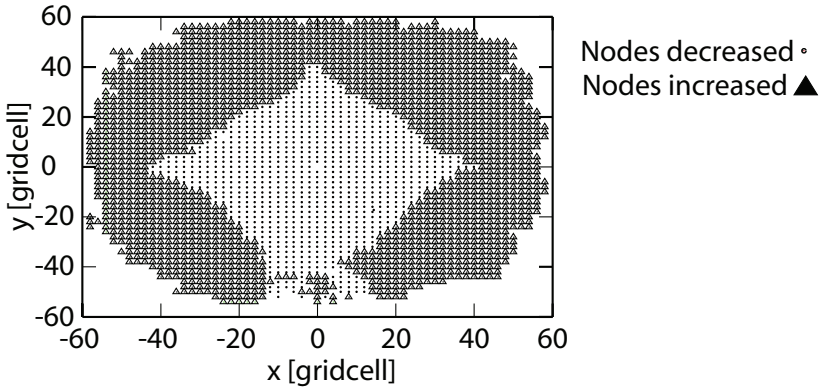


Figure 54. x-y Plane View Obtained from the Difference between not using NSS and using NSS with Steering Sets

color shows an area with more nodes when using NSS, while a bright color shows an area with fewer nodes.

Thus, applying NSS to MSP cuts down the node counts in the center, and enlarges the tree on the outside and on the outer nodes. Since an even exchange ratio was used by the NSS and the 3 factors of one branch, memory usage increases. For example, in the case of Fig. 53(b), memory usage is 117 MB. Using NSS, the number of nodes in the central heap decreases by about 40,000, and the distance from the root node reaches to about 17 gridcells.

6.2 Experimental Settings

This section describes the experimental results that verify the accuracy of the precomputation planning using steering sets applied to MSP and the effect of NSS. As well, it is confirmed that using steering sets with precomputation planning

improves the speed and optimality of the paths when using an indoor environment map.

In the experimental simulation, a PCS is made with 30 generations using steering sets as the first three branches. The root node is placed at the center of the map. The length of one branch is about 2 gridcells. PCS is applied to the prune processing based on the predefined tree limits of 20 MB for each generation after the 12^{th} generation. The experimental map is a square map of 110 grids on one side. The map includes a static obstacle, the size of which is a 40-cm square based on the size of a stool (for example, Fig. 55). The number of obstacles in a map is based on the predefined ratio to the environmental map size. Experimental maps with obstacle ratios from 0% to 20% were prepared based on the previously defined dispersions in Section 2. Obstacles were scattered on the map.

The starting position of planning was specified as the origin point of the experimental map, and 700 goal positions were randomly selected.

The CPU used for these experiments was a Intel(R) Core2 X6800 2.93 [GHz] with 4 GB of memory, running Linux.

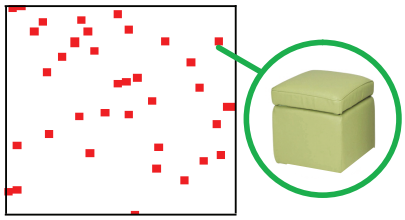


Figure 55. Experimental Map with 5% Obstacle Rates

6.3 Effect of MSP

This section describes the execution of a precomputed search tree that was pruned using MSP. Figure 56 shows the accuracy evaluation result of path planning using PCS with MSP, as measured by the percentage of goals reached for random goals. In Fig. 56, the vertical axis is the "*obstacle rate*',' and the horizontal axis is the distance from the start node. The success rate is shown by differences in color. The bright color represents 100%, with darker shades representing a lower success rate.

The vertical blue line indicates the position of the beginning of pruning, which converts the generation based on MSP settings into the distance of the path. Even though the precomputed tree is incomplete when it is being pruned by the MSP, the planner reliably generates a path, and its success rate is 100% for obstacle rates from 0% to 20% and up to a 40-grid distance. A success rate of 5% of obstacle rate is 100% up to 50 grid of distance. As Fig. 56 shows, the success rate decreases in the case of a large distance on a crowded map. On a map with many obstacles, it is thought that the obstacles that are out in the open after the planner starts to trace back the nodes cut off the link between the nodes that the planner has followed up to then. It would appear that the more pruned tree is not sufficient for getting a complete path against the far goal point on the more congested map. It is also believed that the inefficient spread nodes caused by the turning-around expanded branches decrease the heap of nodes around distant places.

