Autonomous Navigation and Mapping of Snake Robots for Urban Search and Rescue

Syed Izzat Ullah
Department of Computing Sciences
Texas A&M University-Corpus Christi
Texas, USA
sizzatullah@islander.tamucc.edu

Tallat Mahmood
Control, Automotive and Robotics Lab
NCRA, Rawalpindi, Pakistan
tallatmahmood98@gmail.com

Anayatullah
Control, Automotive and Robotics Lab
NCRA, Rawalpindi, Pakistan
anayat.ullah@buitms.edu.pk

***Abstract*—In an Urban Search and Rescue (USAR) situation, under extreme time pressure, rescue workers have to locate and extract the trapped people in collapsed structures. Due to the lack of medical treatment, food, and water, the victim's mortality rate dramatically increases over time. Rescue operations for both rescue workers and victims might be as dangerous as the initial event. For such situations, snake robots which are inspired by their biological counterparts, are shown to be a good option in the literature, to help the rescue workers in positioning the victims or delivering life-saving drugs to extend the life of the victims for some time. However, current research mainly focuses on mechanical design, control mechanisms, and gait generation. To alleviate this concern, we have integrated state-of-the-art methods to develop an autonomous snake robot that can navigate in an unknown environment while also generating a 3D map, to provide a better idea of the environment to the rescue workers. A simulated maze environment is implemented and demonstrated by using the CoppeliaSim simulation, running on Robot Operation System (ROS) and Linux OS. The simulation result shows the effectiveness of the proposed autonomous navigation system for the snake robot to plan an obstacle-free path from the robot's current position to the goal position without an apriori knowledge of the environment.**

Index Terms*—Snake robot, navigation, Mapping, Path Planning, ROS, CoppeliaSim, USAR, RRT

I. Introduction

In an urban search and rescue (USAR) operation, the focus of the rescuers is to locate and extract the trapped people in collapsed or damaged structures. Owing to the exposure and absence of medical treatment, water, and food; the victim mortality rate dramatically increases after 48 hours [1]. For both rescuers and victims, rescue operation might be as dangerous as the initial event. Moving through the collapsed structure or enlarging entry points for people and rescuing equipment may cause further collapse, endangering the lives of trapped survivors or rescuers. Besides that, the possibility of gas leakage and explosions can also increase the threat of survivors' mortality. In this situation, a rescue dog can aid in exploring smaller void spaces that a human cannot, but they cannot substitute professional structural assessment equipment or a camera system that provides live situations to the rescue workers. A small lightweight robot with perceptional and navigational capabilities to explore the area can be a good option for an urban search and rescue operation while significantly limiting the danger to both humans and dogs. For example, the robot could

- Localize the survivor's limbs to prevent rescue workers from damaging them during extraction,
- Place specific sensors into the rubble to accurately position the survivor,
- evaluate the structural damage by collecting visual and seismic data,
- deliver communication devices or a small amount of food and lifesaving drugs to the survivor, and
- help guide the use of jaws-of-life and other rescuing tools.

Towards that end, studies have been conducted on the research and development of various mobile robots for USAR with different levels of capabilities. These include, for instance, an aerial vehicle, tracked robots, bipedal robots, and modular re-configurable robots. Despite having much wider terrain adaptability than wheeled robots, these robots still cannot fit all types of terrain. The exceptional mobility of snakes in a variety of environments, including desert, water, rocky mountains, and trees, has compelled researchers to develop snake robots that move and looks like biological snake and exhibits locomotion without requiring wheels or legs [2]. Snake robots are well suited in this case to help the rescue workers speed up the recovery of trapped survivors. The application of snake robots is not only restricted to USAR but they are also a suitable option for operating in numerous applications such as surveillance, firefighting, inspecting pipes, and hazardous tasks in industrial and nuclear plants where human operations are not preferred. To alleviate this concern, high intelligence of the snake robots is apparently required, which proposes autonomous control techniques and task adaptive path following strategy. Tanaka et al. [3] have shown a semi-autonomous snake robot with passive wheels that can avoid collisions with obstacles. They have used a 2D LiDAR and a camera mounted on the head of the robot for SLAM and frontal obstacle avoidance, while ultrasonic sensors at the sides are used for lateral collision avoidance. However, 2D LiDAR and passive wheels restrict the robot to navigate on a planner

Syed Izzat Ullah was with the Control, Automotive and Robotics Lab (CARL), National Centre of Robotics and Automation, Rawalpindi, Pakistan. He is now pursuing a Ph.D. in the Department of Computing Sciences at Texas A&M University-Corpus Christi, TX, USA

surface only. Furthermore, the use of multiple sensors will increase computational complexity and reduce the operation time due to excessive power consumption. Similarly, Li et al. [4] has used an overhead camera mounted on the top of a custom-built arena for localization and a front-facing camera on the robot head for collision avoidance. Sartoretti et al. [5] have proposed a combined velocity and heading control-based navigation strategy in simulation for a cluttered environment where instead of avoiding the obstacles, it has to be exploited to support and direct the motion of the snake robot. Yang at el. [6] has investigated a perception-aware path-finding strategy where a 2D LiDAR is used to model the obstacle map and then a modified RRT is used as a path planner to find a feasible path to the goal position. Machine learning-based approaches, including a deep convolutional neural network coupled with reinforcement learning, are also being used in contemporary techniques for snake robot navigation [7] [8]. These approaches tend to tackle the object tracking and control pipeline in an end-to-end manner, based on reinforcement learning.

