2022 IEEE International Conference on
Systems, Man, and Cybernetics (SMC)

Learning Hierarchical Traversability Representations for Efficient Multi-Resolution Path Planning *

Reza Etemadi Idgahi[1] and Manfred Huber[1]

2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC) | 978-1-6654-5258-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SMC53654.2022.9945075

***Abstract*—Path planning on grid-based obstacle maps is an essential and much-studied problem with applications in robotics and autonomy. Traditionally, in the AI community, heuristic search methods (e.g., based on Dijkstra, A*, or random trees) are used to solve this problem. This search, however, incurs a high computational cost that grows with the size and resolution of the obstacle grid and has to be mitigated with effective heuristics to allow path planning in real-time. This work introduces a learning framework using a deep neural network with a stackable convolution kernel to establish a hierarchy of directional traversability representations with decreasing resolution that can serve as an efficient heuristic to guide a multi-resolution path planner. This path planner finds paths efficiently, starting on the lowest resolution traversability representation and then refining the path incrementally through the hierarchy until it addresses the original obstacle constraints. We demonstrate the benefits and applicability of this approach on datasets of maps created to represent both indoor and outdoor environments to represent different real-world applications. The conducted experiments show that our method can accelerate path planning by 40% in indoor environments and 65% in outdoor environments compared to the same heuristic search method applied to the original obstacle map, which demonstrates the effectiveness of this method.**

I. INTRODUCTION

Path planning on a static grid is a well-known and common problem in AI and Robotics for which a large variety of solution methods have been proposed. These methods can be divided broadly into two categories. One is global path planners that compute a complete path before starting navigation, such as Dijkstra, A*, or Rapidly Exploring Random Trees (RRT). The other is local path planners, which determine local navigation actions without first computing a complete path, such as potential fields or reactive navigation strategies.

Local path planning generally requires less computation time and is able to work while only having knowledge of the local environment around the current position. However, these methods are prone to fail and get stuck in local minima and thus may not be able to find a path.

On the other hand, global path planners rely on heuristic search in the state-space induced by the grid cells and can guarantee to find a path given sufficient information about the environment. However, since they require significantly more information, these methods often struggle to find a path in a reasonable amount of time as the size of the maps increase. While efficient heuristics can accelerate path construction in these algorithms, the derivation of effective heuristics for a specific environment is a significant challenge. Most traditional approaches thus use generic heuristics that are largely independent of the environment, such as Cartesian Distance or Search Node Density, thus still requiring large amounts of time in cluttered and ill-structured environments. This huge drawback makes these methods challenging for applications with stringent time constraints.

This work introduces a novel method to derive a hierarchical traversability representation using a stackable convolutional neural network architecture that can serve as an efficient heuristic for global pathfinding approaches. The proposed approach finds a path between two positions on a large map while reducing the time of path planning. To do this, we translate the obstacle grid map into a directional traversability representation. We then learn a model in a supervised fashion that can reduce the resolution of traversability grid maps and create a new set of traversability maps of smaller size that preserves most navigation information. This model is created using a convolutional neural network that can be expanded without retaining variable-size inputs while achieving a constant factor size reduction. This enables us to stack the learned convolutional kernel an arbitrary number of times without additional re-training, yielding an increasingly deep network and creating a growing hierarchy of abstracted maps of increasingly smaller size. On this hierarchy, we then apply a global path planner in a multi-resolution framework, starting by finding a path in the smallest size (i.e., coarsest resolution) abstracted map and then incrementally refining it on the increasingly higher resolution maps on each level of the hierarchy, including the original grid map which results in the final path. The use of the traversability hierarchy here ensures that paths found on low-resolution maps retain most aspects of the true path and can thus be easily refined, yielding faster overall planning times than planning on the full resolution map. To demonstrate this, we show results using both Dijkstra's algorithm and A* for path planning and refinement.

The main contribution of this work is to demonstrate that the model is able to learn a new hierarchical representation of physical features in a map in terms of obstacle and non-obstacle cells. The learned model allows the algorithm to reduce the maps' resolution accurately to perform path planning in a shorter time. Furthermore, our experiments show that our algorithm is more efficient than the plain Dijkstra or A* and can significantly reduce the overall time of finding a path in a grid map.

