Article

Hierarchical Path-Planning for Mobile Robots Using a Skeletonization-Informed Rapidly Exploring Random Tree*

Hyejeong Ryu

Department of Mechatronics Engineering, Kangwon National University, Chuncheon KR24341, Korea; hjryu@kangwon.ac.kr; Tel.: +82-33-250-6376

Received: 26 September 2020; Accepted: 3 November 2020; Published: 5 November 2020

Abstract: An efficient, hierarchical, two-dimensional (2D) path-planning method for large complex environments is presented in this paper. For mobile robots moving in 2D environments, conventional path-planning algorithms employ single-layered maps; the proposed approach engages in hierarchical inter- and intra-regional searches. A navigable graph of an environment is constructed using segmented local grid maps and safe junction nodes. An inter-regional path is obtained using the navigable graph and a graph-search algorithm. A skeletonization-informed rapidly exploring random tree* (SIRRT*) efficiently computes converged intra-regional paths for each map segment. The sampling process of the proposed hierarchical path-planning algorithm is locally conducted only in the start and goal regions, whereas the conventional path-planning should process the sampling over the entire environment. The entire path from the start position to the goal position can be achieved more quickly and more robustly using the hierarchical approach than the conventional single-layered method. The performance of the hierarchical path-planning is analyzed using a publicly available benchmark environment.

Keywords: path-planning; mobile robots; rapidly exploring random tree; grid map segmentation; environmental graph structure

1. Introduction

Path-planning for mobile robots involves identifying an efficient path from a start position to a goal position on a given environmental map using online sensor data. The path-planning algorithms are divided into those facilitating local and global planning. Local path-planning (“obstacle avoidance”) prevents collisions, even when unexpected (unregistered) obstacles are encountered [1,2]. Global path-planning generates waypoints that connect the start to the goal using the given map [3,4]. In practice, local and global plannings are integrated as the robot proceeds via various waypoints to a goal, while the local planner calculates the actions to arrive at each waypoint using current sensor data. Therefore, the efficient waypoints in safe regions that can constitute the shortest path are essential for higher level applications.

Most global planners initially extract safe/navigable regions from a global map and then find an appropriate path between the safe regions. Safe regions can be represented as empty/unoccupied cells of a grid map or as the nodes of a graph. Grid map-based planning is applicable in two-dimensional (2D) environments because it is possible to directly recognize safe regions and their relationships when planning possible paths between them via regularly decomposed grid cell representation [5,6]. However, to solve scalability issues in high-dimensional applications with limited computational power and time, the grid-based planners require additional effort like the cost approximation [7].

Graph-based methods were developed to handle path-planning in large environments. The computational burdens are lower than those of grid-based methods. The two most widely used approaches are the probabilistic roadmap (PRM) [8] and the rapidly exploring random tree (RRT) [9,10]. These are termed sampling-based methods, because they sample/generate nodes stochastically in empty regions [11,12]. PRM planners first construct a graph (a roadmap) that covers the entire environment and includes collision-free paths. Then, they compute the shortest path from a start position to a goal along the roadmap. Although the multiple-query approach of a PRM planner aids path-planning in complex or high-dimensional spaces [13], many practical path-planning problems can be solved using a single-query approach such as that of an RRT. RRT planners basically expand a tree via random sampling of empty regions from the start position and stop when the tree attains the goal. The RRT variants differ in terms of how they optimize the initial feasible solution or how they modify the tree structure. RRT-Connect simultaneously grows two RRTs rooted at the start and the goal toward each other until they meet, reducing the computational time required for a converged solution [14]. RRT* asymptotically converges to an optimal solution by additionally rewiring the tree [15,16]. RRT*-Connect combines RRT-Connect and RRT*, yielding asymptotically optimal solutions faster than either algorithm alone [17]. Although various RRT algorithms deal with high-dimensional and complex state spaces by ensuring probabilistic completeness and delivering optimal asymptotic solutions, convergence to the optimal solution is slow for environments with narrow passages and when non-solution areas in the entire configuration space are (unnecessarily) subjected to dense sampling [18,19]. The informed RRT* (IRRT*) was developed to improve the convergence rate and the final solution quality by restricting the sampling region to an ellipsoid defined by the solution of the previous optimization step [20]. It obtains a final optimal solution in less time than the basic RRT*. However, as the IRRT* employs a vanilla RRT* sampling method (the sampling target is the entire space) to achieve an initial solution, the IRRT* suffers from problems similar to those of the RRT*.

To overcome the slow convergence rate and reduce the overall processing time by quickly obtaining an initial solution, we previously developed a skeletonization-informed RRT* (SIRRT*) for mobile robots moving in 2D environments [21]. The IRRT* obtains the initial solution using the RRT* algorithm. By contrast, the SIRRT* generates an initial solution by skeletonizing a grid map and constructing a tree on the skeletonized image using Prim's algorithm [22]. As the convergence rate of the IRRT* algorithm depends on the quality of the initial solution computed by random sampling of the RRT* over the whole space, the IRRT*'s performance is unstable even when the start and goal positions do not change within the same environment. The SIRRT* utilizes the deterministic property of grid map image skeletonization to find the initial solution more robustly and rapidly than the IRRT*. The total computational time required to attain a final solution by the SIRRT* is lower and more repeatable than that required by the IRRT*.