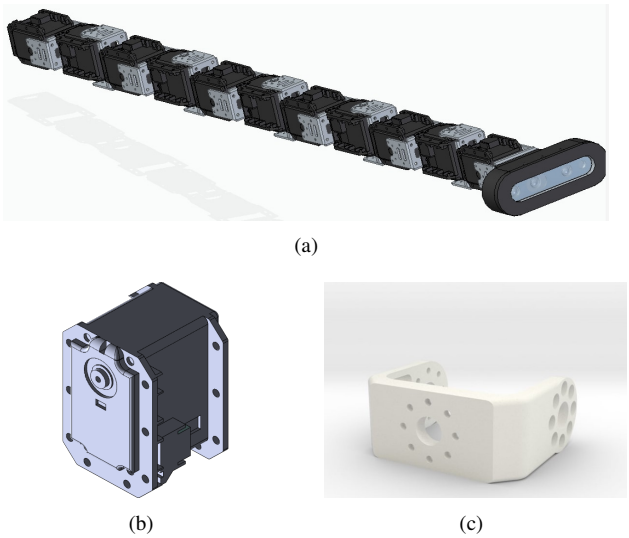
Our work is more inspired by a real USAR scenario where the robot has to explore the environment for victims and as well as generate a 3D map to reel back once the victim is identified. We have used only a single stereo camera as the only perception sensor for both mapping and obstacle avoidance. In detail, our framework consists of three implementation procedures. First, the stereo camera is utilized to perceive the environment around the robot. Then, we iteratively create a local map from the stereo point cloud using the octomap [9] package of the ROS, by which an Informed RRT* [10] is used to find a feasible path from the robot's current position to the goal position. Finally, to navigate over the path, we have developed a path following strategy as explained in section III-A5, to navigate the snake robot using the three locomotive gaits; rectilinear, clockwise, and counter-Clockwise.

In this work, the snake robot localizes itself within the simulation framework using the simulation ground truth, hence, we assume that the robot has the knowledge of its current position and the global goal position at all times.

II. Background

A. Mechanism of the Snake Robot

The construction of the snake robot in this paper bears a strong similarity to Monsalve et al. [11], consisting of similar modules as shown in Figure 1 and a newly proposed head module. Each module is equipped with a COTS one degree of freedom (DOF) actuator, known as ROBOTIS MX-28T. These actuators are electrically connected serially via a 3-pin cable to each other. The head module is especially designed to accommodate an Intel realsense D435 stereoscopic camera for the reconstruction of the environment. The tail module is assumed to be connected to a remote static PC through a three-pin cable for power supply and signal transmission. The robot consists of a total of 11 joints and together these joints accumulate to 11 degrees of freedom mechanism.



(a)

(b) (c)

Fig. 1: Illustration of the mechanical structure of the snake robot developed in a simulation environment. (a). The complete body of the robot with a custom-designed head (b). A 3D model of the servo motor, and (c). the connecting bracket

B. Locomotive Gait Generation

The design of the snake robot determines its capability to perform 2D and 3D locomotion. Various gaits have been developed and proposed for the locomotion of the snake robot. As described by equation 1, locomotive gaits such as rectilinear (linear progression), clockwise, counter clock-wise, sidewinding, turning, and rolling can be generated using a sinusoid function generator. In this work, we have mainly utilized rectilinear, clockwise, and counterclockwise gaits to navigate the snake robot from its current position to the global goal position.

As mentioned in the previous sub-section II-A, the mechanical structure of the snake robot consists of orthogonally connected, 1 d.o.f joints, where odd joints are ascribed to modules that bend in the horizontal plane, while even joints to modules that bend in the vertical plane. For a rectilinear gait, only the even modules engage in the mechanism when a vertical sinusoid wave perpendicular to the ground is passed; the odd modules remain stationary. On the other hand, only the odd modules will engage in the overall locomotion when a horizontal sinusoid wave is passed through the mechanism, and the even modules will stay stationary. Hence, orientation knowledge of the odd and even modules with reference to the plane is important for such mechanisms. In the case of clockwise and counter clock-wise gaits, the intention is to rotate the snake robot about its centroid. These gaits are important when the robot has to set its orientation toward a particular direction.