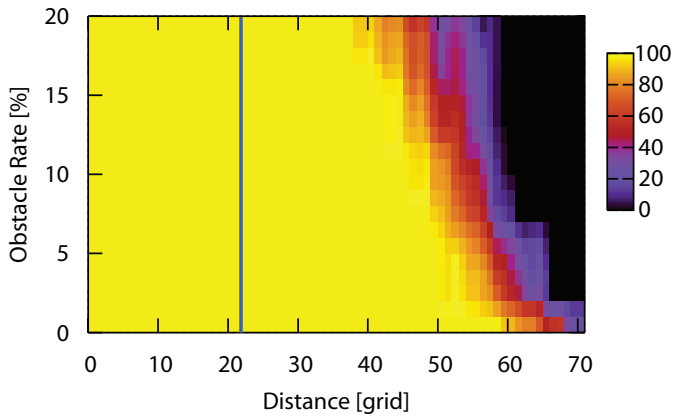


Figure 56. Success Rate of the Precomputation Planning using MSP

6.4 Effect of NSS

This section shows the effect of the strategic selection of nodes NSS applied to MSP. In this experiment, the exchange ratio of NSS settings is 1:2. The other settings for goal counts and experimental maps are the same as above. Figure 57 shows the result of the success rate for path planning using PCS applying

NSS to MSP. Comparing the success rate using NSS (Fig. 57) with the rate for not using NSS (Fig. 56), the color of the success rate is brighter than that for not using NSS in the range between 0% to 10% on the vertical axis and after 50 grids of distance in Fig. 56, Using NSS to correct the success rate contributes to improving the performance on a map with many obstacles and distant goal points. The success rate with NSS is virtually 100% for obstacle rates from 0% to 10% on a map, whereas the success rate is under 20% in the case of not using NSS. Therefore, applying NSS to MSP is useful in improving the performance of the path planning using a precomputation tree.

The obstacle rate of real indoor environments is less than 5%, as was shown in Section 2. Furthermore, it has been shown that a living room has the highest obstacle rate. A precomputed search tree in this experiment can cover a living room area, and its success rate is always 100% when the obstacle rate is less than 5% of the map.

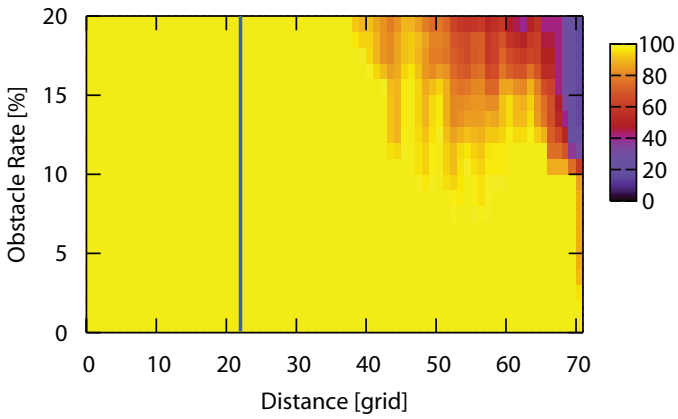


Figure 57. Success Rate is Improved with NSS

6.5 Speeding Up of Searching

This section relates the effect of speeding up the search for the precomputation planner. The precomputation planning with the existing steering sets is compared to planning without precomputation.

6.5.1 PCS With NSS

Figure 58 shows the extracted results of the planning time for the experiment that is executed as in Section 6.4 above and the runtime of path planning without any precomputational techniques. It is a semilog plot of the planner runtime with the vertical axis being the planning time in milliseconds and the horizontal axis being the distance. The upper two lines show the runtime of steering set planning when the obstacle rate is 0% and 13% of the map, while the other lines show the precomputed planner runtimes when the obstacle rate is 0%, 5%, 12%, and 13%. The values of the precomputed planner runtime for obstacle rates less than or equal to 12% are constantly below the value of the existing planning without precomputation. The planning speed of the steering set planning depends on the distance from the start to the goal. The precomputation planner runtime can increase when the obstacle rate increases. It was found that planning using the proposed search method is faster than the existing planning that does not use the precomputed trees (see, Fig. 58). When the obstacle rate is less than 13% of the map, the runtime of the precomputation planning is more than two orders of magnitude faster than the existing planning method without precomputation.