The global path-planning approaches mentioned above assume that an environmental map is available and process the entire map. Thus, path-planning is performed in a single-layer manner. When a part of the map is changed, replanning includes both the unchanged and changed regions. Moreover, we can encounter the situation when the current start and goal positions differ slightly from the start and goal positions of the previously solved problem. The conventional path-planners require sample nodes covering the entire map, although most of the two final solutions may be identical. If we can divide the map and obtain the adjacency between map segments, we can efficiently handle such situations. Sampling-based planners require ever more samples (associated with longer computational times) of larger environments to obtain optimal solutions. If we can construct an environmental graph that contains local path information for each segment and if the graph can be re-used to deal with future path-planning problems, the sampling space can be focused to compute paths more efficiently. Thus inspired, we here propose a hierarchical path-planning method using the SIRRT*.

Earlier global planners have sought to guarantee optimal solutions and reduce computational times; few studies have explored hierarchical path-planning. The focused hierarchical D* (FHD*) is a grid-based,

hierarchical, path-planning algorithm seeking to optimize paths by placing bridge nodes on the potential field of the grid map [23]. A three-level hierarchical structure exploiting particle swarm optimization was suggested to yield feasible, safe, and optimal paths [24]. However, its hierarchy means the algorithmic structure is composed of map decomposition and path optimization; the method does not reveal the hierarchical geometric structure of the environment. A hierarchical roadmap (HRM), similar to what we develop here, can be incrementally constructed via normalized graph-cut segmentation of a local sonar grid map [25].

Here, a two-layered, hierarchical path-planning method is developed. The combined method of the previous work, the SIRRT*, and the well-known image segmentation method, the watershed segmentation, is presented. A global path, thus a sequence of path regions from a start region to a goal region, is firstly obtained through the segmented grid map. Within each path region, the SIRRT* is applied to compute a local path between two junction nodes. An entire path from the start to the goal is generated by combining these local paths. Figure 1 illustrates the proposed hierarchical path-planning method. The local grid maps (A, B, and C) and the junction nodes (J1, J2) between adjacent local maps constitute a graph. The local paths within local grid maps, defined via intra-regional searching, are combined using the inter-regional path. The proposed approach can be applied to most mobile robots operated in 2D environments, including wheeled mobile robots and legged mobile robots. The proposed hierarchical planning is a kind of divide-and-conquer method, and it suggests the required sequence and its corresponding components from environmental segmentation to database construction. The contribution of this paper lies in the overall sequence for hierarchical path-planning. Although divide-and-conquer or local-metric/global-topological approaches have already been discussed in other research areas, we need to consider additional components to apply them to practical path-planning problems. Especially, a navigable graph structure, which is a database of local paths, is constructed by the SIRRT*. The previous works, for SIRRT* planning, show robust performance for multiple running applications, and it is efficient for constructing the navigable graph structure.

Figure 1. Hierarchical path-planning.

The remainder of the paper is organized as follows. In Section 2, a navigable graph of an environment is constructed; we segment the grid map and extract junction nodes between two adjacent segments. In Section 3, the hierarchical path-planning algorithm using the navigable graph is developed; we employ the Dijkstra graph-search for inter-regional path-planning and the SIRRT* intra-regional path-planning to this end. Section 4 describes the experimental results using a well-known (benchmark) environment to verify the efficacy of the proposed hierarchical approach. Section 5 contains conclusions and future directions.

2. Navigable Graph of an Environment

This section shows how to construct a navigable graph of an environment. The grid map image is firstly segmented using a watershed segmentation algorithm. Then, junction nodes lying on edge grid cells separating adjacent segments are defined.

2.1. Grid Map Segmentation

Many studies have explored environmental segmentation, and most have sought to construct topological maps by extracting the nodes and edges of empty areas defined using various metrics by reference to a generalized Voronoi graph (GVG) [26–28]. Using the positions of nodes, the environment can be easily segmented so that each grid cell is assigned to the nearest node; cells assigned to the same node are in the same segment. Other studies have focused more on dividing a grid map [29]. RoomSeg is a watershed, grid map segmentation method used to guide autonomous cleaning robots in large environments and is a practical end-to-end implementation even if the grid map is noisy [30]. It is possible to divide a consecutive, local sonar grid map into two regions via cell decomposition and the normalized graph cut method and incrementally segment the environment to construct a connected graph [31,32].

A watershed algorithm that is easily implemented and that yields reliable segmentation results is applied to the grid map segmentation in this paper. We engage in hierarchical path-planning using a navigable graph of segmented local grid maps; we do not engage in grid map segmentation per se. The watershed transform detects catchment basins in a grayscale image by considering the image as a topographic relief; the height of each pixel is proportional to its gray level [33]. As the grid map is considered a black (occupied)-and-white (empty) binary image, it must be converted to impart various intensities to empty cells. A distance transform operator is used to calculate the distance to the closest boundary (an occupied cell) from each empty cell. Figure 2a shows a grid map image of a public benchmark environment [34], and the grid map image is obtained by constructing a grid map using laser measurements and converting it to a binary image. Figure 2b shows the normalized distance transform image computed using the Euclidean distance metric.

