Trajectory Planning and Control of Bathy-drone: A Drone Towing a Boat equipped with Sonar for Bathymetry Mapping

Andres Pulido, Antonio Diaz, Andrew Ortega, Peter Ifju, Jane Shin

October, 2022

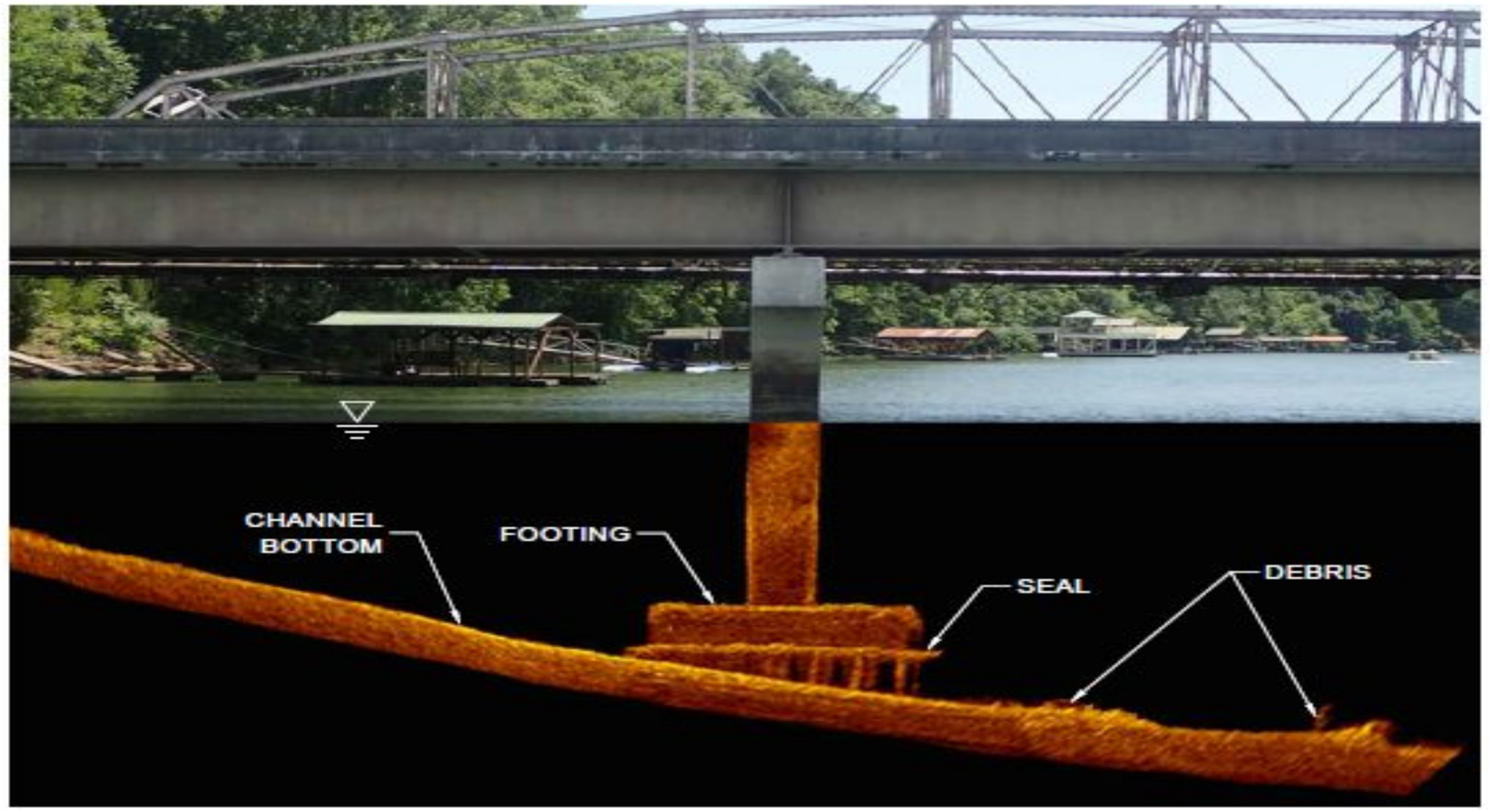
UF UNIVERSITY of FLORIDA

SCITECH

APRILab ACTIVE PERCEPTION AND ROBOT INTELLIGENCE LAB

Bathymetry: Potential Applications

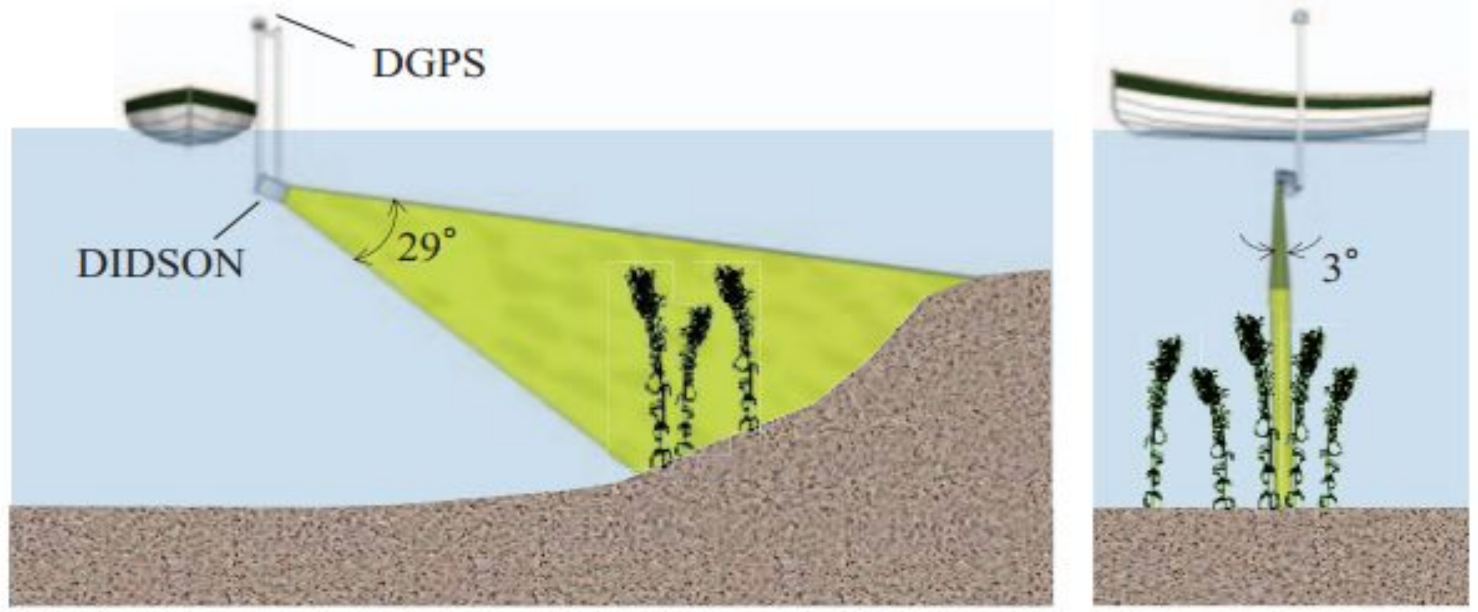
Artificial object inspection (pipes, bridges)