Locomotive gaits can be described by joint angles. A sine function generator with an offset is used to model each joint angle. As given by equation 1, the joint angle of the nth module at time t, is:

$$\Theta(n,t) = \begin{cases} A_{even} Sin(\omega t + n\delta) + \alpha_{even}, & n = even, \\ A_{odd} Sin(\omega t + n\delta + \phi) + \alpha_{odd}, & n = odd, \end{cases} \quad (1)$$

where, A_{even} and A_{odd} are the amplitude of even and odd joints, ω is the angular frequency of the sinusoid joint motion, δ is the spatial frequency that controls the lag between two consecutive joints, and α_{even} and α_{odd} are the offsets of the even and odd joints, respectively. The selection of these parameters changes the nature and type of gait generated by the robot.

C. Mapping

For the autonomous navigation of the snake robot, 3D structural information such as the obstacle region and free region is required. Various mapping technique has been proposed in the literature that maintains a volumetric representation of the environment in the form of free space, occupied space, and unknown space. In this paper, we utilize Octomap [9]; an open-source probabilistic technique for 3D reconstruction of the environment. OctoMap is an octrees-based efficient technique in terms of the storage of 3D models as it can capture huge environments in compact maps. OctoMap employs a particle filter and a probabilistic technique to effectively deal with measurement noise and update the map. Each node in an octree approach reflects the occupied space in a cubic volume, also known as a voxel, employing a hierarchical data structure for spatial subdivision in three dimensions. Each voxel is recursively divided into eight sub-voxels until the minimum voxel size is reached, which determines the resolution of the octree. This will allow controlling the resolution of the map by limiting the depth of the octree. Figure 2 shows an illustration of the voxel structure and its octree representation.

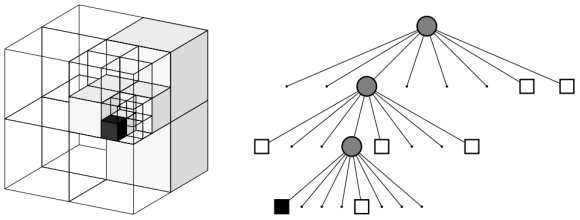


Fig. 2: An illustration of the Octomap octree representation, source of Fig. is [9]

The main advantages of using an octomap data structure in autonomous navigation and exploration are:

- The space is defined as occupied, free and unknown areas
- No prior knowledge of the environment is required and maps based on the current 3D information acquired from the noisy sensors
- Estimate and maps the space in a probabilistic manner that facilitates the dynamic environment and noisy sensors information
- The built map can be saved, shared, updated, and elongated as per requirement in one or multi-robot settings at any instant of time
- The Robot Operating System (ROS) package allows building the map in required resolutions
- memory and computationally efficient for mapping large environments

D. Path Planning

Autonomous navigation depends on the planning of collision-free paths. Path planning is concerned with finding a collision-free path and generating a list of waypoints. It is mostly performed in the configuration space which has dimensions equal to the degree of freedom of the robot, and each point in the configuration space represents the state of the robot. The configuration space is divided into two subspaces; free space, which has collision-free configurations, and collision space, which reflects the configurations of the robot being in a collision with the environment or itself. The main objective of the path planning algorithm is to find a path from the start configuration to the goal point configuration in the free space.

Sampling-based motion planning, which avoids the requirement to compute the entire configuration space, has been the subject of recent research to offer effective solutions to path planning. Rapidly Randomized exploring Tree (RRT) is famous among the sampling-based methods, that are probabilistically complete, meaning they provide a (sub)optimal path from start to goal position when an infinite time is provided [12]. RRT* [13] is an improved version of the RRT that is asymptotically optimal but is computationally expensive due to its inconsistency with the single-query nature in high dimensions.

In this paper, we utilized a variant of the RRT*; known as informed RRT* [10]. Informed RRT* performs the same as RRT* until the first (sub)optimal solution is found, afterwards it only samples from the subset (i.e. “an ellipsoid that encompasses the initially found path”) of the search space. A comparison of the RRT* and Informed RRT* is illustrated in Figure 3. A robot should be integrated with re-planning capabilities in order to traverse the space safely in an unknown environment with little or no knowledge of the free and collision space. As the robot moves in the environment, it perceives more information through the perception sensors and eventually applies collision checking to ensure the rest of the configurations found in the path are collision-free. In case any collision is detected, a re-planning takes place to plan a new collision-free path from the robot position to the goal position.

III. Proposed Method

This section explains the overall methodology of developing an autonomous navigation framework for the snake robot.