Figure 2. Cont.

Figure 2. Grid map segmentation using the watershed algorithm: (a) The grid map image. (b) The normalized distance transform. (c) Over-segmentation after application of the basic watershed algorithm. (d) Markers labeled using the connected component labeling algorithm; the different colors indicate different labels. (e) Predetermined markers for marker-controlled watershed segmentation; the white areas are undetermined.

Use of the basic watershed segmentation algorithm can cause over-segmentation. Figure 2c shows the segmentation result of the distance transform image of Figure 2b, computed using the basic watershed algorithm. To overcome this problem, a marker-controlled, watershed segmentation algorithm has been developed [33,35]. This uses markers to indicate initial segment information, thus the number and approximate positions of objects or foregrounds that can be definitely distinguished from backgrounds. Markers are computed using morphological operations such as erosion and dilation. Prior to the watershed application to the grid map, the definite background pixels (occupied cells) are easily recognized from the grid map image; this is not the case for other gray-scale images. When path-planning, we are interested in safe navigable areas (empty cells) and calculate the distance transforms for all such cells. The pixels lying further from occupied cells are considered as definitive foreground segments. The empty cells that exceed the threshold value of the normalized distance transform serve as marker regions. Markers are labeled using the connected components labeling (CCL) algorithm (which assigns unique labels to all pixels in each connected segment). Figure 2d shows the labeled markers; these are combined with the empty cell information of Figure 2e. The marker-controlled watershed algorithm gradually grows regions and decides the segment to which each non-labeled empty cell belongs. Marker-controlled segmentation is shown in Figure 3a.

Figure 3. The navigable graph structures of the segmented grid maps: (a) The segmented grid maps and the junction nodes (yellow dots). (b) The adjacency matrix of grid map segments. (c) A navigable graph.

2.2. Junction Node Extraction

The watershed segmentation algorithm produces several grid map segments with different labels, separated by edges/boundaries one pixel in width. To construct a navigable graph using the segments, the junction nodes between adjacent segments must be defined. In the proposed path-planning method, these nodes serve as sub-goals/waypoints, allowing movement to the next path regions, defined as local regions through which the robot must pass.

Algorithm 1 shows how to extract junction nodes, *Node*, and how to construct the adjacency matrix, *Adj*, of the grid map segments. Algorithm 1 requires two inputs, $gridmap_{dist}$ and $gridmap_{WS}$. $gridmap_{dist}$ is the normalized distance transform of the grid map, and $gridmap_{WS}$ is the watershed segmentation result where each pixel value shows the label identification of the corresponding segment. The algorithm first inspects all edges of cells, $grid_{edge}$, in the watershed segmentation result and combines the edge cells that share neighbors into a matricized form, Adj^{cells} , wherein each component is a group of cells lying between the same adjacent segments (Lines 4–10). *Adj* is the adjacency matrix among grid map segments. If there are M segments, the size of the symmetric matrix *Adj* is M . If there are any edge cells between two segments i and j , $adj_{i,j}$ and $adj_{j,i}$ are one (Line 8). Each component $adj_{i,j}^{cells}$ of Adj^{cells} stores all edge cells between two segments i and j (Line 7).

For each group of edge cells, we inspect and obtain the connected edge segments using the CCL (Line 13), which can deal with the situation in which two adjacent grid map segments can be connected via several separate edge areas. The junction node should be the safest grid cell; thus, we select the cell exhibiting the maximum, normalized distance transform value with respect to one of the connected edge segments (Line 16). Finally, we obtain the junction nodes and the adjacency matrix of the grid map segments (Line 20).

Algorithm 1 Junction node extraction.

Require: $gridmap_{dist}$, $gridmap_{ws}$

Ensure: junction nodes $Node = \{node_{i,j}^n | 1 \leq n \leq N, N \text{ is the number of junction nodes.}\}$, adjacency

matrix among grid map segments Adj

```

1: function SELECT-JUNCTION-NODE( $gridmap_{dist}$ ,  $gridmap_{ws}$ )
2:   Set  $Adj^{cells} = \{adj_{i,j}^{cells} | adj_{i,j}^{cells} = \emptyset, 1 \leq i, j \leq M, M \text{ is the number of segments.}\}$ 
3:   Set  $Adj = \{adj_{i,j} | \text{the identity matrix of size } M \times M, M \text{ is the number of segments.}\}$ 
   ▷ Grouping edge cells that have the two same neighboring segments and calculating the adjacency
4:   for all  $grid_{edge} \in gridmap_{ws}$  do
5:     Inspect eight neighbor cells around  $grid_{edge}$ .
6:     if there are two different labeled neighbors, and their labels are  $i$  and  $j$  then
7:        $adj_{i,j}^{cells} \leftarrow adj_{i,j}^{cells} \cup \{grid_{edge}\}$ ,  $adj_{j,i}^{cells} \leftarrow adj_{i,j}^{cells}$ 
8:        $adj_{i,j} \leftarrow 1$ ,  $adj_{j,i} \leftarrow adj_{i,j}$ 
9:     end if
10:  end for
11:   $n \leftarrow 0$ 
   ▷ Selecting junction node cells for each group of edge cells in  $Adj^{cells}$ 
12:  for all non-empty  $adj_{i,j}^{cells} \in Adj^{cells}$  do
   ▷  $adj_{i,j}^{cells,1} \cup \dots \cup adj_{i,j}^{cells,K} = adj_{i,j}^{cells}$ ,  $K$  is the number of disconnected edge segments in  $adj_{i,j}^{cells}$ .
13:     $\{adj_{i,j}^{cells,1}, \dots, adj_{i,j}^{cells,K}\} \leftarrow \text{connected\_component\_labeling}(adj_{i,j}^{cells})$ 
14:    for all  $adj_{i,j}^{cells,k}$  do
15:       $n \leftarrow n + 1$ 
      ▷ The safest cell with the maximum distance to obstacles is selected as a junction node.
16:      Select one cell  $grid_{d_{max}}$  in  $adj_{i,j}^{cells,k}$  with the maximum  $gridmap_{dist}$ .
17:       $node_{i,j}^n \leftarrow grid_{d_{max}}$ 
18:    end for
19:  end for
20:  return  $Node$ ,  $Adj$ 
21: end function

