Selective Autonomous Exploration

Fan Ding
University of Minnesota
ding0322@umn.edu

December 18, 2021

Abstract

Selective Autonomous Exploration (SAE) is a method designed for autonomous exploration when resources are limited. It selectively explores the environment. The main ideas is shorten the exploration path, and increase the exploration speed by ignore some inefficient exploration. Compared with completeness, SAE pays more attention to efficiency. SAE prioritizes exploring more volumetric paths in a shorter time. Experiments show that compared with TARE exploration planner, the coverage rate of SAE has increased by 10.4% on average.

1 Introduction

Autonomous exploration is about using autonomous robots to explore and map previously unknown environments. Autonomous exploration is challenging because it needs to update the representation of the environment to track the explored area, and search the representation and find a continuous traversable path to guide the vehicle to the unknown area. In addition, the complex environment, cluttered space, narrow passages, and rugged terrain make autonomous exploration more challenging. In some cases, we have to make trade-offs on completeness and efficiency. For example, when resources are limited, we are unable to explore the whole environment. Therefore, we trade completeness for efficiency. We can selectively explore the environment. Therefore, I introduce selective autonomous exploration (SAE), which is a method designed for autonomous exploration when resources are limited. In this report, Section 2 talks about the motivation and purpose of this project. Section 3 describes SAE's foundation stones: Autonomous Exploration Development Environment and TARE Exploration Planner. Section 4 introduces the methodology and the logic behind SAE. Section 5 describes my experiments and results. Section 6 mentions challenges and potential future work. Section 7 provides a brief conclusions.

2 Motivation

This project is inspired by the paper "TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments"[1]. It won the Best Paper Award and Best System Paper Award at the 2021 Robotics: Science and Systems Conference (RSS). TARE is a hierarchical framework for highly efficient exploration. And it has been deployed in DARPA Subterranean Challenge in Satsop Nuclear Plant, WA. The vehicle traveled more than 886m in 1458 seconds, exploring the

environment completely autonomously. In addition to nuclear power plants, in other dangerous and inaccessible environments, robots can be very helpful to replace humans and perform tasks intelligently. TARE explores the entire environment. But in some cases, we are not able to explore the whole environment because of limited resources, such as fuel or time. For example, after an accident, we need to explore the environment within a limited time before dispatching rescue robots. One possible method is to selectively explore the environment. For example, if exploring a small corner requires the robot to take a much longer detour, the robot should ignore the corner and take a shorter path. Therefore, I decide to make a few changes in the original TARE's algorithm, which explores the entire environment to selectively explore the environment. Thus, I would like to propose Selective Autonomous Exploration. This is a method designed for autonomous exploring when resources are limited. It improves exploration efficiency by discarding insignificant local environmental information. This project is a variant of the original TARE paper. And the goal of the project is to propose a new concept "Selective Autonomous Exploration" and give slight insights into further researches on Autonomous Exploration.

3 Related Work

This project is based on Autonomous Exploration Development Environment[2] and TARE Exploration Planner[5], both from Carnegie Mellon University Robotics Institute. The Autonomous Exploration Development Environment provides different simulators and many built-in modules, which are used in our Selective Autonomous Exploration method. TARE Exploration Planner is a hierarchical framework that can effectively explore the environment. Our Selective Autonomous Exploration method is a variant of TARE.

3.1 Autonomous Exploration Development Environment

The autonomous exploration development environment aims to leverage system development and robot deployment for autonomous navigation and exploration. It contains a variety of simulation environments, collision avoidance, terrain traversability analysis, waypoint tracking, and other autonomous navigation modules, as well as a set of visualization tools. Users can develop autonomous navigation systems and then transplant these systems to real robots for deployment. With those build-in modules, the autonomous exploration problem is simplified to finding a sequence of waypoints to form a trajectory. By connecting those waypoints, we have a path for the robot to follow. After following the path and visiting each waypoints, the robot should be able to build a map of the environment with its onboard sensors. Figure 1 shows the robot using collision avoidance module. Figure 2 shows the simplified model. In this report, we focus on how to find the waypoints to form the exploration path.