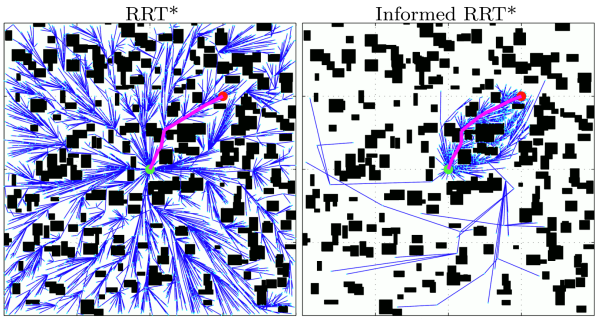


Fig. 3: An illustration of the comparison between RRT* and Informed RRT* in a random world. RRT* and Informed RRT* work the same until a first solution is found, where the informed RRT* focuses its search to only a subset of the space that includes the first (sub)optimal solution, hence, Informed RRT* consumes less computational resources and time

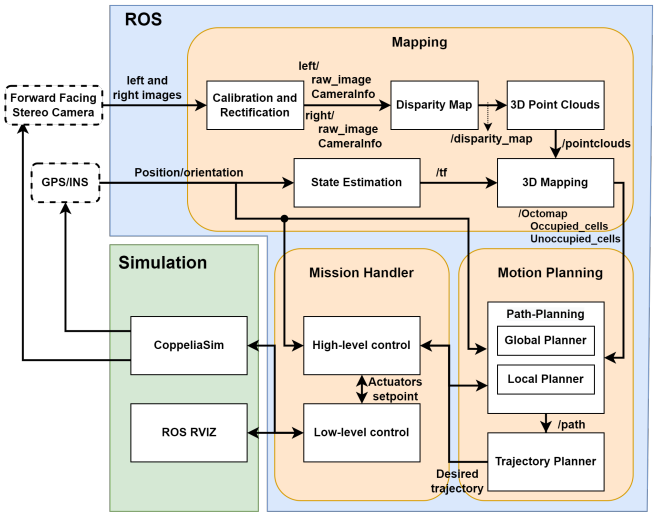


Fig. 4: Overall systems architecture of the proposed framework

A. System Architecture

As shown in Figure 4, the overall system architecture of the proposed framework consists of two main parts; simulation, and software implementation using ROS. We have simulated the autonomous navigation of the snake robot in CoppeliaSim [14]; a robotics simulator that is explained in detail in section III-A1. In this work, we assume a perfect localization of the snake robot from GPS/INS in the simulation environment. Initially, the stereo images are calibrated and rectified to compute disparity images and 3D point clouds using ROS *stereo image proc* package [15]. The 3D point clouds coupled with the transformation information are then subscribed by the ROS *Octomap* package to generate the 3D map of the part of the environment perceived by the stereo camera. Using the localization of the snake robot, these local maps are transformed into a global frame of reference for a complete map generation. The octomap publishes the obstacle map (i-e occupied cells for visualization on RVIZ) and complete binary map which includes a compact binary stream encoding free and occupied cells. We then implemented Informed RRT*; an improved version of the RRT* on the octomap binary map for an obstacle-free path from the robot's current position to the goal position. A linear interpolation technique is applied on the path provided by the path planner to smooth the path. Finally, a trajectory tracking controller is implemented to navigate the robot on the desired path. In the later sub-sections, we explain the main blocks of the system architecture in detail.

1) Simulation Environment: The performance of the proposed framework for the autonomous navigation of the snake robot is evaluated in a simulated environment using CoppeliaSim [14] and ROS. CoppeliaSim is a flexible, and general-purpose robotics simulator. It has distributed control architecture, due to which, each object in CoppeliaSim can be controlled using ROS nodes, remote API clients, embedded scripts, plugins, or custom solutions. Furthermore, CoppeliaSim has a nice integration with SDKs written in C/C++, Java, Python, Matlab, and Lua, which make it easy to use and implement the realization of any robotics idea. As shown in figure 6(a) and 6(c), we have created a maze environment that includes walls as an obstacle. The location of the start and goal position, coupled with the structure of the maze environment is set in such a way that evaluates the continuous planning and re-planning of the path planner while the snake traverses the goal point. The goal location is represented with a dummy object on a green circle that the snake robot has to reach while avoiding obstacles.

2) Environmental Perception: A stereo camera is mounted on the head module of the snake robot to enable visual feedback of the environment while navigating unknown terrains. In particular, because of the limited space and overall weight distribution, a relatively small stereoscopic vision system is embedded in the snake robot. In the proposed framework, we have chosen a low-cost COTS Intel Realsense D435 stereoscopic camera as the only perception sensor. Intel Realsense provides calibrated left and right images and is ideal for depth estimation in the range of 0.3 to 3 meters. For processing the stereo images, we have used the *stereo image proc* package of the ROS. The *stereo image proc* package subscribes to the left and right images of the stereo camera, estimates the depth information, and publishes a 3D point cloud.