```

Figure 3 shows the graph structure of the segmented grid map. In Figure 3a, the yellow dots are the junction nodes, and the numbers identify the segments. Figure 3b is the corresponding adjacency matrix.

In Figure 3a, Segments “6” and “7” are structurally connected in two separate places. Two segments in Figure 3c feature two edges and two junction nodes.

3. Hierarchical Path-Planning

In this section, the hierarchical path-planning method using the graph-search algorithm and the SIRRRT* is developed. Previously, a navigable graph of the environment consisting of local grid map segments, the adjacency matrix, and the junction nodes between all adjacent segments was constructed. We now use the SIRRRT* to compute intra-regional paths among the junction nodes of each grid map segment. Figure 4 shows the SIRRRT* result for Segment “9”. The initial solution path is computed using the Harris corner nodes of the skeletonized grid map image and Prim’s algorithm (Figure 4a–c). The final path is obtained via informed sampling optimization (Figure 4d).

Figure 5 shows the overall procedure of the proposed method. The first layer in path-planning is an inter-regional search using the navigable graph structure of the segmented grid maps; we find a start region and a goal region (A). Then, the inter-regional path from the start region to the goal region is computed using the adjacency matrix and the graph-search algorithm (B). The Dijkstra graph-search algorithm is employed to this end; all adjacent segment pairs are equally weighted during the inter-regional search. When the inter-regional path is defined, the junction nodes are selected (C) to yield the SIRRRT* paths among those nodes for each path region. At this time, several junction nodes for two adjacent segments can be selected because obstacles may lie between the segments. Thus, two adjacent segments can be connected at several nodes. To ensure that the junction node sequence from the start to the goal is efficient, we construct a cost matrix. The SIRRRT* yields the intra-regional paths from the start to the junction nodes of the start region and from the goal to the junction nodes of the goal region (D). The cost matrix is calculated by the costs of the SIRRRT* paths (E). By using the cost matrix and the Dijkstra graph-search algorithm, the node sequence from the start position to the goal position can be computed (F). The graph-search algorithm selects the cheapest junction nodes. Finally, a global path is achieved by combining the intra-regional paths with reference to the node sequence (G).

Figure 4. Cont.

Figure 4. An example of local path-planning for Segment “9” using the skeletonization informed rapidly exploring random tree* (SIRRT*): (a) Skeletonization of the grid map segment. (b) Local nodes (blue dots) extracted by the Harris corner detector; the red dot is the start position; the green dot is the goal. (c) The initial tree (blue line) and the initial path (yellow line). (d) The final tree (blue line) and the optimized path (red line).

Algorithm 2 is the pseudocode of the hierarchical path-planning, and it describes the procedure after obtaining the inter-regional path from the start region, seg_S , to the goal region, seg_G . Seg is the sequence of the path regions according to the inter-regional path, and K is the number of the path regions. Algorithm 2 also requires the navigable graph structure including the junction nodes, $Node_{JC}$, and the intra-regional paths, $Path$, of all segments. The junction nodes, $Node_{path}$, are selected according to the inter-regional path, Seg , in Lines 2–5. The intra-regional paths, $Path_S$, from the start position to the junction nodes of the start region are computed by the SIRRT* (Lines 6–10). The intra-regional paths, $Path_G$, from the goal position to the junction nodes of the goal region are also computed by the SIRRT* (Lines 11–15). The cost matrix, $Cost_{JC}$, among the selected junction nodes, the start position, and the goal position is obtained in Lines 16–26. If M is the number of the selected junction nodes, $Node_{path}$, the size of the symmetric cost matrix is $M + 2$. The cost values between the start/goal to the selected junction nodes should be added to the cost matrix. Using the cost matrix, the Dijkstra graph-search algorithm computes the node sequence, $Node_{SG}$, from the start to the goal. Here, the indices of the start and the goal in the cost matrix are $M + 1$ and $M + 2$, respectively (Line 27). By connecting the intra-regional paths in series, we obtain the final path, $Path_{SG}$ (Line 28).

Figure 5. The steps of hierarchical path-planning.

