Heterogenous Autonomous Robotic Exploration (HARE)

Jackson Parker, Allen Spain, Caleb Adams

***Abstract*— The purpose of this research is to prove that subtask division can be accomplished based on knowledge of robot capabilities. To prove this, we are developing an exploration system in which robots with different hardware and capabilities will work together to accurately explore and map a complex region of space. Through the definition of individual attributes and the sharing of those attributes, each robot will search and map regions of space that it is capable of maneuvering. Instead of simple indexed mapping of the region where each cell on the map is represented by and index, each robot will mark the region that it has explored and can maneuver through this a specific identifier. The robots share sensed terrain using map node which defines the region in terms of terrain. This gives the state space provides the following information that can enable a robot to determine it's terrain, traversability (as determined by groups attributes). After the region has been sensed it is also marked as explored which is factored in when determining tree state. See table below for implemented terrain definitions, see table I in appedix. Once a robot comes to an obstacle or region that it cannot maneuver around, it will evaluate which team members are capable of doing so and mark that space for the capable robots to explore. Due to time limitations on this project, instead of relying on complex perceptive capabilities to evaluate which robots could explore that region, the utilization of terrain identification which can be surveyed by the robot's sensors to determine if a region is traversable, this is designed statically in this experiment as a proof of concept. However, the proposed concept allows robots to compare their own attributes with those required to traverse a terrain in the sensed region. The robot attributes for the general case is shown in table III.**

Fig. 1: The wild hare known for its swiftness, stamina and adaptability.

IEEE, IEEEtran, journal, LaTeX, paper, template, multi-robot systems, exploration, autonomous mobile robots, heterogenous, breadth-first search,path planning, behavior trees

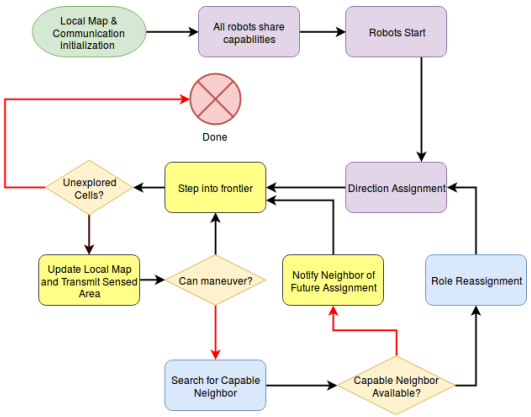


Fig. 2: High level block diagram for HARE

I. Introduction

A. Overview

As an individual working in a cooperative operation, knowing the capabilities of you and your teammates is necessary to the division of tasks. This idea provides the basis for this proposed research. As homogeneous multi-robot systems are inherently limited based on the capabilities of the specific robot, it makes sense

to venture down a heterogeneous avenue. A lot of research has gone into heterogeneous multi-robot systems in the past few years, but a common trend amongst most of them is that the model is built for the specific platforms that are to be used. This proposed research is meant to provide the groundwork for a heterogeneous model that allows for the insertion of a robots with different skill sets with the purpose of enhancing the overall capability of the system. This framework would allow for a variety of models to be built on top of it, with a variety of hardware. Researchers wanting to build a specific system could allocate funds and resources to build each member to accomplish a portion of the overall task, possibly allowing for more cost efficiency and remove unnecessary redundancy. The main challenge in this research will be finding the optimal way to reassign tasks based on defined capabilities of each robot in the system. This brings up a secondary, more basic, challenge of defining robot's characteristics in a simple yet comprehensive manor. All-in-all, this research acts as a proof of concept for task division based on member attributes, or individual robots capabilities, and will allow for a high degree of heterogeneous utilization and multi-robot model expansion.