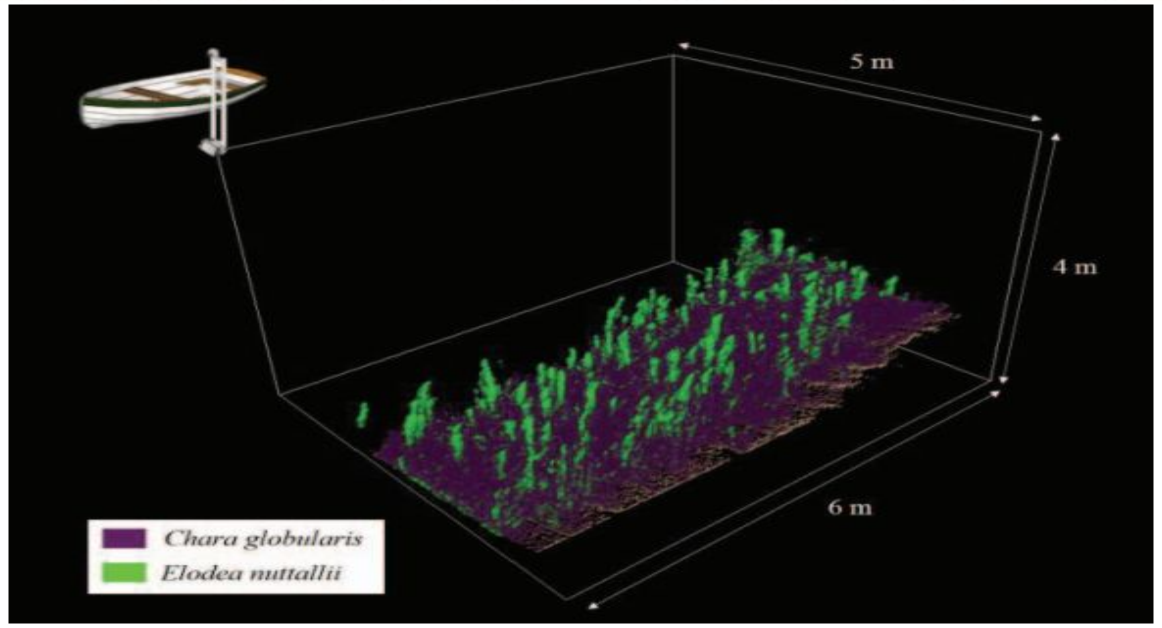


Source: Tennessee Department of Transportation

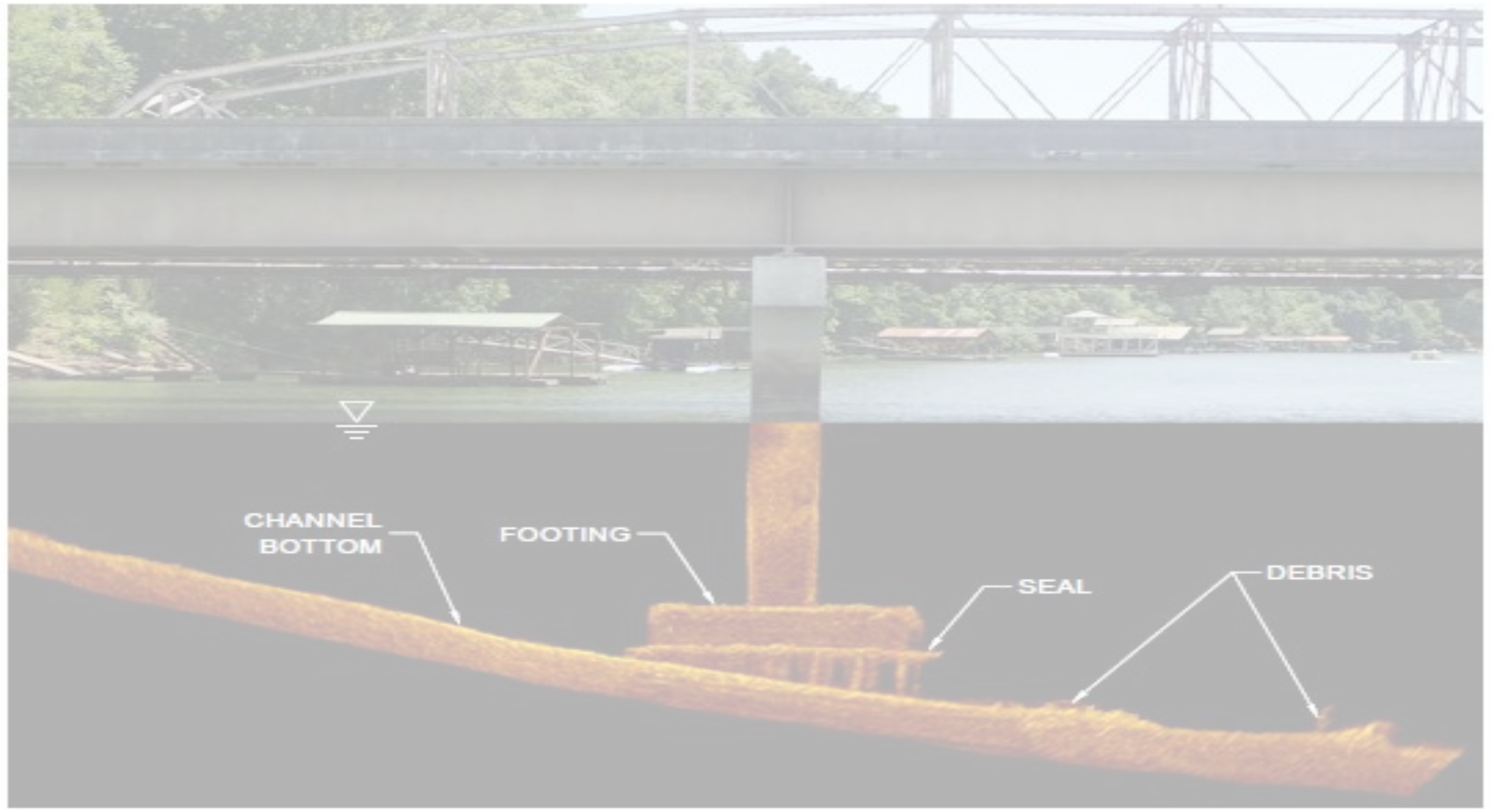
Bathymetry: Potential Applications

- Artificial object inspection (pipes, bridges)





[2]

