Multiple Autonomous Underwater Vehicle Tracking in Low Communication Environment

Muhammad Aadil Khan
Mechanical Engineering
Stanford University
Stanford CA, USA
maadilk@stanford.edu

Aubrey Kingston
Dept. Aeronautics/Astronautics
Stanford University
Stanford CA, USA
aubreyk1@stanford.edu

Laura Lee
Mechanical Engineering
Stanford University
Stanford CA, USA
leeane@stanford.edu

***Abstract*—In this paper, we look at multiple autonomous underwater vehicles tracking a target with uncertain communication between the robots. We model these robots with realistic dynamics based on the Cornell University 2019 competition vehicle. We are able to successfully maintain formation around a moving target by utilizing Extended Kalman Filters to estimate robot positions when communication is lost within the network. The target tracking is stable in simulation when the target rapidly changes speed and direction. Some stability is lost in the formation when the speed of the target is large or when the number of robots exceeds six.**

***Index Terms*—multi-robot, underwater robotics, leader, follower, distributed formation control, robots**

I. Introduction

Autonomous Underwater Vehicles are deployed in a wide variety of environments for a range of applications. Some AUVs are deployed in shallow water for hydrographic surveying, pipeline inspection, and scientific sampling. Other AUVs are deployed in deep water, up to 6000 meters, for mapping, environmental surveying, and search and salvage operations. Underwater vehicles allow us to explore the shallows and depths of the ocean with minimal distribution to the ecosystem and without the need for a more costly Human Operated Vehicle (HOV).

One of the main challenges in underwater exploration is the lack of communication. When underwater, the vehicle has no Global Positioning System (GPS) or Wi-Fi communication abilities. The most reliable form of communication underwater is through the use of acoustic modem transducers. However, acoustic communication is range limited (often limited to just one or two kilometers) and has a severely limited bandwidth. When AUVs explore deep water they have to be deployed without communication to any ground or boat station due to the severely limited communication range. These acoustic communication limitations make deploying a centralized control algorithm for multiple underwater vehicles highly challenging, necessitating the need for a decentralized control that is robust to communication drops and limited sensor measurements.

II. Background & Related Work

A. Problem Background

For a long time, researchers have been interested in studying the habits, patterns and survival tactics of different underwater animals. Life underwater is something that remains a mystery to this day as humans are incapable of fully exploring the depths of the oceans. Tracking animals such as sharks, turtles, and more recently whales, has allowed us to better observe animal behaviors previously unknown and to track migration patterns. With recent advancement in technology, robots have been used to overcome this problem to some extent. Thus far, underwater animal tracking has been limited to only one AUV, with the animal being the leader and the robot being the follower.

Distributed algorithms and control techniques have opened up new avenues to carry out underwater research. The use of multiple AUVs to track a single target could allow us to gather more information about the animal and its surroundings, provide redundancy of data collection, as well as to be able to better track large animal species. It is also of importance when tracking an animal that the following AUVs do not interfere with the animal and maintain a reasonable distance.

For our project, we were interested in using distributed techniques to track the movement of a shark in open water. The main difficulty in this task was keeping up with the speed of the shark which are capable of swimming at speeds of up to 31 mph or 13.85 meters per second. Another obstacle was dealing with the lack of consistent communication underwater without which it would be hard to maintain a robust formation around the shark.

B. Literature Review

The deployment of multi-Autonomous Underwater Vehicles is not a widely adopted practice due to the many challenges of these types of systems. With communication being the main complicating factor of deploying multiple AUVs, Sotzing and Lane propose a multi-agent mission architecture in order for the vehicles to be robust to communication break downs while still completing their overall mission. Using acoustic communication is highly prone to corrupt messages between vehicles and often times the vehicles will go out of range of each other for prolonged periods of time. In [1], the authors compare the performance of acoustic transmissions based on different aspects like quantization and cycle time on the frequency response of the system. They noted that to track high dynamic target, there needs to be a tradeoff

between quantization for low cycle time. The authors claim that "Oceanographic pursuit" is likely moving towards group autonomy as is already being done in surface and aerial vehicles. In [2], the authors implement a decentralized multi-agent structure where each vehicle independently does mission planning of the other agent when a message has not been successfully received after a set time frame. This prediction adds overhead to the computation of each vehicle, but allows for successful mission completion even when goal marking messages are not received. It is of note that Sotzing and Lane found that the mission was prone to failure in their experimental trials when prediction was not accurate.

