Mapping Crowd-based Dynamic Blockages for Navigation in Indoor Environments

Alleff Deus[†] Guilherme Daudt[†] Renan Maffei[†] Mariana Kolberg[†]

Abstract— **In indoor environments, the presence of people and dynamic objects is a common occurrence. In autonomous navigation, this proposes a challenge since such objects have unpredictable behavior. This affects the robots' navigation, often requiring re-planning. Having the knowledge about which regions of the map are more densely occupied by those objects can help in decision-making. In this paper, we propose a method that creates a Crowd-Based Dynamic Blockage layer using semantic information. This map represents regions that are potentially difficult to navigate due to their concentration on those obstacles. We also propose a strategy to track such obstacles and update the map continuously. Finally, the robot can decide whether a region is suitable to navigate or whether alternative paths must be chosen. Our experiments show that the Crowd-Based Dynamic Blockage can successfully improve planning in this environment.**

I. Introduction

During the last decades, autonomous robotics has evolved from static environments, built specifically for a given application, to environments with more dynamic objects, such as warehouses, offices, healthcare houses, markets, and others. In those places, the robot must adapt to them since the possibility of interaction with people and other objects is more likely to occur.This adaptation can bring more autonomy, robustness, and efficiency to the robotic system [1].

An autonomous robot needs to obtain information from its environment and translate it into a sequence of actions determined for it to reach its goal [2]. The amount of variable information makes robotic environments highly unpredictable, especially in close quarters, due to operating close to people.In this case, using higher-level information is a way to understand better the environment and the behavior of those in it.

Many branches of research focus on obtaining and maintaining information about the position of dynamic objects in the environment to use them in the robot's mapping and navigation. In [3], semantic information is used for grid map construction and marking identified objects. Similarly, in [1] and [4], semantic information is used more straightforwardly to navigate the environment and mark what has been seen. Likewise, approaches to a complete system are referenced in [5] and [6], where information is used to create layers and make navigation decisions on specific situations.

Besides knowing the position of the objects, it is necessary to know the class of the objects that are identified, to know the characteristics of the objects and how long they usually stay in the environment, e.g., a heavy object (such as large boxes or refrigerators) in the middle of a path tends to stay there for an extended period of time as opposed to the presence of a person who stays for a short period of time. Thus, to maintain a well-known environment for autonomous robots, it is prudent to keep information about the type of object and its location, which can be done by fusing the data obtained through the robot's sensors. Therefore, combining this information in a semantic map and using it for autonomous robots' navigation, task planning, and exploration is possible.

Areas with dynamic objects can cause partial blockages and delay the robot's navigation. This information hasn't been thoroughly explored in previous works. The idea is not to know precisely the position of dynamic objects, as they tend to change frequently. Rather, roughly estimate the region in which they are located and assess the impact it may cause on navigation in terms of delays if the robot travels there. This is the focus of this work. In this paper, we propose a method for constructing dynamic blockages using semantic information about people in the environment. These blockages, called Crowd-Based Dynamic Blockages (CBDB), are weighted for different configurations of people and objects and can be used by traditional planners to make informed decisions. Our main contributions are an approach to maintaining updated CBDB information and a planning strategy using CBDB.

The paper is structured as follows. Section II presents the academic background, Section III describes a system overview of our method, Section IV shows the results of conducted experiments, and Section V summarizes our work and suggests future works.

II. Related Work

Semantic mapping is a widely researched area with differing approaches and for different purposes. In [7], a semantic map is defined as obtaining relevant information in the environment in which the robot is inserted and attaching it to maps being created or other types of objects created. In some cases, its use is for open environments, like in [8], where the authors create a semantic map of the sidewalk and the road. In indoor environments, in [9], the authors build a semantic map from the robot's sensor information and use the robot's information to be able to navigate the environment and plan tasks. This information is assembled into a multi-hierarchical representation. Similarly, the strategies in [3] and [10] perform object detection and classification with the

[†]Institute of Informatics, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil adpdeus, gdaudt, rqmaffei, mariana.kolberg@inf.ufrgs.br

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

YOLO technique [11] [12] and place the information in a grid map.

A different approach to constructing the semantic map is made by [13], where the authors address the problem that the robot knows that there are objects around it but does not know what type of object is detected. In [4], a different type of semantic map is built with information from door numbers; thus, a robot can be directed to rooms within the known part of the environment. In [14] and [1], the authors use the object's semantic information to associate locations and orientations where these objects should be to devise a dynamic navigation strategy.

