Multi-Robot Search and Rescue Operation

Sara Honarvar
ENME 808T-Final Project
Mechanical Engineering Department
University of Maryland, College Park
MD, USA
honarvar@umd.edu

***Abstract*—In this paper, control strategies used for multi-robot search and rescue operations are discussed. The control strategies are designed for three objectives. First, rendezvous and ensuring that all agents can communicate with another while avoiding collisions. Upon rendezvous, the agents would switch their objective to enter the search area while avoiding Obstacles. Once at the facility, the agents would switch to exploration mode to search for survivors. The simulation and experimental results are provided to verify the control strategies.**

I. Introduction

Networked control systems are a class of systems where individual units interact with each other, and one's dynamics affect its neighbors'. Due to the high cost of fast communication and low reliability of centralized methods, distributed control has attracted much attention among researchers and in the industry recently. By using distributed methods, one can use the redundancy in the system to improve reliability.
This paper aims to explain and evaluate some distributed control methods and their implementation on real robots in the context of multi-robot search and rescue operations. In this scenario, we needed to get a team of robots to search for and rescue the survivors. The accomplishment of three tasks was needed for this mission: Initial deployment, Navigation to the cluttered environment, and finally, search and rescue. In the following, the submission for each task and the controller used to complete the mission will be explained. We verify the performance of the explained methods using simulation and experimental results.

II. Initial deployment

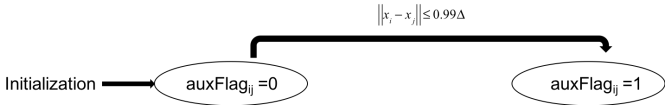
In this part, the robots had to rendezvous to ensure that the agents can communicate with one another while avoiding collision using minimum safety distance, δ, and maximum interaction distance, Δ. The interaction between agents was modeled using a Δ-disk graph.
To accomplish this objective, we used the following edge tension function. This function tends to infinity at δ or Δ. This edge tension function with some hysteresis was used to ensure connectivity in our Δ-disk graph and achieve consensus.

$$\epsilon_{ij} = 10^{-3}\left(\frac{\|x_i - x_j\|}{(\Delta - \|x_i - x_j\|)(\|x_i - x_j\| - \delta)}\right)^2$$

The consensus weights were given by:

$$\begin{aligned} w_{ij}(\|x_i - x_j\|) &= \frac{1}{\|x_i - x_j\|}\frac{\partial \epsilon_{ij}(\|x_i - x_j\|)}{\partial(\|x_i - x_j\|)} \\ &= 10^{-3}\left(\frac{\|x_i - x_j\|^2 - \delta\Delta}{(\Delta - \|x_i - x_j\|)^3(\|x_i - x_j\| - \delta)^3}\right.\end{aligned}$$

Hysteresis was included through the use of the auxFlag variable to mitigate for large total energy jumps due to the addition of new edges.



$$\dot{x}_i = -\sum_{j\in\mathcal{N}_i}\frac{\partial \epsilon_{ij}^T}{\partial x_i} = \sum_{j\in\mathcal{N}_i} w_{ij}(\|x_i - x_j\|)(x_j - x_i)$$

This energy design led robots to successfully rendezvous in both simulation and experiment. The parameters were chosen based on the trial and error in the simulation environment.

III. Navigating the Cluttered Environment

To accomplish this objective, we used the following edge tension function. This function tends to infinity at δ or Δ. This edge tension function with some hysteresis was used to ensure connectivity in our Δ-disk graph and achieve consensus. The objective of this part is for agents to enter the search area while avoiding obstacles. They need to switch to a leader-follower formation mode to accomplish this mission. It is worth mentioning that only the leader has access to the target, which is the last known location of the robots. In this part, we also need to avoid collisions between agents. Nevertheless, as in the real world of robotics there is only so much we can ask from the robots to do, we decided to use a less restricted formation such as a line graph topology while completing the mission. For all agents, including both followers and the leader, $j \in \mathcal{N}_i^{\lceil}$, we used the following design:

First, we defined the adjacency matrix for the line graph as follows.

$$Formation = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Then using the agents' ID, we found the neighbors and created the required edges, according to the network topology for the line graph, to maintain connectivity.