II. RELATED WORK

A variety of search algorithms have been proposed in the context of path planning, such as A* [1], Anytime Repairing A* [2], Rapidly-exploring Random Tree (RRT) [3], Hybrid-A* [4], and two-stage RRT [5]. Generally, these methods use generic, handcrafted heuristics, such as Manhattan distance, Euclidean distance, and length of Reeds-Shepp paths. Additionally [6] shows that cost functions can be created

*This work was supported in part by NSF under grant IIS-1724248

[1]Reza Etemadi Idgahi and Manfred Huber are with the Department of Computer Science and Engineering at the University of Texas at Arlington, Arlington, TX, 76019-0015. reza.etemadiidgahi@mavs.uta.edu, huber@cse.uta.edu

from the topology of the state and can be learned from demonstration [7]. Although it is possible to achieve optimal solutions with a well-founded heuristic, the complexity of the environments can make these methods unpractical, especially when the length of the path or the dimension of the state space increases. ARA* [2] reduces the complexity by using a scaling factor and adjusting the upper bound of the cost of a path. Hybrid A* [4] uses the maximum of two different heuristics to guide grid cell expansion; however, it neither preserves completeness nor guarantees an optimal solution. Two-stage RRT [5] utilizes two RRTs where an upper RRT produces waypoints, which guide the other lower RRT to address the robot kinematics.

Researchers have developed several hierarchical path planning methods to improve the quality of the planned path in complex environments [8], [9], [10], [11], [12]. Fujimura and Samet [11] proposed a hierarchical system for path planning in an environment with moving obstacles, in which time was included as one of the dimensions of the model world. However, this approach leads to having a large search space. In [12], a hierarchical approximate cell decomposition method was introduced where different resolution decompositions were used. Additionally, path planners based on fuzzy systems and optimization methods have been introduced to overcome the challenging properties of environments such as unknown and dynamic areas [8], [13]. In [13], a multi-objective path planner based on particle swarm optimization is proposed for navigation in uncertain environments. By contrast, [8] proposed a hierarchical planning strategy with two layers, where fuzzy logic was used for motion planning. However, the optimization and generalization abilities of these planners still need to be improved.

Fueled by the increasing popularity of artificial neural networks, the AI community has started to utilize these computational models in path planning and to use their powerful features such as strong function approximation. In particular, deep reinforcement learning [14] which is a machine learning framework for sequential decision making, has been used to learn a policy that can generate the shortest path. Prior RL algorithms, like Q-learning and Sarsa [14], work well with discrete state spaces. Konar et al. [15] have applied Q-learning to path planning of a mobile robot. However, if the state spaces become very large or continuous, these algorithms will be computationally expensive and impractical. In [16], [17], [18], [19] researchers introduced various approximation methods to deal with large or continuous state spaces. Inspired by the least-squares temporal difference learning algorithm [16], Lagoudakis and Parr [17] proposed least-squares policy iteration (LSPI), in which linear architectures were used to approximate the value functions in continuous state spaces. In [20] a hierarchical planning approach based on A* and LSPI is presented in which a two-level structure was used. In the first level, the A* algorithm was used to find several path points as subgoals for the next level and in the second level, LSPI was used to learn a near-optimal local planning policy. However, these methods require significant training in order to be practical for real-world applications.

In contrast, the approach presented here uses deep neural networks to explicitly learn reductions of the world model that can be used as a heuristic by a path planner to reduce complexity. The goal here is to combine the benefits of a global path planner in terms of completeness and correctness with the advantages of machine learning to identify traversability patterns that indicate successful path attributes.

III. METHOD

The method developed here attempts to provide efficient guidance to a hierarchical path planning and refinement approach by creating a cascade of increasingly lower resolution representations of the environment that maintain its main characteristics in terms of path construction with minimal refinement requirements. For this purpose, the maps must maintain broad traversability characteristics rather than obstacle locations or densities. For example, while an orchard of trees and a field with walls might have the same average obstacle density, the traversability of the orchard is relatively universal (i.e., there is a path through the orchard in any direction), in contrast to the traversability of the walled field which is severely limited. To abstract these two pieces of environment in a way that maintains approximate paths, they need to be represented in a way that their characteristics are preserved. To capture this, we propose the concept of directional traversability maps and utilize supervised learning with convolutional networks to identify the features that indicate particular traversability characteristics.