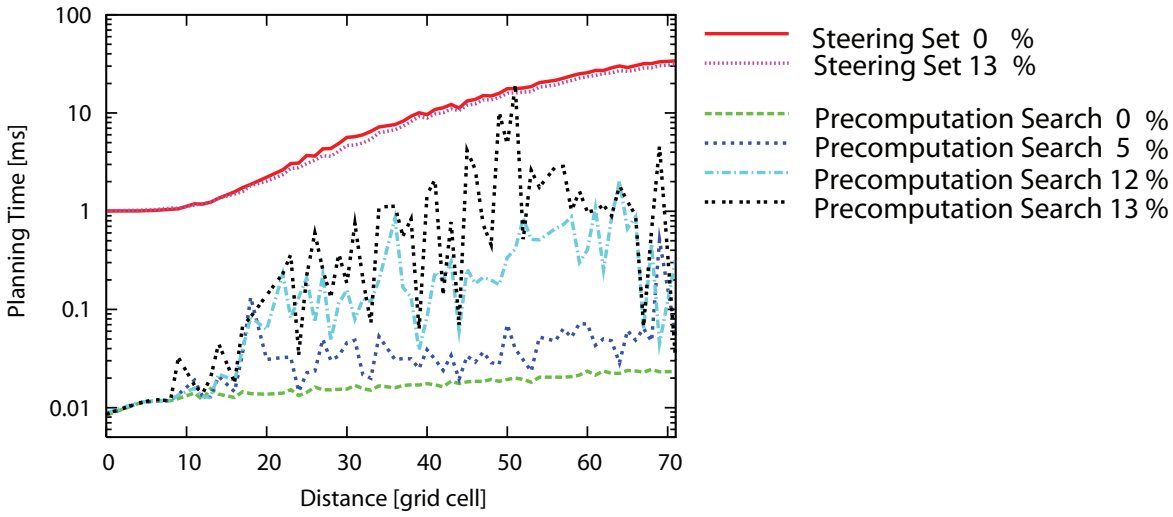


Figure 58. Planning Time of PCS With NSS and Steering Set

6.5.2 PCS Without NSS

Figure 59 shows the extracted results of the planning time for the experiment that did not use NSS and the runtime of the steering set planning without any precomputational techniques. Without NSS, the planning time is slightly shorter than planning with NSS, but it could not find paths greater than 60 gridcells, since the tree without NSS is small.

PCS without NSS is suitable for fast planning in a small space.

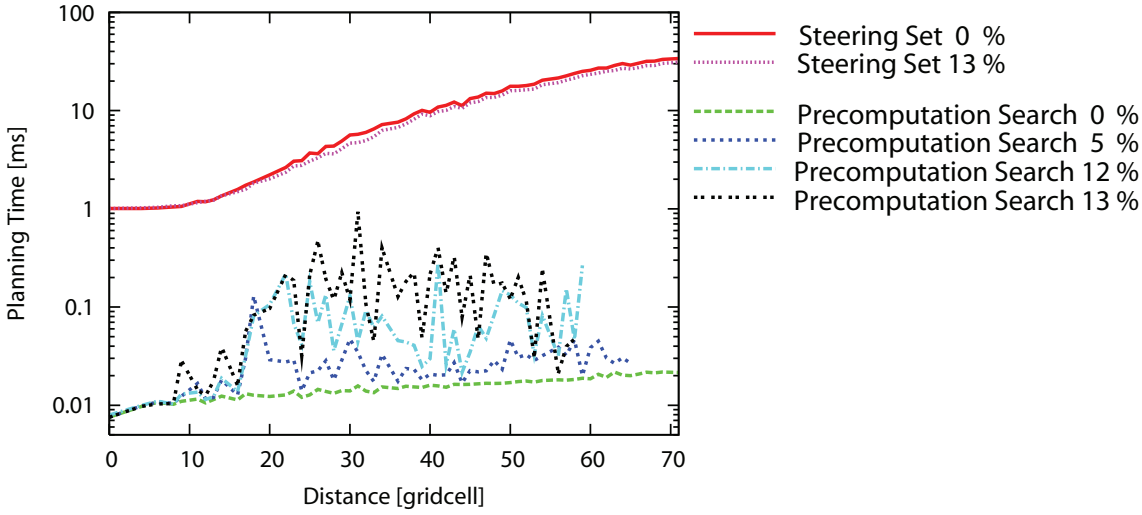


Figure 59. Planning Time of PCS Without NSS and Steering Set

6.6 Path Optimality

This section describes the evaluation of path optimality that is provided by the precomputation planning.

6.6.1 PCS With NSS

The optimality is defined by the percentage of the path length of the precomputation planning as compared to the path length of the existing steering set

planning without precomputation. The existing planning that was introduced in Section 5.1 produces the shortest paths, which are used as the reference values.

The experimental setting is the same as for Section 6.2. Figure 60 shows the path optimality for every obstacle rate from 0% to 20% on the experimental maps. The horizontal axis is the obstacle rate and the vertical axis is path optimality. The asterisk is the average path optimality, and the dotted-line is the standard deviation. The rectangle is the mode value.

As shown in the average path length in Fig. 60, the precomputation planning finds an optimal path that is less than 110% of the length of the shortest path on a map with an obstacle rate of less than 12%. On maps with an obstacle rate from 13% to 20%, the proposed planner finds an optimal path that is less than 120% of the length of the shortest path. Every mode value is 100% of the length of the shortest path in all the trials.