Tracking of an animal can be describe most closely to be a leader-follower architecture where the shark is an autonomous leader, tagged with an acoustic pinger, and the pursuit robots are the followers. In [3], Xiang et al. explore a leader-follower based control algorithm for deploying a formation underwater robots for pipeline inspection. In this paper, follower AUVs follow a parallel path altering speed to provide separation. Simulations involve only three robots but have the advantage of only requiring communication links between the follower and leader and not communication between followers. Obstacle collision avoidance is not considered. Obstacle avoidance is taken into consideration in the work conducted by Lin et al in [4]. They deployed two REMUS 100 AUVs to track a tiger shark. Because of the unreliability of the acoustic communication between the shark and each AUV, they implemented a particle filter on the relative distance between each agent and the target. For the tracking control, they used two different controllers in order to plan a collision free path to a boundary around the shark and then another controller to circumnavigate the shark to collect data. In order to achieve collision avoidance between the agents circumnavigating the target, they implemented A* path planning on each agent.

The work done in [5] describes a leader-follower formation control algorithm for multiple underwater vehicles. The paper uses surface vehicles to conduct the experiment but enforces constraints that would be faced by underwater vehicles. Only the leader broadcasts its own position that is used by the followers to maintain a formation; the formation also rearranges itself if one of the followers is lost or a link breaks. Another interesting feature of this algorithm is that if the communication completely disrupts, the robots start acting as independent robots and carry out individual tasks. The algorithm can scale up with the number of robots; however, the only limitation is based on the range of the acoustic range of the leader.

Another key aspect of leader follower distributed control deals with maintaining consensus given information transmission failure. This is a likely event in an underwater environment as we have noted previously. In [6], Pan et al. shows sufficient conditions for the convergence of a distributed leader-follower control architecture given information packet drop off. Their results show that even with high probability of data loss, consensus is still met albeit with a sometimes significant time penalty. Furthermore, the leader-follower architecture in underwater robotics has to deal with redundancy of formation with loss of communication between robots. One approach taken by [7] uses a multiple pseudo-leader architecture. In this system there is only one true leader, but many pseudo leaders that maintain connections with subgroups of robots. This paper shows provable disturbance rejection properties of a H_2/H_8 feedforward controller compared to that of more typical LQR which exhibits considerable drift under communication errors.

Significant research has been done in this problem like in [8] where the authors used optimal control to deploy a network underwater to track an unknown target. Geometric traversals are used to track the coverage of the sensor network. The authors also employ dynamics of the underwater robots and sensors in this paper. The sensors are modeled as disks that have varying radius to account for the changing dynamics underwater. The control technique is also implemented and the results show that optimal control increases the performance of the sensor network. The optimal network are capable of generating Pareto optimal trajectories that minimize energy consumption.

A key challenge to the modeling of AUVs underwater are the dynamical uncertainties in the environment. Attempting to model these effects are computationally intensive and struggle when used on systems with many robots. In [9], the authors propose an adaptive PD controller scheme for multi-robot underwater systems. With this, they use a Lyapunov-like function combined with a 6-DOF simulation to prove stability characteristics of this controller. With only a model of the gravitation and buoyancy characteristics underwater they are able to derive this controller. They simulate their control algorithm on the Omni Directional Intelligent Navigator (ODIN), a 6-DOF underwater robot from the University of Hawaii.

A large majority of previous work on limited communication multi robot systems is in the application of coverage exploration. Mosteo et al work with limited communication over a large area by ensuring that no agent is out of range of any other agent in [10]. This method of deployment works by keeping the network connected by ensuring that messages can be chain passed to other robots that may need the information. However, this method assumes that if an agent is within communication range of another agent, there is reliable and high bandwidth communication between the two, which is not necessarily the case in the application underwater.

III. Problem Formulation

We set out to achieve three main goals for our project. Firstly, we wanted to model the dynamics of AUVs realistically so that the distributed techniques could be built and deployed in real scenarios. Instead of using simple single integrator dynamics, we employed full six degree of freedom dynamics that also included the hydrodynamics of the robots. Secondly, we wanted to implement a distributed controller for tracking of the sharks. The controller would allow the robots to stay a set distance away from the shark and maintain the maximum possible distance between one another for best coverage around the shark. Thirdly, low communication is a

major challenge underwater. We wanted to test cases where the robots lose communication between one another but still maintained their positions around the shark. For this, we used Extended Kalman Filters that would estimate the state of the robots around the shark when communication was lost.

A. Vehicle Dynamics

Many distributed controllers for multi-robot networks uses simplified dynamics, such as single integrator or holonomic drive robots. The dynamics of Autonomous Underwater Vehicles is more complicated due to the six degrees of freedom and hydrodynamic constraints. AUVs in development have adopted many different shapes and control methods. For example, the REMUS vehicle developed at the Woods Hole Oceanographic Institute utilizes an active ballast for depth control, a single thruster at the rear for propulsion, and pivoting fins for direction control. The REMUS has been successfully deployed for shark tracking missions, but lacks the high degree of maneuverability we desire for higher speed tracking and formation control. New AUVs have been in development that utilize a more drone like design, utilizing thrusters for all axes of control. For this project, we simulate the dynamics of the AUV built by Cornell University in 2019 for use in the AUVSI RoboSub competition, Figure 1.