To address varying size maps, the developed learning approach utilizes a convolutional kernel that reduces the resolution of the traversability maps by a constant factor. Using a convolutional kernel allows us not only to expand the layer to arbitrary input sizes but also to stack the same kernel to form a deeper network, producing increasingly coarser-resolution traversability maps with each additional layer.

Using this hierarchy of representations, path planning works back up the hierarchy, starting by constructing a path in the lowest resolution (and thus smallest size) map and then incrementally refining it by considering the next higher resolution representation. Figure 1 shows an overview of the heuristic construction and path planning approach.

The obstacle map (top left) is translated into a set of directional traversability maps, which are then transformed into incrementally smaller size representations by applying the convolutional network until the desired smallest size is reached (illustrated left to right at the top). Once this representation hierarchy is formed, the path planner is executed at the bottom of the hierarchy (bottom right), and the result is incrementally refined as moving back up the hierarchy until a path is derived on the original obstacle map (illustrated right to left at the bottom).

A. Directional Traversability Maps

Obstacle maps represent the environment in a grid, where each cell can be either 0, meaning obstacle, or 1, meaning non-obstacle. We introduce the concept of directional traversability maps as a new way of representing the environment where each cell can have a value between 0 and 1, representing its traversability score. Each obstacle map can be converted to 6 traversability maps, each representing traversability scores in a specific direction, namely horizontal, vertical, top-left, top-right, bottom-left, and bottom-right.

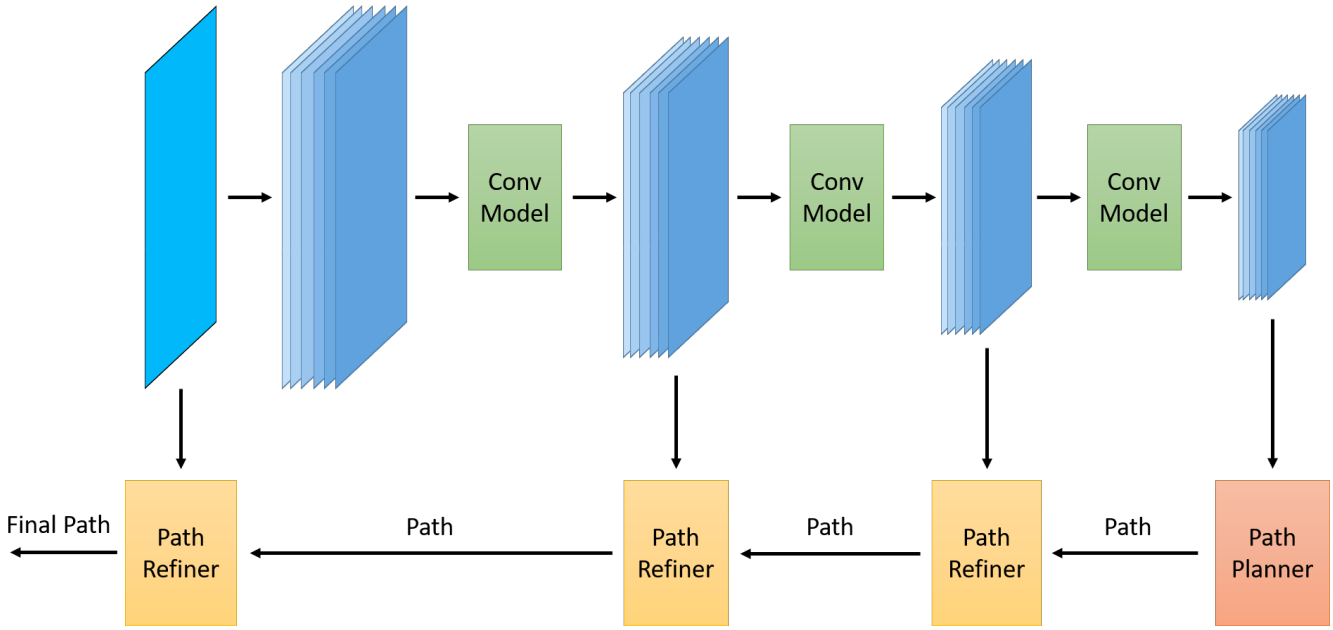

Fig. 1. Overview of the proposed hierarchical, multi-resolution path planning approach. Construction of the reducing resolution directional traversability cascade (top from left to right) is followed by bottom-up path planning and refinement (bottom from right to left).