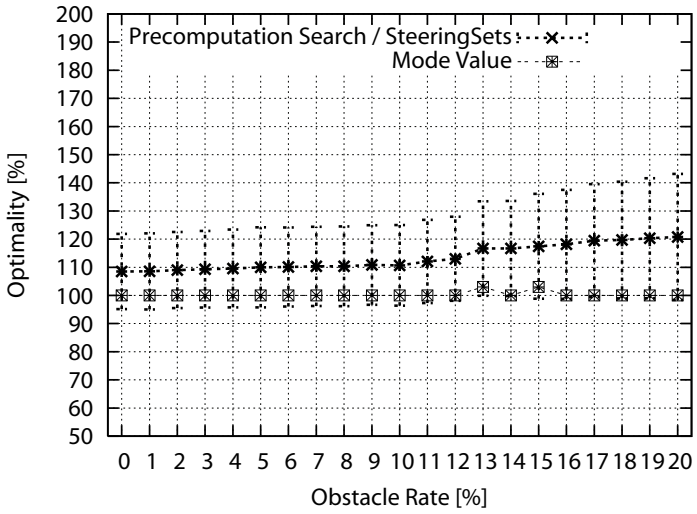


Figure 60. Path Optimality of the PCS With NSS

6.6.2 PCS Without NSS

Fig. 61 shows the path optimality in the case of PCS without NSS as compared to the path length of the existing steering set planning without precomputation. In the case of PCS without NSS, the average optimal path is higher than in the

case with NSS, since PCS without NSS includes many redundant nodes.

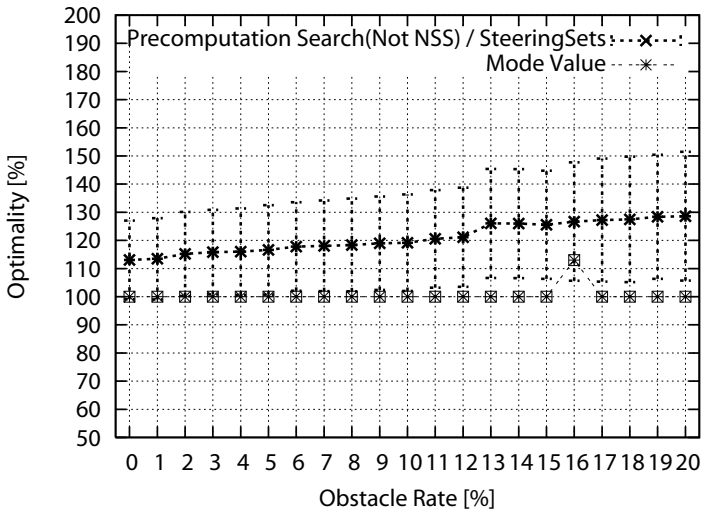


Figure 61. Path Optimality of the PCS Without NSS

6.7 Experiments in an Indoor Environment

The results using real indoor environmental maps are shown in Fig. 62 from A) to D). Figure 62 is the same as the office in Fig. 7 in Section 2. The other experimental settings for the NSS ratio are the same as in Section 6.1. The start position is placed at the center of the maps, and goals are set all over the map.

Figure 63 shows the success maps, in which the black area represents the obstacles, and the light area represents where a planner obtains the path. The proposed planner can successful determine the paths for most areas in A) to D), except for two points in maps A) and C) that have been enclosed by circles, since, in this area, the planner has to pass through a narrow area and to turn into the deep backside of a wall, which PCS cannot perform due to a lack of sufficient branches in the pruned tree.

6.8 Summary

In this chapter, the planning of precomputed search trees with steering set is shown to be effective and to give high-speed performance. The precomputed

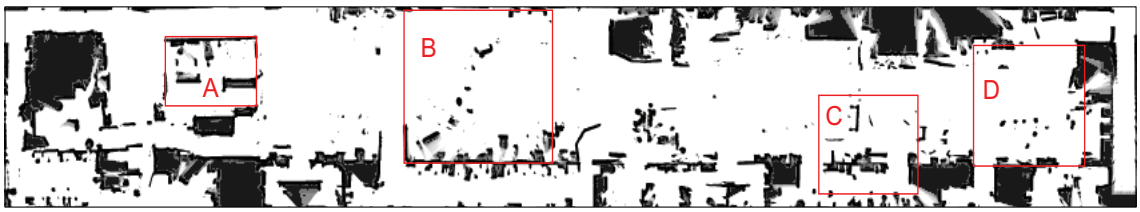
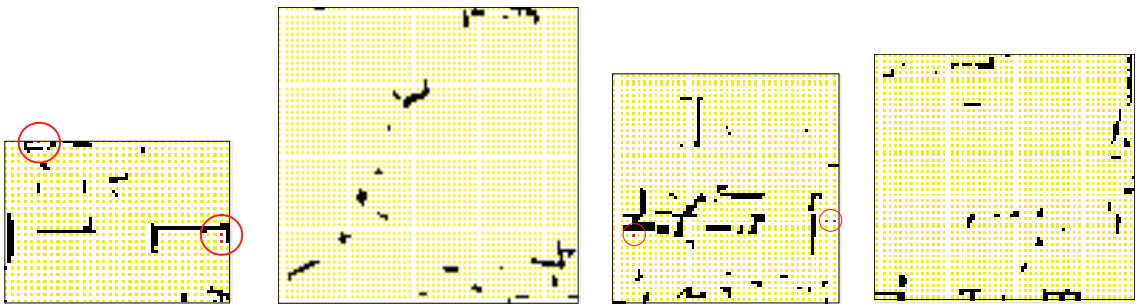