Other works using semantic maps focus on task planning, a well-known area where a sequence of steps is planned for the robot to execute to reach a specific location and fulfill a task. In [15], the authors use a previously created semantic map to solve tasks without human intervention. Instead of having the goal as a set coordinate for the robot, the system sets goals by giving the robot the task of finding an object or entering a room in the environment.

A complete navigation robot system can be made using the strategies above. In [16], the authors built a system using semantic mapping and path planning techniques for use in a wheelchair. The chair can use the grid map to plan its path, and the semantic information is used to iterate between humans and robots, enabling efficient navigation. In [6], the authors use semantic map information when the robot moves to a new destination, considering that at certain times of the day, it must deviate from an area and take a longer path for safety. In [5], the authors utilize Human-Robot interaction to generate layers from observed objects in the environment. This enables the robot to approach individuals and seek their attention or steer clear of areas where a person is engaged in tasks that the robot might disrupt. Similarly, in [17], the authors use layering with spatial information to characterize office spaces to create conceptual representations. Our approach also proposes a complete mapping and planning system, as presented below.

III. System Approach

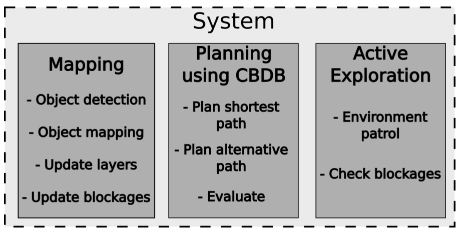


Fig. 1. Overview of the proposed system containing the mapping, the planning, and the exploratory components.

We propose a system for robots equipped with a 2D LIDAR and an RGB-D camera that operate in environments with dynamic objects and agents, such as people. Our system is composed of three major components, as seen in Fig. 1: the mapping of objects in the form of a semantic map describing the CBDB information, a planning strategy using the CBDB where the best path is chosen considering the mapped blockages that cross the robot's way and an exploration strategy which consists of actively moving the robot to the location of current blockages to update them and search for new ones. Next, we present each component in detail.

A. Mapping

The main idea of this component is to map dynamic objects in a 2D representation and then evaluate whether the regions containing them are favorable for navigation. For this, we identify and classify objects, geometrically map the environment using SLAM, mark the regions of interest, and evaluate the costs of going through those regions. For the first two tasks, we use YOLO for object detection using camera images[1], and GMapping for building a 2D occupancy grid map using LIDAR information[2], both being well-established solutions in the field of robotics and automation. For this work, only some objects are considered (e.g., person and luggage); other objects in the environment are considered unknown to this system. Some of these classified objects are shown in Fig. 2.

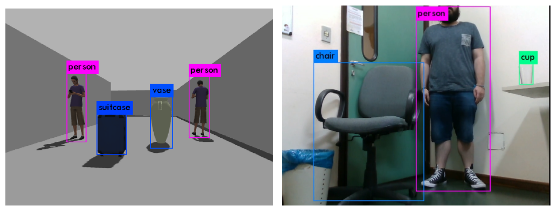
After identifying an object of interest, to obtain its location in the world, we calculate an estimated position based on the object's bounding box, determined by YOLO, and the depth information of the pixels associated with the image patch given by the bounding box. The information about the objects in the environment is used to create a layer that maintains control of the regions containing objects over time. In this work, we call this layer *"Crowd-based Dynamic Blockages"*, an example seen in Fig. 3(a).

The original occupancy grid, a lane map, and a semantic map indicating the objects' positions must also be updated before building the CBDB. A complete overview of the layers[3] used in the system can be seen in Fig. 4(a). When an object is detected, positional and semantic information about the object is marked in the semantic map, i.e. a grid map with different colored marks indicating the semantic value, as shown in 4(b). The information regarding the position, quantity, and types of objects in the environment is utilized to

[1]We use YOLO_v3, a deep learning method for object detection [11].

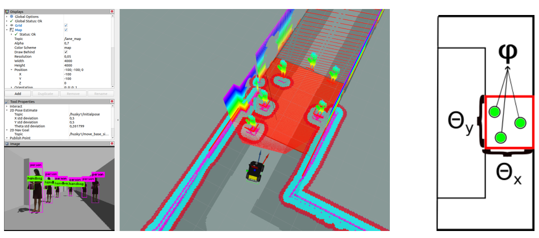
[2]We use the ROS implementation described in `http://wiki.ros.org/gmapping`.