3) Mapping: In this framework, the 3D structural map of the environment is required for autonomous navigation. The map should be maintaining the representation of the environment in the form of free space, occupied space, and unknown space. As mentioned in section II-C, volumetric representations such as the Octomap are one of the 3D map representations that have the advantage of providing this information about the environment. $Octomap - server$; which is a mapping library, to compute and create the 3D maps of an environment using the Octomap technique, is already integrated with the ROS. Hence, in this work, we

take the advantage of Octomap server to build the map. Octomap subscribes to the $3D$ $point$ $cloud$ of the stereo camera explained in section III-A2 and tf $transformation$ that provides the transformation between the stereo frame to the world frame. It then computes the map and publishes:

- $Octomap\_binary$, which is a compacted stream of occupied and free space,
- $occupied\_cell\_vis\_array$, which is the obstacle map/ occupied cells for RVIZ visualization,
- $point\_cloud$ that includes the centers of all occupied voxels, and
- 2D projected occupancy grid mappings

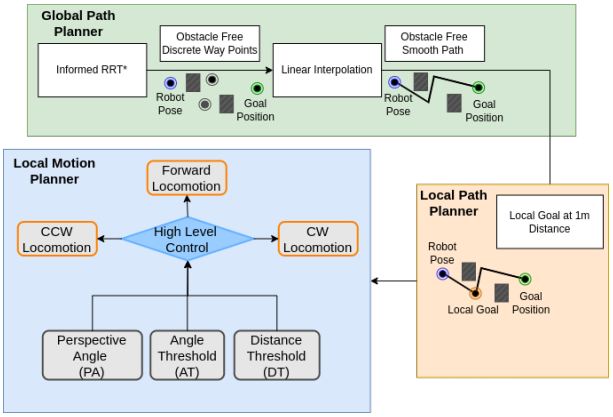
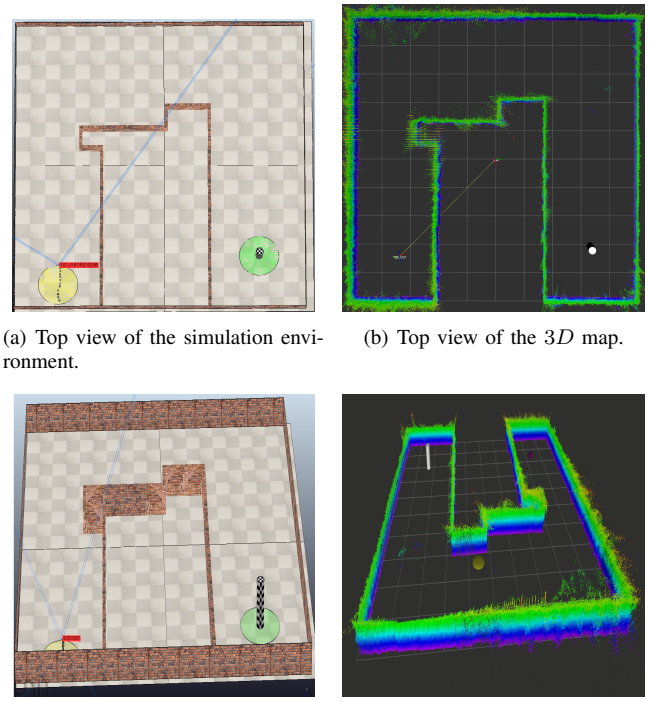


Fig. 5: Flow diagram of the path planning and tracking methods

4) Path Planning: The octomap provides a 3D representation of the environment that includes free and occupied spaces. The next step after mapping is to implement a path planning algorithm that provides an obstacle-free path from the robot's current position to the global goal position. We have implemented a ROS node for path planning that uses a flexible collision library [16] to compute and check the collisions with obstacles and an Open Motion Planning Library [17] to plan paths using Informed RRT*. The path planning node subscribes to the $octomap\_binary$ and tf topics to plan paths and then publishes $waypoints$. Since, octomap provides the advantage of updatability; it can map dynamic objects unless they are in the perception range of the stereo camera, hence, if any object happens to be colliding with the existing waypoints, the planner will re-plan the path to avoid collisions.

5) Autonomous Navigation: The complete architecture of the path-following strategy is illustrated in figure 5. After receiving the planned path from the planner, we then apply linear interpolation on the waypoints to form a continuous low-cost trajectory from the current position of the snake robot to the goal position. This interpolated trajectory is then divided into path patches with a maximum length of 1 meter. Each patch serves as a local path with the start point as the current position of the robot and the end point as the local goal. Here, we consider the head module as the reference point for path planning as the stereo camera is mounted on the head module, which perceives the environment. Also, from tail to head, we calculate the heading vector of the snake robot. The objective of the path-following algorithm is to generate a sequence of gaits based on the Perspective Angle (PA), Angle Threshold (AT), and Distance Threshold (DT), that can move the snake robot toward the local goal. PA, AT, and DT, respectively, are angles between the robot heading and the current local goal, a threshold for PA to actuate forward gait, and a goal-reaching threshold distance. We use Eq. 2 for AT and $0.5m$ for DT. The CD in Eq. 2 is the current distance between the local goal and the robot's location. Depending on the perspective angle and distance to the local goal, at a particular time, anyone from the three gaits (i-e rectilinear, clockwise, and counterclockwise) will be selected which reduces the error between the planned and tracked path.

$$AT = \max(20, 20 + (1 - CD) * 60) \tag{2}$$



(a) Top view of the simulation environment.