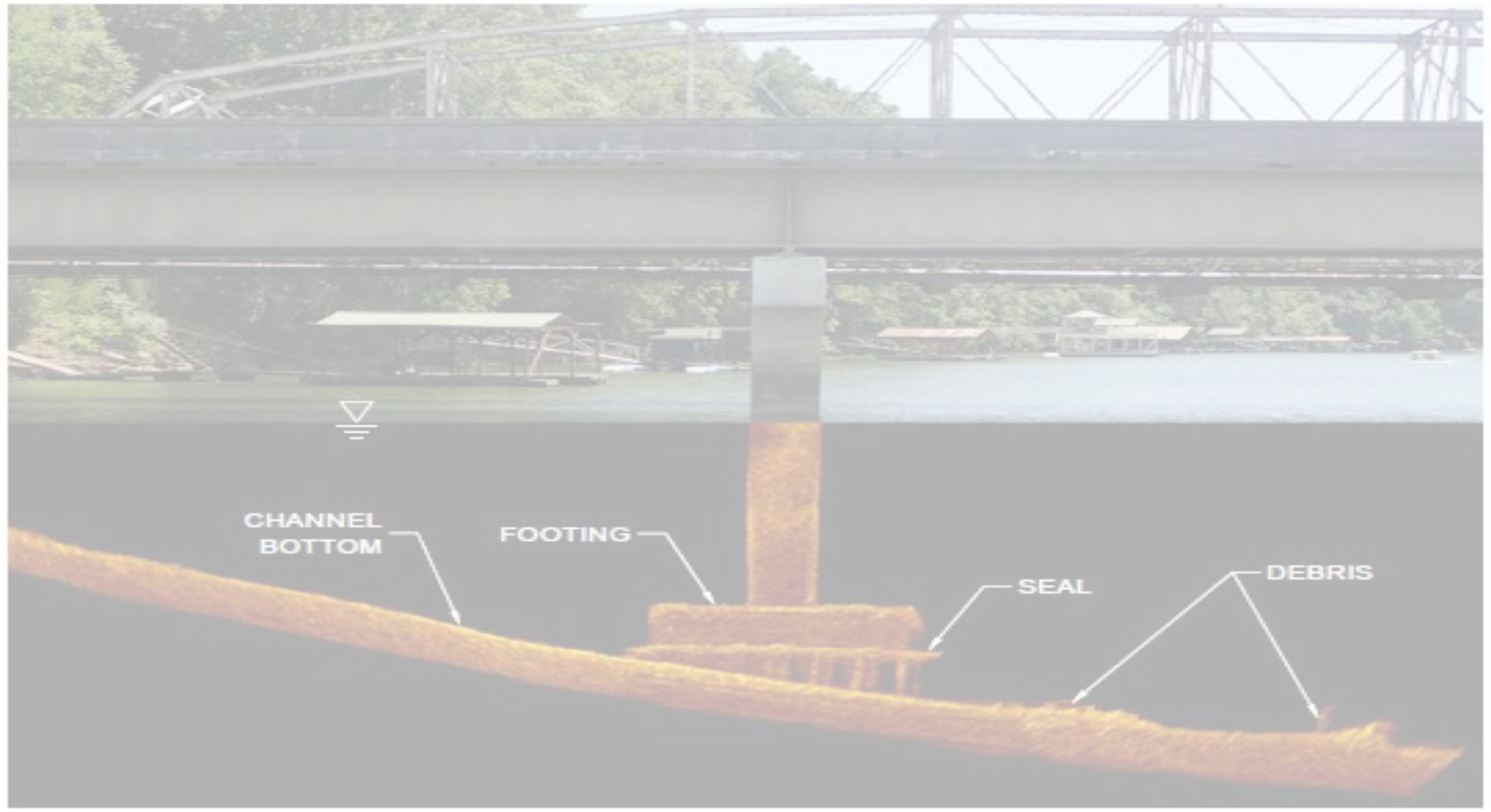


Source: Tennessee Department of Transportation

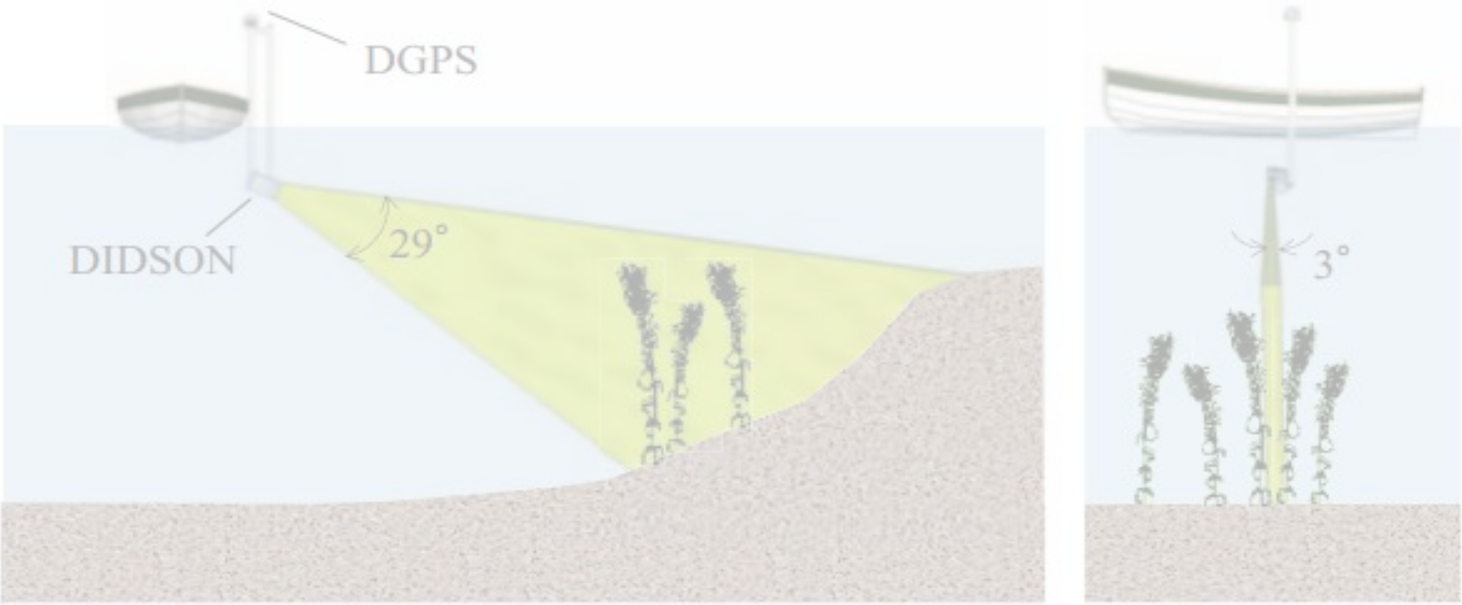
