Inter-IIT Tech Meet 13.0

2024

Team 84

Bharat Forge Presentation

Inter-IIT Tech Meet 13.0

2024

TABLE OF CONTENTS

Introduction

The PS is divided in two major parts

Map Exploration

Task Allocation

Exploration

Step I

Generating map using SLAM and exploring the map manually. Hence, making a map using SLAM toolkit

Step II

Making a system to have the bot autonomously explore any given environment and using that to generate the map for the system

Step III

Adding multiple bots into the simulation and making them explore the map

Step IV

Merge the maps obtained from multiple bots and making a global map

Step V

Extending the autonomous exploration and map merging using multiple bots

Manual Exploration – Single Bot

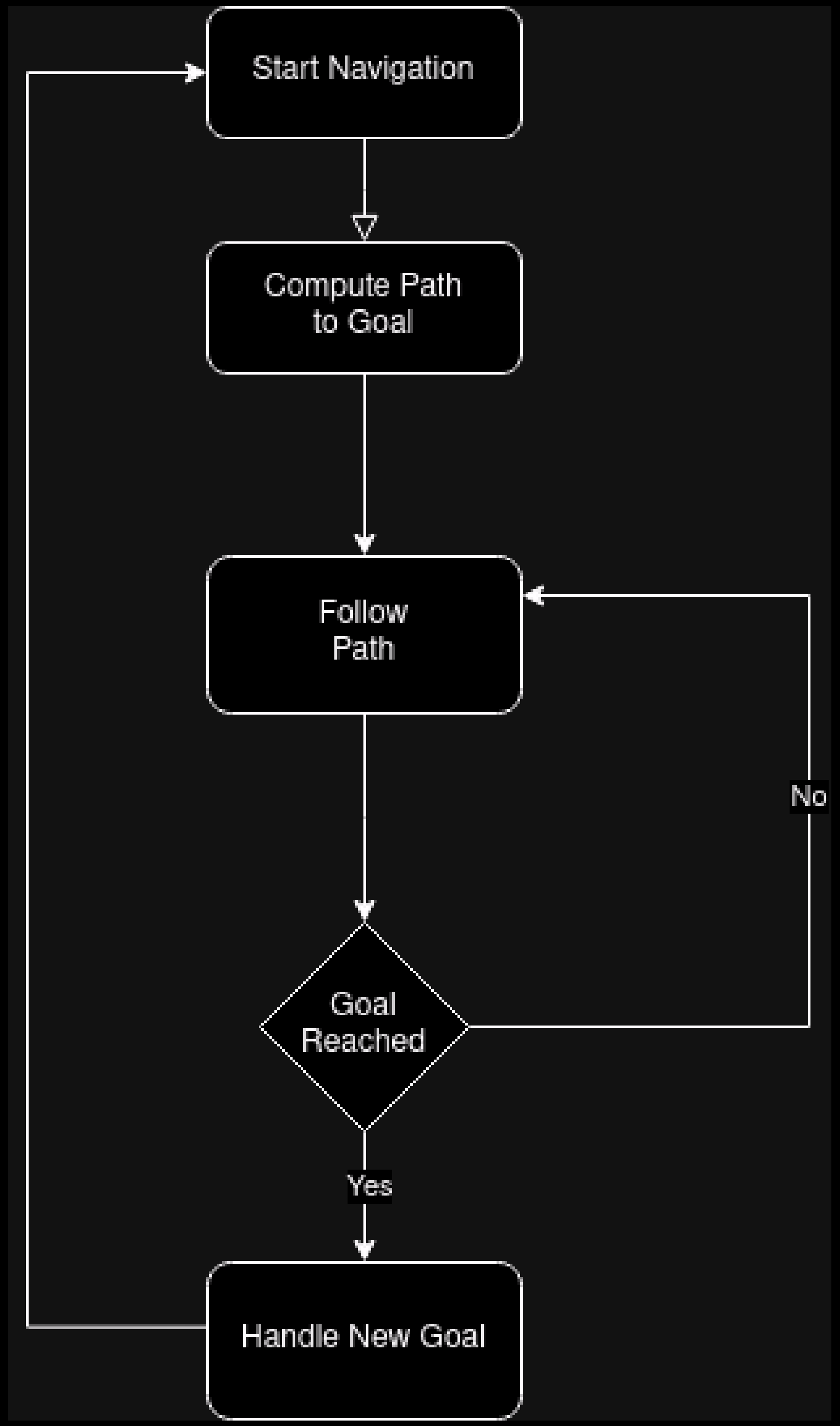
```