The traversability score for each direction represents the degree to which a path exists that traverses the underlying patch of the environment in the corresponding direction. Figure 2 shows an example of an obstacle map region and the corresponding traversability representations.

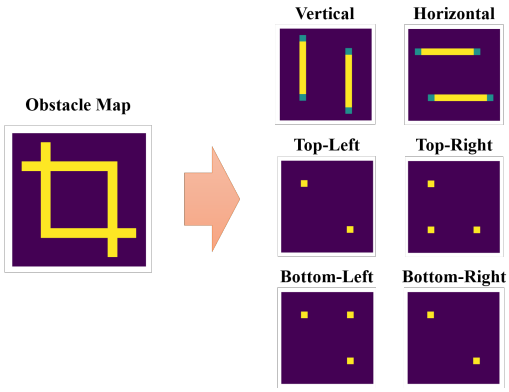

Fig. 2. Obstacle map and corresponding directional traversability representations for the six traversal directions. Purple, yellow, and green colors show obstacles (non-traversable), free (fully traversable), and partially traversable areas, respectively.

To manually convert an obstacle map to equal-sized traversability maps, we calculate the traversability scores for each cell in the obstacle map. If the cell is an obstacle, then traversability scores in all directions are 0. Otherwise, we look at the pair of adjacent neighbors of that cell in each direction. For example, the left and right neighbors are considered for horizontal, while the top and bottom for vertical traversability. The traversability score in each direction is set to 0 if both neighbors are obstacles, 1 if both are non-obstacles, and 0.5 if one is an obstacle and the other is a non-obstacle.

B. Dataset

To evaluate the approach in varied settings and derive training data for the learning approach to hierarchical resolution reduction of directional traversability maps, an environment generator was developed that constructs arbitrary worlds with particular characteristics. We created two sets of 256x256 resolution maps to represent indoor and outdoor environments. Indoor maps consist of rooms and hallways with few random obstacles representing furniture or other items in a building. By contrast, outdoor maps consist of forest, field, and jungle patches which are large numbers of small obstacles scattered through an area with different densities, as well as random walls and obstacles of a bigger size. During dataset creation, parameters such as the size of hallways, the number of random obstacles, and the size and density of the jungles were specified randomly to represent a variety of settings. Figure 3 shows one example map for each of the two sets.

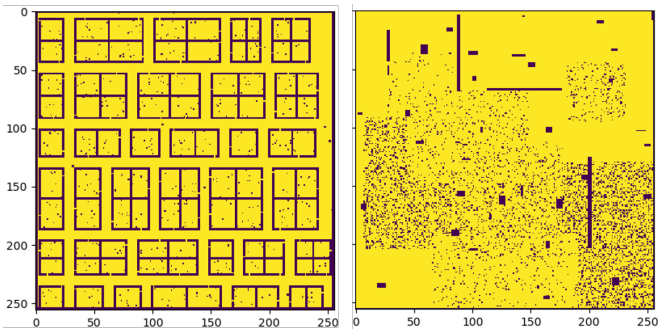

Fig. 3. Examples of obstacle maps for indoor environments (left) and outdoor environments (right).

C. Model for Map Size Reduction

a) Model Architecture: The core of the proposed work is the construction of a cascade of resolution (and thus size)

reduced directional traversability maps representing the same environment. To achieve this, we trained a convolutional kernel to reduce 4x4 patches of the traversability maps to abstracted 2x2 patches while preserving the main traversability characteristics. The convolution kernel can be replicated in a variable-size convolution layer covering the entire input map, making it flexible to train and apply on arbitrarily large maps. To avoid underfitting, the proposed model utilizes two convolutional layers. The first layer has three 4x4 filters with a stride of 2 and relu activation functions for each directional traversability map. The outputs of this layer are then stacked, and as a result, the input traversability maps with a size of WxHx6 are converted to W/2xH/2x18 maps. The second layer is a 3D convolutional layer with six 1x1x18 filters with a stride of 1 and clipped relu as the activation function to generate valid traversability scores between 0 and 1. In the end, the output of this model is six directional traversability maps with half resolution of the input. Figure 4 shows the architecture of the proposed model.