Figure 6a,c,e shows the hierarchical path-planning results derived using three different starts and goals, and Figure 6b,d,f is the corresponding cost matrices for the selected junction nodes along the inter-regional paths. As shown in Figure 6a, the inter-regional path is $16 \rightarrow 10 \rightarrow 9 \rightarrow 6 \rightarrow 7$. The cyan squares are the selected junction nodes, or $Node_{path}$ of the path regions in Algorithm 2. As Segment “7” is connected to Segment “6” through two junction nodes, both nodes were selected when computing the cost matrix. The second-to-last entries of the cost matrix are the SIRRT* distances from the start to the junction nodes of the start region and the last entries the SIRRT* distances from the goal to the junction nodes of the goal region. Such path-planning chooses the most efficient junction nodes when computing a global path using the cost matrix and the graph-search algorithm.

Figure 6. Hierarchical path-planning results: (a,c,e) The global paths from starts (red dots) to goals (green dots); the cyan squares are the selected junction nodes. (b,d,f) The cost matrix for the chosen junction nodes, the start, and the goal.

Algorithm 2 Hierarchical path-planning.

Require: the inter-regional path from the start region, seg_S , to the goal region, seg_G , $Seg = \{seg_k | 1 \leq k \leq K, seg_1 = seg_S, seg_K = seg_G\}$, junction nodes $Node_{JC} = \{node_{i,j}^n | 1 \leq n \leq N\}$, where N is the number of junction nodes, $node_{i,j}^n$ is the junction node between seg_i and seg_j , and the SIRRT* paths among the junction nodes $Path = \{path_{p,q}^{seg_i} | p, q \in Node_{JC}, seg_i \in Seg\}$

Ensure: the final path $Path_{SG}$ that is a sequence of waypoints from the start to goal positions

- 1: **function** HIERARCHICAL SIRRT*($Seg, Node_{JC}, Path$)
- 2: Set $Node_{path} = \emptyset$
 ▷ Selecting the junction nodes ($Node_{path}$) according to the inter-regional path
- 3: **for** $seg_k \in Seg, 1 \leq k \leq K - 1$ **do**
- 4: Find all $node_{k,k+1}^n \in Node_{JC}$. $Node_{path} \leftarrow Node_{path} \cup \{node_{k,k+1}^n\}$
- 5: **end for**
 ▷ Computing the intra-regional path ($Path_S$) from the start position to the junction nodes
- 6: Set $Path_S = \emptyset$.
- 7: **for all** $node_{1,k}^n \in Node_{path}$ **do**
- 8: Compute the SIRRT* path $path_{S,n}^{seg_1}$ from the start position to $node_{1,k}^n$ of $seg_1 = seg_S$.
- 9: $Path_S \leftarrow Path_S \cup \{path_{S,n}^{seg_1}\}$ ▷ $n \in Node_{path}$
- 10: **end for**
 ▷ Computing the intra-regional path ($Path_G$) from the goal position to the junction nodes
- 11: Set $Path_G = \emptyset$.
- 12: **for all** $node_{k,K}^n \in Node_{path}$ **do**
- 13: Compute SIRRT* path $path_{n,G}^{seg_K}$ from the goal position to $node_{k,K}^n$ of $seg_K = seg_G$.
- 14: $Path_G \leftarrow Path_G \cup \{path_{n,G}^{seg_K}\}$ ▷ $n \in Node_{path}$
- 15: **end for**
 ▷ Calculating the cost matrix among the selected junction nodes and start/goal positions
- 16: Set a zero-matrix $Cost_{JC} = \{cost_{i,j} | 1 \leq i, j \leq M + 2\}$
- 17: **for** $seg_k \in Seg, 2 \leq k \leq K - 1$ **do**
- 18: Find all pairs of $node_{k-1,k}^i, node_{k,k+1}^j \in Node_{path}$. ▷ $Node_{path} \subset Node_{JC}$
- 19: $cost_{i,j}$ and $cost_{j,i} \leftarrow$ the cost of $path_{i,j}^{seg_k} \in Path$
- 20: **end for**
- 21: **for all** $path_{S,n}^{seg_1} \in Path_S$ **do**
- 22: $cost_{n,M+1}$ and $cost_{M+1,n} \leftarrow$ the cost of $path_{S,n}^{seg_1}$
- 23: **end for**
- 24: **for all** $path_{n,G}^{seg_K} \in Path_G$ **do**
- 25: $cost_{n,M+2}$ and $cost_{M+2,n} \leftarrow$ the cost of $path_{n,G}^{seg_K}$
- 26: **end for**
- 27: A node sequence $nodes_S$ to $node_G$, $Nodes_{SG} \leftarrow$ Dijkstra_Graph_Search($Cost_{JC}, M + 1, M + 2$)
- 28: Obtain the final path, $Path_{SG}$, by combining the intra-regional paths according to $Nodes_{SG}$.
- 29: **return** $Path_{SG}$
- 30: **end function**

4. Experimental Results

The proposed hierarchical path-planning method was analyzed to verify its stable and efficient performance. All experiments were performed on an Intel Core i7-9700K (eight cores of 3.6 GHz) with 32 GB of memory in the same benchmark environment; several rooms were connected by corridors, as in the real world.