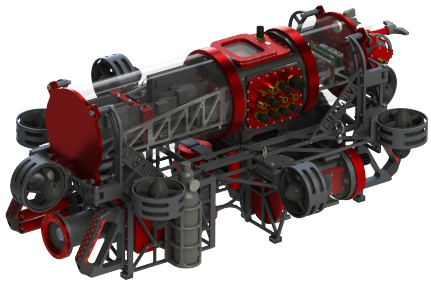

Fig. 1. Cornell University 2019 AUV, Odysseus

This AUV is fitted with eight *Blue Robotics* T200 thrusters for control, two *iDS* uEye cameras, three *Teledyne Marine* Reson hydrophones, a custom acoustic transducer, a *Microstrain* 3DM-GX4 -25 Inertial Measurement Unit (IMU), and a *Teledyne Marine* Pathfinder Doppler Velocity Log (DVL). For this project we will be working under the constraint that the acoustic transducer can transmit data at a rate of at least 100 bytes per second. We are also going to assume that the DVL is able to accurately provide the vehicle position at an accuracy of 200mm.

In order to simulate accurate dynamics of the vehicle, we designed a low level Linear Quadratic Regulator (LQR) controller for each vehicle to run onboard at a faster rate than the distributed formation controller. The physical properties of the vehicle that were used to develop the LQR controller are:

$$m = 31.75\,kg \tag{1}$$

$$V = 0.0193\,m^3 \tag{2}$$

$$C_d = 0.05 \tag{3}$$

$$\rho = 998.2\,\frac{kg}{m^3} \tag{4}$$

$$I = \begin{bmatrix} 2.03 & 0 & 0 \\ 0 & 2.13 & 0 \\ 0 & 0 & 0.7625 \end{bmatrix} \tag{5}$$

where m is the mass, V is the volume, C_d is the drag coefficient, ρ is the water density, and I is the moment of inertia tensor. The T200 thrusters are able to provide up to 5 kgf of thrust in the forward direction, and up to 4 kgf of thrust in the reverse direction. For the controller, we will be using thruster values interpolated from empirical data gathered from the thrusters at 16V, as seen in Figure 2.

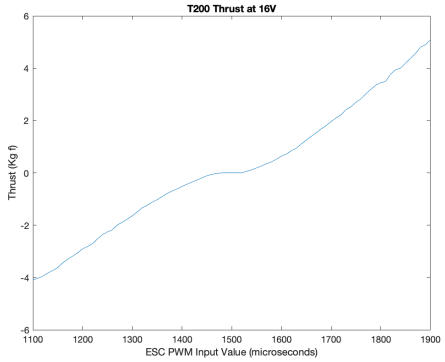

Fig. 2. *Blue Robotics* T200 thruster curve at 16V, gathered from empirical test data.

The mass matrix of the vehicle is as follows:

$$M = M_{rb} + M_A \tag{6}$$

$$M_{rb} = \begin{bmatrix} mI_{3x3} & 0 \\ 0 & I \end{bmatrix} \tag{7}$$

$$M_A = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_p & 0 & 0 & 0 & Y_r \\ 0 & 0 & Z_w & 0 & Z_q & 0 \\ 0 & 0 & 0 & K_p & 0 & 0 \\ 0 & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & 0 & 0 & N_r \end{bmatrix} \tag{8}$$

The hydrodynamic damping matrix is as follows:

$$D = D_l + D_q \tag{9}$$

$$D_l = diag\left(\begin{bmatrix} X_u & Y_v & Z_w & K_p & K_q & N_r \end{bmatrix}\right) \tag{10}$$

$$D_q = diag\left(\begin{bmatrix} X_{uu} & Y_{uu} & Z_{ww} & K_{pp} & K_{qq} & N_{rr} \end{bmatrix}\right) \tag{11}$$

Where the hydrodynamic constants are: $X_u, Y_v = 0$, $Z_w = 50$, $K_p, K_q = 0$, $N_r = 10$, $X_{uu} = 85.625$, $Y_{vv} = 80$, $Z_{ww} = 150$, $N_{rr} = 15$, and $K_{pp}, K_{qq} = 0$.