(b) Top view of the $3D$ map.

(c) Isometric view of the simulation environment.

(d) Isometric view of the $3D$ map.

Fig. 6: Results of the mapping of complete maze environment, (a). and (c) are, respectively, the top view and isometric view of the ground truth. (b) and (d) are the respective $3D$ maps generated using the octomap mapping.

IV. Experimental Setup and Results

To validate the performance of the snake robot to map the urban search and rescue environment and plan an obstacle-free path, we have tested our method in a maze-like environment. The environment spans $100m^2$ area with $(10m \times 10m)$ dimensions. Figure 6(a) illustrates the test environment, which consists of walls, plain flooring, a snake robot, a start (shown

as a yellow circle), and a global goal location (shown as a green circle).

A. Mapping

Figure 6 shows the actual ground truth and the generated map of the complete maze environment using our method. For creating a map of the maze environment, no prior information is required, except the real-time localization of the snake robot in the world frame of reference. As can be seen from Figure 7, the local map, created through the local perception of the stereo camera is iteratively transformed into the global map as the robot traverses the environment over time. Because of the stereo camera range limitations, the map gets distorted when the distance between the camera and an object is less than 0.3 meters, which is the lowest range that the stereo can capture depth.

B. Path Planning

The performance of the path planning and autonomous navigation technique is evaluated in two different situations, depending on the level of map detail. When the snake robot is assigned a task to autonomously navigate in the USAR region and explore the environment for victims, the robot does not bear any information about the environment at that time. The only information it has is self-localization and the object (i.e. victim) to be explored and identified. We call this situation autonomous navigation without an apriori map, where the robot navigates through the environment to reach the global goal position and iteratively creates the 3D map, as shown in Figure 7. This map helps both the snake robot and the rescue workers; the rescue workers receive and identify access through which the victim can be traced and rescued, and the snake robot can directly be reeled back by planning the shortest path from the victim's location to the rescue workers. Furthermore, this map can be utilized in the later stage to deliver food, water, and medication to victims in the shortest possible path and time. Navigating from the victim's (i-e goal) position to the rescue worker's (i-e start) position while having a complete map of the environment is our second situation and we call it autonomous navigation with an apriori map. We performed five experiments, where the environment, start, and goal position was kept the same, and the results are provided in Figures 8, 9, and Table I. In this work, we assume the snake robot has an apriori knowledge of its localization and the location of the object(i-e. victim) to reach.

1) Autonomous Navigation without an apriori Map: In this situation, as shown in the temporal representation in Figure 7, the map of the environment is updated in real-time to include all the possible static and dynamic obstacles that happens in the path of the robot. The path planners also replan if the existing path leads to collision with obstacles. The red cube in Figure 7(b) is the current position of the robot, dark magenta cubes are the waypoints provided by the path planner, and red dots show the trajectory followed by the snake robot. This situation is also shown in Figure 8, which shows the comparison of the actual planned path (dotted blue curve) and the traversed trajectory (continuous red curve) by the snake robot. As mentioned in table I, the total planned distance from start to goal location is 19.19 meters, using the three locomotive gaits, and the snake robot took around 7 minutes (i-e. 426 seconds) to traverse the planned path. The robot iteratively reaches the local goals (blue circles) to reach the global position.

2) Autonomous Navigation with an apriori Map: The robot reaches the global goal in the first situation explained previously, and has a complete map of the environment. In this situation, we replace the start and global location of the goal. Fig. 9 illustrates the results of this situation where the blue and red curves, respectively, represent the optimal planned path and robot's footprints to the final goal.