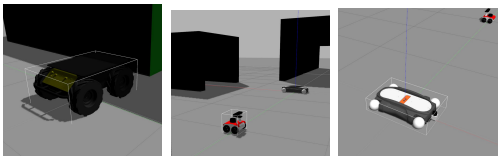
B. Background

The basis of this research is in the definition of attributes which is detailed in the methodology section. This section is meant to detail previous research that will be used in proving that the fundamental idea of task division based on individual capabilities. As this research is meant to strictly provide a basis for heterogeneous task division and capability utilization, the model built on top of it could take many forms. This project was inspired by multiagent decision making using behavioral trees constructed to guide operations and define command sequences at a high level like proposed in [1]. To implement such a system the DSL buzz will be used which provides a level of abstraction that allows for focus on high level cooperation, information sharing, and simple behavioral definitions. The researchers that developed this language have created a ROS package that will be utilized, called ROSbuzz [2], in this proof-of-concept system. Control mechanisms and communication in multirobot systems have been a center of research for a while and updates/innovation in this corner of the field are practically constant. The Robot Operating System (ROS) [3] provides a large variety of packages that can be put together to form fully functioning and unique control mechanisms as well as providing easy to use communication packages. Exploration and path finding are widely researched and implemented using ROS packages, as components are pretty broad and can help illuminate edge cases in multi-robot systems as agents have to operate without global knowledge of their environment. This research served as a foundation for multirobot decision making, communication and high level design framework required to operate a decentralized swarm. However after experimenting with ROSbuzz and ROS in the gazebo environment, the bottom up approach proved to be a better method for the proposed project. This approach utilizes rospackages for point to point communication but uses a class structure to abstract node creation, sensing, and motion. This paper will discuss more on the testing framework created as well as the results from small scale simulations, which were designed to test the communication architecture and improve the proposed sensing and steering models.

II. Methodology

A. Role Assignment

As aforementioned role assignment was predefined for the robot set. In the project three robots were given static characteristics, these characteristics were defined based on robot attributes. The simplified set of attributes taken into consideration were maneuverability and robot height. These attributes corresponded to terrain traversability, the terrain was indexed as follows: 0 - open terrain, 1 - ramp, 2 - short tunnels, and 3 - tall tunnels. Given the set of robots and attributes capabilities were defined. Robot 1 (Youbot) could traverse terrain 0, 2,



(a) Husky (b) Rosbot (c) Youbot

Fig. 3: Three commonly used robots with very different shapes and sizes.

and 3. Robot 2 (Rosbot) could traverse terrain 0, 1, and 3. Robot 3 (Husky) could traverse terrain all terrains in the terrain set, the robots and terrain is illustrated in figure 4.

As an individual working in a cooperative operation, knowing the capabilities of you and your teammates is necessary to the division of tasks. This idea provides the basis for this proposed research. As homogeneous multi-robot systems are inherently limited based on the capabilities of the specific robot, it makes sense to venture down a heterogeneous avenue. A lot of research has gone into heterogeneous multi-robot systems in the past few years, but a common trend amongst most of them is that the model is built for the specific platforms that are to be used. This proposed research is meant to provide the groundwork for a heterogeneous model that allows for the insertion of a robots with different skill sets with the purpose of enhancing the overall capability of the system. This framework would allow for a variety of models to be built on top of it, with a variety of hardware. Researchers wanting to build a specific system could allocate funds and resources to build each member to accomplish a portion of the overall task, possibly allowing for more cost efficiency and remove unnecessary redundancy. The main challenge in this research will be finding the optimal way to reassign tasks based on defined capabilities of each robot in the system. This brings up a secondary, more basic, challenge of defining robot's characteristics in a simple yet comprehensive manor. All-in-all, this research acts as a proof of concept for task division based on member attributes, or individual robots capabilities, and will allow for a high degree of heterogeneous utilization and multi-robot model expansion.

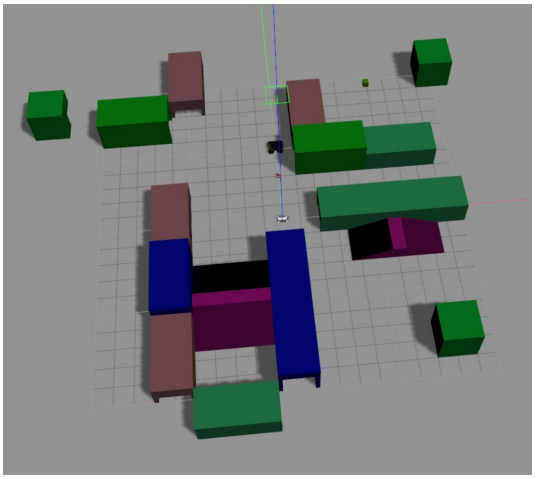


Fig. 4: Gazebo map, with different terrain. Open terrain in gray, short tunnels in red and green, tall tunnels in blue.

