SEMANTIC CLASSIFICATION IN SEARCH AND RESCUE ENVIRONMENTS

by

151220164090 MUHAMMED KOCAOĞLU

151220164064 MUHAMMED ALİ UZUN

151220164012 YUNUS EMRE IŞIKDEMİR

A Graduation Project Report

Electrical Electronics Engineering Department

JUNE 2021

SEMANTIC CLASSIFICATION IN SEARCH AND RESCUE ENVIRONMENTS

by

151220164090 MUHAMMED KOCAOĞLU

151220164064 MUHAMMED ALİ UZUN

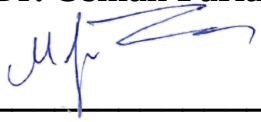
151220164012 YUNUS EMRE IŞIKDEMİR

has been approved by

Supervisory Committee

Asst. Prof. Dr. Burak Kaleci

Prof. Dr. Osman Parlaktuna



Asst. Muhammed Oğuz Taş

Prof.Dr. Gökhan Çınar, Chairperson

ÖZET

Sel, yangın, zehirli madde yayılımı ve deprem gibi afetlerden sonra bina içi ortamlarda yapılması gereken arama ve kurtarma faliyetleri, binanın yapısal bütünlüğünün bozulması ve zehirli madde sızıntısı gibi risklerden dolayı insan ve hayvanlar için hayati tehlike oluşturmaktadır. Bu yüzden, robotların arama ve kurtarma görevlerinde kullanılmasına yönelik çalışmalar son yıllarda oldukça artmıştır. Robotların, bu zorlu ortamlarda kendilerine verilen görevleri yerine getirebilmeleri için yeteneklerini geliştirmeleri gerekmektedir. RoboCup ve DARPA gibi organizasyonlar bu gelişime katkı sağlaması amacıyla her yıl yarışmalar düzenlemektedir. Bu yarışmalara katılan ekipler NIST (The National Institute of Standards and Technology) tarafından belirlenen standart test ortamlarında robotlarının yeteneklerini ölçmektedirler. Bu proje kapsamında, arama ve kurtarma ekiplerine yardımcı olması amacıyla NIST standart test ortamlarına benzer ortamlarda 3B anlamsal harita oluşturulmuştur. 3B anlamsal harita bu projenin yenilikçi yönüdür. Geçmiş çalışmalarda, 3B metrik haritaya nesneler eklenerek anlamsal harita elde edilirken bu projede nokta bulutu şeklinde anlamsal harita elde edilmiştir. Böylece bütün detaylar nokta bazında haritada yer almıştır.

Bu projede ilk olarak, ortamı temsil edecek olan küresel metrik harita (3B doluluk ızgarası) oluşturulmuştur. Küresel metrik harita voksellerden oluşmaktadır. Voxel pikselin 3 boyutta karşılığıdır. Bu haritada yer alan vokseller bilinmeyen, boş ve dolu olarak sınıflandırılmıştır. Robotta bulunan RGB-D kamera yardımı ile robotun algıladığı bölgede yer alan vokseller boş veya dolu olarak güncellenmiştir. Ayrıca, eş zamanlı olarak ortamda bulunan rampalar, zemin ve duvarlar, DGCNN (Dynamic Graph CNN) nokta tabanlı 3B derin öğrenme mimarisi ile anlamsal olarak sınıflandırılmış ve 3B anlamsal haritanın çıkarılmasında kullanılmıştır. Anlamsal haritanın ürettiği rampa, duvar ve zemin bilgileri kullanılarak topolojik harita üretilmiştir. Bu bilgiler kullanılarak üretilen topolojik haritadaki düğümler Minimum Spanning Tree (MST) algoritması kullanılarak birleştirilmiştir, böylece robotun otonom olarak gezebileceği yol planının hazırlanması sağlanmıştır. Optimum yol planını belirlemek için Dijkstra algoritması kullanılmıştır. Afetzedeler ise, You Only Look Once (YOLO) ile tespit edilmiştir.

Projede üretilen 3B anlamsal harita ile, arama ve kurtarma ekipleri çalışmalarına başlamadan önce ortam ile ilgili detaylı bilgileri elde edilmiştir. Bu sayede, hayati risklerinin azalması sağlanmıştır. Ayrıca, ortaya çıkan yazılımın katma değeri yüksek bir ürün olarak yurt dışına ihraç edilmesi söz konusu olabilir.

Anahtar Kelimeler: *arama ve kurtarma robotları, 3B anlamsal haritalama, 3B metrik haritalama, topolojik haritalama, afetzede tespiti, nokta bulutu verisi.*

ABSTRACT

Search and rescue activities that should be carried out in indoor environments after disasters such as flood, fire, toxic substance spread, and earthquake may cause a life-threatening danger for humans and animals due to risks such as structural integrity of the building and toxic substance leakage. Therefore, studies that utilize robots in search and rescue missions have increased considerably in recent years. Robots need to improve their abilities in order to perform the tasks assigned to them in these challenging environments. Organizations such as RoboCup and DARPA organize competitions every year to contribute to this development. The teams participating in these competitions measure the abilities of their robots in standard test environments determined by NIST (The National Institute of Standards and Technology). Within the scope of this project, a 3D semantic map was created in environments similar to NIST standard test environments in order to assist search and rescue teams. The main contribution of this project is producing 3D semantic map. In previous studies, a semantic map was obtained by adding objects to the 3D metric map, while in this project a semantic map in the form of a point cloud was obtained. Thus, all the details are located on the map on a point basis.

In this project, firstly, a global metric map (3D occupancy grid) was created to represent the environment. 3D occupancy grid consists of voxels. Voxel is a 3D form of a pixel. The voxels in this map are classified as unknown, empty, and occupied. All voxels are initialized with unknown state. With the help of the RGB-D camera in the robot, the voxels in the robot's region of view have been updated as empty or full. In addition, ramps, terrain, and walls in the environment simultaneously were semantically classified with the DGCNN (Dynamic Graph CNN) point-based 3D deep learning architecture and used in the extraction of the 3D semantic map. The topological map was produced by using the information produced by the semantic map. The nodes in the topological map produced by using that information were combined using the Minimum Spanning Tree (MST) algorithm, thus it was provided to prepare a route plan that the robot could navigate autonomously. The Dijkstra algorithm was used to determine the optimum path plan. The victims were identified with YOLO.

With the 3D semantic map produced in the project, detailed information about the environment was obtained before the search and rescue teams started their work. In this way, vital risks can be reduced. In addition, it may be possible to export the resulting software abroad as a product with high added value.

Keywords: *search and rescue robots, 3D semantic mapping, 3D metric mapping, topological mapping, victim detection, point cloud data.*

ACKNOWLEDGEMENT

We would like to thank our supervisor, Asst. Prof Burak Kaleci for his support and guidance in the project with his knowledge and experience. We are grateful to Research Assistant Muhammed Oğuz Taş and Research Assistant Kaya Turgut for their support, guidance, and contribution to the project.

This project is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) 2209-B Industrial Oriented License Graduate Thesis Support Program, Grant No: 1139B412000223

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	**Explanation**
NIST:	The National Institute of Standards and Technology.
AFAD:	Afet ve Acil Durum Yönetimi Başkanlığı.
DGCNN:	Dynamic Graph CNN
PCL:	Point Cloud Library
RANSAC:	Random Sample at Consensus
RTAB-Map:	Real-Time Appearance-Based Mapping
YOLO:	You Only Look Once
DARPA:	Defense Advanced Research Projects Agency
MST:	Minimum Spanning Tree

1. INTRODUCTION

1.1. Objectives

Within the scope of this project, it is aimed to create a 3D semantic map that search and rescue teams can benefit from. Creating 3D map is one of the subjects which increased its popularity in the field of robotics with the development of 3D perception technologies. Especially in recent years, in RoboCup competitions, 3D map quality has started to be evaluated as a success criterion. Semantic map and metric map are the ways of representing environments as 3D. For both metric and semantic maps, the point cloud data of the environment is required. There are many mapping packages available in ROS to obtain point cloud data of the environment. RTAB-Map and OctoMap are among the packages that are used for this purpose. In the scope of this project, we aim to provide our own mapping method to gather point clouds of the environment. Eventually, metric and semantic maps will be created with the help of the point cloud data. A metric map is obtained by representing a table, chair, wall, etc. which exists in the environment with voxels. In this project, it is planned to create a metric map with a predetermined size and resolution. Also, a point-based 3D semantic map will be created to derive the planes that are ramps, walls, and terrains. It is also aimed to create a topological map from a semantic map. After the topological map is created, its nodes will be used to navigate in the environment. In order to detect the victims in the environment, the YOLO algorithm will be used.

1.1. Novelty

With the project is successfully completed, the targeted product will be a software capable of 3D mapping in indoor environments after the disaster. This software is well suited for these environments as it will be fed with point clouds. Software that successfully 3D mapping using object recognition and image data are available in previous studies. However, the success of these studies directly depends on the quality of lighting. There will probably be no electricity in the indoor environments after the disaster. In this case, problems in the lighting of these environments will cause the software running using existing image data to fail. Two main problems are still waiting to be solved in software that creates exploration

and 3D semantic maps using point cloud data. The first problem is that the computational cost increases due to the large number of points in the point cloud. In previous studies, it has been tried to create a 3D occupancy grid by keeping the point cloud data in data structures such as an 8-tree structure (octree). However, this technique may not be efficient. In order to increase the efficiency, we have created a 3D array in which we stored the voxels with their centers. The sizes of the array and the resolution of the voxels were predetermined. Initially, all the voxels were assumed to be unknown. As the robot moved, the voxels were labelled as free or occupied. With this method, we accelerated the creation of the metric map.

Deep learning architectures, which have been very popular in recent years, are used to determine the semantic class of each point on the map. Thus, semantic classification is made on point basis with high accuracy. Moreover, we obtained the topological map with the help of the ramps, walls, and terrain information of the semantic map.

