3D Trajectory Planning Method for UAVs using Probabilistic Roadmaps and A* Algorithm

Vishal Gattani[1] and Antranik Anto Malkastian[2]

***Abstract*— This report includes the implementation of 3D Probabilistic Roadmaps (PRM) for the purpose of multiple trajectory planning for a swarm of Unmanned Aerial Vehicles (UAVs).**

Keywords:* UAV, 3D probabilistic roadmaps, trajectory planning, multi-robot systems, A

I. Introduction

For autonomous UAVs to accomplish a set of tasks, it is necessary to establish safe trajectories that allow the UAV to maneuver through the environment, avoiding static as well as dynamic obstacles. Therefore, trajectory planning becomes crucial for such systems to undertake complicated mission statements provided by a user. Once the user has establish what the task should be performed by the autonomous system, the generation of safe and efficient trajectories considering obstacle avoidance and collision avoidance methods need to be developed.

This work presents the implementation procedure followed in [1] through a different process. Similar to [1], the work presented in this report focuses on exploring a given environment in 3-Dimensional (3D) space and generates an occupancy map for the same. Once the occupancy map is generated, the development of a trajectory finding algorithm based on Probabilistic Roadmaps (PRM), a solution is provided for the movement of a UAV through the structured environment. The mission statement is considered along the lines of search and rescue work or urban emergencies related tasks.

This document is structured as follows. Section 2 focuses on describing the implemented work and comparing it with the work proposed in [1]. Section 3 provides the experiments results and observations. Finally, Section 4 contains the conclusions and details of further developments.

II. 3D Planning based on PRM

From [1], the general idea is to present a framework for algorithms that allow UAVs to plan trajectories within an environment from a 3D occupancy map. In order to do this, [1] creates an occupancy map of the environment using the Binvox library [2] and the Octomap library [3]. Using these libraries, an octree or an occupancy map is obtained which is used to generate the PRM of the possible paths to reach any desired location.

However, in this project, a similar structure is followed i.e., an occupancy map is generated using a 3D model of a urban structure (see Figure 1 using Trimesh [4] software.

As shown in 5, the process to obtain a 3D occupancy map involves converting the 3D model file (.obj) format to a voxelated mesh and then reconverting it into a mesh for computational purposes. This voxelated mesh is used as an input to the exploration of the environment unlike in [1] where the process involves converting the model file into a binvox file formal following which an octree is obtained using the octomap library.

Figure 6 shows the process when the visualization is viewed from the side.

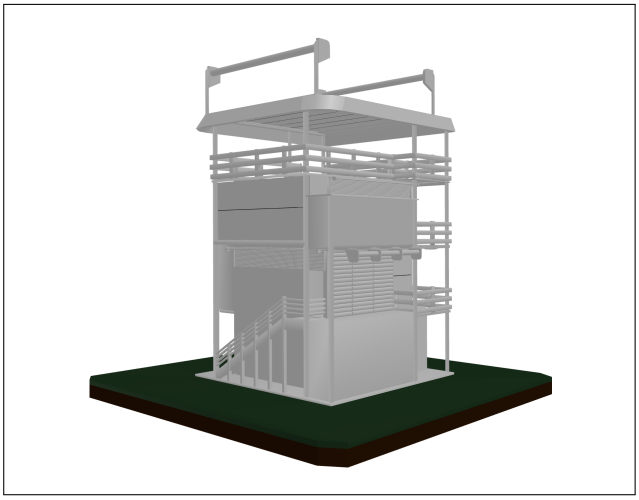

Fig. 1: Environment Model.

Once the 3D occupancy map is obtained, a probabilistic algorithm is implemented to explore the work environment to obtain shortest paths to a set of possible target coordinates. To obtain these shortest paths, the A* algorithm [5] is employed.

As the main aim is to establish an algorithm based on the probabilistic framework, this method, according to [1] aims to reduce the computational time taken to explore the large environment. In simpler terms, once the exploration is performed, the A* algorithm is used to find the shortest possible paths to the target coordinates which are also considered during the sampling using probabilistic methods. It is worth noting that it is likely that there may not be any possible solution due to incomplete sampling or down sampling of the area being considered.