B. Algorithms

For navigation we used two methods. The first is a Breadth First Search (BFS) which explores the cells adjacent to the robots current positions, before exploring the second level of adjacent cells. After determinining explorable cells method getPath as described in Algorithm 1 will compute the a hueristic based on the manhattan distance to determine a set of cells that make up an optimal path. For robot exploration getPath can be called for exploration by simpling passing in a goal represented as the nearest unexplored node a summary of the path planning algorithm can be called with the goal being the nearest global unexplored node as shown in the algorithm in figure 5. The use cases for this method are as follows: a) robot needs to swap locations due to percieved danger, b) robot finds an obstacle (tunnel) it cannot explore and needs another robot to explore it 3) the robot's tree-state is not being beconed will return to performing exploration.

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
came_from[start] = None

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      if next not in came_from:
         priority = heuristic(goal, next)
         frontier.put(next, priority)
         came_from[next] = current
```

Fig. 5: An overview of the A* Algorithm [4]

C. Entities

The Entity class is the base physical structure for this testing package. This could represent a static or one of the two derived classes: Neighbor and Robot Entities hold major variables like state information, initial position and odometry. The Neighbor and Robot classes are very similar to each other as they both represent robots, the Neighbor class is just used to hold the information for the Robot's neighboring robots as they do not need to keep track of their entire path and known map, like they are doing for themselves.

D. Steering

The challenge using a centralized class stucture like the one proposed is the necessity to steer all robots in set using a single set of steering methods. Each method was generalized to control a robot independent of its motion model. To achieve this four steering methods were utilized, 1) goBackward(), goForward(), goRight(), goLeft(), turn(angle), face(goal). The idea is such that the robot locomotion would not be defined by a new set of steering methods, but instead would rely on a different sequence of steering calls to perform the same motion. For instance, to turn right a differential robot like the rosbot would call function goRight() followed by goForward(), while a holonomic robot would be ale to directly call goRight(). The robot motion attributes as described in table III will be used to better tune this steering model.The addition of robot attributes could be added to a yaml file, uses and configuration of yaml files is briefly described in section Adding Robots. This simplified experiment focused on the youbot, husky, and rosbot. These robot locomotion models were far from idealistic which gave us a better testing platform for later porting on the physical platforms, but made experimentation challenging. In experiment the huskies linear and angular control actions were greatly impeded due to the frictional force induced on its large outdoor wheels, to mitigate this affect we overloaded the turn method to include a turn radius. In hindsight, we could have applied a gain factor to the turn method to provide a larger control input. Alternatively, the rosbot wasn't greatly effected by the frictional force when applying an angular velocity, however it did suffer from aggregate error, whereby the robot would divert from the direction of its input trajectory after some time. The youbot responded optimallly to the steering sequences, and would serve as a good candidate for beginner user testing and education in this framework.

E. Adding Robots