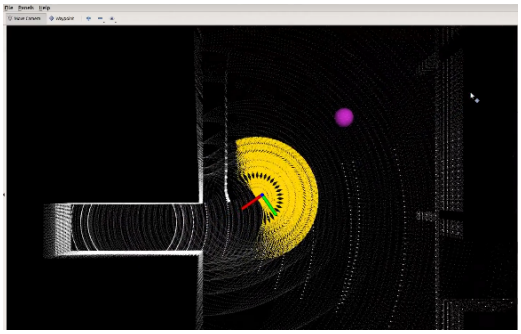

Figure 1: A robot automatically avoid obstacles and follow the purple waypoint in Autonomous Exploration Development Environment.

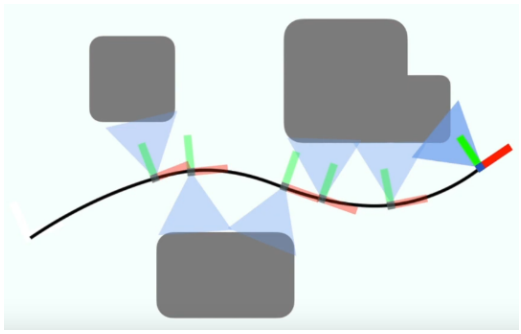

Figure 2: Simplified autonomous exploration model. Black line represents the robot trajectory. Grey objects are obstacles. Blues are robot view area.[4]

3.2 TARE Exploration Planner

TARE planner involves a hierarchical framework for effective exploration. It contains local and global levels. See Figure 3 for details. The local level of the framework maintains dense data and calculates detailed paths within the scope of local planning, while the global level maintains data sparsely and calculates rough paths within the global scope. The two paths are connected together to form an exploration path. The insight of the framework is that detailed processing close to the vehicle is the most effective, while rough processing far away from the vehicle provides sufficient utility. The framework exchanges details on a global scale in exchange for computational speed. In other words, the framework gives priority to the exploration of the surrounding environment of the vehicle, while taking into account the overall situation. Pseudo code are included. Algorithm 1 is TARE local planer. Algorithm 3 is TARE global planner.

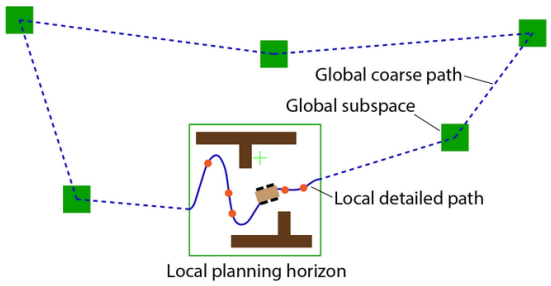

Figure 3: TARE Framework. The local level(green square in the middle), data is densely maintained and a local detail path(dark-blue line) is computed. At the global level, data is sparsely maintained in the distant subspaces(green solid squares) and a global coarse path(dot-blue) is computed. The local path and global path are connected together to form the exploration path.[3]

4 Methodology

SAE is a variant of TARE. The different is that TARE explores entire environment, but requires a long time, while SAE selectively explore the environment in a shorter time with a higher exploration efficiency. The main ideas behind SAE is ignoring some time-consuming exploration to shorten travel path. Figure 4 shows the different between TARE and SAE. TARE explored the entire environment. To covered every surface, the robot made a detour to cover the right bottom surface. SAE obtain a much shorter path and reduce exploration time by ignoring some surface. In another word, SAE trades completeness for efficiency. As we mentioned in simplified autonomous exploration model, we need to find a sequence of waypoint to from the exploration path. Thus, SAE's goal is to find waypoints that form the shortest path that covers most of the surface. "Most of the surface" will be defined in the later SAE Local Planner part. SAE uses the same global planner in TARE, but slightly changed the local planner.