The SIRRT* was compared with the IRRT* to verify that the SIRRT* is more efficient in constructing the proposed navigable graph structure that needs multiple intra-regional planning runs among junction nodes. The stability of the SIRRT* in terms of computational time and the final costs is shown in Table 1. The start and goal positions are given in Figure 6a. The IRRT* and SIRRT* performed 1000, 2000, and 3000 informed sampling optimizations after obtaining an initial solution. For each optimization number, one-hundred experiments were performed. In the SIRRT*, the initial solution is deterministically computed by skeletonization employing the Harris corner detector; random sampling is not in play. However, the IRRT* algorithm uses RRT* sampling to achieve an initial solution throughout the entire environment. The total number of IRRT* iterations is the sum of the RRT* samplings required until an initial solution is obtained and the number of informed samplings required for optimization.

Table 1. A comparison of the IRRT* and the SIRRT*.

		# of Iterations for Optimization					
		1000		2000		3000	
		IRRT*	SIRRT*	IRRT*	SIRRT*	IRRT*	SIRRT*
Computational time (s)	Mean	8.01	0.51	8.74	0.83	7.46	1.19
	Std.	5.86	0.08	6.59	0.11	7.39	0.12
	Min.	1.02	0.37	1.45	0.59	1.23	0.93
	Max.	32.08	0.72	40.08	1.14	49.04	1.44
Final cost (pixels)	Mean	1441.27	1555.33	1451.28	1535.60	1442.94	1516.71
	Std.	61.90	16.53	64.49	21.65	54.54	20.52
	Min.	1389.36	1506.31	1388.81	1475.96	1390.23	1465.13
	Max.	1680.39	1583.75	1676.32	1577.36	1630.01	1563.89
# of total iterations	Mean	7611.26		7469.60		7106.37	
	Std.	2684.09		2353.94		2895.54	
	Min.	3758.00		3610.00		3669.00	
	Max.	16,027.00		15,124.00		20,554.00	

As the IRRT* performs the RRT* sampling to obtain an initial solution, the computational time is negatively affected. If initial solution computation is slow, the number of RRT* samplings increases, as does the computational time. The mean IRRT* computational times were 8.01, 8.74, and 7.46 s. However, the differences were not significant; the SDs were very large (5.86, 6.59, and 7.39 s). Thus, the IRRT* performed very differently during each trial, although the start and goal positions were the same. By contrast, the SIRRT* computational times were faster and more stable (means 0.51, 0.83, and 1.19 s) than those of the IRRT*. The SDs were relatively small at 0.08, 0.11, and 0.12 s. The SIRRT* generated initial solutions faster than did the IRRT*, and the total computational time for the final solution was lower and more robust than that of the IRRT*. The final costs of the SIRRT* were relatively larger than those of the IRRT*. However, the IRRT* SDs were larger than those of the SIRRT*. Sampling-based

algorithms such as the RRT variants are more likely to achieve an optimal solution as the number of samples increases. The larger number of the total IRRRT* iterations reduced the mean final cost, but at the expense of large SDs. Constructing the navigable graph structure requires conducting the local planning algorithm multiple times to obtain the intra-regional paths among junction nodes. Therefore, the SIRRT* that shows more consistent results (low SDs) is efficient for the hierarchical planning approach.

The hierarchical path-planning method consists of two phases: we construct a navigable graph of the environment featuring grid map segmentation and the SIRRT*-mediated path computation among the junction nodes of each segment, then we engage in inter- and intra-regional path-planning using the graph. Table 2 shows the computational times required for the construction of navigable environmental graphs using different numbers of the SIRRT* optimization iterations. The number of such iterations is the number of ellipsoidal-informed samplings performed after obtaining an initial solution via skeletonization. For each fixed iteration number of the SIRRT* optimization, thirty experiments were performed. The environment was divided into 20 local segments, and twenty-eight junction nodes were registered. As all intra-regional paths were defined using the SIRRT*, the SIRRT* algorithm totally ran 117 times for each experiment. Therefore, the computational times in Table 2 reflect 117 SIRRT* path-plannings. The SIRRT* algorithm exploits the deterministic feature of skeletonization and the random nature of the RRT*, thus affording more stable performance than the other RRT* variants. Even after many trials, the standard deviations (SDs) were small; it is appropriate to construct the navigable graph via multiple computations of local paths in each grid map segment.

Table 2. Computational times (s) for constructing a navigable graph structure when the numbers of path optimization iterations varied.

# of Iterations for Optimization								
1000			2000			3000		
Mean (Std.)	Min.	Max.	Mean (Std.)	Min.	Max.	Mean (Std.)	Min.	Max.
44.33 (0.90)	42.70	45.78	108.13 (1.89)	104.82	111.36	196.05 (4.48)	189.12	202.16

Table 3 shows the hierarchical path-planning results including the inter- and intra-regional path-planning. The starts and goals are those of Table 1. The pre-constructed navigable graph featured 3000 SIRRT* iterations; we performed 100 runs for each iteration number of the SIRRT* optimization. The iteration number of the SIRRT* optimization reveals intra-regional paths from the start and goal positions to the junction nodes. Computationally, global path calculation time increased as the number of iterations increased. When sampling-based planners were used, the time required and the sample number increased as the search area increased. In the experiments of Table 1, we searched the entire environment; optimization required many samples. However, hierarchical intra-regional searches of start and goal positions were performed on environmental segments. Intra-regional searches do not require large numbers of samples. After only 100 iterations, the converged final path was apparent. Cost was reduced if the navigable graph featured over 3000 iterations. Moreover, the proposed hierarchical planning based on the SIRRT* yielded smaller SDs.