TABLE I: Trajectory Distances

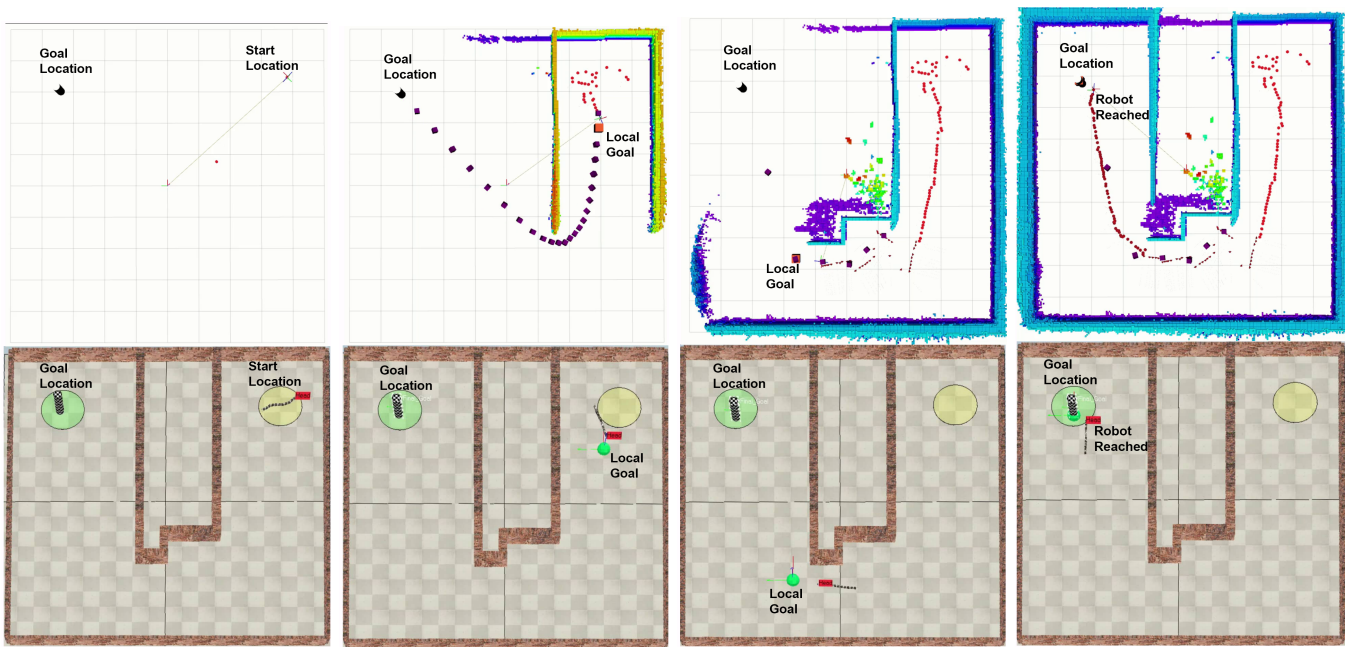
Experiment	Planned Distance (m)	Followed Distance (m)	Time (sec)
Without an apriori Map	19.193	22.534	426
With an apriori Map	13.979	16.752	346

Since the robot already has a complete map from its current position to the rescuer's (i-e. goal) position, therefore, the path provided by the planner is also short as compared to the previous situation. Compare to the previous situation, the snake robot takes 346 seconds to navigate over 13.97 meters of distance. Tab. I contains the information on trajectory distances in meters and traversal time in seconds for navigation with and without an apriori map. Planned distance is the distance from the start to the goal position, planned by the path planner. While followed distance is the trajectory traversed by the snake robot while navigating over the planned path. From Tab. I, we can see that the path planner returns the shortest path (i-e 13.979m) with an apriori knowledge of the map and a slightly longer path without an apriori map. This difference in distances is due to the motion planning and control strategy. It also shows the importance of building a map of the environment for SAR operations, so that the victim can be approached in the shortest time possible to supply medical aid and food, using the map.

TABLE II: Trajectory Following Error (m) Statistics

	Experiment	Mean	Std	Min	Max
At local goal points	Without Map	0.102	0.098	0.002	0.379
	With Map	0.199	0.108	0.012	0.391
At planned path waypoints	Without Map	0.059	0.058	0.000	0.318
	With Map	0.046	0.0454	0.000	0.142

The proposed motion planning method provides acceptable results. The statistics of deviation from the asymptotically optimal path are provided in Tab.II. We define the deviation of the robot from the planned path as an error. We show the error statistics at planned path points and over each local goal point. The mean error at planned path waypoints and followed trajectory by the snake robot, with and without an apriori map, respectively is 0.046 and 0.059, with a standard deviation of 0.0454 and 0.058. Similarly, the mean error between the points



(a) t=00:00 (m:sec) (b) t=01:10 (c) t=04:10 (d) t=07:09

Fig. 7: Temporal illustration of the online map generation, path planning, and navigation of the snake robot towards the goal location autonomously without a map.

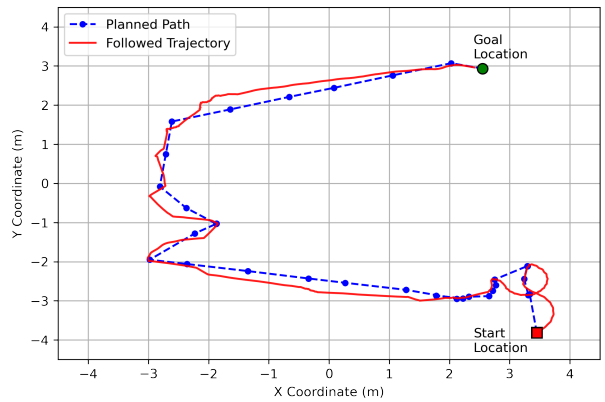


Fig. 8: Results of the path planning algorithm when map is created while the snake traverses the environment, (a). blue circles show the actual waypoints provided by the path planner, (b). the blue dotted line shows the interpolation of the waypoints to form a smooth trajectory, and (c). the red line is the actual trajectory traversed by the robot