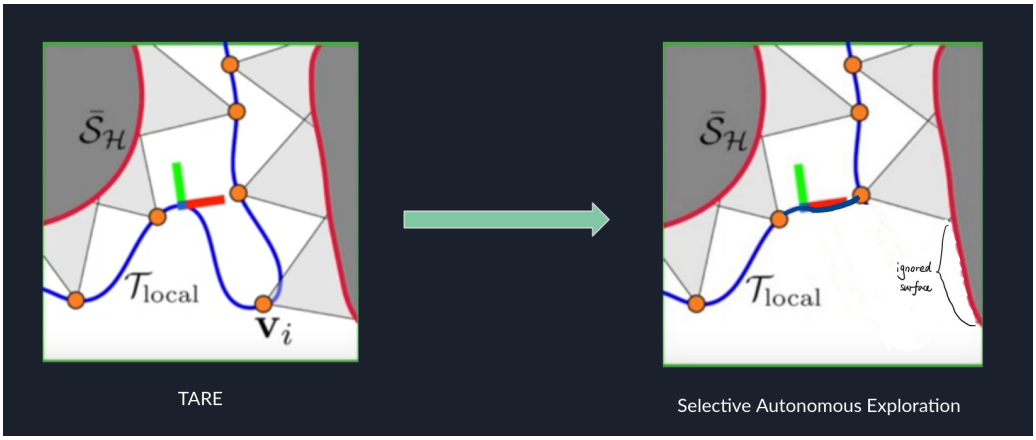

Figure 4: SAE shorten exploration path by ignoring some surface. Red lines represent environment surface. Blue line is robot trajectory.

4.1 SAE Local Planner

SAE local planer is a variant of the TARE local planner. See Algorithm 1 and Algorithm 2 for comparison. See Figure 5 for simplify algorithm. SAE first uniformly generates candidate viewpoints and puts them into a priority queue sorted by its surface coverage. Viewpoints that cover more surface will be at the front of the priority queue. Then iterate K times to find the shortest path that can cover most of the local space. In each iteration, the local planner probabilistically selects viewpoints one by one. High coverage viewpoint has higher possibility to be picked. After each selection, the surface coverage of remaining viewpoints in the priority queue will be updated and the priority queue will be resorted. The selection loop stops when the surface coverage of the first element in priority queue is less than a constant threshold called max_ignore. This means that no matter what viewpoint we choose from the remaining viewpoint pools, it will not increase our total coverage by max_ignore. Therefore, it is not worth adding those viewpoints to our selected viewpoint sets. We stop the while loop for selection. Then calculate the TSP tour that visits all selected viewpoints. Track the shortest TSP tour of all iterations.

- Uniformly generates candidate viewpoints.
- Select viewpoints sequence:
  Iterate K times:
  - Probabilistic select viewpoints one by one (high coverage viewpoint has higher possibility) until reminding viewpoint candidates are not worth to add.
  - Compute a TSP tour visiting all selected viewpoints.
  - Keep track of the shortest TSP tour.

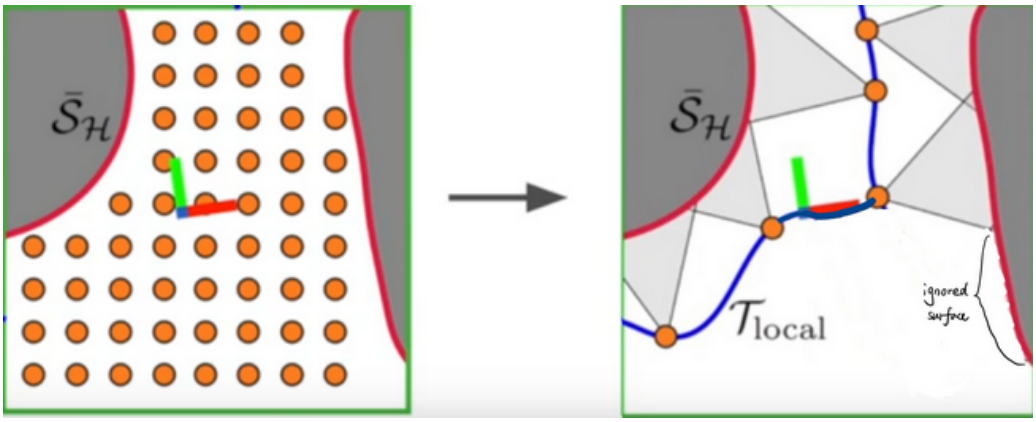

Figure 5: SAE Local Planner select viewpoints. Yellow dots are viewpoints. Red lines represent environment surface. Blue line is robot trajectory.