2. REQUIREMENTS SPECIFICATION

Significant technological developments in the fields of software and hardware in recent years have increased the studies for the widespread use of robots in many areas. One of these areas is mapping of indoor environments after disasters such as floods, fires, toxic substances spread and earthquakes. Organizations such as RoboCup and DARPA, organize competitions in order to measure whether robots have the necessary hardware and software capabilities to perform this task. RoboCup organization has organized various competitions regarding search and rescue tasks every year since the early 2000s and evaluated the teams participating in the competition according to certain standards and criterias. Kitano and Tadokoro [1], in their study conducted in 2001, evaluated the difficulties that robots may encounter in search and rescue tasks, current technologies and developments that need to be made. In the light of these evaluations, new and more demanding standards and criteria are determined every year, so the capabilities of robots have been improved. In the evaluation made by Sheh et al. [2] in 2016, the improvements made by robots have been revealed thanks to the RoboCup competitions. The competitions held in 2016 and later have focused on mapping. In the previous studies, methods that successfully perform the 3D mapping task using image data has been proposed [3,4]. However, image data is not suitable for post-disaster search and rescue environments where lighting conditions are bad, dusty and low

visibility. Therefore, teams participating in RoboCup competitions generally prefer to use sensors that can generate point clouds such as 2D and 3D laser, LiDAR and RGB-D cameras. For example, Yıldız robot, which was developed by Yıldız Technical University's Yıldız team and earned them first place in the search and rescue class in the RoboCup competition in 2016, has a UTM-30LX laser and a Kinect RGB-D camera [5]. In the past years, teams participating in RoboCup competitions have benefited from various mapping methods using the mentioned sensors for 3D mapping. For example, in the RoboCup organization held in 2018, the Cambridge University team is used the gmapping method, while the Yıldız Team is used the R-SLAM (Simultaneous Positioning and Mapping) method. Besides all these, there are frequently used methods such as hector_mapping [6], OctoMap [7] and real-time appearance-based mapping (RTAB-Map) [8] [9].

Within the scope of this project, it is aimed to perform 3D semantic mapping by using the Asus Xtion Pro RGB-D camera located on the robot. This map will be created for search and rescue teams to have detailed information about the environment before they start their activities. Unlike previous studies, objects such as ceilings, floors, walls, flat and inclined ramps and victims will be included in the 3D map to be created within the scope of this project. In order to achieve this, firstly, the basic information obtained through the robot's sensors will be transferred to a global metric map (3D occupancy grid) by using the RTAB-Map method. 3D occupancy grid will also be created using 3D array. In 3D array, the empty, occupied and free cells will be stored. In order for the robot to create the semantic map, it is required to semantically classify the points as wall, floor, inclined and flat ramp in the point cloud data obtained with the RGB-D camera. To achieve this, it would be a suitable solution to use point-based deep learning architectures. In their study, Turgut and Kaleci [10] implemented the PointNet, PointNet ++, DGCNN, and PointCNN architectures in order to determine the semantic classes of wall, floor, inclined and flat ramp objects in search and rescue areas. And they discussed the positive and negative aspects of these architectures. It was decided to use the DGCNN method by making use of this study. It is aimed to create a path plan in order to reach the determined goal or goals. Topological map will be used to create this path plan and shortest path finding methods will be used. In cases where there is only one target, Dijkstra's [11] method will be used to create the path plan. After the path plan is created, the robot will perform the navigation task to reach the given destination. In order to be successful in this task, appropriate linear and angular velocities should be

determined in the environment where there are ramps. The robot will continuously update its 3D metric. Finally, the robot must detect the victims in the environment and show their locations accurately. Previous studies include architectures that have been successful in identifying people, such as YOLO [12], SSD, RCNN and Faster RCNN. Among these methods, it is decided to use the YOLO method, by considering the detection speed and detection distance criteria.

3. PROPOSED METHODS

In this section, creating metric, semantic and topological maps' steps will be detailly explained. Besides, the path planning and victim detection steps will be mentioned.

3.1. Creating Metric Map

Robots generally need an accurate representation of the environment in order to perform the tasks assigned to them. Metric and topological mapping methods have been used frequently for this representation in previous studies . Metric maps are usually represented by grids in 2D space those divide the environment into equal-sized cells. Although metric maps can be created easily, it requires high processing power when the number of cells are increased. On the other hand, since topological maps are expressed with nodes and edges connecting the nodes, they reduce the required processing power and work more efficiently in real-time applications. The main disadvantage of topological maps compared to metric maps is that they are more difficult to create [18].

An example metric map is shown in Figure 1. In the figure, gray, green, and red colors show unknown, empty, and occupied voxels, respectively.

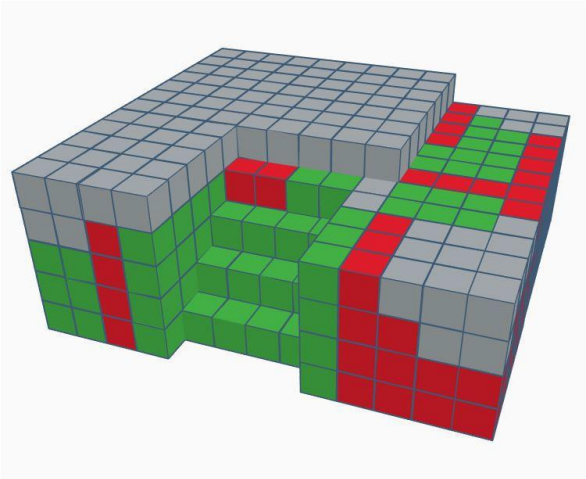


Figure 1. *Metric map*

Within the scope of this project, the metric map of the indoor environment in the search and rescue environment has been created with a 3D occupancy grid. In order to create the 3D metric map, RGB-D camera has been used in the Gazebo simulation environment. Common mapping methods such as OctoMap and RTAB-Map are suitable to create the metric map. Silva et al. [9] discussed the positive and negative aspects of OctoMap and RTAB-Map methods which are used for post-disaster search and rescue tasks in their study. The authors stated that the installation of RTAB-Map and it's use in ROS offers a more user-friendly experience, contrary to RTAB-Map, more intense procedure should be used for most operations to be performed in OctoMap. In addition, while 3D object classification can be done with RTAB-Map method and objects whose classes are determined can be transferred to the map, OctoMap method does not have such a capability. For these reasons, it was preferred to use the RTAB-Map method in this study.

Initially, we used RTAB-Map in order to obtain the metric map [17]. In the study, the point cloud data is converted into voxels using octree data structure. In the creation of the metric map, the semantic map is used as well. From the walls, terrain, inclined and straight ramps, the voxels are labelled as red, yellow, blue, and pink respectively. RTAB-Map was very useful for creating metric map because the point cloud data represents free cells of the environment was already available in the package itself. The point clouds respresent the occupied spaces in the environment were also obtained by using RTAB-Map at the first stage of our study. The robot was moved in the environment with teleoperation method and 160 Point Cloud Data (PCD) has been gathered. Each of these point cloud data

processed seperately and created an metric map with these seperate point cloud datas. In order to create metric map, octree structure available in PCL libraries has been utilized. With the help of the RTAB-Map and PCL, the free and occupied voxels has been generated. We also represent occupied cells with blue, pink, red and yellow in order to show the inclined ramps, straight ramps, walls and terrain in the environment, respectively. Which of the cells belong to which of the class were determined by using semantic map which means that semantic map was used in the stage of creating metric map.

After the metric map with the data obtained from RTAB-Map has been created, we have made our own mappig by which we also created a metric map. As it is stated, the point cloud data which represent free and occupied spaces in the environment was already available in RTAB-Map. However, our own mapping method provides us only the point clouds that represent occupied spaces in the environent. For this reason, we also determined the free cells in the environment with our own algorithm. We tested the algorithm with large environment as well. Without depending the environemnt size, the speed for determining the free and occupied voxels was the same.

In the algorithm, we first determined resolution and sizes of the environment and created a 3D array where the center information about metric map was stored. If the voxel is free, its value is 1, if the voxel is occupied, its value is 0 and lastly, if the voxel is unknown, its value is 0.5. Initially, all voxels in the environment were assumed to be 0.5 which means unknown. As the robot moved around, the voxels are labelled as free or occupied. According to algorithm we designed, constructing a metric map consists of 2 stages. The first stage is to find the occupied voxels and the second stage is to find free voxels. In the first stage, the process is simple. The point cloud data obtained from our own mapping method is used. The points are avaiable with their location information in point cloud vector and must be determined which of the points belong to which of the voxel. A voxel may contain many points and most probably it will since the resolution we determined for metric map is 0.1 m. In order to determine the voxel they belong to a simple process is applied. The global location of the points are divided by the resolution of the metric map and converted to integer. If the point is located in position x = 1.2, y = 3.1, and z = 0.8, this would mean that the point belong to voxel with indexes 12, 31, and 8. The problem here is that, there may be more than 1 point in one voxel. As a result of that, the same voxel may be needed to be