Survey of aquatic plants

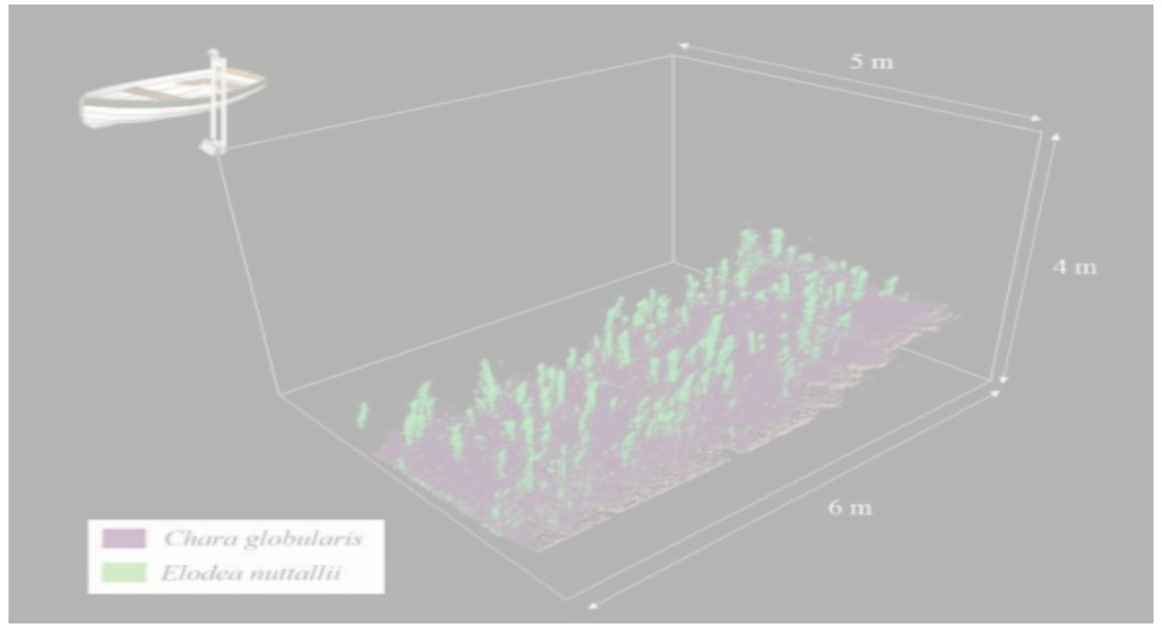
Bathymetry: Potential Applications

- Artificial object inspection (pipes, bridges)