The following Algorithm provides the steps for the exploration of the 3D environment.

Once the parameters have been set for the algorithm, a set of phases are implemented to allow the generation of a finite set of possible paths within the environment.

1) Generation of nodes: Nodes are generated with random

Algorithm 1 3D Probablistic Roadmap

Input: 3D Object Model
Output: 3D Probabilistic roadmap

Input: *N_KNN* Number of nearest neighbors/number of edges near to each sampled point, *MAX_EDGE_LEN* Maximum edge length, *minDist* Minimum edge length, *pitch* size of the voxel cubes, *numPoints* Number of points to be sampled, *objectFile .obj* file
mesh ← "file.obj"
points ← mesh.sampleVolume(count←numPoints)
pcd ← points.PointCloud(points)
voxels ← voxelize(mesh, pitch)
voxels ← VoxelGrid(voxels,fill)
for point in pcd **do**
  if point is inside voxel **then**
    ignore
  else if point is outside voxel **then**
    sampledPoints ← point
  end if
end for
voxelatedMesh ← voxels(centers,pitch)
for counter < sampledPoints **do**
  if inside voxel **then**
    ignore
  else if point is outside voxel **then**
    nodes.pushback(point)
  end if
end for
for i=0;i<nodes.size();i++ **do**
  for j=i;j<nodes.size();j++ **do**
    diffX ← nodes[i].x - nodes[j].x
    diffY ← nodes[i].y - nodes[j].y
    diffZ ← nodes[i].z - nodes[j].z
    euclideanDist ← sqrt(diffX^2 + diffY^2 + diffZ^2)
    if euclideanDist ≥ minDist and euclideanDist ≤ maxDist **then**
      rayOrigins ← (nodes[i].x,nodes[i].y,nodes[i].z)
      rayDirections ← (-diffX,-diffY,-diffZ)
      if rayIntersects(voxelatedMesh) **then**
        do nothing
      else if !rayIntersects(voxelatedMesh) **then**
        possibleEdge(nodes[i],nodes[j])
        edges.pushback(possibleEdge)
      end if
    end if
  end for
end for
Draw PRM

Algorithm 2 A* Algorithm

Input: startNode,goalNode
Output: path

Input: *N_KNN* Number of nearest neighbors/number of edges near to each sampled point, *MAX_EDGE_LEN* Maximum edge length, *minDist* Minimum edge length,*voxelatedMesh* voxelated Mesh, *sampledPoints* Sampled points
openSet, closedSet = dict(), dict()
openopenSet(startNode.Index) = startNode
while True **do**
  if openSet empty **then**
    break
  end if
  cid ← min(openSet,lowestCostNode)
  current ← openSet[cid]
  if current = goalNode **then**
    goalNode.parentIndex ← current.parentIndex
    goalNode.cost ← current.cost
    goalNode.setParent(current.parent)
    break
  end if
  delete openSet[cid]
  closedSet[cid] ← current
  neighbors ← expand(current,sampledPoints)
  for neighbor in neighbors **do**
    node ← Node(neighborCoords,cost(current,neighbor),cid)
    nid ← node.Index
    node.setParent(current)
    if nid in closedSet **then**
      continue
    else if nid not in openSet **then**
      openSet[nid] ← node
    else if openSet[nid].cost ≥ node.cost **then**
      openSet[nid] ← node
    end if
  end for
end while
temp ← goalNode
while temp != startNode **do**
  inode ← temp
  temp ← temp.parent
  fnode ← temp
  possibleEdge(inode,fnode)
  path.pushback(possibleEdge)
end while
Draw Path

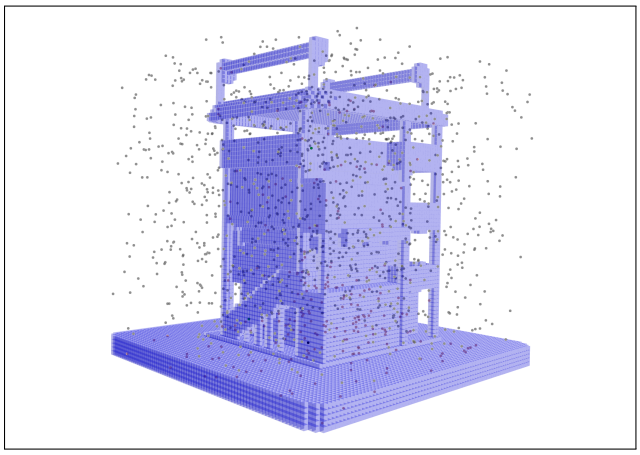

Fig. 2: Sampling of 1500 points.