Figure 62. Office Room



A) 64 × 49 [cell] B) 110 × 98 [cell] C) 76 × 76 [cell] D) 92 × 81 [cell]

Figure 63. Success Maps in a Real Indoor Environment
Area enclosed with circles in A) and C) indicates error points, the root is placed in the middle of the maps; yellow: success point, red: error point

search tree is built using pruning based on the rule of constant memory and the maximum size pruning method (MSP), which uses a preset pruning ratio. Using MSP, a large precomputed search tree with reasonable size was obtained. As well, by applying the node selection strategy (NSS) to MSP, the edge of the tree was extended to enhanced path reachability.

Experiments were performed, which showed that the success rate, speed of path searching, and path optimality were as expected.

From the experiments, it can be concluded that the proposed precomputed search tree finds paths in the map with an obstacle rate of no more than 20%. Even when the goal position was set on an incomplete tree that is pruned, the precomputation planning could successfully find an optimal path. The speed and path optimality of the proposed method were evaluated and compared to the results of steering set planning. It was seen that, for an obstacle rate of 13% on a map, the runtime of precomputation planning was more than two orders of magnitude faster than the planning without using a precomputed search tree. It was found that the paths were less than 110% of the length of the shortest path in a real environment. Every mode value was 100% of the shortest path in all the trials. Planning with precomputed search trees produced an optimal path as long as the shortest path. Since the obstacle rate of real indoor environments is less than 5%, the precomputation planning quickly discovers the path in the real world. Its path optimality is only a 10% increase over the steering set planner.

PCS without NSS is suitable for planning on a map that includes a narrow area or a deep backside, while PCS with NSS is suitable for planning in sparse areas. ABBT has a beneficial effect on both PCSs to speed up planning.

7. Conclusions

This dissertation proposed a high speed and efficient path planning for mobile robot systems by considering the following areas. Firstly, the obstacle rates based on the dispersion scale, as well as the distribution of obstacle rates in real indoor environments was examined. Secondly, a pruning method was described that generated an efficient precomputed search tree that allowed an effective heuristic to trackback a path. Thirdly, the accuracy, speed, and optimality of the generalized precomputed search tree were considered. Fourthly, a smooth path planning method using steering sets that allows a wheeled robot to avoid pedestrians was proposed. This proposed smooth path planning method was validated using experiments. Finally, the performance of a precomputed search tree method using steering sets was examined using a real indoor environment.

Contributions of This Thesis This thesis explored appropriate approaches to planning strategies that can be quickly generated. Based on an investigation of real indoor environments, an appropriate obstacle rate was defined and used as a threshold to select the best planning for a given search space.

Gridcell and edge obstacle rates, both of which express map dispersion, were defined for a given search space in order to evaluate the planned path. As well, these obstacle rates can be used to demonstrate the limitations of precomputation planning.

A pruning method that generates an efficient precomputed search tree was developed. This allowed a heuristic to be defined that effectively tracks back the path. This method compares favorably in terms of effectiveness and efficiency with results obtained in forward planning.

As well, a path planning method is presented that can be used to generate trajectories with smooth changes in direction using a predesigned steering sets including three trajectory types. If it is assumed that a prediction can be made of the trajectories of the moving obstacle 's movements, for example a walking human being, the steering set planner generates a smooth path without smoothing that is suitable for use on a wheel robot.

In summary, the results of this thesis suggest an appropriate reference for evaluating path planning and provide efficient guidelines for selecting the best

planning method given the environment.

It should be noted that search space issues were not considered. Thus, if the input is uncertain, then neither forward planning nor precomputed planning can be used independently. Instead, both are run simultaneously and the appropriate method is selected as soon as possible.

Furthermore, the physical aspects of the robot have been omitted in this thesis, since it has been thoroughly considered as a C-Space problem in previous research [50–53].

Obstacle Rate The obstacle rates that were defined in Section 2 indicate the obstacle quantity and concentration in a search space. The obstacle rates express one of the main issues in the planning problem. The difficulties in the search space consist of dispersion and spatial configuration, which is the issue of finding narrow spaces. In this thesis, this problem has not been considered.

Precomputation Planning In the precomputation experiments, a square experimental map with a maximum of 110 gridcells on each side and a tree with several levels that can fit a living room map in the real environment was used. The grid resolution was 10 cm in the experiment.

Paths can be obtained quickly in a large environment by using a compact tree repeatedly, since the performance of the precomputed search tree is known beforehand. As well, it is possible to estimate the runtime and path optimality in other environments.

Using the proposed planner in a multiple goal situation was not considered. In the case of the same start position and various goal positions, the proposed planner does not have an advantage over traced branches, since it is a backtracking path planning from a goal position.