Source: Tennessee Department of Transportation

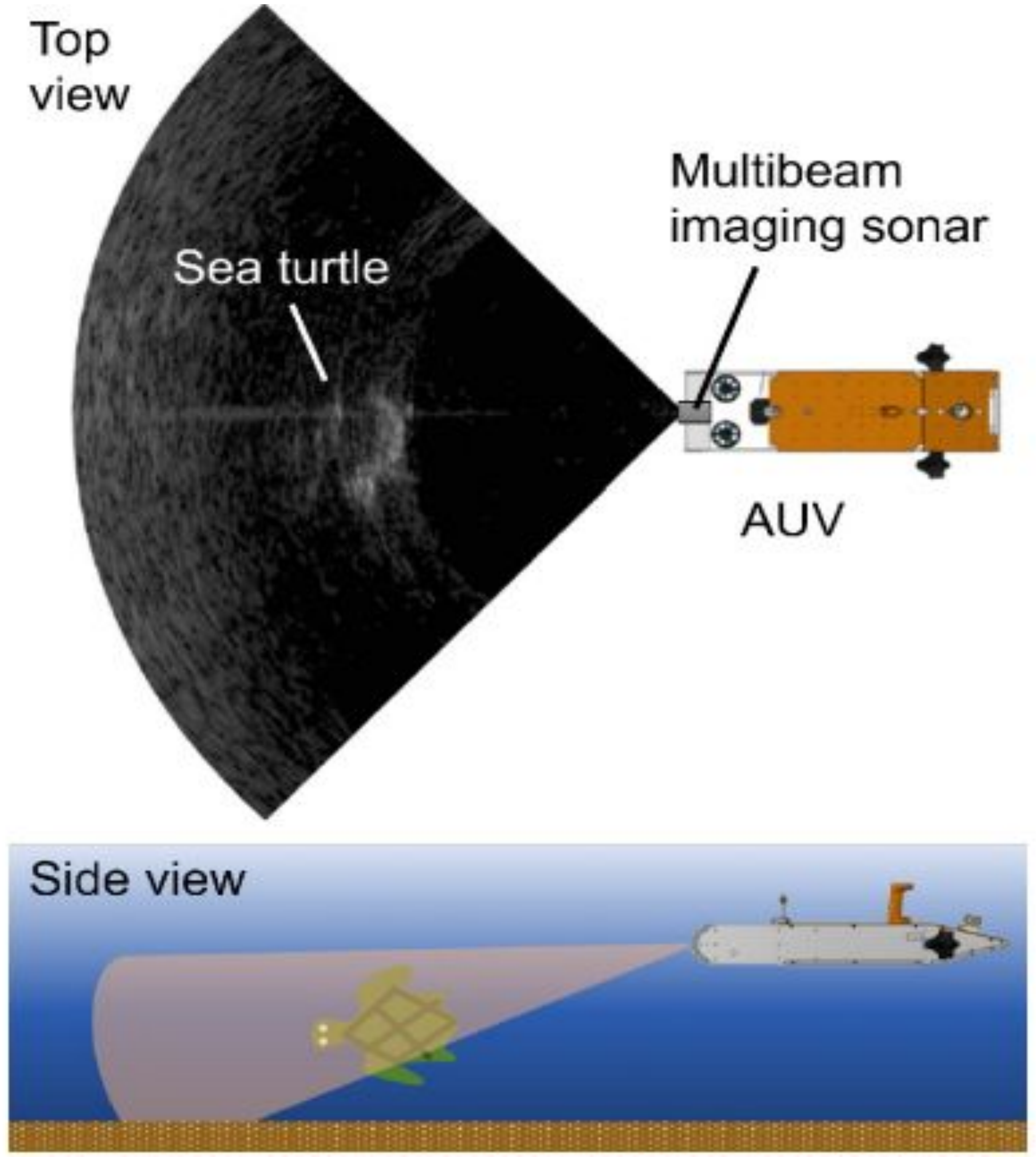




[2]