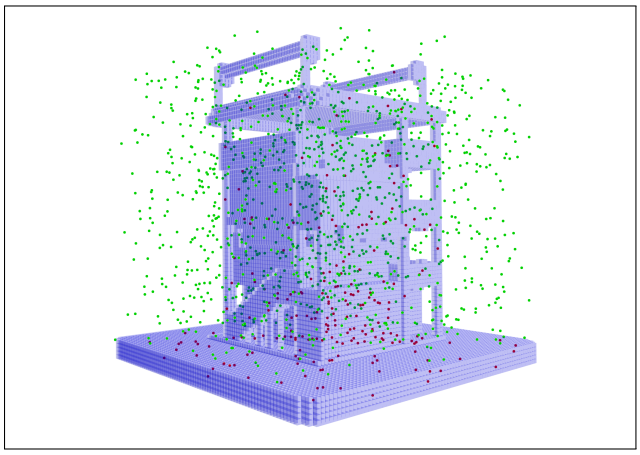

Fig. 3: Occupancy Map.

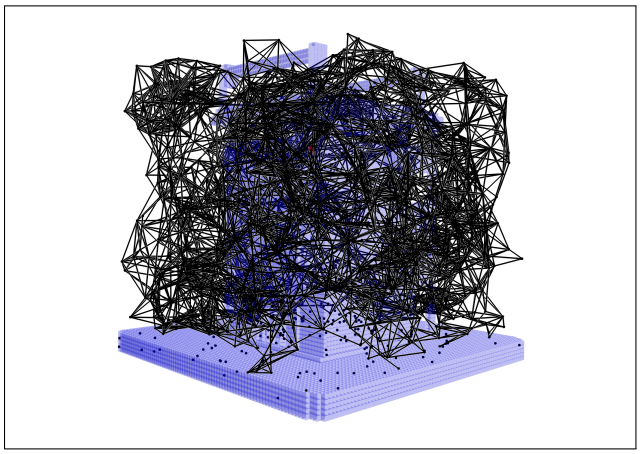

Fig. 4: 3D Probabilistic Roadmaps.

coordinate values. To do this, the software Trimesh allows to sample a volume randomly given the number of points to be sampled.

2) Once the node is generated, the next step is to look for this node inside the occupancy map, and check if it corresponds to free space. If so, it is added to the set of PRM nodes and if it does not belong to free space, it is discarded.
3) Edge generation: The next step is to establish all possible connections between the generated nodes. To achieve this, first the algorithm checks if the nodes are at a valid distance according to the established range, and then verifies if the line connects both nodes is free collision path. This process is carried out using the Trimesh software, which have functions that allow casting rays from a point to another and receiving if the ray is hitting an obstacle or not, and checking that the end of the ray is the destination and not an intermediate point. In addition, this function takes into account the direction in which the beam must be launched and from the originating point, ensuring that the edge generated is between the two nodes generated. Once the check is done, the edge is saved.

The first step is defining a set of variables required by the algorithm:

1) ***N_KNN***: Number of nearest neighbors or number of edges near to each sampled point
2) ***MAX_EDGE_LEN*** Maximum edge length
3) ***minDist*** Minimum edge length
4) ***pitch*** size of the voxel cubes
5) ***numPoints*** Number of points to be sampled
6) ***objectFile*** $.obj$ file

The last step is to establish the optimal path to join the initial position of the UAVs to the goals. This step consists of employing the A* algorithm [5] to find the optimal path once the PRM is generated. The reason for this is the passive character of the PRM algorithm, which randomly explores the environment but is not able to provide a solution.

III. Experiments and Results

The integration of A* algorithm from [5] results in determining the paths that interconnect all the possible sampled points within a certain distance threshold and an additional constraint of number of edges that connect each sampled point.