4.2 SAE Global Planner

The SAE global planner uses the same global planner algorithm in TARE. The global planner maintains sparse information outside the local area. Algorithm 4 shows the pseudo code. And here is the basic idea for SAE local planner:

- Divides the space outside the local area into subspaces.
- Takes the centers of all unexplored subspaces and perform TSP.
- After getting a ordered waypoints and global path from TSP, the planner can concatenate the global path and local path.

Algorithm 1: TARE Local Planner

input : traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, current viewpoint $\mathbf{v}_{\text{current}}$, boundary viewpoints $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$
output: local path $\mathcal{T}_{\text{local}}$

Generate a set of viewpoint candidates $\mathcal{V}$ in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$;
Initialize priority queue Q;
For every $\mathbf{v} \in \mathcal{V}$, estimate coverage $\bar{\mathcal{S}}_{\mathbf{v}}$, then push $\mathbf{v}$ into Q with the priority set to its reward $A_{\mathbf{v}}$;
$\mathcal{T}_{\text{local}} \leftarrow \emptyset$, $c_{best} \leftarrow +\infty$;
for $i := 1$ to K **do**
  $Q' \leftarrow Q$, $\mathcal{V}' \leftarrow \{\mathbf{v}_{\text{current}}, \mathbf{v}^1_{\text{boundary}}, \mathbf{v}^2_{\text{boundary}}\}$;
  while $Q' \neq \emptyset$ and Q' contains at least one non-zero priority **do**
    Probabilistically pick viewpoint $\mathbf{v}'$ in Q', then remove $\mathbf{v}'$ from Q';
    $\mathcal{V}' \leftarrow \mathcal{V}' \cup \mathbf{v}'$;
    Update priorities for all viewpoints in Q';
  end
  Compute smooth path $\mathcal{T}'_{\text{smooth}}$ and cost c'_{smooth}
  if $c'_{\text{smooth}} < c_{\text{best}}$ **then**
    $\mathcal{T}_{\text{local}} \leftarrow \mathcal{T}'_{\text{smooth}}$, $c_{\text{best}} \leftarrow c'_{\text{smooth}}$;
  end
end
return $\mathcal{T}_{\text{local}}$;

Algorithm 2: SAE Local Planner

input : traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, current viewpoint $\mathbf{v}_{\text{current}}$, boundary viewpoints $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$
output: local path $\mathcal{T}_{\text{local}}$

Generate a set of viewpoint candidates $\mathcal{V}$ in $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$;
Initialize priority queue Q;
For every $\mathbf{v} \in \mathcal{V}$, estimate coverage $\bar{\mathcal{S}}_{\mathbf{v}}$, then push $\mathbf{v}$ into Q with the priority set to its reward $A_{\mathbf{v}}$;
$\mathcal{T}_{\text{local}} \leftarrow \emptyset$, $c_{best} \leftarrow +\infty$;
for $i := 1$ to K **do**
  $Q' \leftarrow Q$, $\mathcal{V}' \leftarrow \{\mathbf{v}_{\text{current}}, \mathbf{v}^1_{\text{boundary}}, \mathbf{v}^2_{\text{boundary}}\}$;
  while $Q' \neq \emptyset$ and $Q'.top().A_v > maxIgnore$ **do**
    Probabilistically pick viewpoint $\mathbf{v}'$ in Q', then remove $\mathbf{v}'$ from Q';
    $\mathcal{V}' \leftarrow \mathcal{V}' \cup \mathbf{v}'$;
    Update priorities for all viewpoints in Q';
  end
  Compute smooth path $\mathcal{T}'_{\text{smooth}}$ and cost c'_{smooth}
  if $c'_{\text{smooth}} < c_{\text{best}}$ **then**
    $\mathcal{T}_{\text{local}} \leftarrow \mathcal{T}'_{\text{smooth}}$, $c_{\text{best}} \leftarrow c'_{\text{smooth}}$;
  end
end
return $\mathcal{T}_{\text{local}}$;

Algorithm 3: TARE Global Planner

input : local planning horizon $\mathcal{H}$, traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, exploring subspaces $\hat{\mathcal{G}}$, current viewpoint $\mathbf{v}_{\text{current}}$
output: exploration path $\mathcal{T}$