Based on the gridcell and obstacle rates, PCS without NSS is suitable for planning in an area that includes points to turn into the deep backside of linear obstacles. Using NSS creates compact memory and a large expanded PCS that is useful for getting paths quickly in large and uncrowded areas.

In Sections 5 and 6, the planning problems for wheeled robots using steering sets were considered. It was shown that precomputation planning is faster than a forward planner irrespective of the search space.

References

[1] L.E.Kavraki, P.Svestka, J.C.Latombe, and M.H.Overmars, "Probabilisitic roadmaps for path planning in high-demensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, June 1996.

[2] J.D.Boissonnat and M. Yvince, *Algorithmic Geometry*, 1998.

[3] M. de Berg, M.van Kreveld, M.Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Aplications*, 2nd ed., Springer-Verlag, 2000, berlin.

[4] H.Edelsbrunner, *Algorithm in Combinational Geometry*, Springer-Verlag, 1987.

[5] J.G.Hocking and G.S.Young, *Topology*, Dover, 1988.

[6] D.T.Lee and R.K.Drysdale, "Generarization of voronoi diagrams in the plane," *SIAM Journal on Coumputing*, vol. 10, pp. 73–87, 1981.

[7] D.Leven and M.Sharir, "Planning a purely traslational motion for a convex object in two-dimensional space using generalized voronoi diagrams," *Discrete and Computational Geometry*, vol. 2, pp. 9–31, 1987.

[8] M.Sharir, *Algorithmic motion planning*, ser. Handbook of Discrete and Computational Geometry, Chapman and Hall/CRC Press, pp. 1037–1064, 2004.

[9] "Finding narrow passages with probabilisitc roadmaps: The small-step retraction method," *Autonomous Robots*, vol. 19, no. 3, pp. 301–309, 1 2006.

[10] N.J.Nilsson, "A mobile automaton: An application of artificial inteligene techniques," in *1st International Conference on Artificial Intelligence*, 1969, pp. 509–520.

[11] Subir Kumar Ghosh and David M. Mount, "An output sensitive algorithm for computing visibility graphs," in *SFCS '87: Proceedings of the 28th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1987, pp. 11–19.

[12] Joseph S. B. Mitchell, "Shortest paths and networks," pp. 445–466, 1997.

[13] Kurt Konolige and Ken Chou, "Markov localization using correlation," in *IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 1154–1159.

[14] Andrzej Lingas, "The power of non-rectilinear holes," in *Proceedings of the 9th Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 1982, pp. 369–383.

[15] M.Bern, *Algorithmic motion planning*, ser. Handbook of Discrete and Computational Geometry, Chapman and Hall/CRC Press, pp. 563–582, 2004.

[16] J. H. Conway, N. J. A. Sloane, and E. Bannai, *Sphere-packings, lattices, and groups*. New York, NY, USA: Springer-Verlag New York, Inc., 1987.

[17] A.Yershova and S.M.LaVall, "Deterministic samplinf methods for spheres and so(3)," in *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[18] J.Matousek, *Geometric Discrepancy*, Springer-Verlag, 1999, berlin.

[19] M.C.Lin, D.Manocha, J. Cohen, and S.Gottschalk, *Collision detection: Algorithms and applications*, A.K.Peters, pp. 129–142, 1997.

[20] S.Quinlin, "Efficient distance computation between nonvonvex objects," in *IEEE international Conference on Robotics and Automation*, 1994, pp. 3324–3329.

[21] M.C.Lin and J.F.Canny, "Efficient algorithms for incremental distance computation," in *IEEE International Conference on Robotics and Automation*, 1991.

[22] B.Miritich, "V-clip: fast and roust polyhedral collision detection," *Mitsubishi electronics Research Technical Report TR97-05*, 1997.

[23] M.G.Coutinho, *Dynamic Simulations of Mutibody Systems*, Springer-Verlag, 2001, berlin.

[24] N. Nilsson, *Principles of Artificial Intelligence.* Tioga Publishing Company, 1980.

[25] Steven M. LaValle, *Planning Algorithms*, Cambridge University Press, ch. 5, pp. 185–186, 2006.

[26] Simon Thompson and Satoshi Kagami, "Smooth trajectory planning with obstacle avoidance for car-like mobile robots," *The23rd Annual Conference of the Robotics Society of Japan*, 2005.

[27] D.Ferguson and A.Stentz, "The field d* algorithm for improved path planning and replanning in uniform and non-uniform cost environments," *Robotics Institute, CMU*, vol. CMU-RI-TR-05-19, June 2005.

[28] Lotfi A.Zadeh, "Fuzzy sets," *Information and control*, vol. 8, pp. 338–353, 1965.

[29] Philippe Garnier and Thierry Fraichard, "A fuzzy motion controller for a car-like vehicle," in *In Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots andSystems*, 1996.