We define the vehicle state with positive x being forward and positive z being up in depth. θ, ϕ, and ψ are the roll, pitch, and yaw of the vehicle respectively.

$$x = [x \quad y \quad z \quad \theta \quad \phi \quad \psi \quad \dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\theta} \quad \dot{\phi} \quad \dot{\psi}]^T \tag{12}$$

Lastly, the control input distances (m) from the center of buoyancy are defined in a matrix:

$$L = \begin{bmatrix} 1.0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 01.0 & 1.0 & 0 & & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 0 & 0 & 0.0222 & 0.0222 & 0.2169 & 0.2046 & -0.2046 & -0.2169 \\ 0.0302 & 0.0302 & 0 & 0 & 0.2201 & -0.3705 & -0.3705 & 0.2201 \\ -0.3205 & 0.3205 & 0.5178 & -0.3078 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

For simplicity, we are assuming that the Coriolis effects are small and can be interpreted as $C = I_{6x6}$. The linearized system matrices can then be approximated as:

$$A = \begin{bmatrix} 0_{6x6} & I_{6x6} \\ 0_{6x6} & -M^{-1}(C+D) \end{bmatrix} \tag{14}$$

$$B = \begin{bmatrix} 0_{6x8} \\ M^{-1}L \end{bmatrix} \tag{15}$$

$$C = \begin{bmatrix} I_{6x6} & 0_{6x6} \end{bmatrix} \tag{16}$$

$$D = 0 \tag{17}$$

With these linearized system dynamics matrices, we solve the continuous Ricatti equation and obtain the LQR K matrix for our controller. The performance of the resulting LQR controller can be seen in Figure 3. It can be seen that when two states (x and y) change, this also adversely affects the other states as well. Moving in x and y causes a temporary drop in depth and slight deviations in roll, pitch, and yaw. This is physically reasonable to what is seen on a real vehicle in the water. The vehicle is able to successfully achieve the new desired state over approximately 20 seconds.

B. State Estimation

In order for our system to work in a limited communication environment, we must deal with the possibility of loss of communication between robots. Each robot transmits its global position to be received by the other AUVs. This information updates the state estimates of the robot such that it can predict the position of the other robots and position around the shark to ensure we have distributed and equitable coverage of the target.

In this project, we assume that the robots use an acoustic pinger tagged to the shark's dorsal fin to triangulate their relative distance to the shark. They broadcast their own positions to the other robots so that they can optimally adjust their positions around the shark. Each robot keeps an estimate of each of the other robots. This additional computation onboard adds computational time to the entire system, which increases with the number of robots in the network.

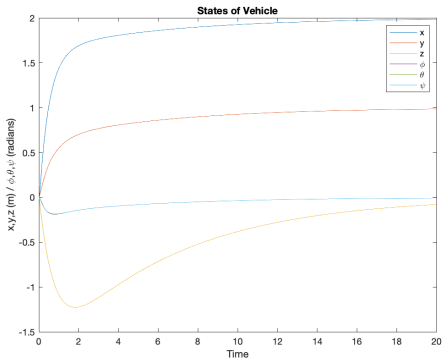

Fig. 3. States of the vehicle using the LQR controller with a reference command of $x = 2$, $y = 1$.

To estimate the state of the robots that lose communication with other robots while tracking the shark, we employ an Extended Kalman Filter (EKF). This filter was chosen due to the nonlinear dynamics of the AUVs. The filter represents the posterior state belief at t, $bel(x_t)$, with a Gaussian distribution with mean μ_t and covariance σ_t. When a Gaussian prior is propagated through non-linear dynamics, there is no guarantee that the results are Gaussian as well. Due to this, the dynamics are linearized around a point that is the most likely argument of each function, namely the mean. Propagating through linearized functions with a linearization point far away from the means results in the accumulation of error.

When there is successful communication between the robots and a new measurement of a robots state is received, both the predict and update steps of the EKF are executed to update the estimate of the robot. When a robot either stops transmitting data or there are errors receiving the correct data, only the predict step of the EKF is executed to best predict how the other robots are moving. This new state estimate is then used to calculate the optimal positions around the target. The predict and update equations are below, where $f()$ is the nonlinear dynamics function and $g()$ is the nonlinear measurement function.

Predict:

$$\begin{aligned} \mu_{t|t-1} &= f(\mu_{t-1|t-1}) \\ \Sigma_{t|t-1} &= A_{t-1}\Sigma_{t-1|t-1}A_{t-1}^T + Q_{t-1} \end{aligned} \tag{18}$$

Update:

$$\begin{aligned} K_t &= \Sigma_{t|t-1}C_t^T(C_t\Sigma_{t|t-1}C_t^T + R_t)^{-1} \\ \mu_{t|t} &= \mu_{t|t-1} + K_t(y_t - g(\mu_{t|t-1}, u_t)) \\ \Sigma_{t|t} &= \Sigma_{t|t-1} - K_tC_t\Sigma_{t|t-1} \end{aligned} \tag{19}$$

For the purposes of our project, in order to have a difference in the nonlinear dynamics function used by the simulation

to propagate the dynamics of each vehicle forward in time and the dynamics function used by the state estimator, the propagation time step used to propagate the dynamics in the estimator is much larger than that used for dynamic propagation. This change between the two dynamic functions is in an effort to make the simulator more realistic. Using the same timestep causes the predict step to be nearly perfect in the absence of noise, which is unrealistic for a physical system.