As seen in figure 2, the environment olume is sampled randomly and then the sampled points are checked against the voxelated mesh object of the work eonvironment to determine if they are within the occupied voxels or not. 3 shows the valid points which do not occupy any voxels in green color and the occupied voxels are shown in red color.

4 shows the generation of all the edges that are within the threshold distance of the neighboring points and also, each point is constrained to having only few edges for computation purposes.

Figure 8 shows the final path (in white) which is the solution of the prm from start(green) to goal(red) using 2 algorithm.

IV. Conclusion

This work presents a method of 3D trajectory planning for a UAVs on PRM. The obtained results from simulation

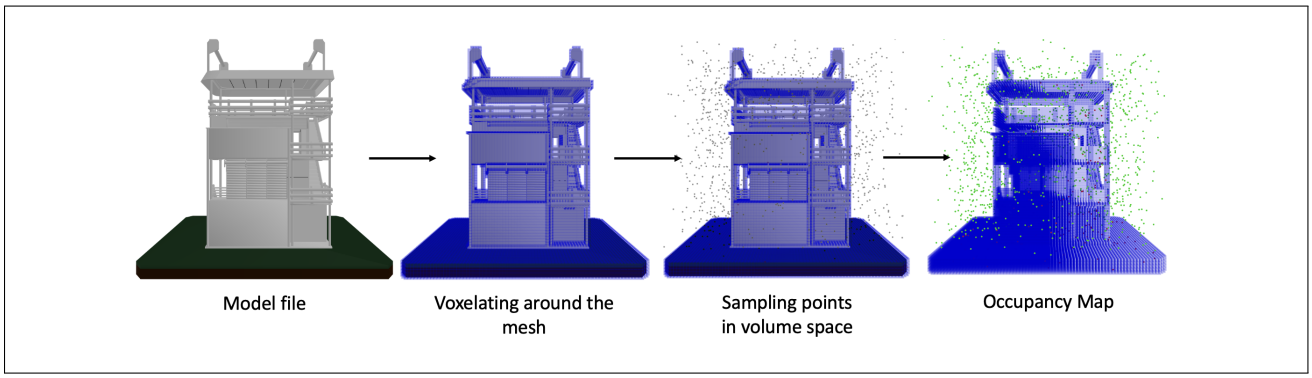

Fig. 5: The process to obtaining an Occupancy Map from a object file.

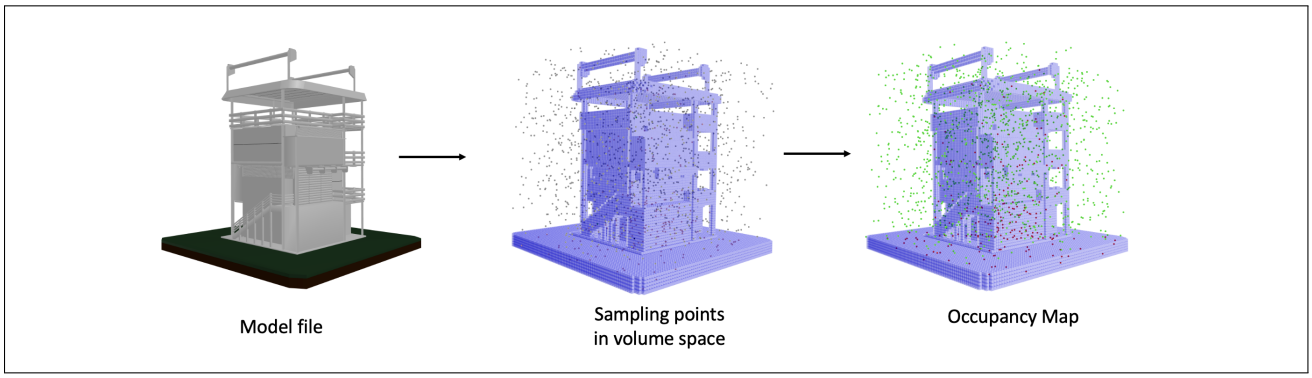

Fig. 6: The process to obtaining an Occupancy Map from a object file.