[30] Yumiko Suzuki, Satoshi Kagami, and James J. Kuffner, "Path planning with steering set for car-like robots and finding an effective set," in *ROBIO,2006 IEEE International Conference*, 2006.

[31] Anthony Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings IEEE International Conference on Robotics and Automation*, May 1994.

[32] ——, "The focussed d* algorithm for real-time replanning," in *Proceedings of International Joint Conference on Artificial Intelligence*, August 1995.

[33] Anthony Stentz and Martial Herbert, "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol. 2, no. 2, pp. 127–145, 1995.

[34] Sven Koenig and Maxim Likhachev, "D* lite," in *Proceedings of the National Conference of Artificial Intelligence (AAAI)*, 2002, pp. 476–483.

[35] Takehiro Horiuchi, Masatomo Kanehara, Satoshi Kagami, and Yoshihiro Ehara, "A probabilistic walk path model focused on foot landing points and human step measurement system," in *SMC,2006 IEEE International Conference*, 2006.

[36] Jae Hoon Lee, Takashi Tsubouchi, Kenjiro Yamamoto, and Saku Egawa, "Pepople tracking using a robot in motion with laser range finder," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2006.

[37] T.Tsubouchi and S.Arimoto, "Behavior of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple moving obstacles," *Robotics and Automation*, vol. 3, pp. 2470–2475, May 1994.

[38] James J. Kuffner and Steven M. LaValle, "Rrt-connect:an efficient aporoach to single-query path planning," in *IEEE Int'l Conf. on Robotics and Automation (ICRA'2000)*, San Francisco, CA, April 2000, pp. 995–1001.

[39] Lydia Kavraki and Jean-Claud Latombe, "Randomized preprocessing of configuration space for path planning: Articulated robots," in *IEEE/RSJ/GI International Conference on Intelligent Robtots and Systems(IROS)*, 1994, pp. 1764–1772.

[40] L. E. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration space for fast path planning." San Diego, CA: IEEE Press, 1994, pp. 2138–2139.

[41] Jared Go, Thuc Vu, and James J. Kuffner, "Autonomous behaviors for interactive vehicle animations," *International Journal of Graphical Models*, 2005.

[42] Emilio Frazzoli, Munther A. Dahleh Y, and Eric Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 25, pp. 116–129, 2002.

[43] Manfred Lau and James J. Kuffner, "Precomputed search trees: Planning for interactive goal-driven animation," in *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Sept. 2006, pp. 299–308.

[44] Makoto Matsumoto and Takuji Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[45] Mutsuo Saito and Makoto Matsumoto, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Springer Berlin Heidelberg, ch. 2, pp. 607–622, 2008.

[46] James J.Kuffner, "Efficient optimal search of euclidean-cost grids and lattices," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, September 2004.

[47] J.A.Reeds and L.A.Shepp, "Optimal path for a car that goes both forwards and backwards," vol. 145, no. 2, 1990.

[48] Jean-Claude Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[49] Takehiro Horiuchi, Simon Thompson, Satoshi Kagami, and Yoshihiro Ehara, "Pedestrian tracking from a mobile robot using a laser range finder," in *Proceedings of 2007 IEEE International Conference on Systems, Man, and Cybernetics(SMC2007)*, Montreal, Canada, 10 2007, pp. 931–936.

[50] T.Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 10, pp. 681–689, 1981.

[51] T.Lozano-Perez and M. A. Wesley, "An alforithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 11, pp. 560–570, 1979.

[52] T.Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transacrions on Computing*, vol. C-32, no. 2, pp. 108–120, 1983.

[53] J.C.Latombe, A. lazanas, and S. Shekhar, *Robot motion planning*, Kluwer, ch. 2, 1991.

Acknowledgements

I am grateful to my advisor, Takeo Kanade, for his support and guidance throughout my graduate school career.

In addition, I am grateful to my advisor, Tsukasa Ogasawara, for his support and guidance throughout graduate school.

Satoshi Kagami inspired me to see the bigger picture of intelligent robotics and guided me through many difficult times with unwavering integrity and wisdom.

Furthermore, I am grateful to my advisors, Masatsugu Kidode and Jun Takamstu, for their useful advice.

I would like to thank James Kuffner, who introduced me to motion planning. This opened up wonderful opportunities for collaboration.

I would also like to thank Simon Thompson, who helped me with the programming and writing English papers.

I would like to thank all the members of the Digital Human Laboratory, especially Koichi Nishiwaki who inspired me to pursue robotics and welcomed me into the lab, and Joel Chestnutts, who introduced me to robotics and provided useful information, advice, and friendship!

I would like to thank my honest, funny friends, Takashi, Osamu, and the Robotics Laboratory people, who inspired, listened to, and cared for me.

Finally, I would like to express my sincere gratitude to my father who raised me to be a engineer.