C. Formation Control

To position the robots optimally around the shark, we implement a scoring method to find the optimal positions of each robot around the target. Each robot knows its own relative position to the shark and has an estimate of the location of all the other robots. With this information, each robot calculates its optimal position around the shark by giving a score to every position at the desired distance around the shark. This score is calculated by summing the distances from the robots current position to the optimal position it is checking, and the smallest distance of all the other robots position estimates to the tracking circle around the shark. The lowest scoring position is assumed to be the optimal point on the tracking circle for that robot to track to. This scoring is done by each robot in the network with its respective state estimates for the other robots.

When all the robots agree on the same optimal positions for all the robots, the system is stable. This scoring method works successfully even when not all the robots agree on the same optimal positions. When there is variance in the calculated optimal positions between the robots, one or more of the robots may move to a location on the tracking circle that was not anticipated by the other robots, but this is easily adapted for in the next time step by all the robots. This causes the robots to "rotate" in formation around the shark, but formation is successfully held around the target.

Algorithm 1 Formation Control Algorithm

Input: Relative position to the shark; broadcasted position of other robots
Output: Optimal position around the shark
  for $iteration = 1, 2, \ldots$ **do**
    for $robot = 1, 2, \ldots, N$ **do**
      Calculate the desired positions around the shark
      Calculate distance to desired positions and give them scores corresponding to the distance
      opt_pos = *des_pos*(min(score))
    end for
  end for

IV. Experimental Results & Simulations

Our simulation of the multi AUV system tracking a target was developed using *MATLAB*. The nonlinear vehicle dynamics described in Section III-A were simulated using *ode45* with time steps of $\delta t = 0.001s$ for dynamic propagation and $\delta t = 0.05s$ for the dynamics function $f()$ used in state estimation. In our simulation, the robots broadcast their positions to the other robots every one second. The target shark was able to be controlled in the simulation by user input using the computer's arrow keys. This allowed for us to easily find the limitations of the tracking capabilities of the network.

When there is no communication loss within the robot network, the robots successfully maintain a very stable formation around the shark. The optimal positions around the shark for each robot does not change in time and each robot maintains its own relative position around the shark. This is seen in Figure 4 where the shark is represented in green and the following robots are seen in blue. The shark in this simulation maintained a constant speed throughout the simulation but changed directions.

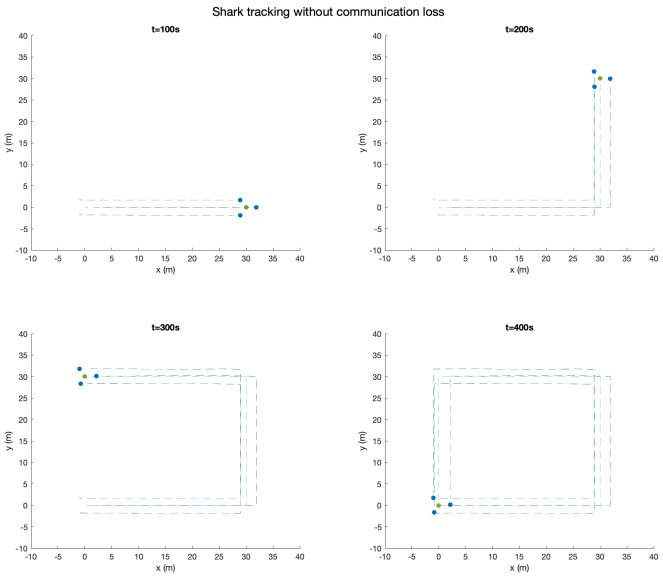

Fig. 4. Tracking of the shark (green) by three robots (blue) with constant target speed.

For this simulation, we defined communication loss as a robot that fails to successfully transmit its position to the other robots. For comparison, we ran the same leader path simulation as above but added in two different times of communication loss. One robot stops successfully transmitting data at 50 seconds and a second at 200 seconds. The results of this simulation can be seen in Figure 5. Whenever a robot loses communication, all the robots appear to rotate in formation around the shark. This rotation behavior appears to continue until a steady state formation is once again found. This behavior could arise because, once a robot loses communication and all the robots must rely on their state estimates of that robot to find the optimal positions, if any of the robots try to go to a different optimal position around the shark, all the other robots must adapt to the change in formation. This change continues until all the robots once again agree on the same optimal positioning around the shark. In our simulations, all three robots were able to successfully maintain a constant

distance around the shark in a relatively stable formation.