[3]As our proposal focuses on the mapping of dynamic objects, the static mapping of the environment using Gmapping and the creation of preferred lanes for robot navigation were created prior to our experiments.



(a) Simulation (b) Real scenario

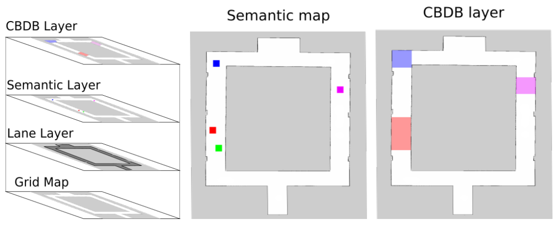
Fig. 2. Example of types of objects detected by the proposed system in simulation and real environments evaluated in our experiments.



(a) (b)

Fig. 3. Example of a CBDB applied over a grid map based on the location of detected objects. (a) Grid with a full blockage marked in red. (b) Values used to compute the weight of a blockage region, where θ_x and θ_y are the dimensions of the area affected by the objects and φ is the number of objects in this region.

create a blockage in the region, as depicted in Fig. 4(c). This blockage is assigned a weight, which will be subsequently used for planning.



(a) (b) (c)

Fig. 4. Configuration of layers used in the system. (a) All layers in the system. (b) Closer view of the semantic map. (c) Closer view of the resulting CBDB based on the information in the semantic map. Clustered objects generate larger blocking regions, as shown in red.

The dimensions of the blockage region, given by θ_x and θ_y, are defined as sides of a bounding box containing the positions of objects detected in front of the robot. Information about the number of objects detected by the robot(φ) is required as well, as exemplified in Fig. 3(b). Next, we compute the weight of the region as follows,

$$\gamma = \frac{\varphi}{\alpha\theta_x\theta_y} \tag{1}$$

where α is a weighting factor that sets the difficulty of crossing an area in terms of the number of objects per m^2. The weight is higher in proportion to the number of obstructions in the robot's path, as more turns are required and additional time is spent on navigation.

B. Planning

We convert the detected objects into weighted blockages within the CBDB layer and provide this information to the robot for planning purposes. It helps the robot determine whether it's worthwhile to navigate through a crowded path or if taking an alternate route would be better. First, our system uses A* as the standard planner, but the algorithm is interchangeable, to get the shortest possible path P_{orig}. The robot can choose the path by considering whether the area is blocked in the CBDB layer, as illustrated in Fig. 5.

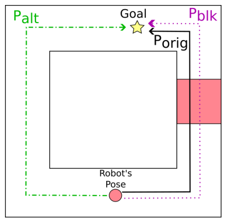


Fig. 5. Example of paths used in the planning process considering the CBDB. P_{orig} is the original path given that no blockage exists. P_{alt} is an alternative path given that the region has a full blockage. P_{blk} is the same path as P_{orig} but with the weight of the partial blockage, w_{blk}.

The impact of the blockage over P_{orig}, which we denominated w_{blk}, is obtained by applying the weight γ over the path section associated with the region, as follows.

$$w_{blk} = \frac{w_R}{1-\gamma} \qquad \gamma < \gamma_{max} \tag{2}$$

When determining the route from P_{orig}, we consider the weight of the blocked region (w_R) that the route passes through, incorporating the weight assigned to the blocked region. Considering that the robot needs a feasible space to travel, the variable γ_{max}[4] is used to determine how crowded a region can be before being considered totally blocked.

The updated path regarding the blockage region, P_{blk}, is P_{orig} with an updated cost, as given by

$$cost(P_{blk}) = cost(P_{orig}) + (w_{blk} - w_R). \tag{3}$$

Then, we analyze if the updated path is sufficiently better than an alternative path, P_{alt}, that completely blocks the designed region, as shown in Fig. 5. The decision follows Eq 4. If, after weighing the blockage, the cost of P_{blk} is higher than P_{alt}, we select the alternative path. Before selecting P_{blk}, we ensure that its cost must be lower than P_{alt} by a threshold w_{diff}[5]. This parameter was added after observing that deciding between narrow differences produced inconsistent results.

$$Path = \begin{cases} P_{alt} & if\ cost(P_{alt}) < cost(P_{blk}) + w_{diff} \\ P_{orig} & otherwise \end{cases} \tag{4}$$

C. Navigation and Object Position Update

In order to ensure the accuracy of blockage region information, the robot maintains a list of previously detected objects. This list is regularly evaluated to determine if the objects are still present and to update their positions if they have moved. An example of objects that need to be analyzed can be seen in Fig. 6, each having a different color assigned to them.