$ ~ ros2 run teleop_twist_keyboard teleop_twist_keyboard

This node takes keypresses from the keyboard and publishes them
as Twist/TwistStamped messages. It works best with a US keyboard layout.
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Every complex system is built on top of small simple things. So, we started out simple...

1. Teleop Twist Keyboard

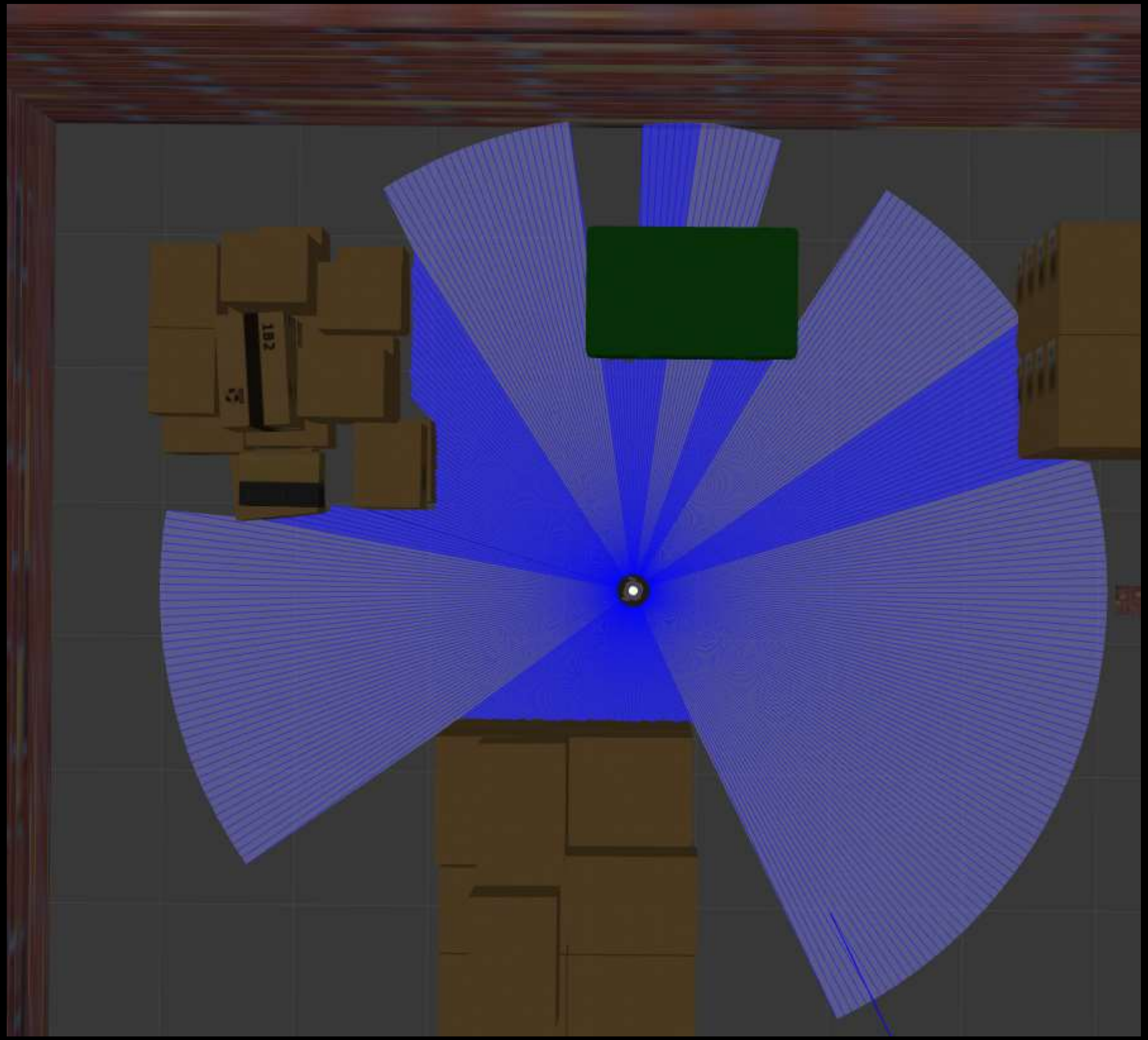
2. Turtlebot3

Autonomous Exploration – Single Bot



We played with exploring the unknown without any human senses. We used the frontier algorithm to explore the unknown

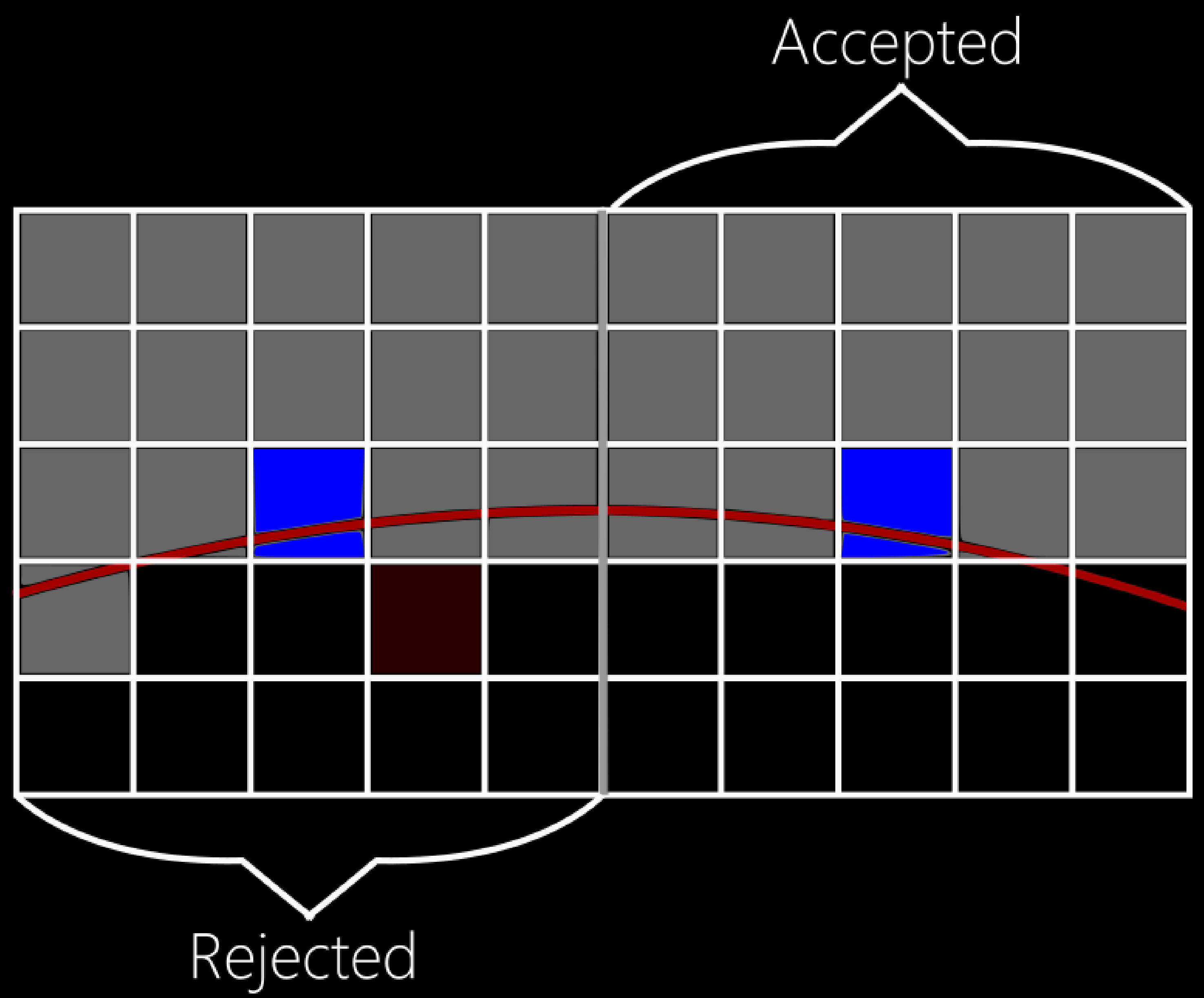
Frontier Algorithm



Frontier algorithm works by evaluating a boundary between the known and unknown boundaries of an area.