This framework allows for easy addition of robots by simply adding the details for the new robot in the xml robots launch file. The mutlirobot launch file is called robots.launch this xml sets up the robot models 3. This file calls .yaml which stores the data for the robot controller scripts, these files define the parameters needed for the motion models for each robot. This launch file also utilizes .xacro files which configures the robot form for use in the gazebo simulation environment and physics engine. To add any additional information yaml files can be used to set rosparams and can be queried in any rosnode as long as the yaml file was specified in one of the launch files as a rosparam. Utilization of the yaml files is essential to adding and sharing information like hardware characteristics before Robots enter

the main loop. So all in all, By configuring the launch file with the proper robots, one can edit the one robot launch file to have each robot launch any set of nodes that would exist in each robot's namespace.

One thing that was essential to streamline and allow for flexibility was the instantiation of robots via partitioned launch files. By defining separate roslaunch files instead of one, it is easy to see where to put new robots as well as ensure that one point of failure is taken care of. The one robots launch file launches nodes that are to be in every robot, who are defined by namespaces in the context of ROS. The robots launch file is where robot descriptions are set along with locations of urdf files containing meshes and other simulation specific data for the individual robot. The global multi-agent launch file used is hare sim.launch this file calls robots.launch this file calls one robot.launch which launches the hare node and spawns the robots in the gazebo simulator.

F. Multi-Agent Testing Framework

After exploring a set of options to test this idea, the best place to go was to the drawing board. Taking a step back it was decided that pushing to get straight into hardware testing was a poor decision and the underlying concept of heterogeneous task reallocation based on capabilities needed to be tested at the lowest of levels. To do this, the creation of a testing framework in roscpp was necessary. This framework allows for easy addition of robots by simply adding the details for the new robot in the xml robots launch file. To add any additional information yaml files can be used to set rosparams, just as was done here for robot capabilties. By configuring the launch file with the proper robots, one can edit the one robot launch file to have each robot launch any set of nodes that would exist in each robot's namespace. This allows for testing of a completely decentralized algorithm in a centralized system. The reason this was done was because of the difficulty of getting a multimaster architecture working within gazebo.

III. Results

A. Exploration

Our results were limited to test run outside of gazebo. We were able to test the breath first search algorithm which provided an obtacle free path to a defined goal. The results of this test are shown in figure 6. We also were able to test the robots motion model, the results of these tests revealed that the youbot was an ideal candidate since it was orientation agnostics and was fully holonomic. The one limitation to the youbot platform was its ability to traverse ramps, this was factored in when designing static characteristics as described in the methodology section. The motion model for the husky and the rosbot were far from ideal due to path deflection and turn radius.

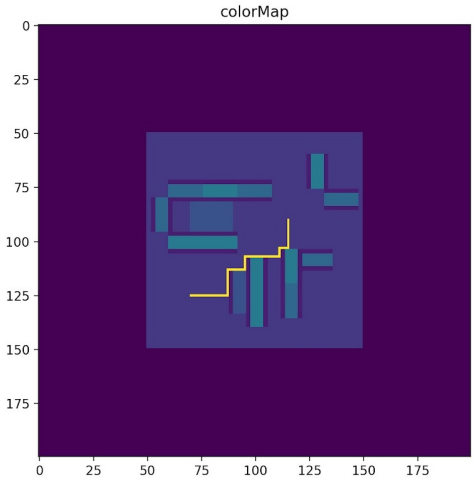


Fig. 6: BFS simulated path planning

IV. Conclusion

A. Discussion

This project served as a foundation to enable future research and testing. This framework is going to be used more as a testing environment for future work such as multiagent micromouse simulation in a realistic 3d environment allowing the user to easily test their Algorithms, with sensing and realistic robot maneuvering. Before this system is launched we will provide a more

robust steering mechanissdm which will take into account the robot model, this will abstract out the difficulty associated with navigating to a goal or following a line. In addition to abstracting out the motion models, this framework is designed to support complex multiagent decision making like those represented in behavioral trees, however the current project uses a simple switch statement to determine its next action.