Compute shortest paths between centroids in $\hat{\mathcal{G}}$ and $\mathbf{v}_{\text{current}}$, then create distance matrix $\mathbf{D}$;
Compute global path $\mathcal{T}_{\text{global}}$ by solving TSP using $\mathbf{D}$;
Extract $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ as the intersections between $\mathcal{T}_{\text{global}}$ and the boundary of $\mathcal{H}$;
Compute local path $\mathcal{T}_{\text{local}}$ using Algorithm 1;
Concatenate $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ to generate $\mathcal{T}$;
return $\mathcal{T}$;

Algorithm 4: SAE Global Planner

input : local planning horizon $\mathcal{H}$, traversable C-space $\mathcal{C}^{\mathcal{H}}_{\text{trav}}$, uncovered surfaces $\bar{\mathcal{S}}_{\mathcal{H}}$, exploring subspaces $\hat{\mathcal{G}}$, current viewpoint $\mathbf{v}_{\text{current}}$
output: exploration path $\mathcal{T}$

Compute shortest paths between centroids in $\hat{\mathcal{G}}$ and $\mathbf{v}_{\text{current}}$, then create distance matrix $\mathbf{D}$;
Compute global path $\mathcal{T}_{\text{global}}$ by solving TSP using $\mathbf{D}$;
Extract $\mathbf{v}^1_{\text{boundary}}$ and $\mathbf{v}^2_{\text{boundary}}$ as the intersections between $\mathcal{T}_{\text{global}}$ and the boundary of $\mathcal{H}$;
Compute local path $\mathcal{T}_{\text{local}}$ using Algorithm 3;
Concatenate $\mathcal{T}_{\text{local}}$ and $\mathcal{T}_{\text{global}}$ to generate $\mathcal{T}$;
return $\mathcal{T}$;

5 Experiment

The experiment is conducted in simulators. The goal of this experiment is to show that selective autonomous exploration methods can improve exploration efficiency compared to TARE. In this project, I use coverage rate as the indicator of exploration efficiency. Coverage rate is defined by total_exploration_volume/total_exploration_time. This experiment uses the University of Minnesota cse-hk4250 computer. The operating system is Ubuntu 20.04, the memory is 31.3GB, the processor is Intel Core i7-4790, the graphic is Intel HD graphics 4600, and the disk is 500GB. I run both

TARE and SAE on simulators provided by the autonomous exploration development environment. The autonomous exploration development environment also contains many navigation modules, such as collision avoidance, terrain traversability analysis, and waypoint tracking, as the middle layer. While the exploration algorithm is on the top layer. In our local-global framework, The local planning scope is composed of 5×5×3 blocks, and block size is set to 8m×8m×5m. The vehicle is located in the center block. The point cloud is used to simulate the surface, maintaining a resolution of 0.2m.

5.1 Observation Interface

During the experiment, there are 4 major windows for us to observe the performance. Figure 6 is a screenshot during run time. The top left is the vehicle_simulator.RViz. The top right is planner_simulator.RViz. The bottom left is real-time metrics. The bottom right is the terminal showing exploration status.

- vehicle_simulator.RViz(top left) - shows the current vehicle view. The purple line is the vehicle trajectory. Yellow lines are free paths (without hitting obstacles). The purple dot is the next way point the robot should move to.
- planner_simulator.RViz(top right) - shows the overall exploration. Blue areas are explored space. White areas are unexplored spaces. Green solid box: uncovered global subspace. Green square: local planner scope.
- metrics(bottom left) - shows exploration volume(m^3) and traveled distance(m).
- terminal(bottom right) - shows exploration status. started, or finished.