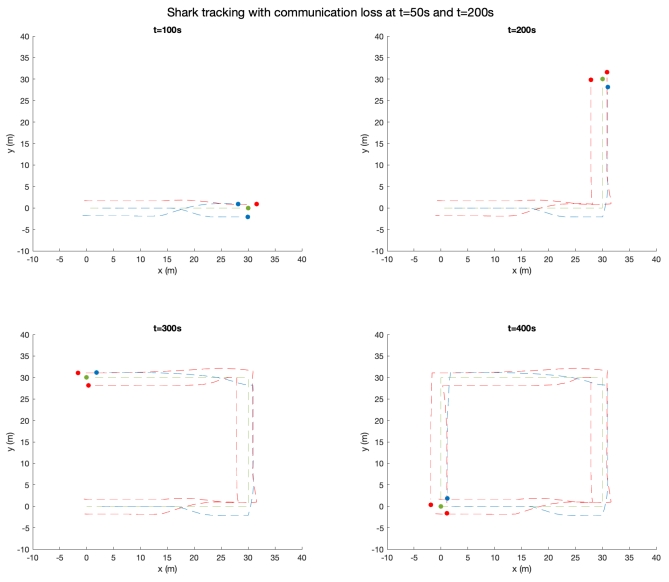

Fig. 5. Tracking of the shark (green) by three robots (blue) with constant target speed with two instances of communication loss, at $t = 50s$ and $t = 200s$.

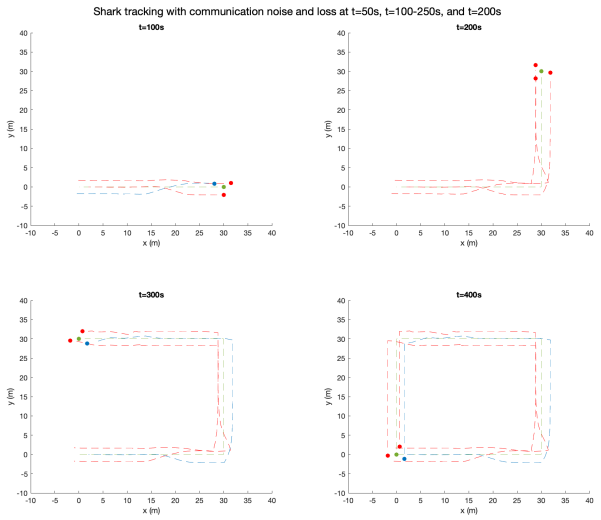

Fig. 6. Tracking of the shark (green) by three robots (blue) with constant target speed with two instances of communication loss, at $t = 50s$, $t = 100-200s$, and $t = 200s$. Gaussian noise is also added to the data transmitted between the robots.

To test the robustness of our formation controller in the presence of more communication loss as well as measurement noise between the robots, we ran the same three robot tracking controller with additional communication loss of the third robot from $t = 100 - 250$ seconds, such that there is 50s where all three robots are not communicating with each other. There was also added Gaussian noise in the data transmitted between the robots, with a mean of zero and a variance of 0.01. The results of this simulation can be seen in Figure 6.

Even when all three robots lose communication with each other, the formation is successfully held among all three robots. In the presence of noise in the transmitted data, the EKF was successfully able to filter out most of this noise. However, it can be seen in the paths of the robots that there is slight wavering of the robots about their nominal optimal paths. This wavering does not greatly affect the robots agreeing on the same optimal positions around the shark when the communication network between the robots is not changing. However, as can be seen in the path of the robots at the top of the trajectories, two of the robots suddenly changed paths around $t = 260s$. This change in the path happens right after a robot comes back into communication with the network. Two of the robots appeared to try to go to a different formation than the third robot, but the formation was able to quickly recover after approximately $15s$. This behavior was also seen in other simulations where robots reestablished communication with the network, even without noise in the communication. This behavior of optimal formation disagreement is likely due to the state estimate of the robot reestablishing communication being updated with the latest measurement. The added noise in the measurement did not greatly affect the overall performance of the system.