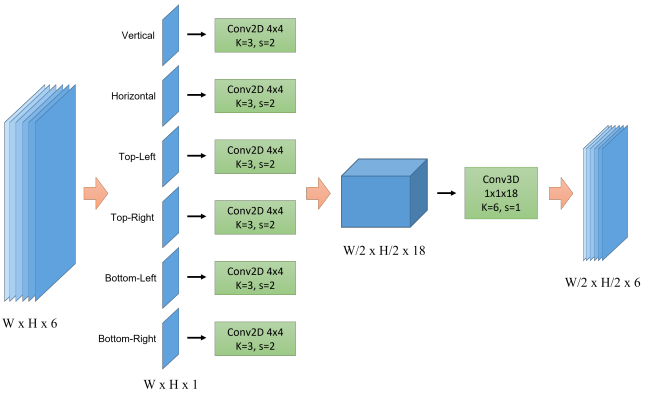

Fig. 4. The CNN model architecture used to reduce the resolution of the traversability maps.

b) Training Data Generation: To train the convolutional network for one step of size reduction, we generated a training set of obstacle maps and their traversability representation (as explained in III-A), with corresponding resolution-reduced traversability labels provided by an automated labeling process. As mentioned in Section III-B, the obstacle maps in this work are 256x256 grids. Therefore, the generated labels are six 128x128 maps, where each cell represents the traversability score of the corresponding 4x4 patch on the obstacle map in the given direction. Each patch shares two columns or rows with its neighbor patch. Having this overlapped area ensures that the traversability across multiple patches is reflected in their traversability scores. This score is a real number between 0 and 1, where 0 means there is no path in the specified direction in that patch and 1 means there are paths from any point on one side to any point on the other side in the specified direction. Values in between represent the fraction of location pairs on the entrance and exit side of the area that can be connected through the area with a collision-free path. Loosely this corresponds to a measure indicating how easy it would be to find a path traversing the area in the given direction.

c) Hierarchical Scale Reduction: Using the derived training data, we train a convolutional kernel that can efficiently derive a traversability representation of the same area but with half the resolution. The convolution kernel can be replicated in a variable-size convolution layer covering the entire input map, making it flexible to train and apply on arbitrarily large maps. This, together with the convolutional kernel utilizing the same input and output representation in terms of six traversability values, enables us to stack multiple layers with identical kernels to achieve an incremental reduction in size down to the desired target size. Therefore we can train the model once and then use it for all levels of abstraction, minimizing the training required while obtaining an approach that is applicable to arbitrary size maps. The concept of this multi-layer application is shown as an overview in Figure 1 and on an example map in Figure 5.

D. Hierarchical Path Planning

After creating the dataset, the supervised learning approach is used to train a model to convert traversability maps to lower resolution maps while maintaining the essential features for path planning. This model enables us to use it on a map multiple times in a hierarchy to reach a very small map where a global path planner can perform reasonably well. In this paper, we use Dijkstra's algorithm to explain the planning and refinement principles and then employ both Dijkstra and A* in the experiments to demonstrate the generality and benefit of the approach. After a path at the lowest resolution has been found, it is incrementally refined in the context of the higher resolution traversability maps until a path in the original environment is found. For this, it is necessary to have a version of the path planner that can be executed on the traversability maps. We developed a modified version of Dijkstra that can find a path in the traversability maps and then refine it at the next hierarchy level, where more map details are available.

a) Modified Dijkstra: Dijkstra is used here with modifications to allow the use of traversability maps. The input is six directional traversability maps with values between 0 and 1. To cope with this multi-dimensional input, the following heuristic function is introduced in order to include the traversability scores:

$$c_{dir} = d + \lambda(1 - t_{dir}) \tag{1}$$

where c_{dir} is the traversal cost of the current cell in direction dir, d is the length of the current shortest path from the start to the entry of the current cell, t_{dir} is the traversability score of the current cell in this direction, and λ specifies the importance of the traversability score compared to the length of the path. If the traversability score is 0, that cell will not be considered for path planning because it is either unreachable or untraversable, meaning we can reach it from one side but not exit from the other side. This rule has an exception at the start and end where we only want to reach the cells and not traverse them.

b) Path Refinement: After a path is found on a low-resolution map, it should be refined on the next higher-resolution map to become a valid path. The method we used for path refinement consisted of reducing the search space and re-running the path planner. As such, instead of searching the entire map, the path planner only scans a small portion of it corresponding to the found path to