Table 3. Hierarchical path-planning using the pre-built map; 3000 iterations.

		# of Iterations for Optimization						
		50	100	200	500	1000	2000	3000
Computational time (s)	Mean	0.28	0.36	0.56	1.49	3.93	11.20	21.94
	Std.	0.02	0.02	0.03	0.06	0.07	0.14	0.28
	Min.	0.25	0.32	0.52	1.40	3.83	11.06	21.51
	Max.	0.32	0.40	0.63	1.63	4.07	11.57	21.51
Final cost (pixels)	Mean	1420.60	1420.00	1420.00	1420.00	1420.00	1420.00	1420.00
	Std.	2.18	0.00	0.00	0.00	0.00	0.00	0.00
	Min.	1420.00	1420.00	1420.00	1420.00	1420.00	1420.00	1420.00
	Max.	1431.00	1420.00	1420.00	1420.00	1420.00	1420.00	1420.00

Figure 7 shows the comparison of the IRRT*, the SIRRT*, and hierarchical path-planning (the H-SIRRT*). Each graph component presents the mean, minimum, and maximum values for 100 runs. As mentioned earlier, because the results of the IRRT* have large differences for each run, it cannot be suitable for practical real-time applications. Moreover, even with the low number of optimization iterations after obtaining the initial solution, the IRRT* required too many iterations to achieve the initial solution, making the total computational time high. As shown in Figure 6a, the start region and the goal region have two junction nodes. The hierarchical path-planning includes the inter-regional graph-search and four intra-regional path-plannings using the SIRRT*. Therefore, the computational time of the hierarchical planning is higher than that of the SIRRT*. The sampling areas of the hierarchical planning's intra-regional planning are restricted as the small parts of the entire environment. In contrast, the sampling area of the SIRRT* is the whole environment. The hierarchical planning can achieve the lower cost solution using only a lower number of optimization iterations.

Figure 7. A comparison of the IRRT*, the SIRRT*, and the hierarchical path-planning using SIRRT* (H-SIRRT*): (a) Computational time. (b) Final cost.

5. Conclusions

A hierarchical path-planning method using a navigable graph structure of the environment evaluated via the SIRRT* was proposed in this paper. Conventional global path-planning algorithms find

optimal paths by processing entire single-layer environments. In the proposed approach, paths are identified on the intrinsic hierarchical structure; inter- and intra-regional searches are performed. The navigable, environmental graph structure using local map segments and safe junction nodes is defined in the hierarchical path-planning. The inter-regional paths from start regions to goal regions are revealed when navigable graph structures are explored using the Dijkstra graph-search algorithm; the intra-regional paths are computed using the SIRRT*. The hierarchical approach assured stable and efficient performance. The SIRRT* was compared with the IRRT*; the SIRRT* was optimal for the proposed hierarchical path-planning because the SDs of the computational times were lower than those of the IRRT*. Experimentally, hierarchical path-planning using the SIRRT* (over various iterations) afforded efficient, final path convergences (from the start to the end) using few iterations. The proposed method will aid stable and efficient path-planning in large environments.

The proposed method assumes a general global planning situation when a robot finds a global path from the start to the goal before it starts to navigate. Although we can use local planners in dynamic environments, global planners also need to cope with dynamic situations in real time. The hierarchical approach can efficiently handle such dynamic situations because it can re-plan the path by performing the intra-regional planning in the changed area and connecting the new intra-regional path with the inter-regional path. Detecting dynamic obstacles and integrating them into the navigation framework in highly dynamic environments need to be investigated in the future. In addition, applying the proposed approach to non-grid-based maps can be valuable for a wide range of practical applications.

Funding: This research was supported by a project of the Korean government (No. 10073166).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Molinos, E.J.; Llamazares, Á.; Ocaña, M. Dynamic window based approaches for avoiding obstacles in moving. *Robot. Auton. Syst.* **2019**, *118*, 112–130. [CrossRef]
2. Kim, Y.; Kwon, S. A heuristic obstacle avoidance algorithm using vanishing point and obstacle angle. *Intell. Serv. Robot.* **2015**, *8*, 175–183. [CrossRef]
3. Pérez-Higueras, N.; Jardón, A.; Rodríguez, Á.; Balaguer, C. 3D Exploration and Navigation with Optimal-RRT Planners for Ground Robots in Indoor Incidents. *Sensors* **2020**, *20*, 220. [CrossRef] [PubMed]
4. Kang, G.; Kim, Y.B.; Lee, Y.H.; Oh, H.S.; You, W.S.; Choi, H.R. Sampling-based motion planning of manipulator with goal-oriented sampling. *Intell. Serv. Robot.* **2019**, *12*, 265–273. [CrossRef]
5. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **2010**, *29*, 485–501. [CrossRef]
6. Sturtevant, N.R. Benchmarks for grid-based pathfinding. *IEEE Trans. Comput. Intell. Games* **2012**, *4*, 144–148. [CrossRef]
7. Carsten, J.; Ferguson, D.; Stentz, A. 3d field d: Improved path planning and replanning in three dimensions. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; IEEE: New York, NY, USA, 2006; pp. 3381–3386.
8. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [CrossRef]
9. LaValle, S.M. Rapidly-exploring random trees: A new tool for path planning. *Comput. Sci. Dept. Oct.* **1998**, *98*. Available online: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1853> (accessed on 5 August 2020).
10. Wang, X.; Luo, X.; Han, B.; Chen, Y.; Liang, G.; Zheng, K. Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm. *Appl. Sci.* **2020**, *10*, 1381. [CrossRef]
11. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]