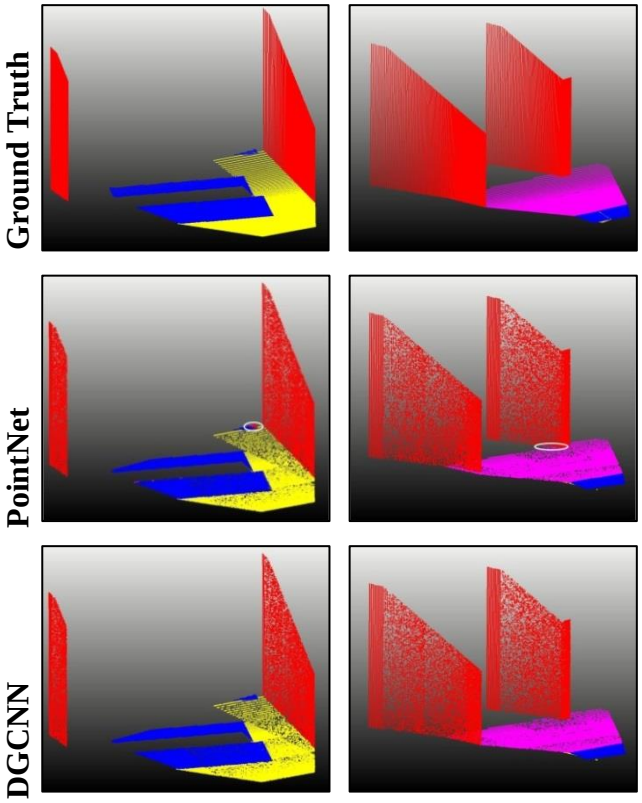
accessed many times in vain. To overceome this problem the voxels' locations are first stored in standard template library set. After all the point are checked, the 3D array is updated. In the next step only the unknowns are taken into account. In this way, the repeated processes are prevented. In the second stage of the algorithm, the free voxels are determined and the array is updated one more time. The process of finding the free voxels a little bit more complicated than finding occupied voxels. First, the odometry information of the robot is required. The camera pose is obtained from odometry info. To do this, the transformation must be applied between odometry and camera. This process is repeated at each step. After the camera pose is obtained, the voxels in free space are found. Secondly, the artificial lines are created between the robot's pose and the obstacles. The points on the line are advanced step by step. The voxels that the points belong to are found. Lastly, the array is updated as free up to the point where occupied voxel is found.

3.2. Creating Semantic Map

Within the scope of this project, semantic map was created mainly to provide detailed information for search and rescue teams in indoor environments after disasters. In addition, it was planned to assign semantic classes to the nodes of the topological map and to use the semantic map during navigation. Thus, this information can be used while planning the path. Point cloud data obtained by the robot's RGB-D camera was used to create a semantic map of the environment. Point-based deep learning architectures was used to decide whether each point in this point data belongs to the floor, wall, inclined or straight ramp classes. In their study, Turgut and Kaleci [10] collected point cloud data from the Gazebo environment given in Figure 2 and created a dataset called ESOGU RAMPS, which contains points belonging to floor, wall, inclined or flat ramp classes. Later, PointNet, PointNet ++, DGCNN, and PointCNN point-based deep learning architectures were employed to train the dataset. PointNet architecture evaluates points independently and individually. Local features are extracted for points using multilayer networks. Then, these local features are summarized with the maximum pooling method and global features are obtained. Finally, semantic class is decided by combining global and local features. PointNet ++ architecture uses local area to extract the properties of points. This local area is always created according to the x, y and z coordinates of the point and the local features of a point are decided by using all points in this region. This local area is expanding in each layer. DGCNN architecture also extracts the

properties of the point by using local regions like PointNet ++. However, in this architecture, local regions are created with K neighbors for each point. Also, the feature space is considered to decide on these K neighbors after the first layer. Unlike other architectures, PointCNN architecture evaluates points together with their neighbors, not individually, in the feature extraction phase. The results obtained with these architectures are given in Figure 2. In the figure, red, yellow, blue and pink show walls, floors, sloped and straight ramps, respectively. White ellipses are used to draw attention to the points that are classified incorrectly. When Turgut and Kaleci [10] evaluated the numerical and visual results together, they concluded that the DGCNN architecture outperformed other architectures.

It was decided to use DGCNN architecture within the scope of this project. Using the model trained for this architecture, the semantic class of each point in the point cloud data that the robot instantly receives will be decided. Then these points were added to the semantic map.



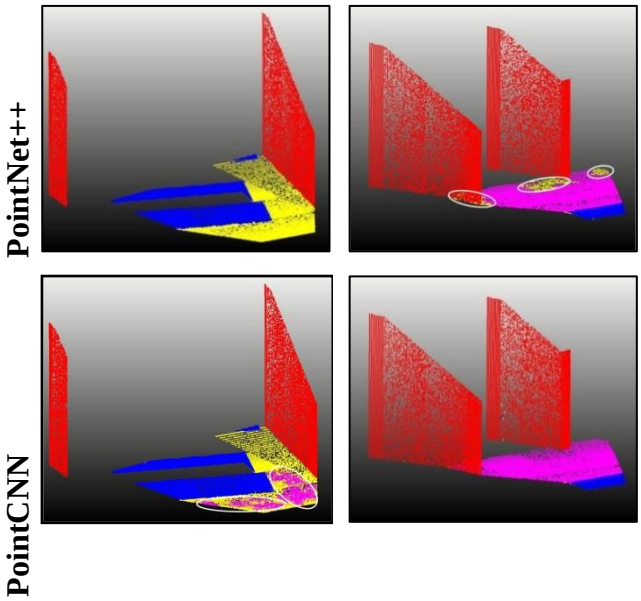


Figure 2. *Semantic classification result [10]*

Within the scope of this project, semantic map was created mainly to provide detailed information for search and rescue teams in indoor environments after disasters [17]. Firstly, point cloud data was obtained by the robot's RGB-D camera that was used to create a semantic map of the environment. Point-based deep learning architectures was used to decide whether each point in this point data belongs to the floor, wall, inclined or straight ramp classes. There are several kind of point-based deep learning approaches, in this scope of this project DGCNN architecture that is more suitable for search and rescuae areas are used. Using the model trained for this architecture, the semantic class of each point in the point cloud data that the robot instantly receives will be decided. Then these points were added to the semantic map and the map will grow step by step. In Figure 3, two adjacent scenes were merged and they were given different colors.

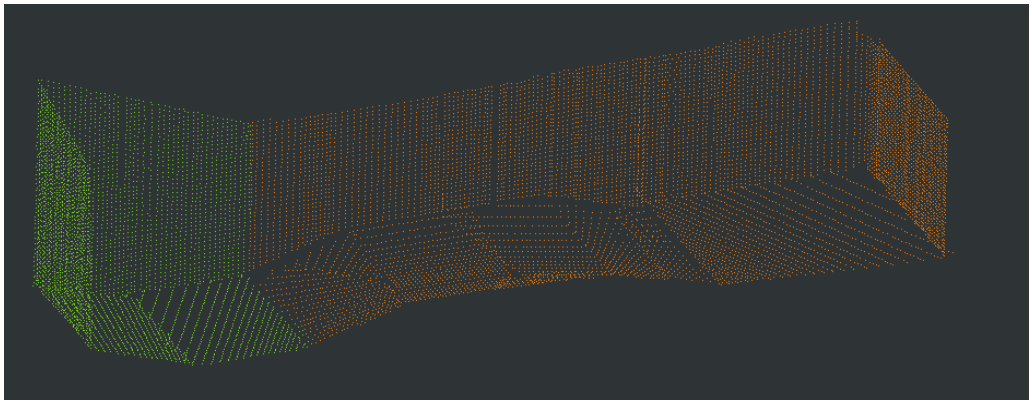


Figure 3. *Semantic map*

In the semantic map, after using DGCNN architecture, different classes were obtained. They were also labeled by giving different colors. Walls, terrains, inclined ramps and straight ramps are assigned to red, yellow, blue and pink, respectively. Figure 4 illustrates that process.

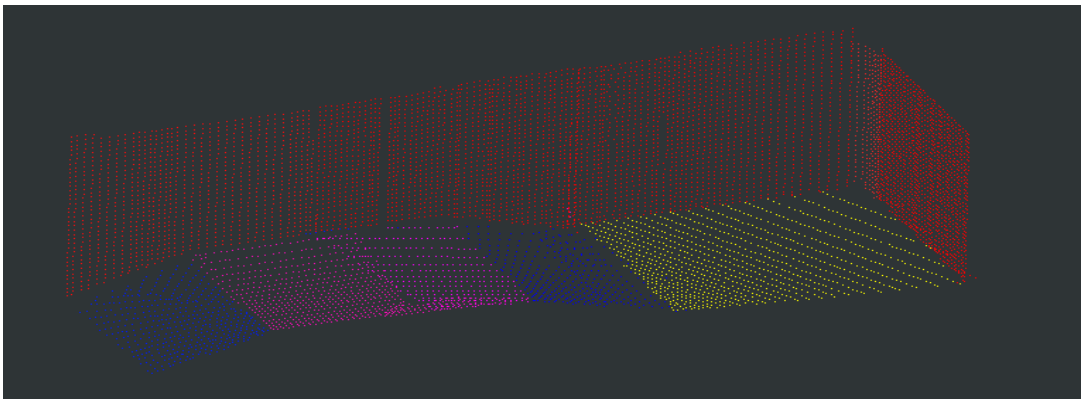


Figure 4. *Clustering planes*

In addition assigned semantic classes are used for generating the nodes of the topological map and to use the semantic map during navigation. Thus, this information can be used while planning the path. In order to achive this process each plane should be determined. In this way terrain will be used to generate the nodes on it also each wall plane helps to determine edge of the minimum spanning tree algorithm. In this way path can be generated to navigate on the map properly. In order to achive this process RANSAC (Random Sample at Consensus) algorithm is used to segment each planes. RANSAC extracts the mathematical equations from planes. However, different planes can be modeled as same mathematical equation. Take into account this problem, Region Growing method is used to separate this planes. Figure 5 depicts that segmented planes with RANSAC.