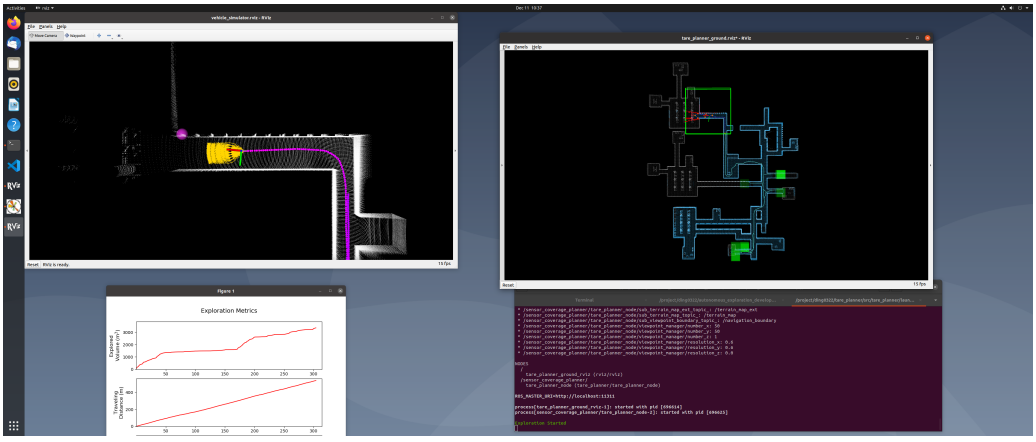

Figure 6: Observation Interface. Top left: current vehicle view. Top right: overall exploration. Bottom left: real time metrics. Bottom right: exploration status.

5.2 TARE and SAE Comparison in Forest Simulator

Both TARE and SAE algorithms run in the forest simulator. Figure 7 shows the resulting data. Compared with TARE, SAE saves 157 seconds of exploration time and only ignores the volume of

$698m^3$. Most of the neglected volumes are located in spaces that are time-consuming to explore, such as corners. See the circled part in Figure 7 for comparison. Blue means that the area is completely covered. White indicates areas that have not been explored. SAE's top left corner is lighter than TARE. Because it's blue and white. Coverage rate is the main indicator used to evaluate these two algorithms. Coverage rate is defined by total_exploration_volume/total_exploration_time. The coverage rate for TARE is 48.52 m^3/s. And the coverage rate for SAE is 58.99 m^3/s. SAE is able to improve the coverage rate by 21.6%, with a loss of 1.8% volume.

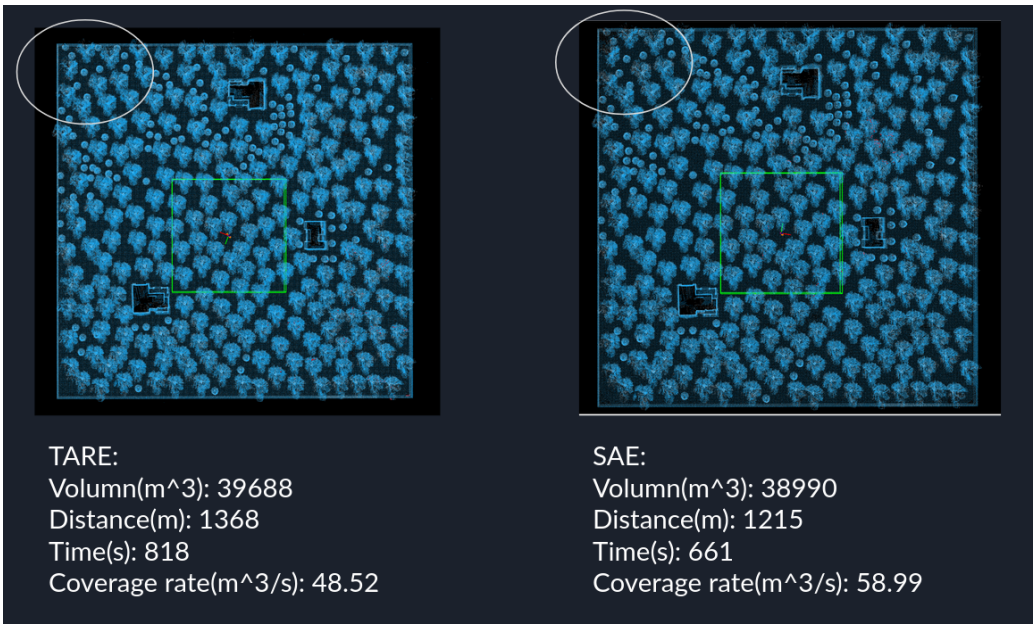

Figure 7: TARE and SAE Comparison in tree simulator. Ignored volume: $698m^3$. Saved time: $157s$

5.3 Different Simulator Environment

Total 5 simulator are used to compare SAE and TARE. They are:

- Forest (150m x 150m): The cluttered environment mainly contains trees and several houses.
- Campus (340m x 340m): As part of the campus of Carnegie Mellon University, the large-scale environment includes undulating terrain and intricate environmental layout.
- Indoor Corridors (130m x 100m): It consists of a long and narrow corridor connected to the lobby area. There are obstacles such as tables and columns. The guardrail increases the difficulty of autonomous exploration.
- Tunnel Network (330m x 250m): A large-scale environment that includes tunnels that form a network. The environment was provided by Tung Dang of the University of Nevada, Reno.
- Multi-storage Garage (140m x 130m, 5 floors): An environment with multiple layers and sloping terrain for testing autonomous navigation in a 3D environment.