We ran the same simulation again but instead of the shark travelling at constant speed and not changing directions much, we controlled the shark to rapidly change both speed and direction. Throughout the simulation, all three robots were able to maintain a formation well around the shark, however, with the rapidly changing speed of the shark, some robots would get closer than the desired distance and others further. This change in distance often happened when the shark quickly accelerated and the robot dynamics limited them by not allowing the robots to change speeds as quickly as the shark did. With all the directional and speed changes of the target, we observed the formation rotating a lot more around the shark, especially when more robots lost communication with the network. Though the overall formation of the network was less stable with a rapidly changing target and more robots not communicating, all the robots were able to mostly stay in formation and no collisions were observed.

The typical cruising speed of a shark is approximately 2 m/s, but they can swim up to 14 m/s. The previous simulations above were conducted at relatively low target speeds, ranging from 0.75 m/s to 2 m/s. In Figure 8, we simulated a shark that travelled faster than cruising speed. When the shark exceeded 3 m/s the following robots were not able to keep up with the shark as the robots maximum speed is not as fast as that of the sharks. This caused our robots to lose formation around the shark. However, when the shark stopped, the robots were able to successfully recover formation around the shark to continue their leader-follower tracking. Our approach does not have any collision avoidance so, in regaining formation around the shark one of the robots (the one which had lost communication), collided briefly with the shark.

Our formation controller works for an arbitrary number of

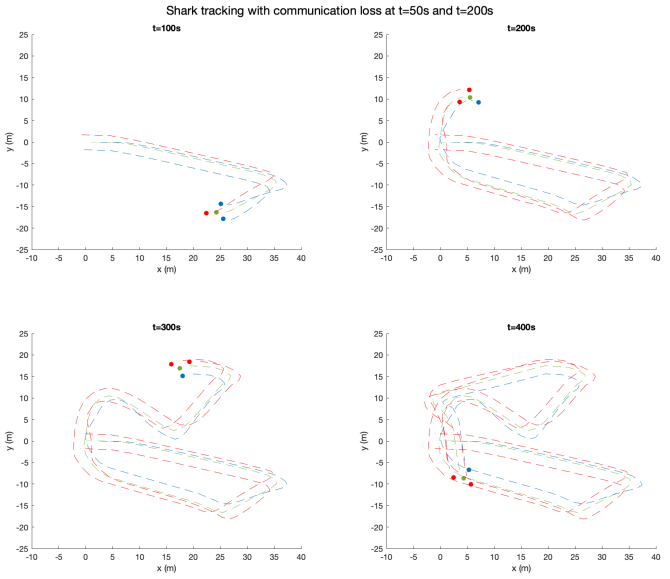

Fig. 7. Tracking of the shark (green) by three robots (blue) with rapidly varying shark speed and direction with two instances of communication loss, at $t = 50s$ and $t = 200s$.

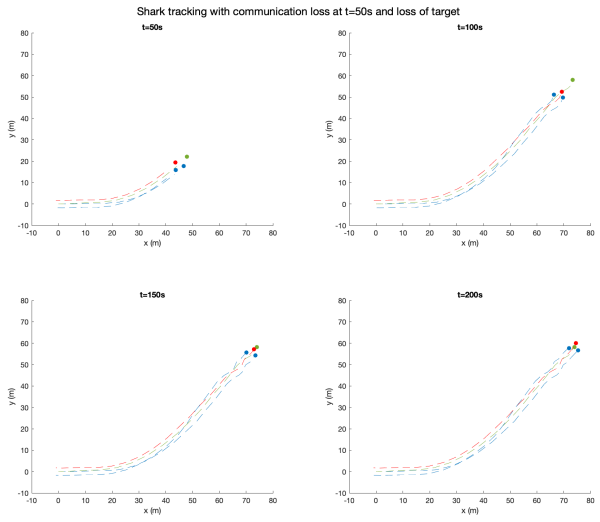

Fig. 8. Tracking of the shark (green) by three robots (blue) with rapidly varying shark speed and direction with two instances of communication loss, at $t = 50s$ and $t = 200s$.

robots, and we have successfully simulated this for up to six robots. With an increasing number of robots, the formation appears to lose stability around the shark. There is more observed oscillations in each of the robots about their optimal position and there is also more rotation around the target. For most of our simulations, the formation is able to be successfully held for shark speeds under 2.2 m/s. However under certain circumstances, a failure in the formation control has been observed, such as in Figure 9. This simulation had up to three of the five robots without communication at a time, and at $t = 300s$ one robot goes back online while another robot loses communication. Shortly after this communication flip two of the robots appear to have converged to the same optimal position around the shark. This disagreement in optimal positioning is normally able to be adapted for by the other robots, as seen in the previous simulations. However, in this simulation, the formation failed and the robots were unable to recover. The exact cause for this formation failure is unknown and we have not been able to successfully recreate it.