Second, for all agents, we wish to ensure no collisions take place. We did that through the use of two edge tension designs. If the agents shared an edge (according to the above-mentioned line graph), the edge tension design was given by:

$$\epsilon_{ij} = 10^{-2} \left(\frac{\|x_i - x_j\|^2}{(\Delta - \|x_i - x_j\|)(\|x_i - x_j\| - \delta)} \right)$$

And the consensus weights were given by:

$$w_{ij}(\|x_i - x_j\|) = \frac{1}{\|x_i - x_j\|} \frac{\partial \epsilon_{ij}(\|x_i - x_j\|)}{\partial(\|x_i - x_j\|)} = 10^{-2}(\frac{2(\Delta - \|x_i - x_j\|)(\|x_i - x_j\| - \delta) + (2\|x_i - x_j\| - \delta - \Delta)\|x_i - x_j\|}{(\Delta - \|x_i - x_j\|)^2(\|x_i - x_j\| - \delta)^2})$$

In the case that agents did not share an edge (again, according to the above-mentioned adjacency matrix), the edge tension design was less restricted, and it was as follows:

$$\epsilon_{ij} = 3 * 10^{-1} \left(\frac{\|x_i - x_j\|^2}{\|x_i - x_j\| - \delta} \right)$$

The consensus weights were given by:

$$w_{ij} = 3 * 10^{-1}(\frac{\|x_i - x_j\| - 2\delta}{(\|x_i - x_j\| - \delta)^2})$$

In order to avoid debris, we used the data from eight range sensors located on the agents. The key here was that the agents must have an additional velocity (control input), which led the robot to move away from the obstacles based on the sensor data. In this scenario, once the robots sense an obstacle, we calculate the direction that gets the robots to the obstacles. Then using this additional input, we ask the robot to move in the opposite direction of that direction. This direction has a weight which is a function of distance to the obstacle and acts like another edge tension energy design given by:

$$\epsilon_{ij} = 10^{-3}(\frac{1}{(d - \delta)})$$

Where d is the distance to obstacles. The consensus weights were given by:

$$w_{ij} = 10^{-3}(\frac{-1}{(d - \delta)^2})$$

Besides, for robots to move toward the target and the search area, we added a term to the velocity of the leader, as it had access to the target location. $u_i = u_i + 3e0 * (x_t - x_i)/max(norm(x_t - x_i), delta)$ This term makes the leader move with constant speed until it reaches the target within δ, at which point it slows exponentially as it approaches the target further. Using the mentioned control law, we could guide the leaders and followers to get to the search area in both simulation and experiment. Again the parameters were chosen based on the trial and error in the simulation environment.

IV. Search and Rescue

In this part, robots need to switch to exploration mode to search for survivors. It is needed for the agents to avoid collisions again. For collision avoidance, the same strategy as the initialization part was used. So, we used the same edge tension function with hysteresis to account for the collision. For covering the area and to roughly specify where the agents should be located in space, a continuous-time coverage algorithm was used. For doing so, we used the following time-varying density function and Lloyd's algorithm.

$$\Phi(q, t) = e^{-(q_x - 2cos(t/10)^2 + (q_y - 2sin(t/10))^2}$$

This time-varying density function helps to ensure the optimal coverage of the domain of interest in our problem. For the optimal coverage, we partitioned the domain into cells and let each robot be in charge of its own region. We let the 5 robots with positions $x_i \in D \subset \mathbb{R}^d$. The domain D itself was divided into regions of dominance, e.g., $P_1, \ldots, P_n$ (forming a proper partition of D), where the idea was to let robot i be in charge of covering region P_i. To measure how well a given point $q \in D$ is covered by robot i at position $x_i \in D$, we can define the following cost function.