- Survey of aquatic plants

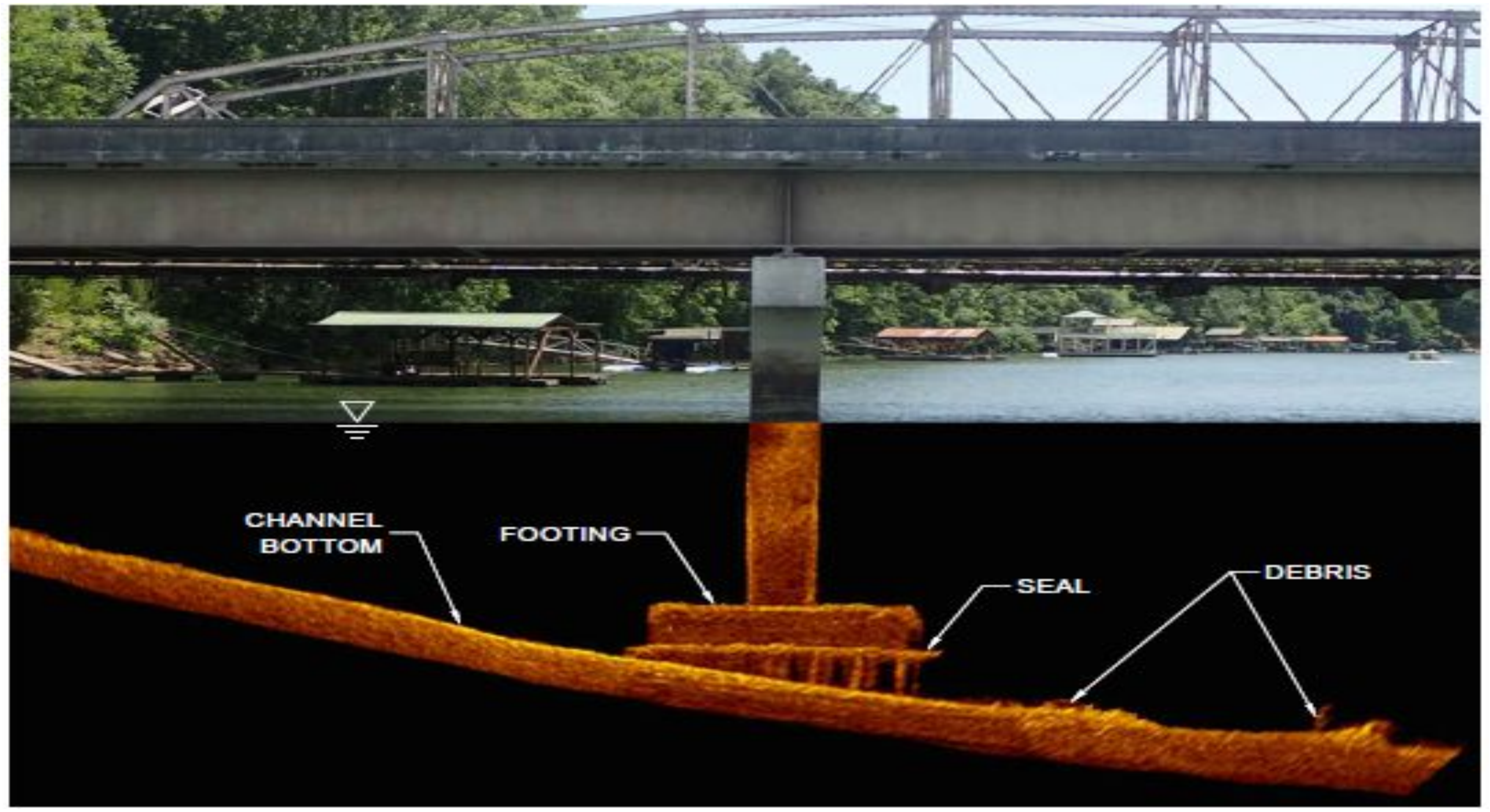