During the exploration phase, visual information is processed and examined. An algorithm describing the decision-making when encountering an object is shown in Alg. 1.

[4]All the experiments executed in this work set the values of $\alpha = 2$ and $\gamma_{max} = 0,7$

[5]In our experiments, the w_{diff} value was taken as a 350 step difference between the two paths.

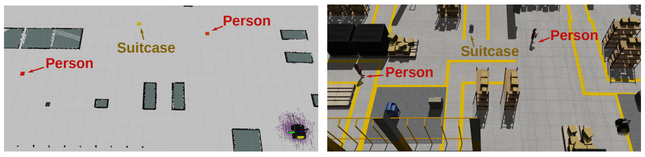


Fig. 6. Example of exploration of desired objects that need to be analyzed and updated. (a) The different classes of objects marked in the grid map. (b) The real position of the objects in the environment.

When the robot performs the exploration, if no object is found, the occurrence in the grid map is cleaned, and the robot goes to the next destination and repeats the process. When the robot sees more than one object, it analyzes all objects.

Algorithm 1: Active Exploration Strategy

```
1 while coverage not completed do
2     if object is detected then
3         if detected object is already mapped then
4             Update object information
5         else
6             Clear old object and update the new one
7     else
8         Clear object information
9 end while
```

IV. Experiments

We ran our experiments with a Pioneer 3-DX mobile robot equipped with a SICK LMS 200 LIDAR and an Intel Realsense D435 RGB-D camera in simulated and real scenarios. Simulated tests were executed using the Gazebo simulation environment with ROS, while tests with the physical robot were performed in the Institute of Informatics building of UFRGS-Brazil. All the simulated experiments were performed in a platform with Ubuntu 20.04 equipped with an AMD Ryzen 5 processor, 48GB RAM, and a GPU Nvidia GeForce RTX 3060. The experiment in the real environment was performed on a platform with Ubuntu 20.04 equipped with an AMD Ryzen 7 processor, 16GB RAM, and a GPU Nvidia GeForce RTX 3060.

Our experimental validation was divided into three parts. First, we performed experiments that analyzed the impact of partial blockages in the robot's path and how knowing such information beforehand leads to better navigation choices. Next, we analyzed the behavior of our algorithm in different situations, observing when the robot decides to take the original shortest path and when it is better to take an alternative longer path depending on the map topology and blockage configurations. Finally, we compared our results with a standard planning strategy[6].

[6]For this, we used the move base package for ROS, which can be found in `http://wiki.ros.org/move_base`

A. Analyzing the impact of people moving in the robot's path

Preliminary experiments were conducted to verify how partial obstructions can hinder navigation, making it slower as the robot needs to adjust its route when encountering obstacles. Simulated tests were done in a corridor-shaped environment, Fig. 7(a). Dynamic obstacles were represented by people walking through the corridor, while stationary obstacles were represented by office furniture and people standing still. A complete scenario is shown in Fig. 7(b). The robot moved 20 times from one side of the map to the other and returned, reacting to obstacles (which moved independently of the robot's motion). We compared the crossing times of the clear path with the partially blocked path in Fig. 7(c).

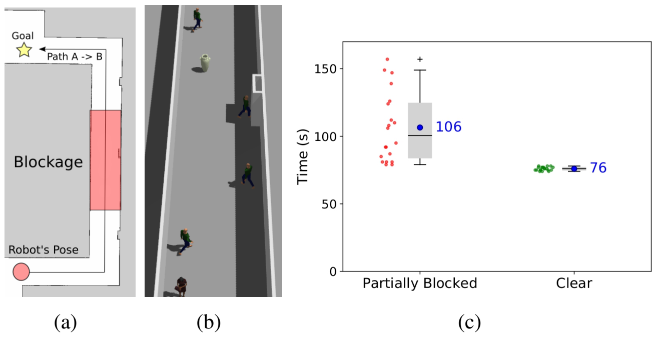
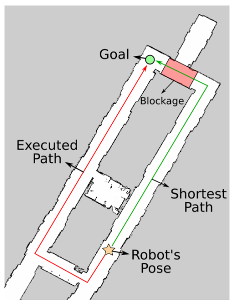