The map is divided into grid containing unit cell. Every unit cell is marked as free, occupied or unknown. The bot chooses a candidate frontier point and goes there.

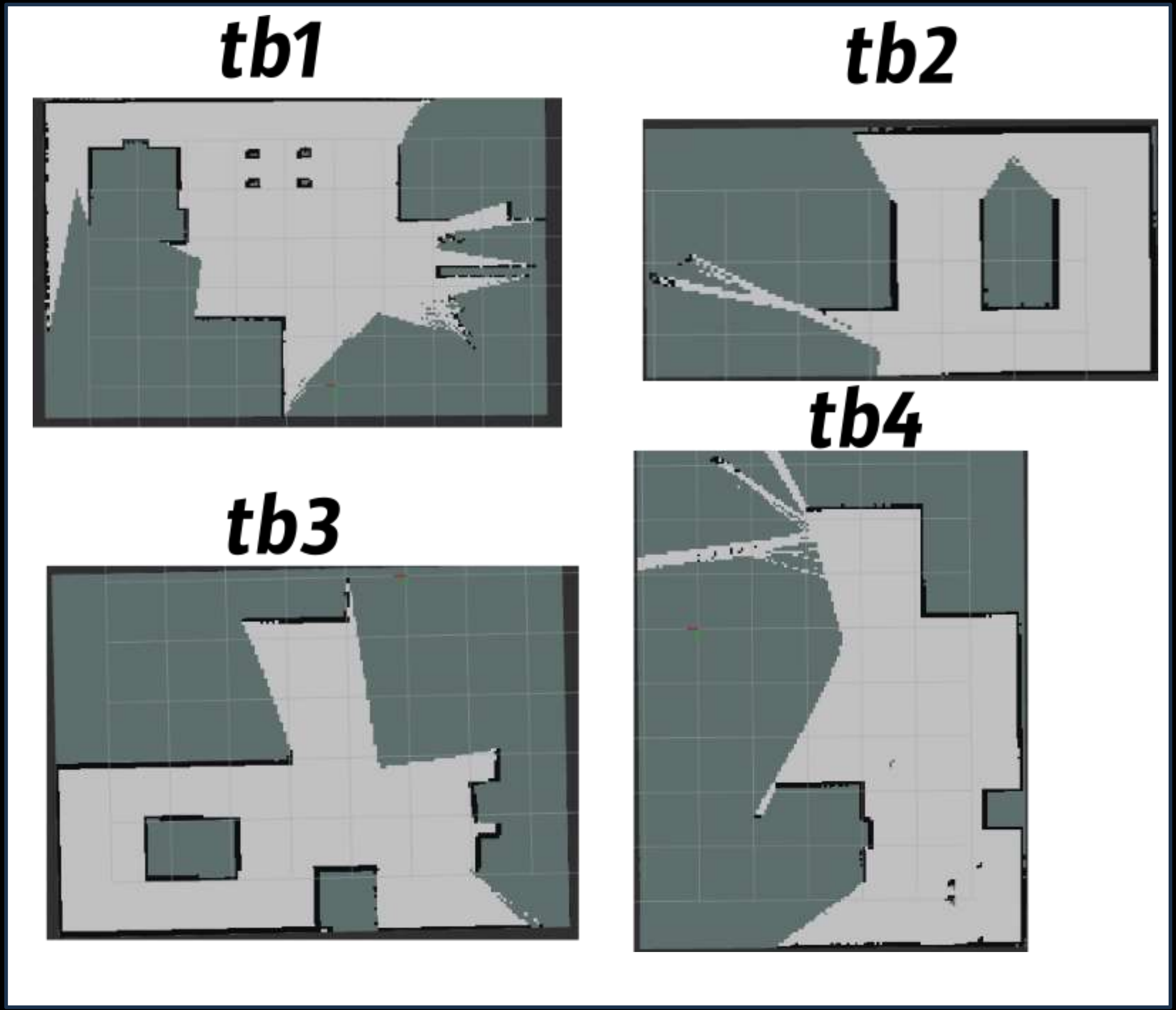
Occupancy Grids



The map merging works by making a grid layout of maps and plotting the known and unknow parts on it using frontier algorithm. Then the grids are divided into 5 × 5 cells. The center of which lies on the frontier.

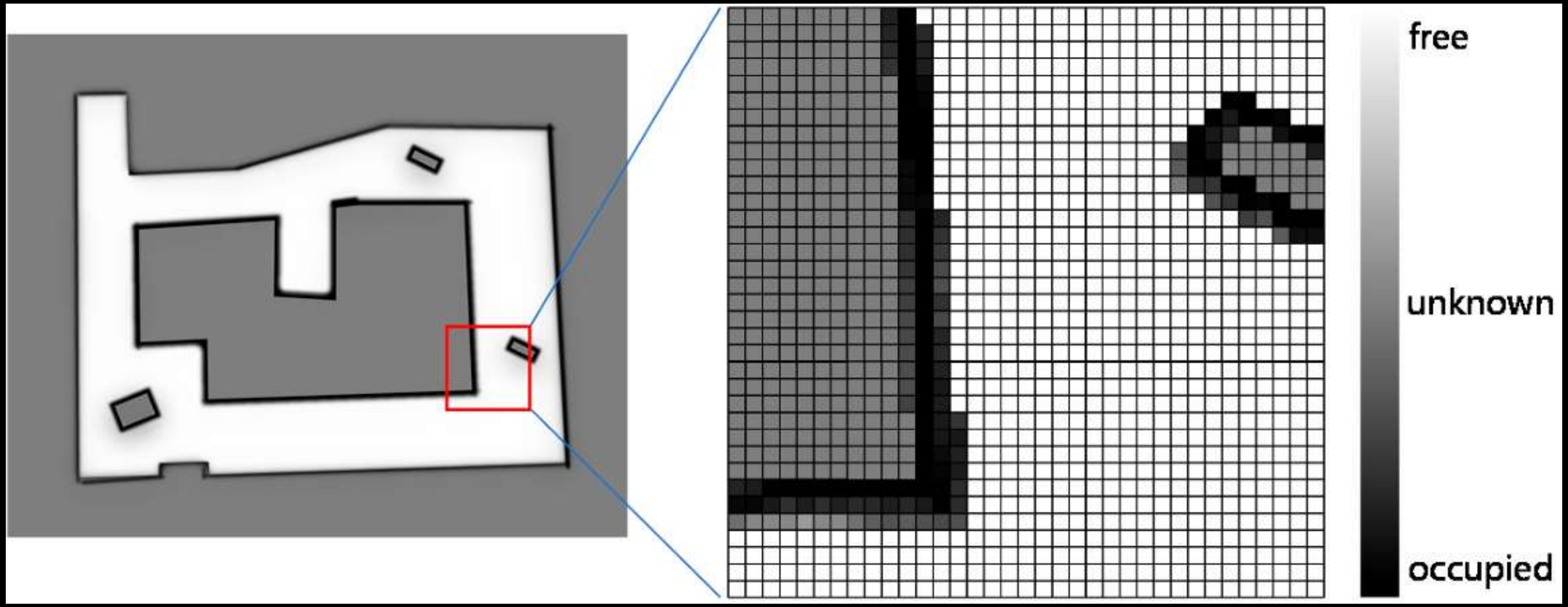
The bot chooses a frontier point as candidate frontier point based on if there is a obstacle in the grid. 5 is chosen as a number to provide enough clearance to the bot from the obstacles.

Manual Exploration – Multiple Bots



We now need a system to explore multiple bots manually, we can work with multiple `/teleop_twist_keyboard` to have system that can explore with any number of bots