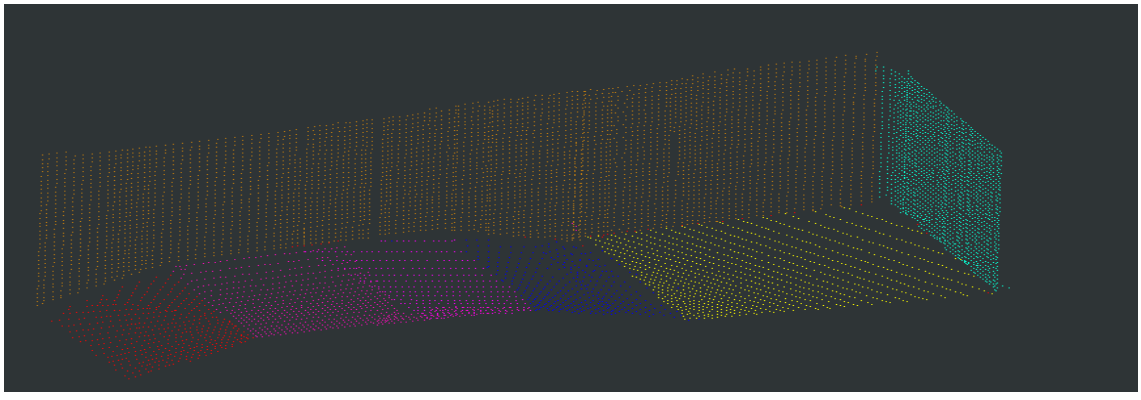


Figure 5. *Segmenting planes*

Storing each segment in the memory is an issue that consumes more memory. In order reduce to memory usage, bounding boxes are stored in the memory. In this way, same process can be performed with less memory usage. Figure 6 illustrates that incremental growing of the map with bounding boxes.

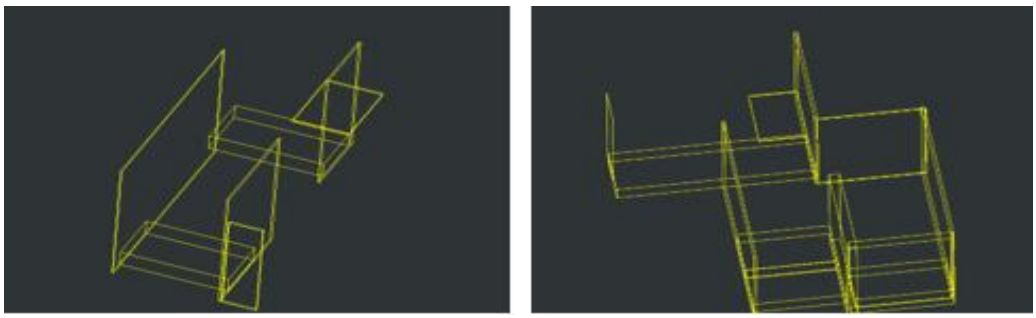


Figure 6. *Bounding box*

3.3. Creating Topological Map

Topological map is a mapping method, that represent the environment with nodes where they place at critical regions and the edges connecting these nodes. Nodes in the topological maps are created to derive a path plan of the environment. By generating nodes in required locations and connecting them by considering the lowest edge weights, efficient path plans can be generated from topological maps. Thus, the required amount of memory is significantly decreasing with the use of more suitable structures for creation of topological map [18]. An example topological map is shown in Figure 7.

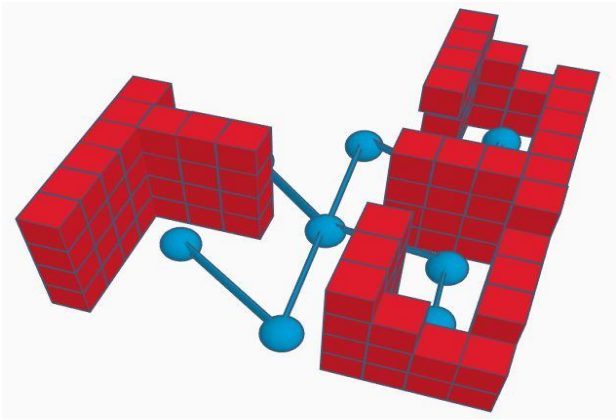


Figure 7. *Topological map nodes*

In this study, different from the studies in the literature, the information produced by the semantic map was used to generate the topological map. In this project, by obtaining the semantic map, we derived the bounding boxes of wall, ramp, and terrain segments in each data. Then, the nodes are created considering the type of the planes. Nodes are located on the middle of the straight ramps and terrains. In contrast of the straight ramps and terrain, nodes are located on the bottom points of inclined ramps. In this way, sliding is prevented when the robot stops on the node for generating a new path plan. During the generation of a new node on the terrain segments, euclidean distances between previously generated nodes are calculated. In this way, as minimum as possible number of nodes are generated. At the same time with generating nodes, we also created the environment by combining and visualizing the bounding boxes by using Point Cloud Library (PCL).

After obtaining the nodes, we pass to the stage of connecting the nodes in the topological map. In order to navigate the environment autonomously, a path plan must be generated. However, considering the locations of the nodes in the environment, it is clear that it is not possible to create a path plan by directly connecting the nodes.

We used the minimum spanning tree method to connect the nodes. The minimum spanning tree is a subset of the interconnected edges that are connecting all the nodes in the environment with the minimum edge weight. In this method, an adjacency matrix is created, then the closest nodes in this matrix are connected and the connection having the lowest cost from the starting node to the target node is created. There is an issue of connecting the nodes that an edge may be intersect with a wall as shown in Figure 8.

In order to obtain collision-free minimum spanning tree, all edges must be checked with the process mentioned above. If an edge does not intersect any wall in the known environment, the nodes are connected. Otherwise, the weight between these nodes is updated with a huge number and the process of generating minimum spanning tree is repeated until we obtain collision-free minimum spanning tree.

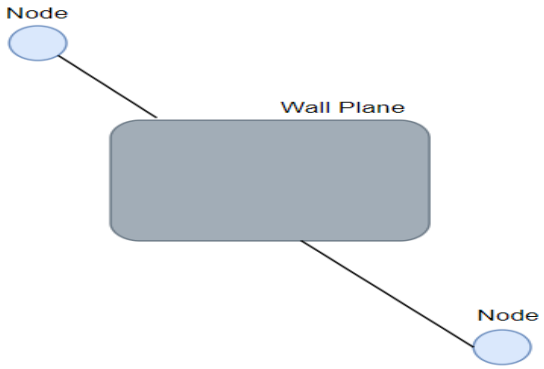


Figure 8. *Wall edge intersection problem*

In order to solve this problem, firstly, parametric equation of the line which is between the nodes illustrated in Figure 9 is calculated using following equations.

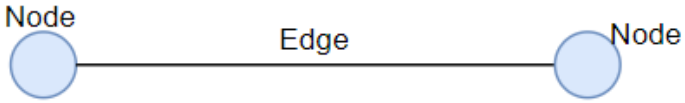


Figure 9. *Connection of two nodes*

$$x = x_0 + (x_1 - x_0)t \quad (1)$$

$$y = y_0 + (y_1 - y_0)t \quad (2)$$

$$z = z_0 + (z_1 - z_0)t \quad (3)$$

After that, plane equation is obtained using RANSAC (Random Sample at Consensus) from the wall plane is illustrated in Figure 10.

Figure 10. *Wall plane*

The related plane equation of the wall in Figure 10 is given in equation 4.

$$Ax + By + Cz + D = 0 \tag{4}$$

Finally, intersection coordinates are obtained by following equations.

$$x_c = x_s - \frac{\mathrm{A}(Ax_s + \mathrm{B}y_s + \mathrm{C}z_s + \mathrm{D})}{A^2 + B^2 + C^2} \tag{5}$$

$$y_c = y_s - \frac{\mathrm{B}(Ax_s + \mathrm{B}y_s + \mathrm{C}z_s + \mathrm{D})}{A^2 + B^2 + C^2} \tag{6}$$

$$z_c = z_s - \frac{\mathrm{C}(Ax_s + \mathrm{B}y_s + \mathrm{C}z_s + \mathrm{D})}{A^2 + B^2 + C^2} \tag{7}$$

3.4. Preparing the Path Plan

After we build the topological map, we consider the shortest path problem to generate the robot's path plan. At this stage, Dijkstra's algorithm, which is one of the most common algorithms used in the solution of the shortest path problem, is used in this study to create the lowest cost path plan. Dijkstra's algorithm is using the topological map to calculate the path plan. First, the robot finds the closest node to itself in the topological map. This node can be called the starting node. Similarly, the closest node to the target is decided. The cost of movement of the robot from the starting node to each node connected to this node is calculated and the lowest cost node is marked. After the first marked node, the cost will be calculated similarly, the following nodes are marked so the lowest cost path plan is created.

3.5. Human Detection in the Map

Identifying the location of victims in disaster environments is very important for search and rescue activities in order to save victim’s life. It may be appropriate to utilize deep learning architectures to accurately and quickly identify the victim in complex

environments such as disaster environments. In this context, it was used the third version of YOLO (You Only Look Once) architecture, which detects objects by passing the image through an artificial neural network at a time. Compared to the existing object detection algorithms in previous studies, YOLO can detect objects quickly as well as successfully, and can recognize objects of different scales and sizes. When the victim has been identified with YOLO, the location of the victim in the environment was also determined. The Figure 11 illustrates the environment with a victim.