$$H(x, P, t) = \sum_{i=1}^{n} \int_{P_i} \|q - x_i\|^2 \Phi(q, t) dq$$

This cost ensures that the performance of sensors deteriorates with a rate proportional to the square of the distance. At a given time t, when a configuration of robots x together with the partition P minimizes the above cost, the domain is said to be optimally covered with respect to Φ. However, it is possible to view the minimization problem as a function of x alone by observing that given x, the choices of P_i that minimize the above cost is $V_i(x) = \{q \in D | \|x_i - q\| \leq \|x_j - q\| \forall j\}$ i.e., all the points closest to i. This partition of D is a Voronoi

tessellation. So, with this choice of region, the locational cost could be rewritten as:

$$H(x,t) = \sum_{i=1}^{n} \int_{V_i(x)} \|q - x_i\|^2 \Phi(q,t) dq$$

To minimize the cost, we can run the gradient descent and we have:

$$\frac{\partial H}{\partial x_i} = \int_{V_i(x)} -2(q - x_i)^T \Phi(q,t) dq$$

Since $\Phi > 0$ we can define the mass m_i and center of mass c_i of the ith Voronoi cell V_i as

$$m_i(x) = \int_{V_i(x)} \Phi(q,t) dq$$

$$c_i(x) = \frac{\int_{V_i(x)} q\Phi(q,t) dq}{m_i(x)}$$

So, the gradient descent becomes

$$\frac{\partial H}{\partial x_i} = 2m_i(x)(x_i - c_i(x))$$

Scaling the gradient descent by the mass yields the following which is known as Lloyds algorithm.

$$\dot{x}_i = -k(c_i(x) - x_i)$$

Using this algorithm, we could cover the space and find the survivors in both simulation and experiment. We used k = -1. Again the parameters were chosen based on the trial and error in the simulation environment.

V. Observations on the performance of controllers

The proposed algorithms were implemented in MATLAB first for simulation and then on the Robotarium. The mission was successfully accomplished in both simulation and experiment. The experimental results on the robots verify the performance of the proposed method in the real world. The result is attached as a video file. First, we implemented how a system with Δ-disk information topology that was initially connected could be guaranteed to remain connected by designing edge tension energies that would not allow existing edges to be removed. Since the objective is to rendezvous, this resulted in a complete graph topology, wherein every agent has access to information from all other agents. The initial part was successfully implemented in the experiment but it is worth noting that we needed to increase the communication range for Delta-disk graph connectivity like up to 0.99*Delta to make sure that robots rendezvous. However, the complete graph topology can be a bit constraining in real-world situations. For example, we may want to purposely disconnect a subset of the agents from the network, e.g., to explore the space or to move around obstacles, and eventually reconnect (e.g., to exchange new information). One interesting thing noticed was that in simulation, the robots could have been assigned to satisfy many constraints, but that was not the case for the experiment. As an example, in the navigation part, the robots could get to the search area even with maintaining a complete graph topology at all the time, staying at a specific distance from each other while avoiding debris. However, in the actual experiment, that was not possible, and there was the time that robots might have run into each other, and the collision was unavoidable. Even tuning the parameters was not enough to satisfy all these constraints. So, in the real world, there is only so much that we can accomplish. To fix the issue, we defined a less strict formation. One other interesting thing was the tuning of the parameters based on trial and error. However, we have to be careful about these tuning to ensure the robustness of results. Finally, for the coverage we needed to use a time-varying density function to make sure that we optimally cover the space for finding survivors which was successfully implemented in the experiment. Overall, the implementation at Robotarium took longer than the simulation which can be due to some factors like noises in the environment or some lags in the motion capture systems. Also, Robotarium implementation takes some re-optimization of gains and parameters again after you're done with the simulation, especially for task 2 it seems because of some difference in robots and environment. Lowering the speed of the robots by decreasing the gains was found to be useful in implementing the experiment.

VI. Conclusion

In this paper, edge-tension energy design, formation, leader-follower control, and coverage control using the Lloyds algorithm were discussed and evaluated. The formulation was reviewed, and then its application in robotics in the context of a multi-robot search and rescue operation was discussed. The simulation and experimental results verified the desired performance of the designed controllers.