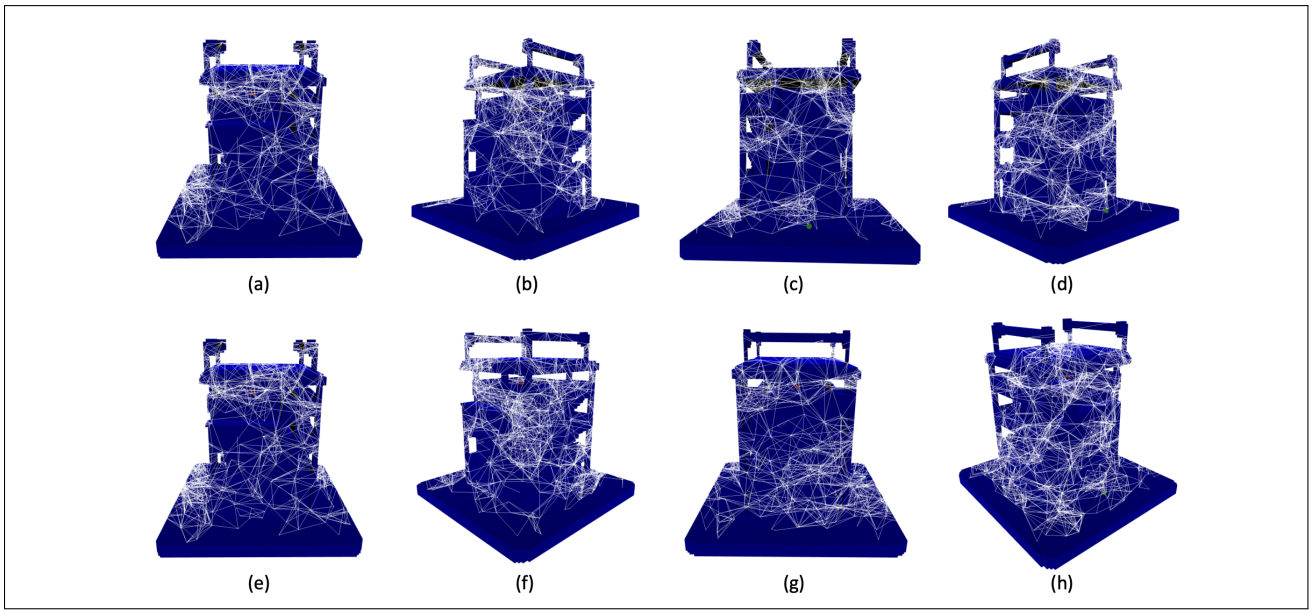

Fig. 7: 3D PRM: 1500 nodes.

demonstrate the scalability of applying this method on multiple robot systems, which has the ability to generate a set of optimal solutions in a short time.

Future research will include ROS architecture for simulation, in which a communication bridge between ROS and MavLink protocol shall be established with a focus on the development of a node responsible for referencing the UAVs' initial and task positions to Gazebo. In addition to this, future research shall also involve environments with dynamic obstacles in real time and modify the preset path if necessary.

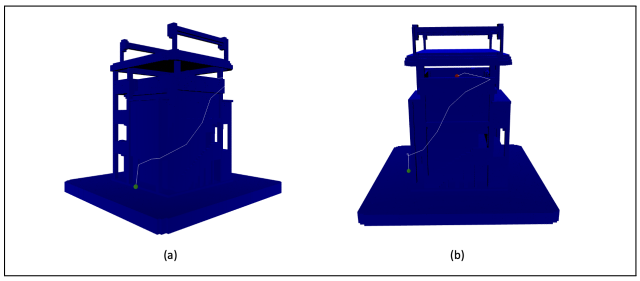

Fig. 8: Resulting solution after A* algorithm.

REFERENCES

[1] Ángel Madridano, Abdulla Al-Kaff, David Martín Gómez, and Arturo de la Escalera. 3d trajectory planning method for uavs swarm in building emergencies. *Sensors*, 20:642, 01 2020.
[2] Patrick Min. [binvox] 3d mesh voxelizer.
[3] University of Freiburg. Octomap - an efficient probabilistic 3d mapping framework based on octrees.
[4] Dawson-Haggerty et al. trimesh.
[5] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.