Fig. 7. Experiments in a simulated corridor environment with partial blockage in the path. (a) The robot repeatedly moves from one end of the corridor to the other and returns. (b) Example of objects present in the partial blockage sector. (c) Box plot comparing elapsed travel time in paths that have a partially blocked zone or clear paths. The red and green dots are the travel times in the partially blocked zone and the clear path, respectively. The blue circle and number indicate the average time for each scenario.

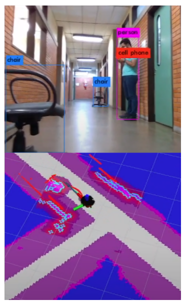
In a fully known environment (e.g., the clear zone), the robot follows the planned path consistently, as seen from the low variance in the times measured. On the other hand, we can see that the presence of dynamic obstacles, even without completely obstructing the path, substantially delays the robot's navigation (in the tested case, by around 40%). Therefore, using the information in the CBDB, the robot can plan whether it is worth trying to cross a potentially blocked area or detour to an alternative path.

B. Analyzing the navigation's decisions using CBDB

For the second part of the experiments, we analyze how different types of maps and blockage locations impact the decisions of our planning strategy. We first tested the approach in the real-world environment of the Institute of Informatics through a few experiments to observe its practical application. An example of a situation encountered in our tests can be seen in Fig. 8(a), where the shortest path goes through a blockage, and our system decides to take the alternative path. In Fig. 8(b), we can see the robot detecting objects in a corridor in a blockage position. In environments with similar corridor sizes, the differences between P_{blk} and P_{alt} are negligible; thus, the method chooses to follow the path without partial blockages, as expected.



(a)



(b)

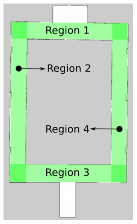
Fig. 8. Experiments performed in the Institute of Informatics. (a) Example of the shortest path calculated and the path executed by the robot after the blockage calculation. (b) Example of dynamic objects observed in the environment that generate a partial blockage in the corridor.

Next, we performed multiple sets of simulated experiments in three different scenarios[7], 20 times each. Scenario A had 192 different configurations[8] of blockages, initial positions, and goals. Due to the complexity of scenarios B and C, we randomly sampled among possible combinations of blockages, initial positions, and goals, these scenarios are shown in Figs. 9(a), 9(c) and 9(e). Each map was divided into regions associated with the corridors. A test configuration was made by randomly selecting regions for the initial pose, the goal, and a blockage region. Given the large number of tested configurations, we focused on the planning process and evaluation of the results, without conducting complete tests involving the physical movement of the robot in the Gazebo simulator.

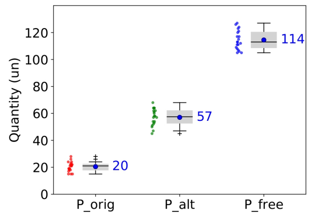
In each test, the robot calculates a path from an origin to a destination. If a blockage is in the way of the original path, our system computes how much this blockage affects the plan. Results are shown in Figs 9(b), 9(d), and 9(f). We can see that path P_{alt} was selected with greater frequency than P_{orig}in all the different maps. Because the generation of blockages in the environment is random, most of the time, the shortest path is not affected by a blockage. In this case, we consider the path chosen as free, or P_{free}. In other situations (which varied from 40% to 45% of the time, depending on the scenario), the blocking region coincided with a section of the shortest path, and the algorithm had to decide between paths. In all scenarios, the algorithm predominantly chose the alternative path to circumvent the partially blocked region. However, the difference decreases when there are fewer options for alternative paths: in scenario C the choice for P_{alt} is 8.5 greater than for P_{blk} (i.e. 68/8); in scenario B it decreases to 6.2 (i.e. 74/12); and in scenario A it drops to 2.9 (i.e. 57/20). Therefore, when there are fewer alternatives, it becomes worthwhile considering traversing the partially blocked regions.

[7]Each scenario is a simplified recreation of the Institute of Informatics with minor modifications in the structure and length of the corridors.