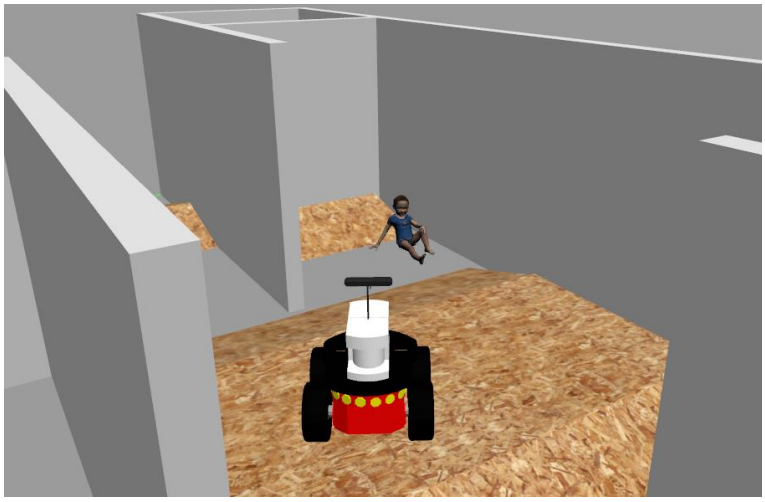


Figure 11. *Gazebo environment with a victim*

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

Before testing the methods developed within the scope of the project using real robots, observing how these methods work in simulation environments such as GAZEBO [13] and whether they produce the desired results are frequently used in the field of robotics. For this purpose, a 6x4 meter environment, similar to the one used in RoboCup competitions (Figure 12(a)), containing objects such as straight ramps, inclined ramps, and wall planes, which are determined by NIST, was created to be used in real robot tests of the project. The same environment was created in the Gazebo simulation environment using the hector_nist_arenas_gazebo [14] package that is shown in Figure 12(b). In addition, the first floor of the 52x15 meter ESOGU Electrical and Electronics Engineering Laboratory building was modeled in the GAZEBO environment. (Figure 13).

(a) (b)

Figure 12. *Real test environment. (a) Gazebo test environment. (b)*

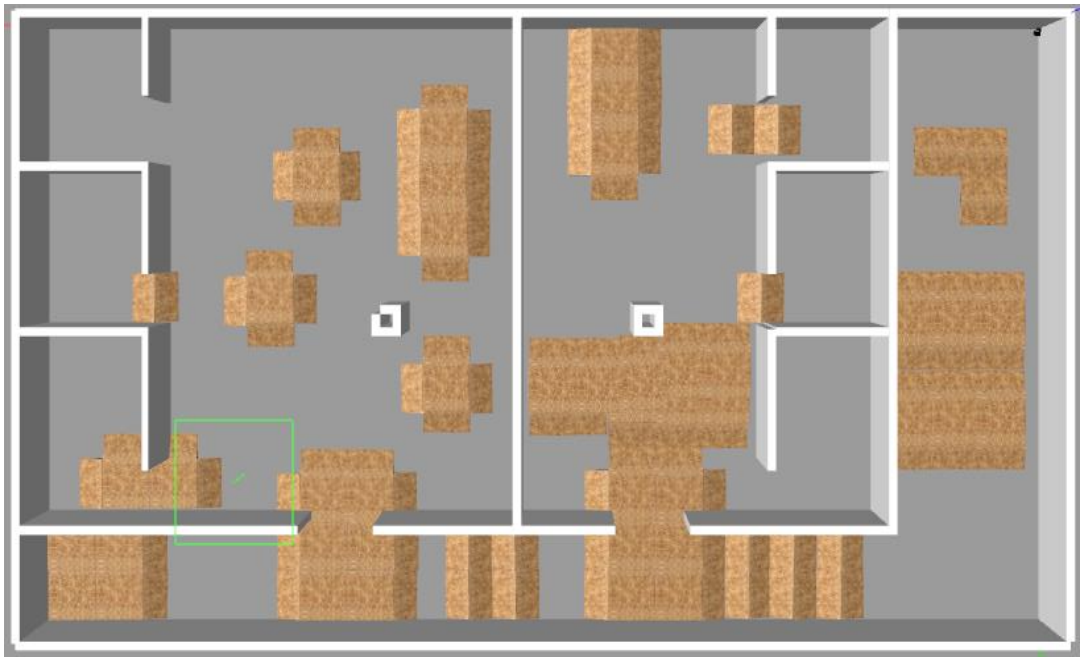


Figure 13. *Esogu electrical and electronics engineering laboratory building gazebo model*

A Pioneer 3-AT mobile robot [15] with Asus Xtion Pro RGB-D camera was launched in Gazebo simulation environment. The Robot Operating System (ROS) [16] was used to receive data from the sensors on the robot and to send data to the robot's motors. At this stage of the project, color visual perception and depth information were obtained with the RGB-D camera.

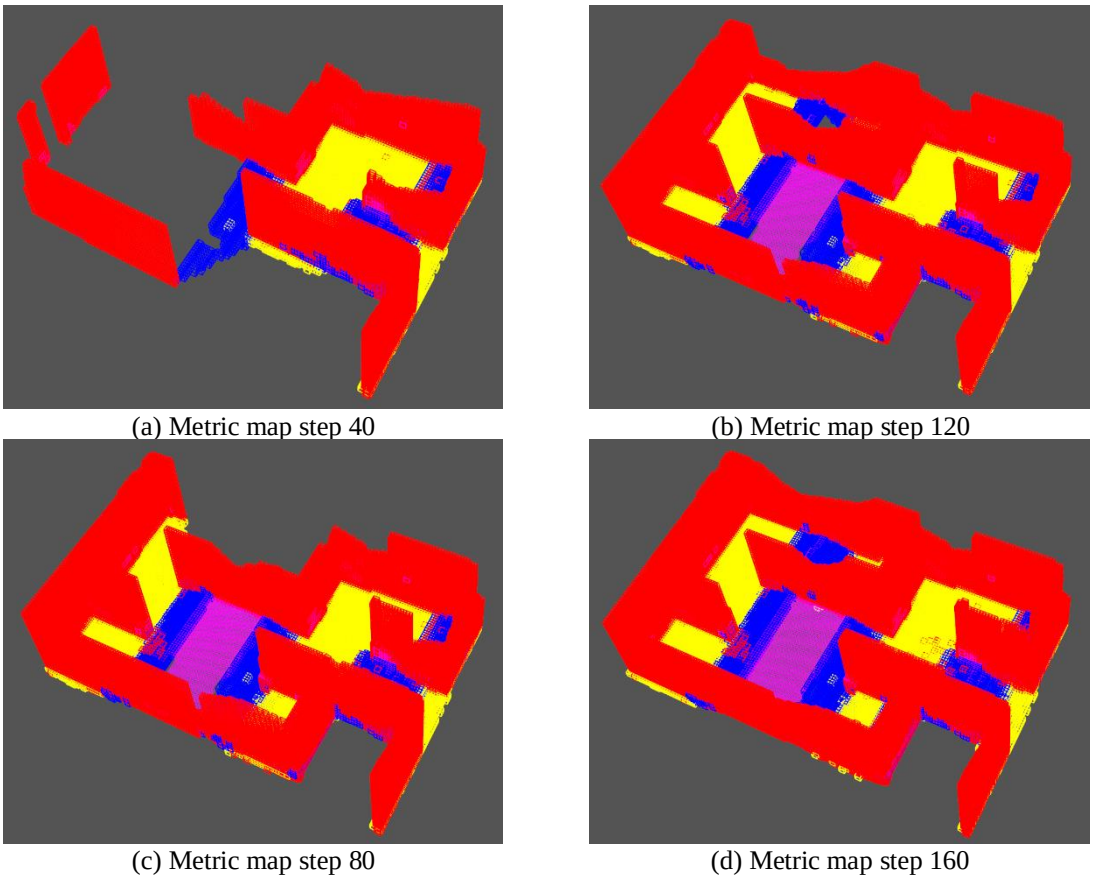
4.2. Metric Map Results

In the scope of the project, metric map was created by using RTAB-Map at first. Then, the metric map was created with out own mapping method.

4.2.1. Metric Map by Using RTAB-Map

The results obtained are shown in Figure 14 [17]. These are the results of the study conducted by Kocaoglu et al. In the study, the metric map is created for 160 sequential steps. The inclined ramps, straight ramp, walls, and terrain are colored blue, pink, red, and yellow, respectively. The information of each plane is obtained from semantic map.

In Figure 14(a), the step 40, in Figure 14(b), the step 80, in Figure 14(c), step 120, and lastly in Figure 14(d), step 160 are illustrated. Here, the results are obtained by using RTAB-Map. The free voxels are not shown since it hides the ramps, and terrains.



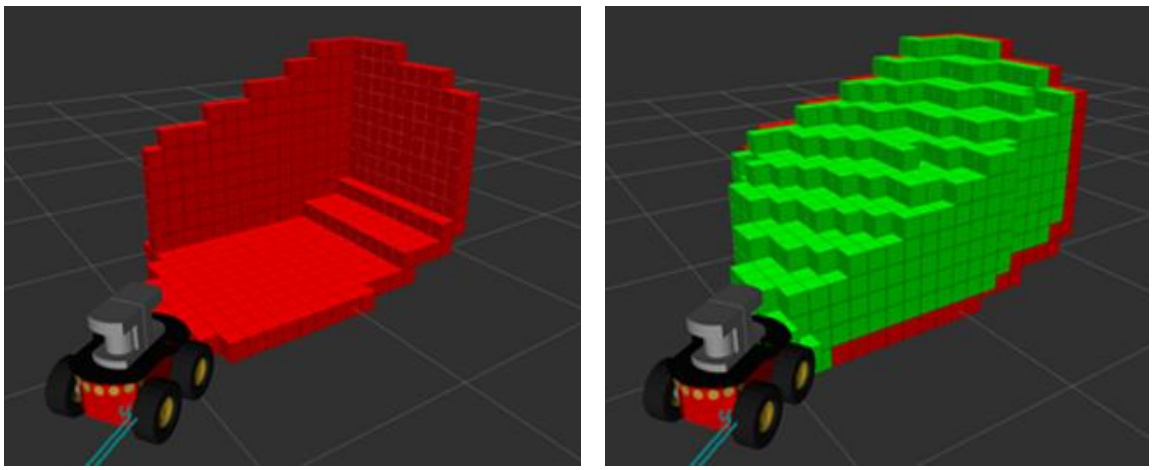
(a) Metric map step 40 (b) Metric map step 120