Map Merging



Map Merging of maps obtained from multiple bots is done using the Occupancy grids. Occupancy grids can work in the following modes-

1. Bots know each other's initial pose
2. Bots don't know each other's initial pose

Occupancy grids works on probability of obtaining an obstacle in a particular grid cell of the map. These probabilities are then combined using the Bayesian updating principle-

$$P(occ_{x,y}) = \frac{odds_{x,y}}{1+odds_{x,y}} \qquad odds_{x,y} = \prod_{i=1}^{n} odds_{x,y}^{i} \qquad odds_{x,y}^{i} = \frac{P(occ_{x,y}^{i})}{1-P(occ_{x,y}^{i})}$$

Map Merging

Transformation Matrix -

$$T_{\delta s} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

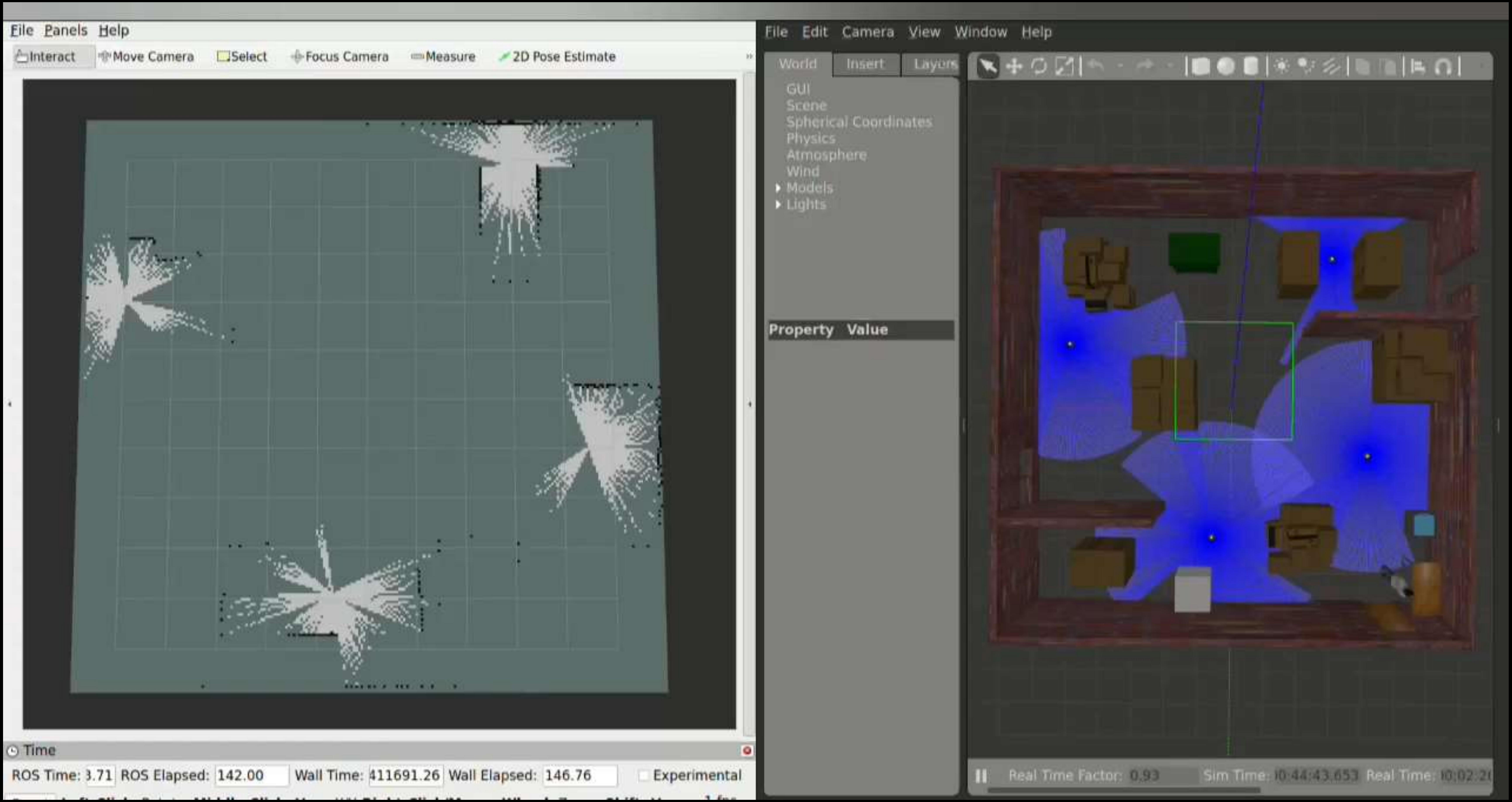
Agreements of Maps obtained-