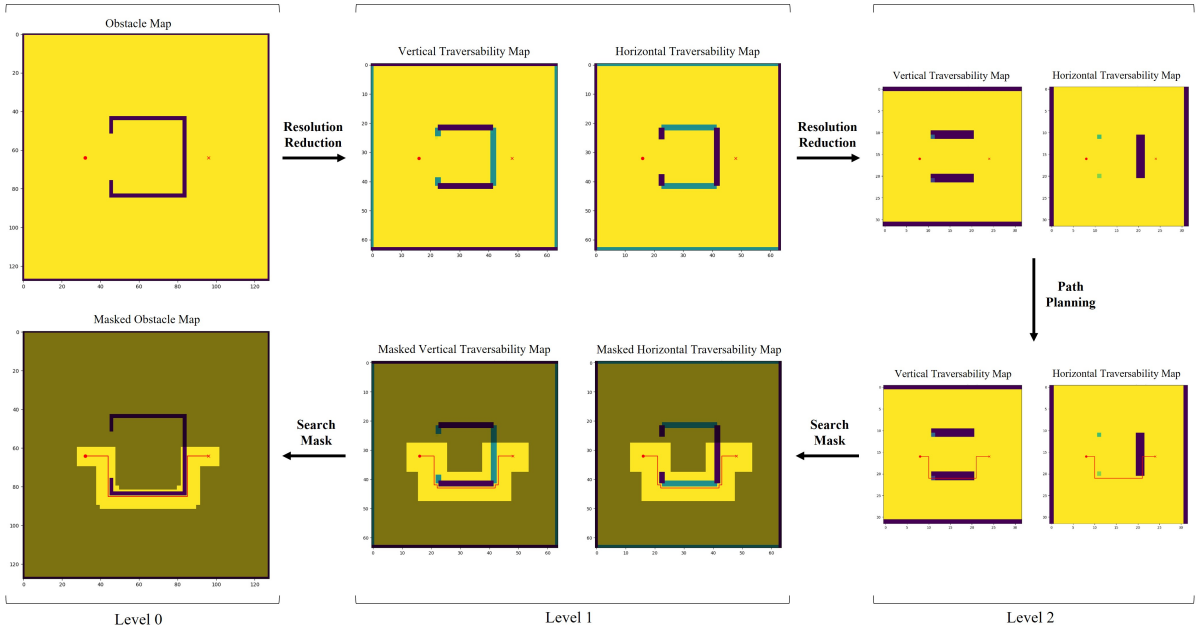

Fig. 5. Example of the hierarchical path planning approach. Hierarchical scale reduction of directional traversability maps is shown at the top (left to right) with corresponding path construction and refinement at the bottom (right to left). Traversability scores are indicated by yellow, green, and brown colors, and the refined path is shown in red. The search space used for path refinement is highlighted on the masked maps, which acts as a heuristic showing a path most likely exists in that area based on the information from previous level. For example, in the last step at level 0, the path planner only searches 9.8% of the map and will quickly find the path.

quickly refine it according to the newly available details. Since the map dimensions are reduced by two at each level, each pixel at level $l+1$ corresponds to a 2x2 window at level l. Therefore, to form the search area, for each pixel and its surrounding pixels on the path at level $l+1$, the corresponding 2x2 windows at level l are added to the search area. The number of surrounding pixels considered for the search area is controlled by a hyper-parameter called margin. Having a large margin helps mitigate the path errors caused by the loss of detail in the low-resolution maps. However, it makes the search area larger and can slow down the path planner. The search area is represented in the form of a mask that sits on the maps and restricts the search space. After that, the path planner is executed to find a path in the specified space. If there is no path, the mask will be removed, and it continues to search the entire map until it finds a valid path. An example of the search mask is shown in Figure 5 on the bottom left map, where the highlighted and shadowed areas represent the inside and outside of the search space.

Algorithm 1 presents a coarse description of the overall approach to hierarchical traversability map-based multi-resolution path planning, and Figure 5 shows an example of the map cascade and hierarchical path refinement with 3 levels.