(c) Metric map step 80 (d) Metric map step 160

Figure 14. *3D metric map, step 120 and 160*

4.2.2. Metric Map by Using Our Own Mapping Method

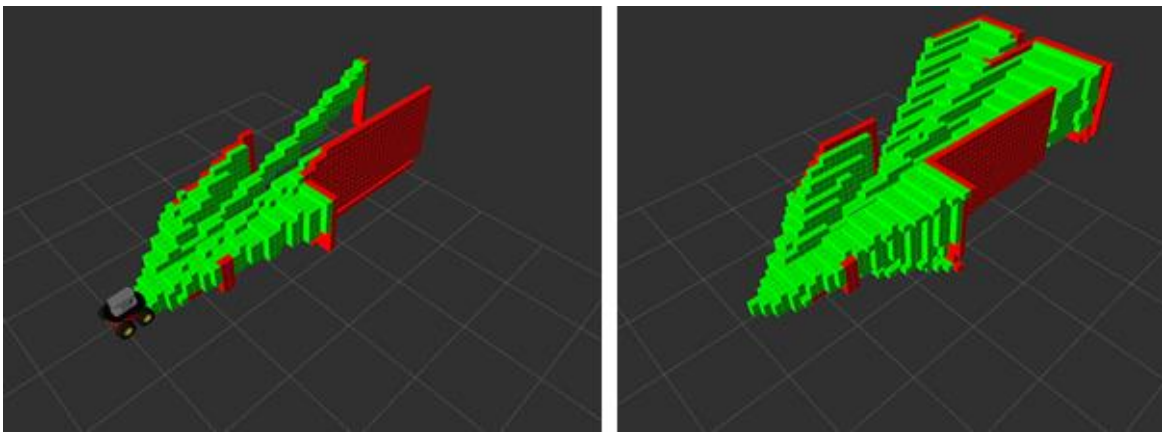
In Figure 15(a) only occupied voxels are shown since free voxels hides the occupied voxels that correspond to the ramps and walls . However, the shape of the ramp is not so clear due to resolution we determined. As the resolution increases, the ramp becomes more visible. The metric map obtained from that scene is shown in Figure 15(b).



(a) (b)

Figure 15. *Metric map with occupied cells. (a) Metric map with occupied and free cells. (b)*

In the Figure 16, the step by step creation of metric map is illustrated. The free and occupied voxels are extracted from predefined array, and then they are visualized by using Rviz.



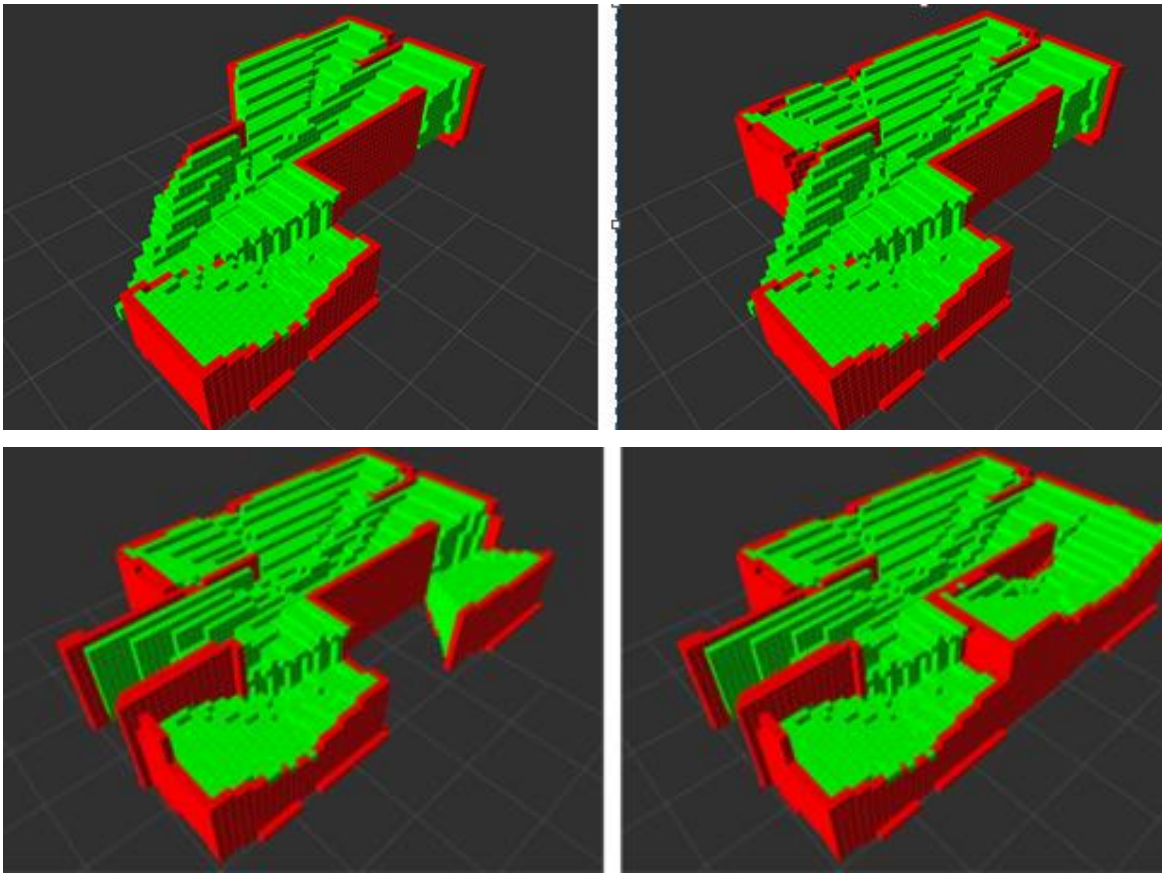


Figure 16. *Metric map creation process*

4.3. Semantic Map Results

In order to generate semantic map, DGCNN architecture was utilized after preprocessing steps. Gathered point cloud data divided into sub blocks as a preprocessing step for taking into account local features. The blocks are divided into 1 m^2 size considering the xy plane. Point-based deep learning architectures require a fixed number of points in each block. In experiments, we selected the fixed number as 4096. However, in some blocks, the number of points are greater or lower than 4096. In order to handle this problem random downsampling or undersampling was performed. The architecture is fed with the data which consists of x, y, z and normalized x, y, z coordinates regardless of color information. Finally, DGCNN point-based deep learning architecture with default parameters for segmentation was utilized. After each scene is semantically classied the entire semantic map was obtained as shown in Figure 17.

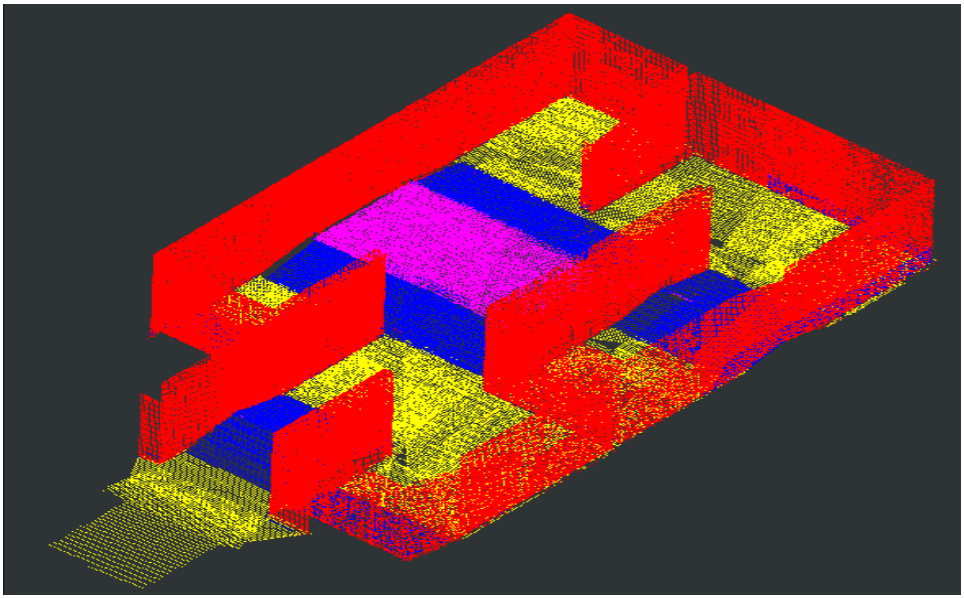


Figure 17. *Semantic map of the environment*

4.4. Topological Map Results

The members of every single segment existing in the environment are taken into account separately during the node generation process. While generating the nodes on the straight ramp, because of its large size, we divided it into two parts and we determined the midpoints of these parts. After that, we created the nodes at these specified points.

In the navigation, robot may stop and update the path plan at any node it visits. Therefore, it is necessary that the ground on which the robot stands should be flat and the robot should not lose it's position by sliding. For this reason, nodes are located at the bottom points of inclined ramps. In this way, the robot can be prevented from sliding due to the slope of ramps and thus the position of the robot may remain the same.

Besides the generating nodes at the locations where they are most needed, it is also important to have the minimum number of nodes as possible as to cover the entire environment. Therefore, we considered the distances between the nodes on inclined ramps and terrain while generating the nodes. If the nodes are created in a similar manner to the previous segments, there will be many nodes on terrains that are unnecessary and slow down the generation of robot path plan. That's why we considered the specified points for the nodes of inclined ramps to apply a node generation algorithm by using the distance between the centers of inclined ramps and terrains.

By applying the mentioned methods, we generated nodes of the topological map for each scene which represents the environment sufficiently and efficiently as seen in the Figure 18.