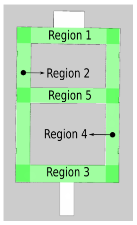
[8]This value is obtained with $f(x) = x! \times 2x$, where x is the number of regions in each scenario



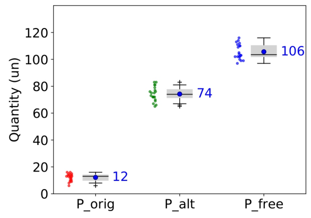
(a) Scenario A



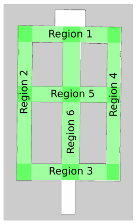
(b) Decisions in Scenario A



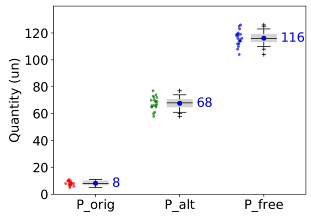
(c) Scenario B



(d) Decisions in Scenario B



(e) Scenario C



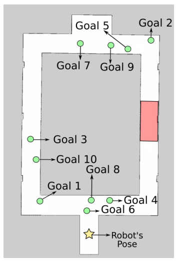
(f) Decisions in Scenario C

Fig. 9. Results of simulated experiments to analyze planning decisions. Left: scenarios tested in the simulated experiments, with each corridor region marked in the maps. Right: box plots showing the decisions made in each scenario.

C. Analyzing the complete planning system

In our final experiment, we assessed the complete planning system by executing the decision-making process and simulating the robot's motion in Gazebo. We compared the results with those obtained from a standard navigation strategy that does not utilize CBDB. The measured run time results, utilizing a configuration system where the blockage is located in a specific map region shown in Fig.10(a), are summarized in Fig.10(b). Column *'CBDB? No'* in Fig. 10(b) describes the results obtained using the standard navigation algorithm, i.e., when the robot does not have the blockage information. Column *'CBDB? Yes'* shows our system's results. We randomly chose 10 positions displayed on the map. The robot starts from its initial position and travels to goal positions sequentially. Only one blocking region exists on the right side of the map, so some shortest paths between goals do not cross the blockage, while others do.

Additionally, two conditions of CBDB information were tested: one with high accuracy and the other with low. In the experiments marked as *'φ set'*, the exact number of obstacles predicted in the CBDB were placed in the blockage area. In the experiments marked as *'φ random'*, a variable amount of obstacles was placed in that region. This means that the number of obstacles likely differed from what was predicted, potentially resulting in a different crossing time than initially expected for the robot.



(a)

	φ set			φ random		
	CBDB?			CBDB?		
#	No	Yes	Choice	No	Yes	Choice
1	26	21	P_{free}	25	22	P_{free}
2	95	84	P_{alt}	100	81	P_{alt}
3	70	73	P_{free}	71	69	P_{free}
4	46	48	P_{free}	46	46	P_{free}
5	74	91	P_{orig}	62	52	P_{alt}
6	80	66	P_{orig}	115	65	P_{orig}
7	87	78	P_{alt}	100	81	P_{alt}
8	85	78	P_{alt}	107	78	P_{alt}
9	130	75	P_{orig}	88	87	P_{orig}
10	74	75	P_{free}	74	75	P_{free}

(b)

	φ set CBDB? [s]		φ random CBDB? [s]	
#	No	Yes	No	Yes
Scenario A	76,7	68,9	78,8	65,6
Scenario B	67,2	60,4	66	56,8
Scenario C	52,8	51,4	52,5	51,8

(c)

Fig. 10. Fig. 10(a) shows the goals used in one of the experiments. The blockage is the red region and the start position of the robot is represented by the yellow star. Fig. 10(b) shows the results obtained in this experiment, where each row is directly related to goals in Fig. 10(a). The highlighted rows are the paths chosen with blocked regions. Fig. 10(c) has the mean times for all configurations on each scenario.

P_{free} represents the choices made when the path to the goal does not pass through the blocked region. P_{orig} represents the choices made when the calculated cost of the detour doesn't reach the threshold, maintaining the original path. P_{alt} represents the choices where the robot detours from the blocked region. The paths followed by both algorithms when choosing P_{orig} are virtually the same. However, the time taken to travel through them may be different, which is justified by the fact that the robot has to make real-time adjustments to avoid unforeseen dynamic obstacles.