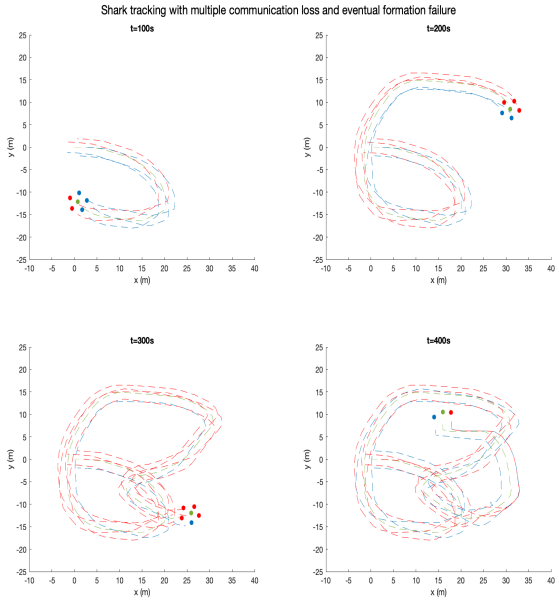

Fig. 9. Tracking with five robots and communication loss at $t = 50 - 300s$, $t = 100s$, $t = 200s$, and $t = 300s$. Around $t = 310s$, the formation is lost and is unable to be recovered.

V. Conclusions

Our leader-follower formation control is successfully able to allow Autonomous Underwater Vehicles to track a shark. For a shark travelling at cruising speeds the formation is able to adapt to communication losses in the network by keeping track of state estimates of each of the robots. The formation stability is lost for increasing speed of the target and for an increase in the number of tracking robots. When formation is lost, the robots are often able to recover and continue successful tracking. The method scales with a larger number of robots but the computation time increases as well.

These results lead to many possible followups and extensions. We would particularly like to extend this implementation to work in a 3D environment. We would also like to integrate orientation of the AUVs into the tracking of the shark in order to best utilize the mobility of the vehicles. Another extension would be collision avoidance in order to guarantee there is no undesirable collisions with either the shark or between the robots. Further analysis can be carried out to determine the optimal number of robots that give the best performance when

tracking a shark including the optimal distance from the shark given our sensor and communication capabilities.

ACKNOWLEDGMENT

We would like to thank Professor Mac Schwager for teaching a highly engaging course and allowing us the opportunity to explore the applications of multi-robot systems. We would also like to thank Joe Vincent and Trevor Halsted for their support in this class.

REFERENCES

[1] B. Reed, J. Leighton, M. Stojanovic, and F. Hover, "Multi-vehicle dynamic pursuit using underwater acoustics," in *Robotics Research*. Springer, 2016, pp. 79–94.

[2] C. Sotzing and D. Lane, "Improving the coordination efficiency of limited communication multi-auv operations using a multi-agent architecture," *J. Field Robotics*, vol. 27, pp. 412–429, 07 2010.

[3] X. Xiang, B. Jouvencel, and O. Parodi, "Coordinated formation control of multiple autonomous underwater vehicles for pipeline inspection," *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 3, 2010. [Online]. Available: https://doi.org/10.5772/7242

[4] Y. Lin, J. Hsiung, R. Piersall, C. White, C. Lowe, and C. Clark, "A multi-autonomous underwater vehicle system for autonomous tracking of marine life: A multi-auv system for autonomous tracking of marine life," *Journal of Field Robotics*, 08 2016.

[5] D. Edwards, T. Bean, D. Odell, and M. Anderson, "A leader-follower algorithm for multiple auv formations," in *2004 IEEE/OES Autonomous Underwater Vehicles (IEEE Cat. No. 04CH37578)*. IEEE, 2004, pp. 40–46.

[6] Y.-J. Pan, H. Werner, Z. Huang, and M. Bartels, "Distributed cooperative control of leader–follower multi-agent systems under packet dropouts for quadcopters," *Systems Control Letters*, vol. 106, pp. 47 – 57, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167691117301056

[7] P. Millán, L. Orihuela, I. Jurado, and F. R. Rubio, "Formation control of autonomous underwater vehicles subject to communication delays," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 2, pp. 770–777, March 2014.

[8] K. A. Baumgartner, S. Ferrari, and A. V. Rao, "Optimal control of an underwater sensor network for cooperative target tracking," *IEEE Journal of Oceanic Engineering*, vol. 34, no. 4, pp. 678–697, 2009.

[9] S. P. Hou and C. C. Cheah, "Can a simple control scheme work for a formation control of multiple autonomous underwater vehicles?" *IEEE Transactions on Control Systems Technology*, vol. 19, no. 5, pp. 1090–1101, Sep. 2011.

[10] A. R. Mosteo, L. Montano, and M. G. Lagoudakis, "Multi-robot routing under limited communication range," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1531–1536.