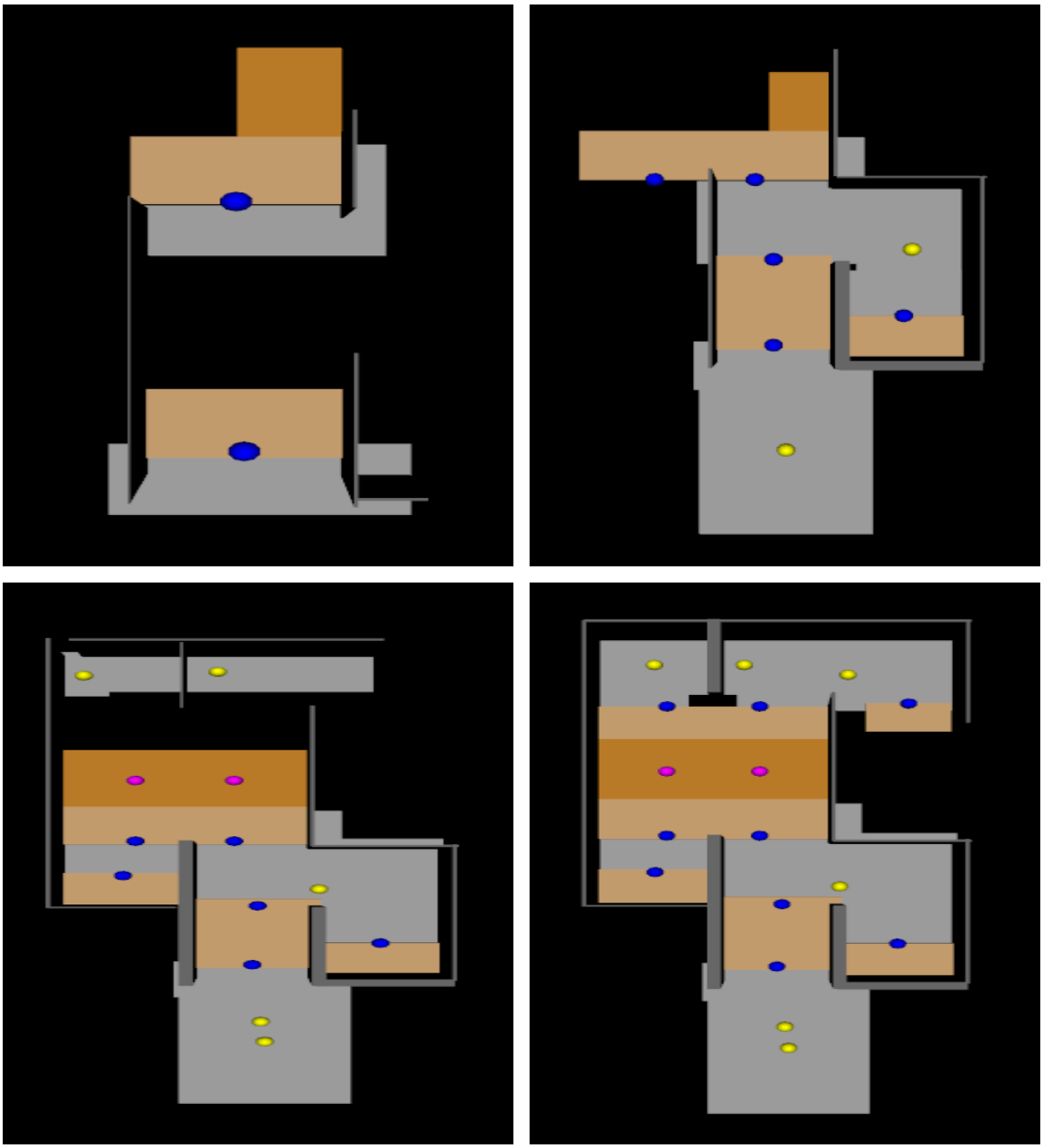


Figure 18. *Node generation steps*

For every step that the information is provided by the semantic map, the point cloud data are processed and the topological map is obtained step by step. The nodes are visualized with yellow, blue, and pink for the terrain, inclined ramp and straight ramp segments respectively. In this way, it is seen on which segments the nodes are generated. At the first

scene, the robot is seen only the ramps, so two nodes on the inclined ramps are generated. At the scenes following the first scene, the produced point cloud data are processed and the remaining parts of the node generation section of topological map is created.

The wall, ramp and terrain segments are visualized by using the bounding boxes that are provided from the semantic map.

In order to obtain the topological map, after the creation of nodes, we connected the closest nodes according to the order of creation of the scenes by using the coordinates of the nodes we produced while generating them. After this stage is completed, it is necessary to check whether there is a wall between the nodes that are connected.

When the robot navigating autonomously, this intersected edges causes to crash. In order to take into account this problem, line – plane intersection was calculated and problematic edges connected to proper nodes.

Mentioned processes in the proposed method section are integrated into minimum spanning tree algorithm in order to avoid improper edge connection. When the edge will be created, firstly it is needed to check is intersect with the wall, if it is not intersecting any wall then connect two nodes. If there is a wall between 2 connected nodes, it is necessary to rerun the minimum spanning tree function by assigning more weight to the value of those 2 nodes in the neighborhood matrix.

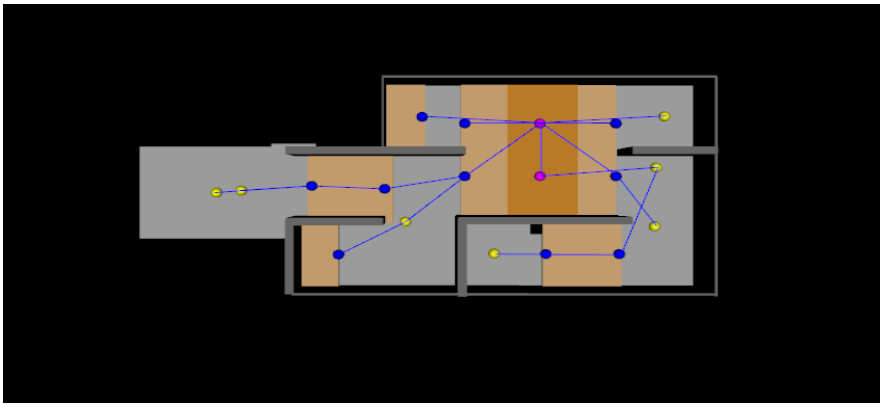


Figure 19. *Topological map of the environment*

After applying the MST method, the topological map of the environment consisting of the connected nodes is created as seen in Figure 19. After applying the node generation and MST algorithms, the topological map of the entire environment is obtained. The adjacency matrix that is produced in this part is provided to Dijkstra's shortest path algorithm to derive the shortest path for the navigation.

4.5. Navigation Results

In the environment where the robot is moving, there are ramps so we need to give the speed low to provide stable movement of the robot or utilize the semantic map in order to decrease the speed when the robot is to climb or descend the ramp. At this part, we gave the robot a low speed that is 0.2 m/s so that we provided the stable movement of the robot. It is also required to give the robot rotating speed when the destination is not in the same direction as the robot's direction. In order to determine how much and to which direction the robot should move, we first created a line in the direction of the robot which is in the x-direction. As the robot moved, the line also moved in the direction of the robot. Then we compared the 2 lines with their angles. The first line is the line we created at the beginning of the navigation, and the second line was created from the robot's position to the determined destination. By doing so we have obtained 2 lines whose angles must be the same in order to make the robot move in the direction of the destination. Different turning speeds and turning directions were given to the robot at the required situation by which the robot became more stable and saved time. For example, if we presume that the robot must turn 90 degrees to line up with the goal, the robot may turn 90 degrees which is logical or the robot may also turn 270 degrees to match up the 2 lines which provide the requirement but illogical. At this point, we gave conditions to make it turn 90 degrees but not 270 degrees. The other condition given to the robot was the turning speed. We adjusted the turning speed of the robot according to the angle difference between the 2 lines. If the angle is wider, we set the turning speed to higher. As the angle difference gets smaller the turning speed was also set to lower speeds.

An example of robot's navigation is shown in Figure 20 and. In the figure, the robot is at location 6 at first, and it is expected to go location 9. The nodes which should be followed by the robot 6, 7, 8, and finally 9. This path plan is obtained by using Dijkstra.

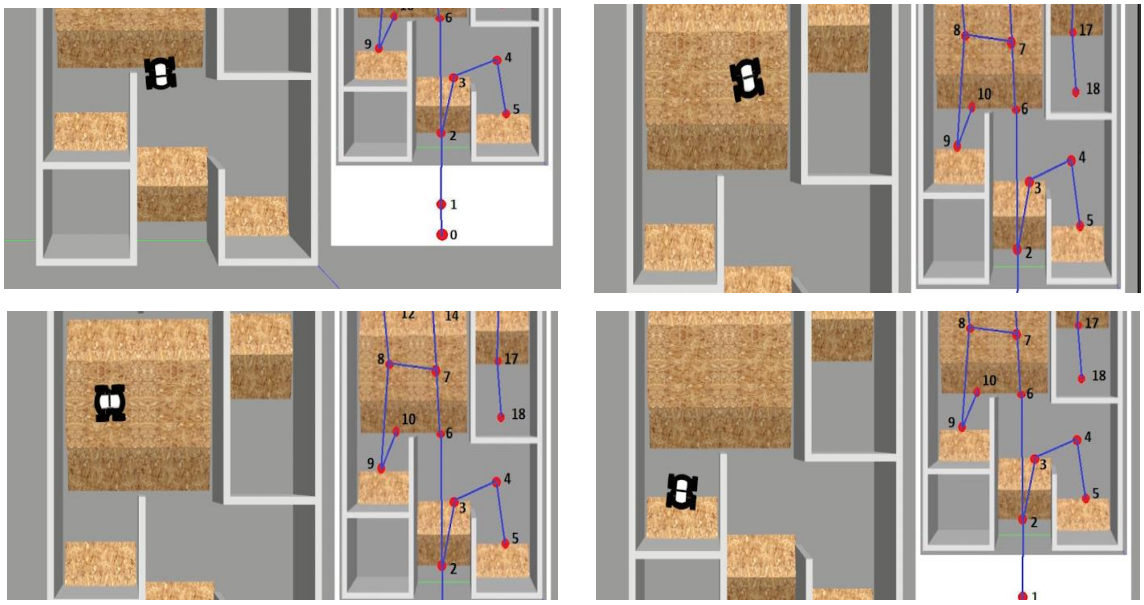


Figure 20. *Navigation process*

4.6. Victim Detection

In order to detect victim in the disaster area, third version of pre-trained YOLO architecture is utilized. Results are obtained as depicted in Figure 21.