$$\omega(m_1, m_2) = \sum_{i-0}^{N-1} \sum_{j=0}^{M-1} Eq(m_1[i,j], m_2[i,j])$$

For relative pose δs, we can use the transformation matrix when we the bots first estimate the relative position when they first encounter each other. Then we use the same $T_{\delta s}$ to transform a bot's map into the perspective of the other.

Then we can use the Posterior Probabilities to merge the probabilities for a particular grid cell.

We then have to maximize the output of the consistency of merging map to get the best results. This is done by maximizing the $\omega\left(m_1, T_{\delta s}(m_2)\right)$ obtained from the above formula


File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
Time
ROS Time: 3.71 ROS Elapsed: 142.00 Wall Time: 411691.26 Wall Elapsed: 146.76
Experimental
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.93
Sim Time:
Real Time:

Task Allocation

Navigation

Bots use `/navigate_to_pose` to set goals, compute paths, retry on failure, and follow the computed path until task completion

Path Planning

The `/compute_path_to_pose` action calculates efficient, obstacle-free paths using Grid-Based Path Planning

Task Management

SQLite3 database tracks available bots, pending tasks, and completed tasks with consistent task ID reuse

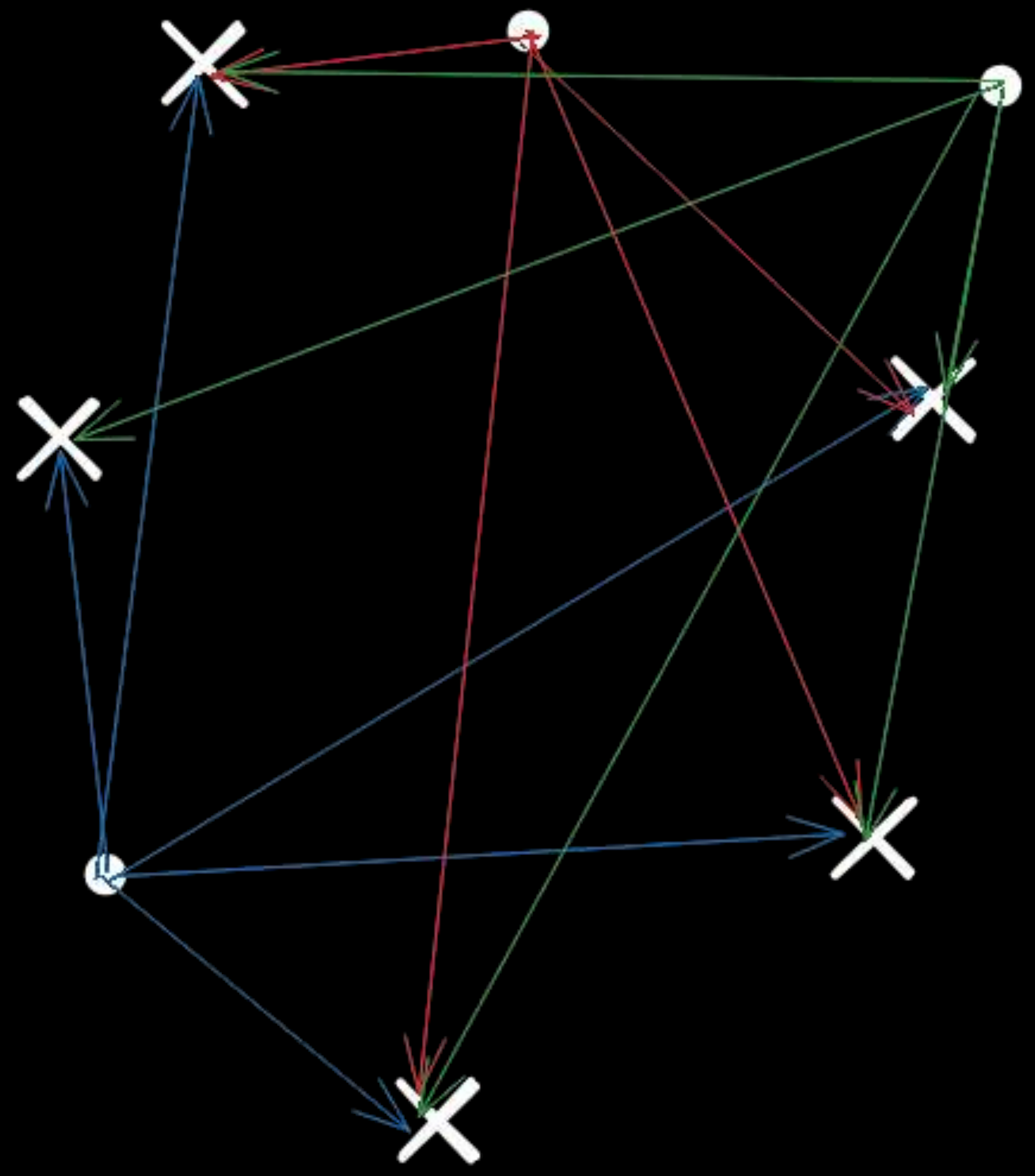
Task Allocation

The Hungarian Algorithm minimizes traversal distance using a cost matrix, handling asymmetry with dummy values (0 or 10^6)

Continuous Workflow

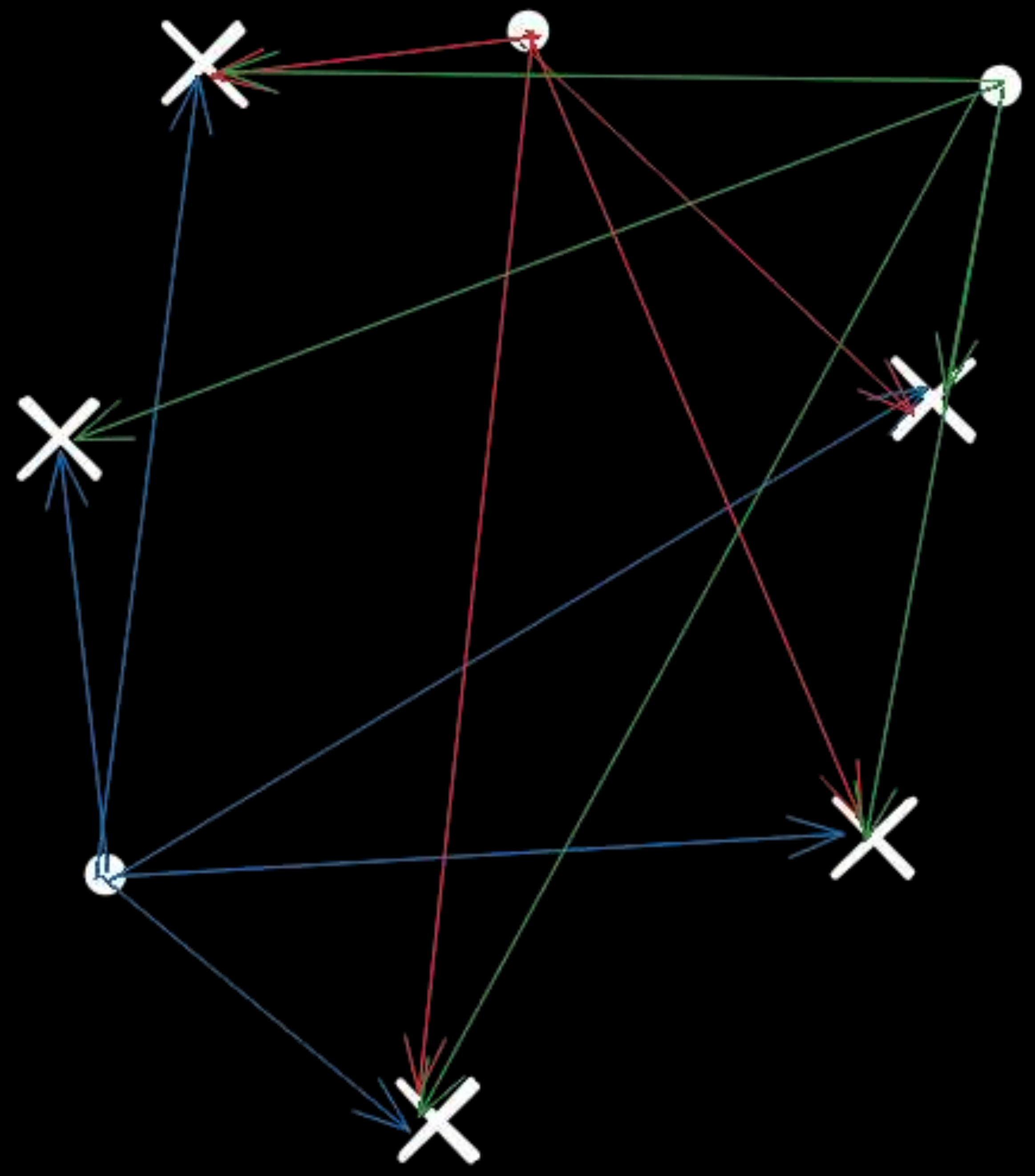
New goals are assigned after task completion for seamless, scalable operations

Hungarian Algorithm