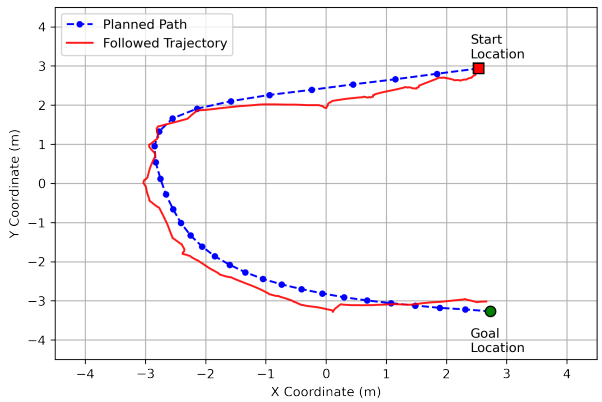


Fig. 9: Results of the path planning algorithm when the snake robot has already reached the victim (i-e. goal) position and has a complete path to the initial position, (a). blue circles show the actual waypoints provided by the path planner, (b). the blue dotted line shows the interpolation of the waypoints to form a smooth trajectory, and (c). the red line is the actual trajectory traversed by the robot

at each local goal over the planned path and trajectory followed by the snake robot with and without an apriori map is 0.199 and 0.102. The robot often crosses the planned path points, resulting in the minimum error of $0m$ in the case of error between the planned path and traversed trajectory. The error lies in the range $(0, 0.379)$ meters in all cases, which is due to the goal reaching the threshold (DT) of $0.5m$.

V. Conclusion

In this paper, we presented an autonomous navigation framework for a snake robot navigating in an Urban Search and Rescue environment. The framework is implemented and developed in a simulation environment using CoppeliaSim and ROS on a Linux OS. We integrated state-of-the-art techniques for path planning, mapping, and implemented an autonomous

navigation system that enables the snake robot to navigate from its current position to the goal position while avoiding collisions with the obstacles. We evaluated our system in five experiments and the results provided have shown that the proposed system can plan an optimal path in a cluttered 3D environment where it is important to compute the 3D structure as the snake traverses over the environment. Future work will include the implementation of an object recognition mechanism rather than goal tracing, and instead of using the localization ground truth from simulation, using SLAM to localize the robot without the requirement of GPS.

VI. ACKNOWLEDGEMENTS

This research is conducted at Control Automotive and Robotics Lab (CARL-BUITEMS), funded by National Center of Robotics and Automation (NCRA) with the collaboration of Higher Education Commission (HEC) of Pakistan.

REFERENCES

[1] R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue," *IEEE Intelligent Systems and their Applications*, vol. 15, no. 2, pp. 14–19, 2000.

[2] X. Xiao and R. Murphy, "A review on snake robot testbeds in granular and restricted maneuverability spaces," *Robotics and Autonomous Systems*, vol. 110, pp. 160–172, 2018.

[3] M. Tanaka, K. Kon, and K. Tanaka, "Range-sensor-based semiautonomous whole-body collision avoidance of a snake robot," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1927–1934, 2015.

[4] G. Li, H. B. Waldum, M. O. Grindvik, R. S. Jørundl, and H. Zhang, "Development of a vision-based target exploration system for snake-like robots in structured environments," *International Journal of Advanced Robotic Systems*, vol. 17, no. 4, p. 1729881420936141, 2020. [Online]. Available: https://doi.org/10.1177/1729881420936141

[5] G. Sartoretti, T. Wang, G. Chuang, Q. Li, and H. Choset, "Autonomous decentralized shape-based navigation for snake robots in dense environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9276–9282.

[6] W. Yang, G. Wang, and Y. Shen, "Perception-aware path finding and following of snake robot in unknown environment," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5925–5930.

[7] Z. Bing, C. Lemke, F. O. Morin, Z. Jiang, L. Cheng, K. Huang, and A. Knoll, "Perception-action coupling target tracking control for a snake robot via reinforcement learning," *Frontiers in Neurorobotics*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2020.591128

[8] S. N. Khan, T. Mahmood, S. I. Ullah, K. Ali, and A. Ullah, "Motion planning for a snake robot using double deep q-learning," in *2021 International Conference on Artificial Intelligence (ICAI)*, 2021, pp. 264–270.

[9] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2878–2883.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2997–3004.

[11] J. Monsalve, J. Leon, and K. Melo, "Modular snake robot oriented open simulation software," in *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, 2014, pp. 546–550.

[12] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. x–xvi, 2019.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *CoRR*, vol. abs/1105.1186, 2011. [Online]. Available: http://arxiv.org/abs/1105.1186

[14] E. Rohmer, S. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," 11 2013, pp. 1321–1326.

[15] J. L. Patrick Mihelich, Kurt Konolige. Stereo_image_proc. [Online]. Available: http://wiki.ros.org/stereo_image_proc

[16] Flexible collision library. [Online]. Available: https://flexible-collision-library.github.io/

[17] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.