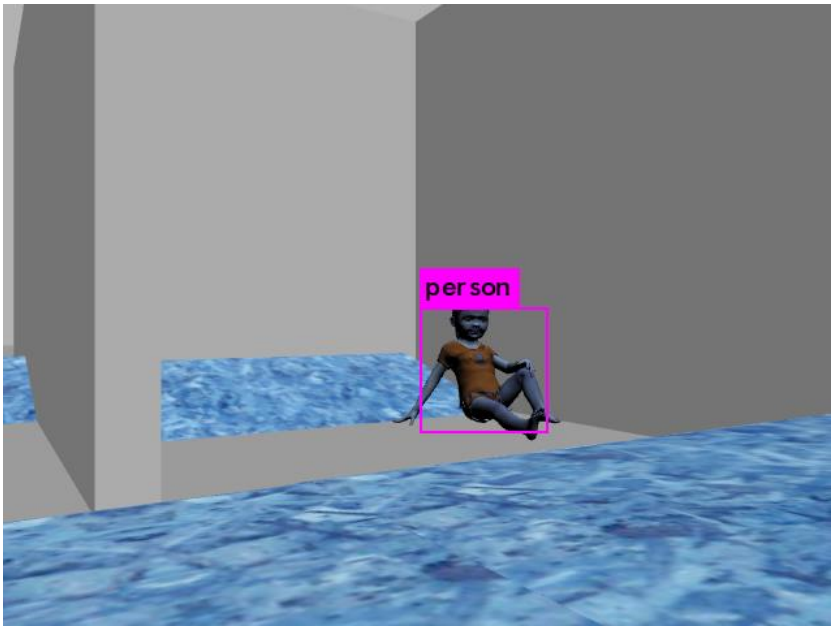


Figure 21. *Victim detection with yolo*

5. Tools, Hardwares and Softwares

- Python
- C++
- GAZEBO
- UBUNTU
- ROS
- RTAB-Map
- PCL
- Cloud Compare
- Tensorflow
- Pioneer P3-AT Mobile Robot
- ASUS XTION RGB-D Camera

6. Project Management

Table 1. *Work packages*

No	Work Packages	Responsible Team Member(s)	Time Intervals	Success Criteria and Contribution to the Success of the Project
1	Preparing the Environment and the Robot for Simulation Tests	Muhammed KOCAOĞLU Muhammed Ali UZUN	1 November – 15 November	When the robot is started in GAZEBO, obtaining point cloud data from RGB-D camera and position information from step counter from the simulation environment will be deemed successful. Its contribution to our project will enable the methods developed before the real environment tests to be experienced in the simulation environment.
2	Creating the Metric Map	Muhammed KOCAOĞLU	15 November – 1 January	The fact that the map shows the ramps and walls in their correct positions and can do this in environments with dimensions of 6x4 and 52x15 meters will be considered as success criteria.When this work package is

				successful, we will have a map where we can create the topological map in our project.
3	Topological Map	Muhammed Ali UZUN	1 November – 1 January	Expressing the entire environment with as few nodes as possible will be considered as a success criterion. Nodes in the map will be produced.
4	Creating Semantic Map	Yunus Emre IŞIKDEMİR	1 January – 15 February	Expressing the entire environment as a point cloud and determining the semantic classes of the points in this point cloud. Its contribution to the project is to obtain a map that can be used by search and rescue teams. It can also be used to create topological map.
5	Preparing a Path Plan for the Determined Target Point / Points of the Robot	Muhammed Ali UZUN Yunus Emre IŞIKDEMİR	15 February – 1 April	Calculation of the shortest path for the robot to reach the given frontier or frontiers will be considered as a success criterion. With this package, the optimal route plan will be created.
6	Navigation for the Implementation of the Path Plan.	Muhammed KOCAOĞLU	15 February – 1 April	The robot's reaching the waypoints and the frontier without tipping will be counted as success criteria. This step will avoid unwanted interruption of the exploration process in the project.
7	Human Detection in the Map	Muhammed KOCAOĞLU Yunus Emre IŞIKDEMİR Muhammed Ali UZUN	1 April – 1 May	The robot's finding all the victims in the environment will be considered as success criteria. Its contribution to the project will require less time for search and rescue teams to reach the victims as they know their location.

6.1. Gannt Chart

The gannt chart is created to plan the process of our studies. It is illustared in Figure 22.

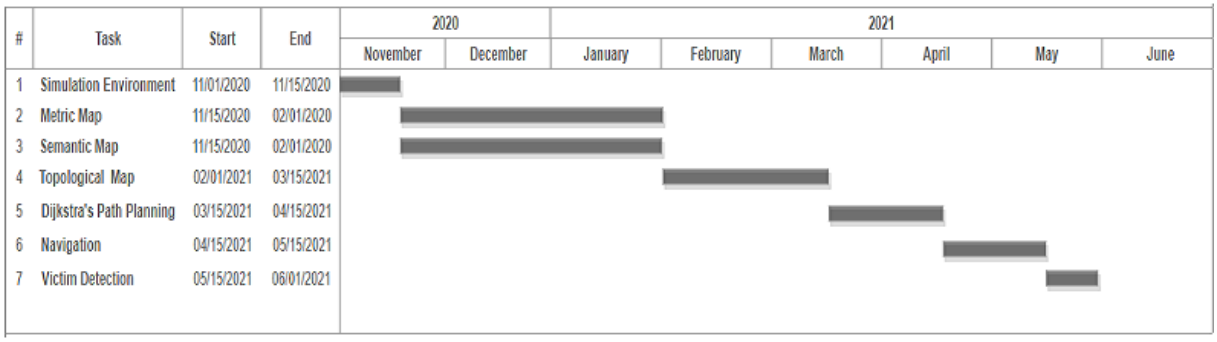


Figure 22. *Gannt chart*

7. CONCLUSIONS

Within the scope of this project, a 3D map of the environment was created in a simulation environment using the Pioneer 3-AT robot in the indoor environment after the disaster. The data collected as point clouds from the environment were classified as wall, floor, straight and inclined ramps, and a semantic map was drawn. Then the metric map was created as 3D array with predetermined size and resolution. Topological map was obtained by using segment information produced in semantic map. After that, by applying MST algoritm to the derived nodes, the connections between nodes were provided. The dijkstra algoritm was used to provide path planning to the robot. With the path planning derived from dijksta, the navigation of the robot was realized.

8. REFERENCES

[1] H. Kitano and S. Tadokoro, "A grand challenge for multiagent and intelligent systems," *AI Mag.*, vol. 22, pp. 39–52, 2001.

[2] R. Sheh, S. Schwertfeger, and A. Visser, "16 years of robocup rescue," *KI-Künstliche Intelligenz*, vol. 30, no. 3-4, pp. 267–277, 2016.

[3] A.J. Davison, Y.G. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," *in Proc. IFAC/EURON Symp. Intell. Auton. Vehicles*, 2004.

[4] T. Lemaire and S. Lacroix, "Vision-based SLAM: Stereo and monocular approaches," *Int. J. Computer Vision*, vol. 74, no. 3, pp. 343–364, 2006.

[5] S. Yavuz, M. Amasyalı, E. Uslu, F. Çakmak, M. Balcılar, N. Altuntaş, and S. Marangoz, "RoboCup Rescue 2016 Team Description Paper YILDIZ.", 2016.

[6] S. Kohlbrecher, “hector_mapping.” Internet: http://wiki.ros.org/hector_mapping, 2013. [Nov. 14, 2020].

[7] “wiki.ros-Octomap-Kinetic.” Internet: http://wiki.ros.org/octomap, [Nov. 25, 2020].

[8] “RTABMap.” Internet: http://wiki.ros.org/rtabmap_ros, [Nov. 28, 2020].

[9] T. De Silva, B. Cooray, J. Chinthaka, P. Kumara, S. Sooriyaarachchi, “Comparative Analysis of Octomap and RTABMap for Multi-robot Disaster Site Mapping.”, 2018.

[10] K. Turgut and B. Kaleci, "Comparison of Deep Learning Techniques for Semantic Classification of Ramps in Search and Rescue Arenas," *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Istanbul, Turkey, pp. 1-6, 2020.

[11] Y. Denga, Y. Chenb, Y. Zhanga, S. Mahadevanc, S, "Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment.", 2012.

[12] J. Redmon, A. Farhadi, “YOLO9000: Better, faster, stronger.”, 2017.

[13] “Gazebo.” Internet: http://gazebosim.org/tutorials, [Dec. 12, 2020].

[14] “Hector_nist_arenas_gazebo.” Internet: http://wiki.ros.org/hector_nist_arenas_gazebo, [Dec. 14, 2020].

[15] “p3at_tutorials.” Internet: https://github.com/Gastd/p3at_tutorial, [Dec. 18, 2020].

[16] “ROS.” Internet: https://www.ros.org/, [Feb. 15, 2021].

[17] M. KOCAOĞLU, Y. E. IŞIKDEMİR, M. A. UZUN, K. TURGUT, M. O. TAS, and B. KALECİ, “A Mobile Robot Application for Constructing Semantic and Metric Maps of Search and Rescue Arenas with Point-Based Deep Learning.” *Journal of Scientific, Technology and Engineering Research*.”

[18] S. Thrun, “Learning metric-topological maps for indoor mobile robot navigation.”, 1998.