robots to share their information states including the enountered terrain in order for the robot team to best explore and traverse a given heterogenous environment. The resultant decision that is derived from an agents neighbor's states namely TREE STATE, this variable represents the robots decision, which could entail further investigation (PROD) or stopping all motion and entering into a low power mode (IDLE). for a full description of table states see table II.

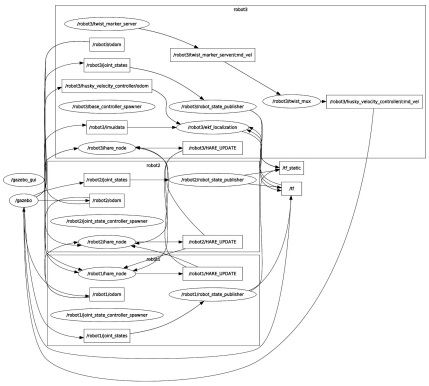


Fig. 7: ROS multi-master communication architecture

B. Future Work

Even though the testing framework was not the overall goal of this research, it turned out to be one of the strong suits of the project. Future work would certainly include iteration on this framework to ensure that it is as easy as possible to randomly generate maps as well as add any number of any kind of robot. The benefit of doing this on Gazebo is the ability to emulate robotic agents much closer to the real world representation than Unity, ARGoS, or any other open source platform capable of multi-robot simulation. Another step would ofcourse be iteration of the tested cooperation model on hardware. This would require much more extraneous simulation to ensure that all edge cases are covered. To properly do this, the testing framework would have to be iterated as described above. Due to Gazebo not inherently supporting multimaster network architectures, there is the possibility that a plugin or new simulation package would need to be developed [5]. The communication architecture with all nodes encapsulated within the robot namsepace this communication archietecture is shown in figure 7.

APPENDIX I
USING THE CODE FROM THIS PAPER

The code written for this paper is available at, this repository will be updated on a periodic basis in order to complete proposed objective. **`https://github.com/uga-ssrl/hare`**.

TABLE I: Terrain index description

Terrain	Index
Unknown	-2
Wall	-1
Open Space	0
Ramp	1
Short Tunnel	2
Tall Tunnel	3

TABLE II: Decision state description

STATE	Decision State
IDLE	Default, low power, malfunction
SEARCH	Explore unknown frontier
RIDE	Seek goal
PROD	Investigation, check tunnel with unknown terrain
DONE	All traversable tunnels prodded, search space explored, out of goals

TABLE III: Robot Characterization

Attributes	Considerations
Dimensionality	directional proximity, 2D, 3D
Sight FoV	horizontal angle, vertical angle
GPS	error in meters
Telemetry	sensor input, telemetry type
Visible Range	distance in meters
Altitude	relative to state space minima
Safe Terrain	submersible, non-submersible, ground, aerial, amphibious, all
Appendages	num, type, joint specifiers, degrees of freedom
Tools	bucket, wedge, shovel, pike, clamp
Maneuverability	3 - holonomic, 2 or 1 nonholonomic, 0
Communication Type	full/half duplex
Reciever FoV	angle horizontal, angle vertical
Transmitter FoV	angle horizontal, angle vertical
Reciever Range	meter distance
Transmitter Range	distance in meters
SNR	ratio
Channel(s)	number
Bandwidth	frequency in Hz
CPU	number cores, clock speed, type
GPU	number, clock speed, type
ASIC	number, type
FPGA	number, type

REFERENCES

[1] M. Colledanchise. (2017, April) Inside the virtual robotics challenge: Simulating real-time robotic disaster response.

[2] MISTLAB. (2019) Rosbuzz.

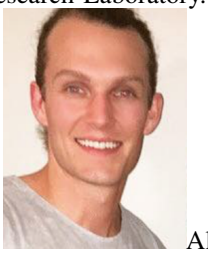
[3] J. M. O'Kane. (2018) A gentle introduction to ros.

[4] "Introduction to the a* algorithm," *Red Blob Games*, 2014.

[5] A. Tiderko, "multimaster fkie," *ROS wiki*, 2015.

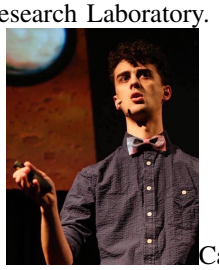
Jackson Parker

M.S. Student with an Emphasis in Electrical and Computer Engineering, currently is the Systems Lead for the UGA Small Satellite Research Laboratory.



Allen Spain

M.S. Student with an Emphasis in Electrical and Computer Engineering, currently is the Lead Hardware Engineer for the Small Satellite Research Laboratory.



Caleb Adams

Ph.D. Student in the department of Computer Science, is the Co-Founder of the Small Satellite Research Laboratory.