Algorithm 1 Hierarchical Path Planning

function HPP($obstacleMap, maxLevel$)
  Produce traverse map from obstacle map
  Append traverse map to traverse list
  for $level \leftarrow 1$ to $maxLevel$ **do**
    $traverseMap \leftarrow$
      $convModel(traveseList[level-1])$
    Append traverse map to traverse list
  $path \leftarrow pathPlanner(traverseList[maxLevel])$
  for $i \leftarrow maxLevels - 1$ to 1 **do**
    $path \leftarrow pathRefiner(path, traverseList[i])$
  $path \leftarrow pathRefiner(path, obstacleMap)$
  return $path$

IV. RESULTS

In this section, we discuss the experiments and the results. A time comparison between our method and the most common path planning algorithms is performed, and the effectiveness of our method is shown in this section. Also, a full breakdown of the failure cases and wasted time on different levels of hierarchical path planning are mentioned at the end of this section.

Our hierarchical path planner (HPP) is evaluated on 2000 indoor maps and 2000 outdoor maps with different numbers of reduction levels in order to measure the saved time as the levels are increased. The results are shown in Table I, where we can see the improvement in performance up to 4 levels, and from that point, adding more levels does not make a significant difference. Additionally, we tested the 4-level HPP using Dijkstra and A* against their baseline algorithms, respectively, and measured the execution time (in seconds) until a valid path was found. Table II demonstrates the advantage of the heuristic provided by our method and the resulted speedup in path planning, leading to 40% improvement in indoor environments and 65% in outdoor environments.

However, due to the fact that at each level of resolution reduction, some of the details are removed, there are some cases in our experiment where at some level, the path refiner fails to refine the given path on the map with more details. The path planner finds a path on the lowest resolution map in these cases. However, the path is not feasible because the map is abstracted multiple times, and some essential features

TABLE I
AVERAGE TIME OF PATH PLANNING USING HPP WITH DIFFERENT NUMBERS OF LEVELS

	Indoor	**Outdoor**
2-Level HPP	0.92951	0.762387
3-Level HPP	0.802333	0.56627
4-Level HPP	0.603032	0.448267
5-Level HPP	0.611608	0.433787
6-Level HPP	0.624602	0.434019

TABLE II
AVERAGE TIME OF PATH PLANNING USING DIFFERENT ALGORITHMS

	Indoor	**Outdoor**
Dijkstra	1.033163	1.313872
HPP using Dijkstra	0.603032	0.448267
A*	0.846951	1.398034
HPP using A*	0.459444	0.392725

for path planning may be lost during the process. If the path refinement fails, the path planner has to plan a new path at the failing level, thus wasting time and potentially consuming longer than the base path planner would. While the data in Table II includes both the cases with and without failures, it would be beneficial to analyze the cost of failures at different levels in order to obtain a better estimate of the expected best and worst-case speedups and slowdowns that can occur.

Table III shows the percentage of failure of the path refiner (number of maps with failure among 2000 maps) at each level and the average wasted time. The wasted time is the time that our method has spent finding a path and refining it before reaching the level at which the failure happens. That path is not feasible, and we have to run the path planner to search the whole map and find a new path on that level. After that, the regular operation continues, and the new path is given to the following levels for refinement.

TABLE III
PERCENTAGE OF FAILURE AND AVERAGE TIME LOSS AT EACH LEVEL

	Indoor		**Outdoor**	
	Failure(%)	Wasted Time	Failure(%)	Wasted Time
Level 0	0.8%	0.36504	0%	0
Level 1	3.2%	0.07783	0.55%	0.08055
Level 2	3.8%	0.02024	0.8%	0.02302

From Table III, it can be seen that this phenomenon has a more significant effect in indoor environments as the number of failures is higher than the outdoor environments. In these maps, doors are small features that provide access between hallways and rooms and play a major role in forming the path. Combining this with random obstacles around these areas can create challenging situations that our model may not accurately represent on a reduced-sized map, leading to finding an infeasible path in abstracted maps. However, the number of failures is still relatively small, and the excess time needed to correct, especially for lower-level failures, is relatively limited, still resulting in competitive performance even in failure cases and significant gains on average, as indicated in Table II.