The Hungarian algorithm minimizes the total cost of assigning bots to tasks by using a cost matrix, and when the number of bots does not match the number of targets, dummy values (0 or a large number like 10^6) are added to balance the matrix for optimal assignment.

Why Hungarian Algorithm?

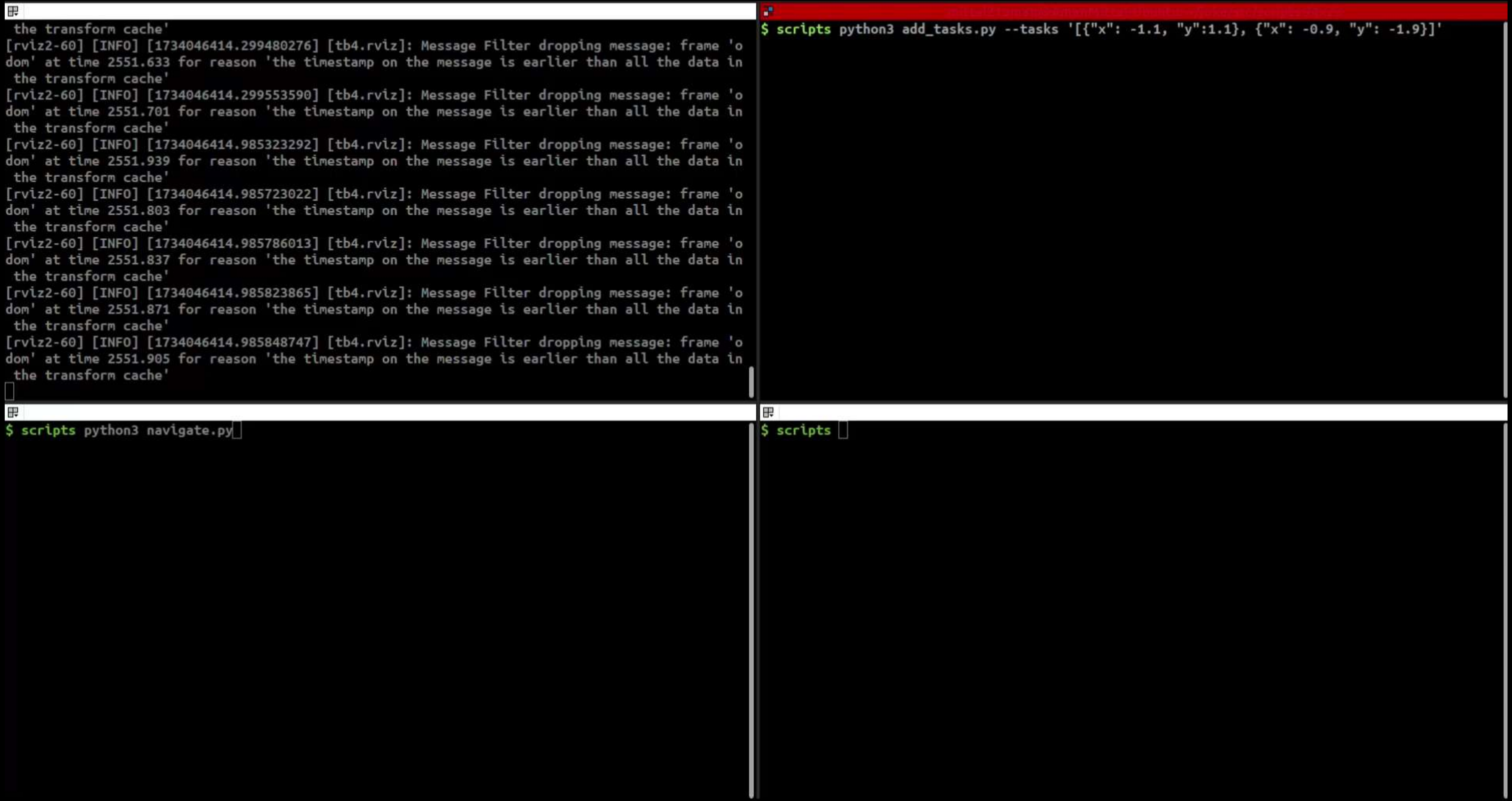


1.It guarantees the best assignment to minimize distance

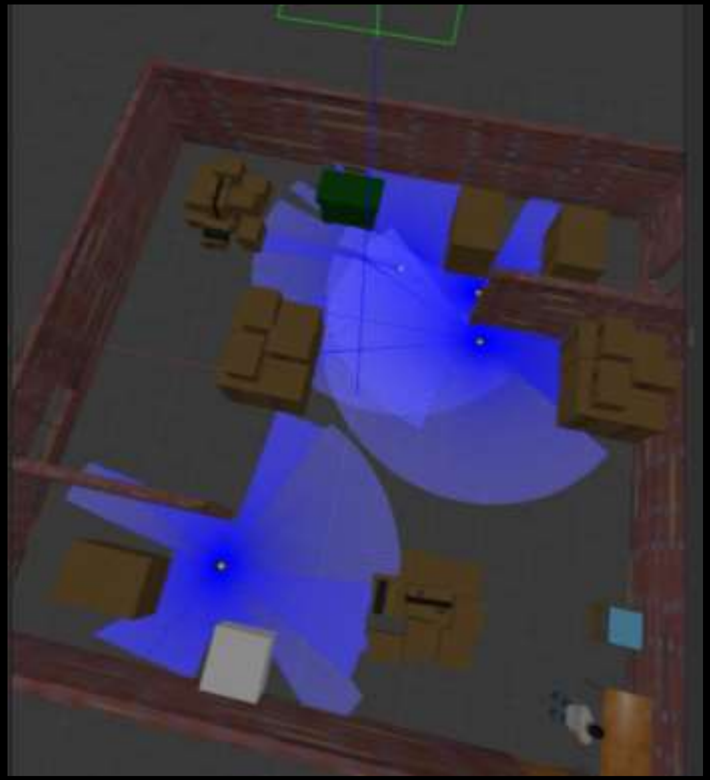
2.Ideal for assigning tasks to bots, as it works with two distinct sets(Tasks and Bots)

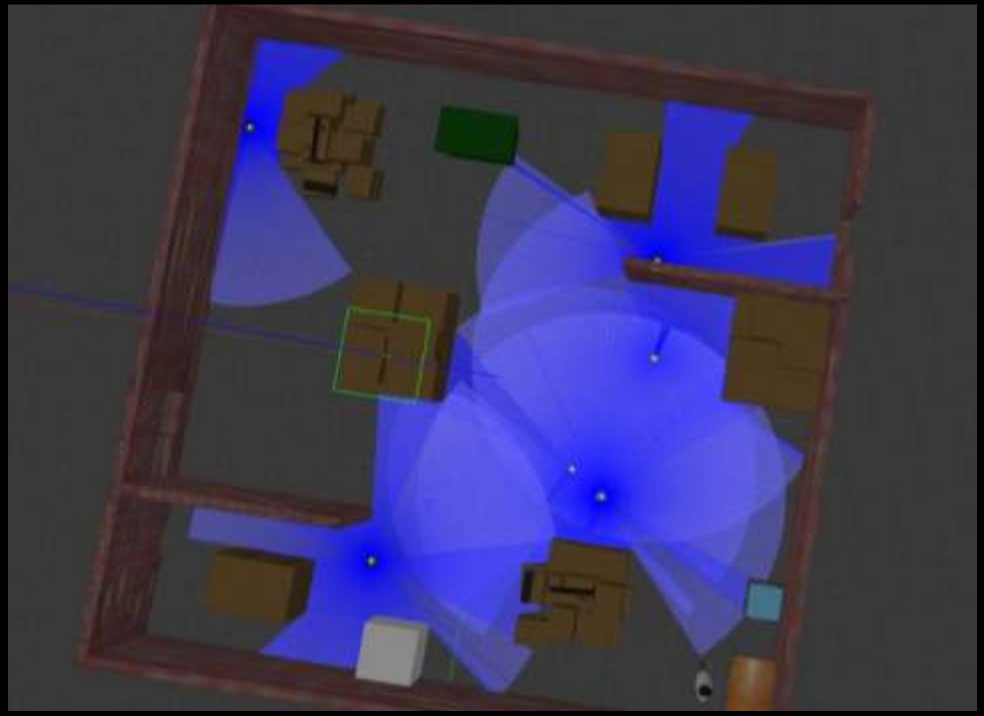
3.It can adapt to different numbers of bots and tasks using dummy values

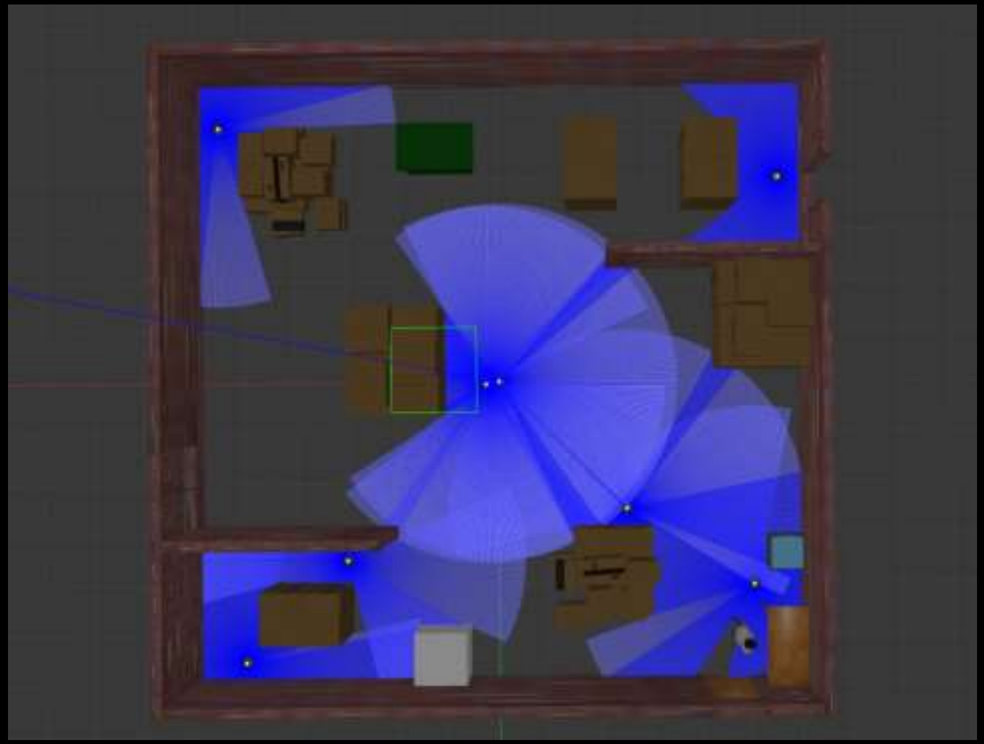
Task Allocation



Scalability







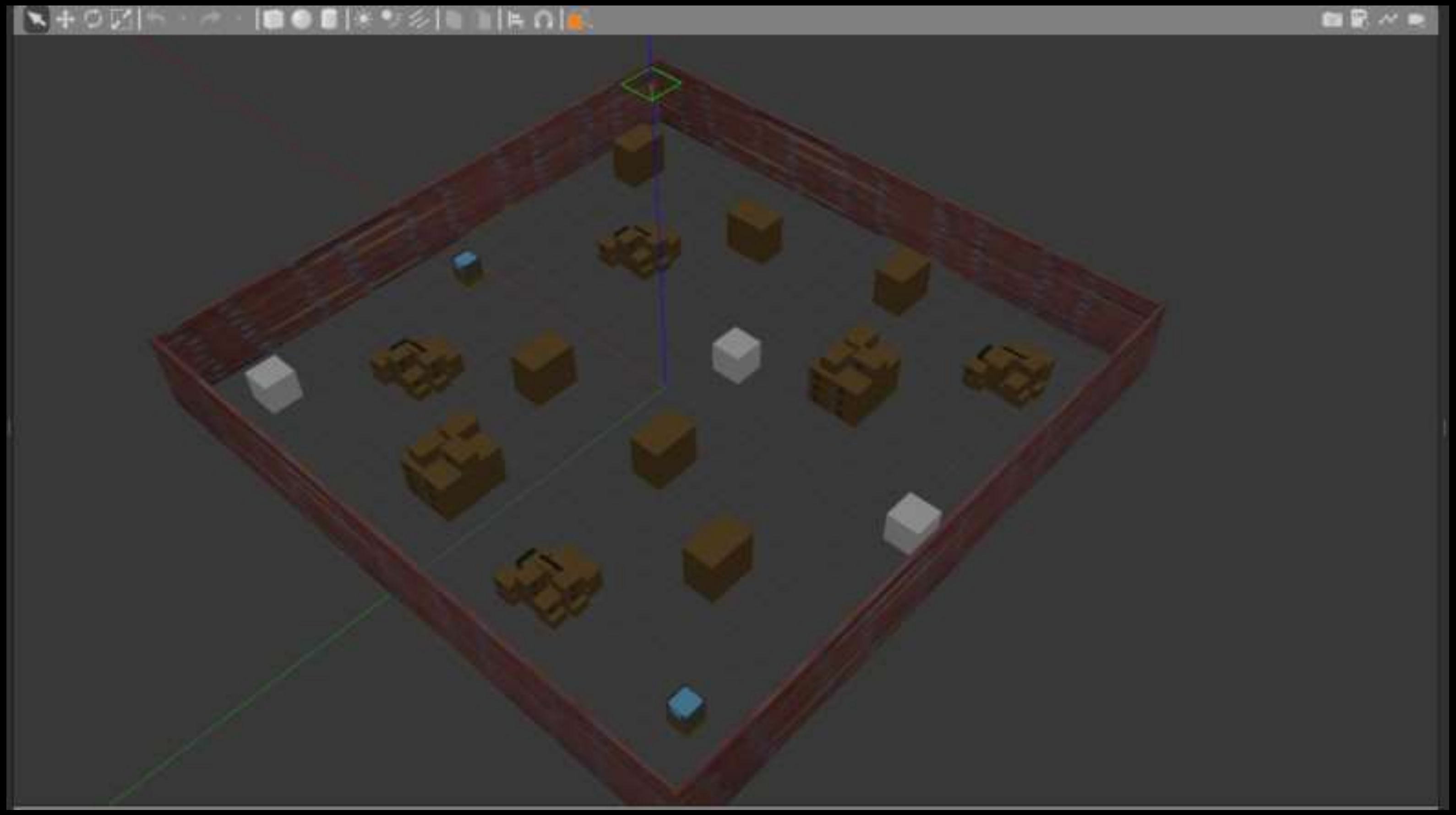
The model we made can be used for all the scenarios as:

1.It can be simulated in any type of environment.

2.Any number of bots can be used simultaneously

3.Any number of tasks can be assigned without needing to know the exact number of bots available in the environment

Scalability



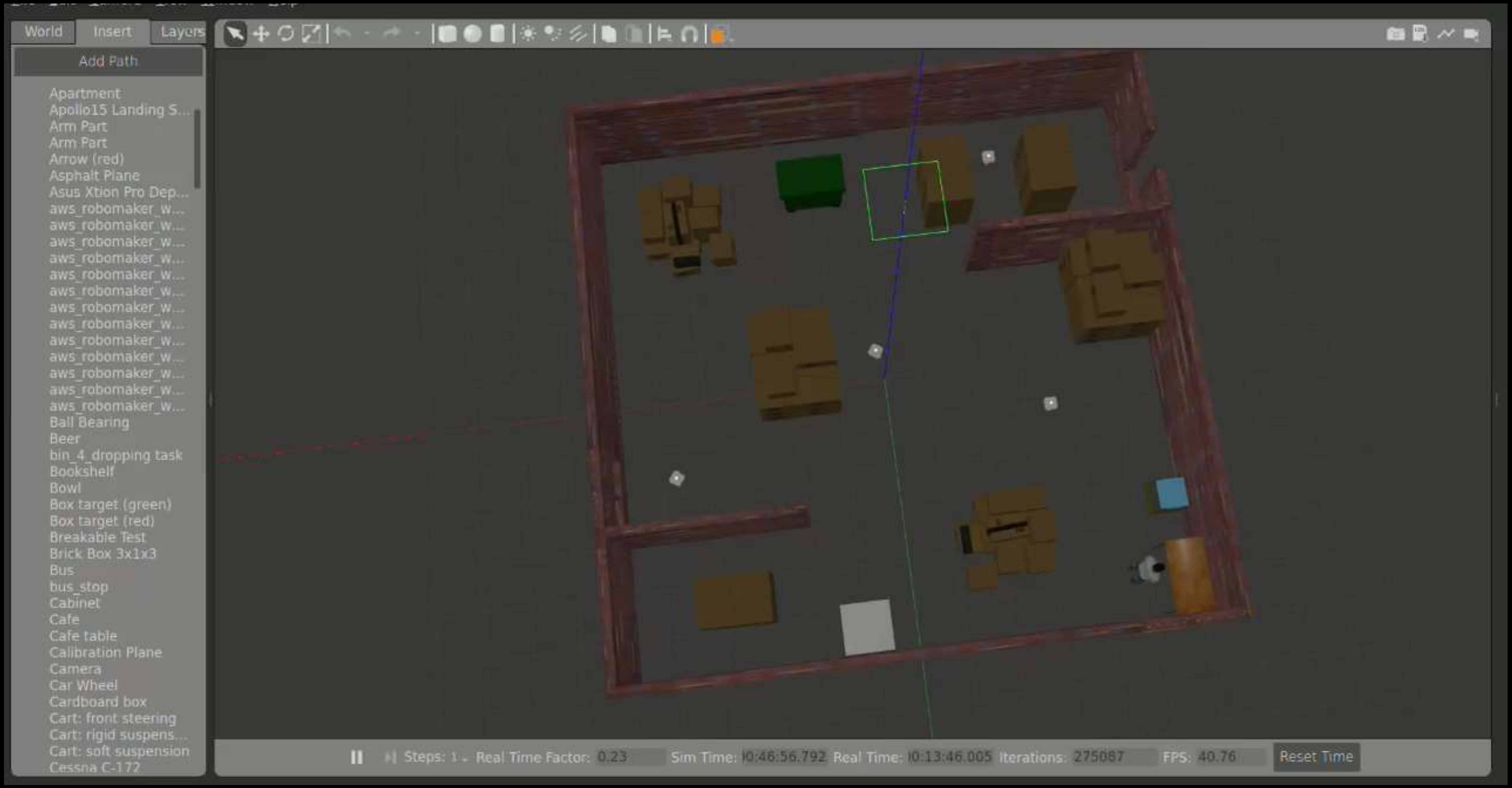
The model we made can be used for all the scenarios as:

1.It can be simulated in any type of environment.

2.Any number of bots can be used simultaneously

3.Any number of tasks can be assigned without needing to know the exact number of bots available in the environment

Dynamic Obstacle Avoidance



Conclusion

A scalable and dynamic system for exploring and mapping out an area without manual intervention.

The model handles multiple tasks and dynamic obstacles with changing environments as well