Fig. 10(c) synthesizes the mean time for each case scenario (Fig. 10(b) being a fragment of the experiment) for all the maps used. With this, we can compare that on the φ set, scenarios A and B have an average time using the CBDB 11% less than not using it (i.e., 68.9/76.7 and 60.4/67.2). However, scenario C's average time is only 3% lower (i.e., 51.4/52.8). When φ is random, using CDBD in scenario A produces a time that is 17% faster (i.e., 65.6/78.8) and 14% faster in scenario B (i.e., 65.6/78.8). In scenario C, the times were only 2% lower (i.e., 51.8/52.5).

With these results, it is possible to note that when the map has different paths that can be chosen, the gain with the usage of CBDB decreases. In contrast, using CBDB in environments with long corridors or limited routes is beneficial.

V. Conclusion and Future Work

In this paper, we proposed a system composed of semantic mapping and planning that uses the Crowd-Based Dynamic Blockage layer, our main contribution. Using this information during planning can avoid wasting time going to a blocked path while saving computational resources by pruning costly paths. With our experiments, we observed that a robot navigating a corridor with dynamic objects might take longer time as opposed to empty corridors. Additionally, using our method, we perceived an improvement when evading blocked regions using previously acquired information. Our method performs similarly to standard navigation methods in areas with wide open spaces since the robot can avoid a blockage without major path alteration. However, it shows better results when operating in narrow hallways or corridors.

For future works, we plan to use our system in a more realistic environment while considering more object classes and vast areas. Additionally, the technique can be improved by using more semantic information, considering types of blockages, and clearing areas through time. We could also use the CBDB layer to adjust the robot's position in the occurrence of poor localization.

References

[1] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe, "Curious george: An attentive semantic robot," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, 2008, from Sensors to Human Spatial Concepts.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[3] H. Qiu, Z. Lin, and J. Li, "Semantic map construction via multi-sensor fusion," in *2021 36th YAC*, 2021, pp. 495–500.

[4] J. Liang, W. Song, L. Shen, and Y. Zhang, "Indoor semantic map building for robot navigation," in *2019 IEEE 8th Joint ITAIC*, 2019, pp. 794–798.

[5] P.-T. Wu, C.-A. Yu, S.-H. Chan, M.-L. Chiang, and L.-C. Fu, "Multi-layer environmental affordance map for robust indoor localization, event detection and social friendly navigation," in *2019 IEEE/RSJ IROS*, 2019, pp. 2945–2950.

[6] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *2016 IEEE ICRA*, 2016, pp. 5729–5736.

[7] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *2017 IEEE/RSJ IROS*, 2017, pp. 5079–5085.

[8] D. F. Wolf and G. S. Sukhatme, "Semantic mapping using mobile robots," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 245–258, 2008.

[9] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernandez-Madrigal, and J. Gonzalez, "Multi-hierarchical semantic maps for mobile robotics," in *2005 IEEE/RSJ IROS*, 2005, pp. 2278–2283.

[10] Y. Xu, Q. Dai, H. Gu, G. Lu, J. Gu, and L. Hua, "Semantic map construction based on monocular vision," in *2019 11th International Conference on IHMSC*, vol. 2, 2019, pp. 126–129.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE CVPR*, 2016, pp. 779–788.

[12] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS," 2016–2018. [Online]. Available: https://github.com/leggedrobotics/darknet_ros

[13] S. Kowalewski, A. L. Maurin, and J. C. Andersen, "Semantic mapping and object detection for indoor mobile robots," *IOP Conference Series: Materials Science and Engineering*, vol. 517, no. 1, p. 012012, apr 2019.

[14] N. Yang, Z. Mi, Y. Guo, B. Sadoun, and M. S. Obaidat, "Fast local map construction of robot using semantic priors," in *2020 International Conference on CCCI*, 2020, pp. 1–5.

[15] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 955–966, 2008, semantic Knowledge in Robotics.

[16] C. Zhao, H. Hu, and D. Gu, "Building a grid-point cloud-semantic map based on graph for the navigation of intelligent wheelchair," in *2015 21st ICAC*, 2015, pp. 1–7.

[17] H. Zender, O. Martínez Mozos, P. Jensfelt, G.-J. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, 2008, from Sensors to Human Spatial Concepts.