12. Li, D.; Li, Q.; Cheng, N.; Song, J. Sampling-based real-time motion planning under state uncertainty for autonomous micro-aerial vehicles in GPS-denied environments. *Sensors* **2014**, *14*, 21791–21825. [[CrossRef](#)] [[PubMed](#)]
13. Marble, J.D.; Bekris, K.E. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE Trans. Robot.* **2013**, *29*, 432–444. [[CrossRef](#)]
14. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000; IEEE: New York, NY, USA, 2000; Volume 2, pp. 995–1001.
15. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime motion planning using the RRT. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; IEEE: New York, NY, USA, 2011; pp. 1478–1483.
16. Salzman, O.; Halperin, D. Asymptotically near-optimal RRT for fast, high-quality motion planning. *IEEE Trans. Robot.* **2016**, *32*, 473–483. [[CrossRef](#)]
17. Klemm, S.; Oberländer, J.; Hermann, A.; Roennau, A.; Schamm, T.; Zollner, J.M.; Dillmann, R. RRT*-Connect: Faster, asymptotically optimal motion planning. In Proceedings of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 6–9 December 2015; IEEE: New York, NY, USA, 2015; pp. 1670–1677.
18. Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 299. [[CrossRef](#)]
19. Noreen, I.; Khan, A.; Ryu, H.; Doh, N.L.; Habib, Z. Optimal path planning in cluttered environment using RRT*-AB. *Intell. Serv. Robot.* **2018**, *11*, 41–52. [[CrossRef](#)]
20. Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; IEEE: New York, NY, USA, 2014; pp. 2997–3004.
21. Ryu, H.; Park, Y. Improved informed RRT* using gridmap skeletonization for mobile robot path planning. *Int. J. Precis. Eng. Manuf.* **2019**, *20*, 2033–2039. [[CrossRef](#)]
22. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2009.
23. Seder, M.; Mostarac, P.; Petrović, I. Hierarchical path planning of mobile robots in complex indoor environments. *Trans. Inst. Meas. Control.* **2011**, *33*, 332–358. [[CrossRef](#)]
24. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Appl. Soft Comput.* **2017**, *59*, 68–76. [[CrossRef](#)]
25. Park, B.; Choi, J.; Chung, W.K. Incremental hierarchical roadmap construction for efficient path planning. *ETRI J.* **2018**, *40*, 458–470. [[CrossRef](#)]
26. Thrun, S. Learning metric-topological maps for indoor mobile robot navigation. *Artif. Intell.* **1998**, *99*, 21–71. [[CrossRef](#)]
27. Burgard, W.; Moors, M.; Stachniss, C.; Schneider, F.E. Coordinated multi-robot exploration. *IEEE Trans. Robot.* **2005**, *21*, 376–386. [[CrossRef](#)]
28. Beeson, P.; Jong, N.K.; Kuipers, B. Towards autonomous topological place detection using the extended voronoi graph. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; IEEE: New York, NY, USA, 2005; pp. 4373–4379.
29. Bormann, R.; Jordan, F.; Li, W.; Hampp, J.; Hägele, M. Room segmentation: Survey, implementation, and analysis. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; IEEE: New York, NY, USA, 2016; pp. 1019–1026.
30. Kleiner, A.; Baravalle, R.; Kolling, A.; Pilotti, P.; Munich, M. A solution to room-by-room coverage for autonomous cleaning robots. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; IEEE: New York, NY, USA, 2017; pp. 5346–5352.

31. Choi, J.; Choi, M.; Nam, S.Y.; Chung, W.K. Autonomous topological modeling of a home environment and topological localization using a sonar grid map. *Auton. Robot.* **2011**, *30*, 351–368. [CrossRef]
32. Choi, J.; Choi, M.; Chung, W.K. Topological localization with kidnap recovery using sonar grid map matching in a home environment. *Robot. -Comput.-Integr. Manuf.* **2012**, *28*, 366–374. [CrossRef]
33. Grau, V.; Mewes, A.; Alcaniz, M.; Kikinis, R.; Warfield, S.K. Improved watershed transform for medical image segmentation using prior information. *IEEE Trans. Med Imaging* **2004**, *23*, 447–458. [CrossRef]
34. Howard, A. The Robotics Data Set Repository (Radish). 2003. Available online: <http://radish.sourceforge.net/> (accessed on 4 June 2020).
35. Parvati, K.; Rao, P.; Mariya Das, M. Image segmentation using gray-scale morphology and marker-controlled watershed transformation. *Discret. Dyn. Nat. Soc.* **2008**, *2008*, 384346. [CrossRef]

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).