Figure 8 shows the resulting data. Env stands for environment, CR stands for coverage Rate, which is calculated by exploration_volume/exploration_time SAE can improve the coverage rate in all environments and the average coverage rate improvement is 10.4%. To compare improvement rate in different environment. New metric improveRate is calculated by (SAE coverage rate - TARE coverage rate) / TARE coverage rate. In the forest environment, SAE can give the highest improveRate be 21.6%. Campus environment also had a high improveRate. It's because the forest and campus are relatively open and wide environments, unlike indoors or tunnels that contain many narrow passages.

Env / CR (m^3/s)	Forest	Indoor	Tunnel	Campus	Garage
TARE	48.52	9.01	7.68	34.83	19.58
SAE	58.99	9.38	8.29	39.69	20.51
ImproveRate	21.6%	4.1%	7.9%	14.0%	4.4%

Figure 8: TARE and SAE Comparison on 5 different environment simulators.

6 Challenge Faced

During the experiment, SAE sometimes missed the "hidden" space. It mistakes a turning point for a dead end. Ignore the turning point will ignore the whole "hidden space". In Figure 7, we can see that the robot is not exploring the top left space. The robot thought that's a dead end and it is not worth to go. Therefore, the robot will not reach the end of the road and miss the hidden room. My current approach is decreasing the ignore_Max value. It will force the robot to explore more surfaces, which gives higher possibility to reach the turning point, thereby reduces the possibility of missing "hidden" space. But this is not a robust solution.

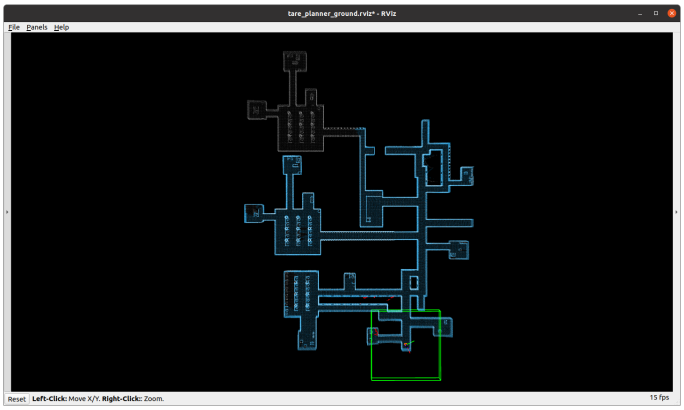

Figure 9: SAE missed a large room(top left).

7 Conclusion

Selective Autonomous Exploration(SAE) is a method designed for autonomous exploring when resources are limited. It's based on Autonomous Exploration Development Environment and TARE Exploration Planner. SAE trades completeness for efficiency. In the experiment on the forest simulator, SAE is able to improve the coverage rate by 21.6%, with a loss of 1.8% volume compared to the original TARE planner. Also, In total five simulators, the average of coverage rate improvement is 10.4%. The goal of the project is to give slight insight to the further research on Selective Autonomous Exploration.

References

[1] Chao Cao et al. "TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments". In: July 12, 2021. DOI: 10.15607/RSS.2021.XVII.018.

[2] *Development Environment*. URL: https://www.cmu-exploration.com/development-environment (visited on 12/16/2021).

[3] Ji Zhang. *ICRA 2021 Talk: Exploring Large and Complex Environments Fast and Efficiently*. May 25, 2021. URL: https://www.youtube.com/watch?v=v9dRqpp5QXQ (visited on 12/16/2021).

[4] Ji Zhang. *RSS 2021 Spotlight: TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments*. July 2, 2021. URL: https://www.youtube.com/watch?v=B0xlEoLVmKU (visited on 12/16/2021).

[5] *TARE Planner*. URL: https://www.cmu-exploration.com/tare-planner (visited on 12/16/2021).