List of Publications

Publications (Peer-review journals)

1. Yumiko Suzuki, Simon Thompson, Satoshi Kagami. Smooth Path Planning with Pedestrian Avoidance for Wheels Robots (Journal of Robotics and Mechatronics, pp. 21-27, Vol.22 No.1 February 2010)

2. Yumiko Suzuki, Simon Thompson, Satoshi Kagami. High-Speed Planning and Reducing Memory Usage of a Precomputed Search Tree Using Pruning (Journal of Advanced Robotics, pp. 903-920, Vol. 24 No.5-6 April 2010)

3. Yumiko Suzuki, Simon Thompson, Satoshi Kagami. Effectiveness Evaluation of Precomputation Search Using Steering Set (Journal of Robotics and Mechatronics, pp. 112-121, Vol.22 No.1 February 2010)

Presentations (International)

1. Yumiko Suzuki, Simon Thompson, Satoshi Kagami (October 2009) High-Speed Planning and Reducing Memory Usage of a Precomputed Search Tree Using Pruning Proceedings of The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3918–3923, St. Louis, USA, Oct., 2009.

2. Yumiko Suzuki, Simon Thompson, Satoshi Kagami (September 2009) Effectiveness Evaluation of Precomputation Search Using Steering Set Proceedings of 18th IEEE International Symposium on Robot and Human Interactive Communication, pp.613–618, Toyama International Conference Center, Japan, Sep., 2009.

3. Yumiko Suzuki, Simon Thompson, Satoshi Kagami (February 2009) Smooth Path Planning with Pedestrian Avoidance for Wheeled Robots: Implementation and Evaluation Paper presented at the 4th International Conference on Autonomous Robots and Agents, Wellington, New Zealand.

4. Yumiko Suzuki, Satoshi Kagami, James Kuffner (December 2006) Path Planning with Steering Sets for Car-Like Robots and Finding an Effective

Set. Paper presented at IEEE International Conference on Robotics and Biomimetics (ROBIO2006), Kunming, China.

Presentations (Peer-review, Domestic)

1. Satoshi Kagami, Takehiro Horiuchi, Yumiko Suzuki, Simon Thompson, James J.Kuffner (March 2007) Short & Middle Term Path Prediction of Human Being and Path Planning by Avoiding Moving Obstacle. Paper presented at the 12th Robotics Symposia, Niigata, Japan

2. Yumiko Suzuki, Simon Thompson, Satoshi Kagami Effectiveness Evaluation of Recomputed Search Tree (March 2010) Paper presented at the 15th Robotics Symposia, Nara, Japan

Presentations (Domestic)

1. Yumiko Suzuki (August 2009) Effectiveness Evaluation of Precomputation Search Using Steering Set Poster presented at Summer Seminar 2009 of the Digital Human Research Center, Japan

2. Yumiko Suzuki, Simon Thompson, Satoshi Kagami (May 2009) The Effectiveness Evaluation of Precomputation Search Poster presented at JSME Robotics and Mechatronics Conference 2009, Fukuoka, Japan.

3. Yumiko Suzuki (August 2008) Reducing memory usage of a Precomputated Search Tree using Pruning. Poster presented at Summer Seminar 2008 of the Digital Human Research Center, Japan

4. Yumiko Suzuki, Satoshi Kagami, Simon Thompson (June 2008) Reducing memory usage of a Precomputated Search Tree using Pruning. Poster presented at JSME Robotics and Mechatronics Conference 2008, Nagano, Japan.

5. Yumiko Suzuki, Simon Thompson, Satoshi Kagami, James J.Kuffner (September 2007) Precomputed Search Tree with Steering Sets. Paper presented at the 25th Annual Conference of the Robotics Society of Japan, Chiba Institute of Technology, Japan

6. Yumiko Suzuki (August 2007) Precomputed Search Tree with Steering Sets Poster presented at Summer Seminar 2007 of the Digital Human Research Center, Japan

7. Yumiko Suzuki, Simon Thompson, James J.Kuffner, Satoshi Kagami (December 2006) Path Planning with Steering Sets for Car-Like Robots in the Presence of Moving Obstacles Poster presented at the 7th Annual Conference of System Integration Division in The Society of Instrument and Control Engineers ,Sapporo ,Japan

8. Yumiko Suzuki, Satoshi Kagami, James J.Kuffner(September 2006) Path Planning with Steering Sets for Car-Like Robots and Finding an Effective Set. Paper presented at the 24th Annual Conference of the Robotics Society of Japan, Okayama University, Japan.

9. Yumiko Suzuki (August 2006) Path Planning with Steering Sets for Car-Like Robots. Summer Seminar 2006 of the Digital Human Research Center, Japan

10. Yumiko Suzuki, Satoshi Kagami, James J.Kuffner (May 2006) Path Planning with Steering Sets for Car-Like Robots. Poster presented at JSME Robotics and Mechatronics Conference 2006, Waseda University, Japan.