Survey of marine life



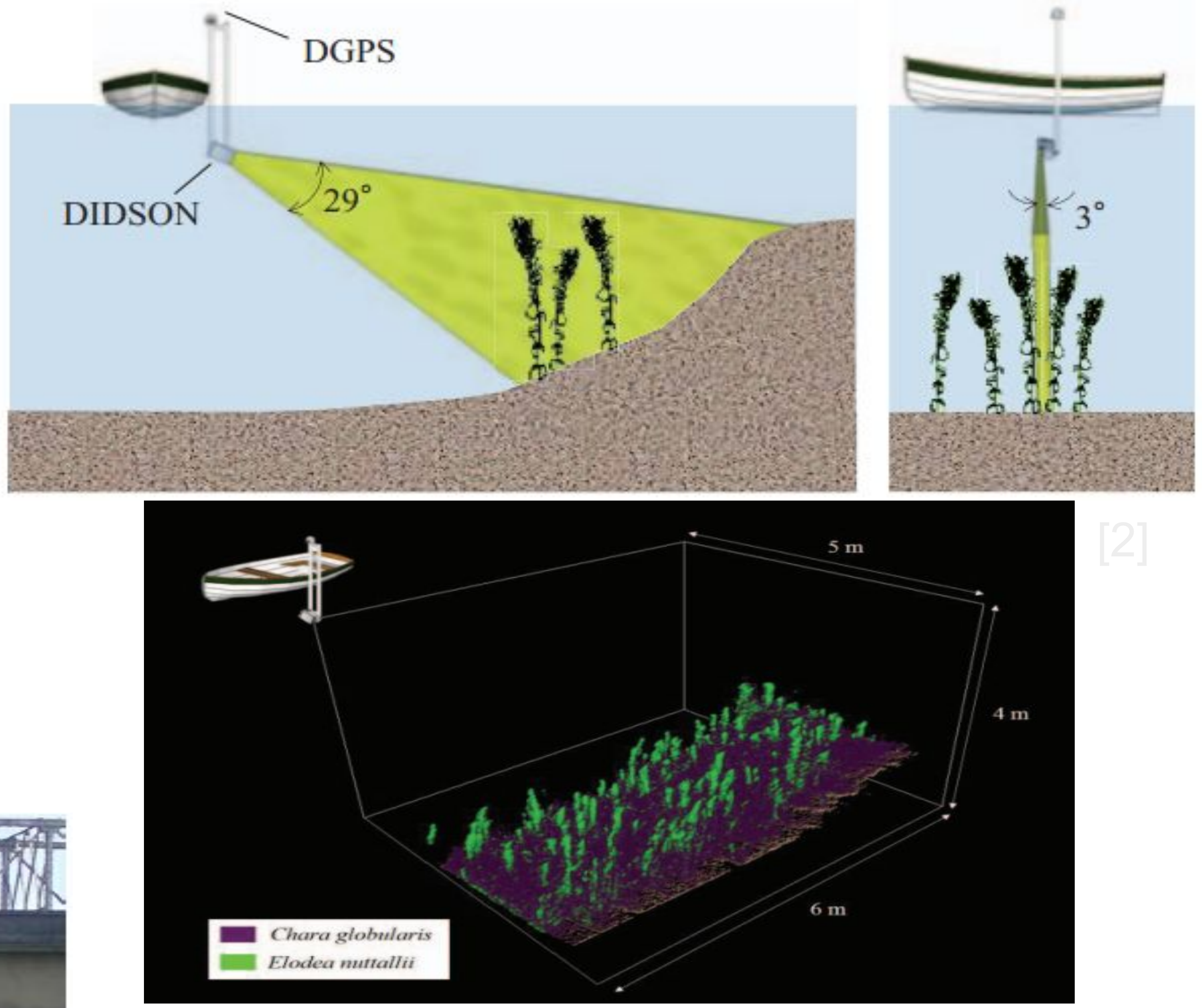
[3]