In the end, we also compared the quality of the paths found by our method against the plain path planners. Since the agent in this work is a point entity that moves in a grid world, we only consider the length of the paths to measure quality, and the optimal solution is defined as the shortest path. Our experiments show that our method can find the shortest path on about 84% of the indoor maps and 64% of the outdoor maps. Although our method may return a longer path than the optimal solution in some cases, the difference in length is relatively small. Table IV demonstrates this comparison in detail. The percentage of the maps that our method returns an optimal solution is shown in the first column labeled as "optimal solution frequency", and in cases that our method returns a non-optimal solution, the average difference in path length compared to the optimal solutions is shown in the second column labeled as "average length difference". As this table shows, even though our method returns a longer path in 35% of the outdoor maps, it is only 2% longer than the optimal solution. In contrast, our method can find optimal solutions in more indoor maps, but errors have higher penalties, leading to 11% longer paths in the case of non-optimal solutions. This is mainly due to the fact that indoor environments have a more coherent structure because of the walls and hallways, so going through a different hallway may significantly increase the length of the path. However, on the outdoor maps, the obstacles are scattered throughout the map, and there is less penalty for taking the wrong direction.

TABLE IV
PATH COMPARISON

	Indoor		**Outdoor**	
	Optimal Solution Frequency	Average Length Difference	Optimal Solution Frequency	Average Length Difference
HPP using Dijkstra	83.55%	-11.4%	62.7%	-2.17%
HPP using A*	84.3%	-11.15%	64.45%	-2.17%

V. DISCUSSION & CONCLUSIONS

We have presented a novel method for path planning in a hierarchical manner. We introduced the concept of directional traversability maps and how we can represent an obstacle map in the form of directional traversability scores. To create the hierarchy, a convolutional model was trained using supervised learning to learn the traversability function as a convolutional kernel and reduce the resolution of its input. This convolutional kernel is replicated into convolutional layers and stacked in a deep network that can derive a cascade of incrementally resolution reduced traversability representations of the environment. On this, a modified Dijkstra and A* was used to find and refine a path throughout the hierarchy and return the final path.

We demonstrated the potential for complexity reduction of our method in experimental results in both indoor and outdoor environments and showed that our method outperforms the raw Dijkstra and significantly improves path planning time in both types of environments. Similarly, we showed that an A* based path planner could achieve the same benefits. Additionally, we discussed the drawback of our method where some of the details of the maps are omitted during resolution reduction, which leads to finding a path that

is either infeasible on higher resolution maps or longer than the optimal solution.

One interesting future direction is to replace the pre-coded concept of traversability and supervised learning with reinforcement learning to derive its own representation of an abstracted map which can be more useful and further improve the performance while reducing the number of failures in the path refiner. Moreover, we are planning to utilize the hierarchical traversability heuristic in the context of RRTs to study its potential in larger environments further.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," *Advances in neural information processing systems*, vol. 16, pp. 767–774, 2003.

[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, 2008.

[5] Y. Wang, D. K. Jha, and Y. Akemi, "A two-stage rrt path planner for automated parking," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017, pp. 496–502.

[6] S. Mahadevan and M. Maggioni, "Proto-value functions: A laplacian framework for learning representation and control in markov decision processes." *Journal of Machine Learning Research*, vol. 8, no. 10, 2007.

[7] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 729–736.

[8] X. Yang, M. Moallem, and R. Patel, "A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1214–1224, 2005.

[9] X.-C. Lai, S. S. Ge, and A. A. Mamun, "Hierarchical incremental path planning and situation-dependent optimized dynamic motion planning considering accelerations," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 6, pp. 1541–1554, 2007.

[10] D. Dolgov, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *I. J. Robotic Res.*, vol. 29, pp. 485–501, 04 2010.

[11] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles (mobile robot)," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 61–69, 1989.

[12] D. Zhu and J.-C. Latombe, "New heuristic algorithms for efficient hierarchical path planning," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9–20, 1991.

[13] Y. Zhang, D.-W. Gong, and J.-H. Zhang, "Robot path planning in uncertain environment using multi-objective particle swarm optimization," *Neurocomput.*, vol. 103, p. 172–185, Mar. 2013. [Online]. Available: https://doi.org/10.1016/j.neucom.2012.09.019

[14] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[15] A. Konar, I. Goswami Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved q-learning for path planning of a mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1141–1153, 2013.

[16] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine learning*, vol. 22, no. 1, pp. 33–57, 1996.

[17] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *The Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[18] X. Xu, C. Liu, S. X. Yang, and D. Hu, "Hierarchical approximate policy iteration with binary-tree state space decomposition," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1863–1877, 2011.

[19] X. Xu, Z. Hou, C. Lian, and H. He, "Online learning control using adaptive critic designs with sparse kernel machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 5, pp. 762–775, 2013.

[20] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on a$*$ and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, pp. 257–266, 2015.