Bathymetry: Potential Applications

Artificial object inspection (pipes, bridges)



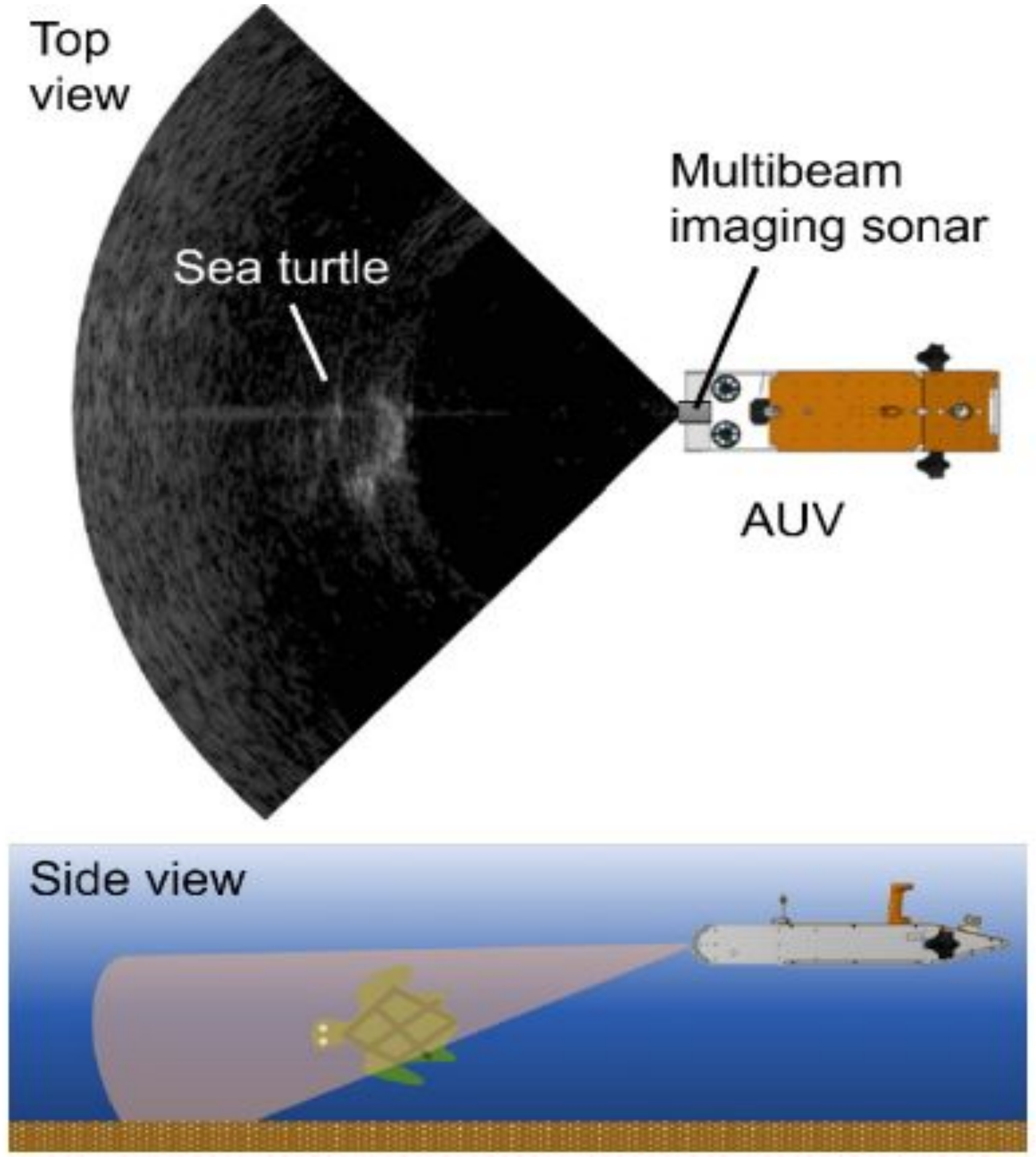
Source: Tennessee Department of Transportation



[2]

Survey of aquatic plants

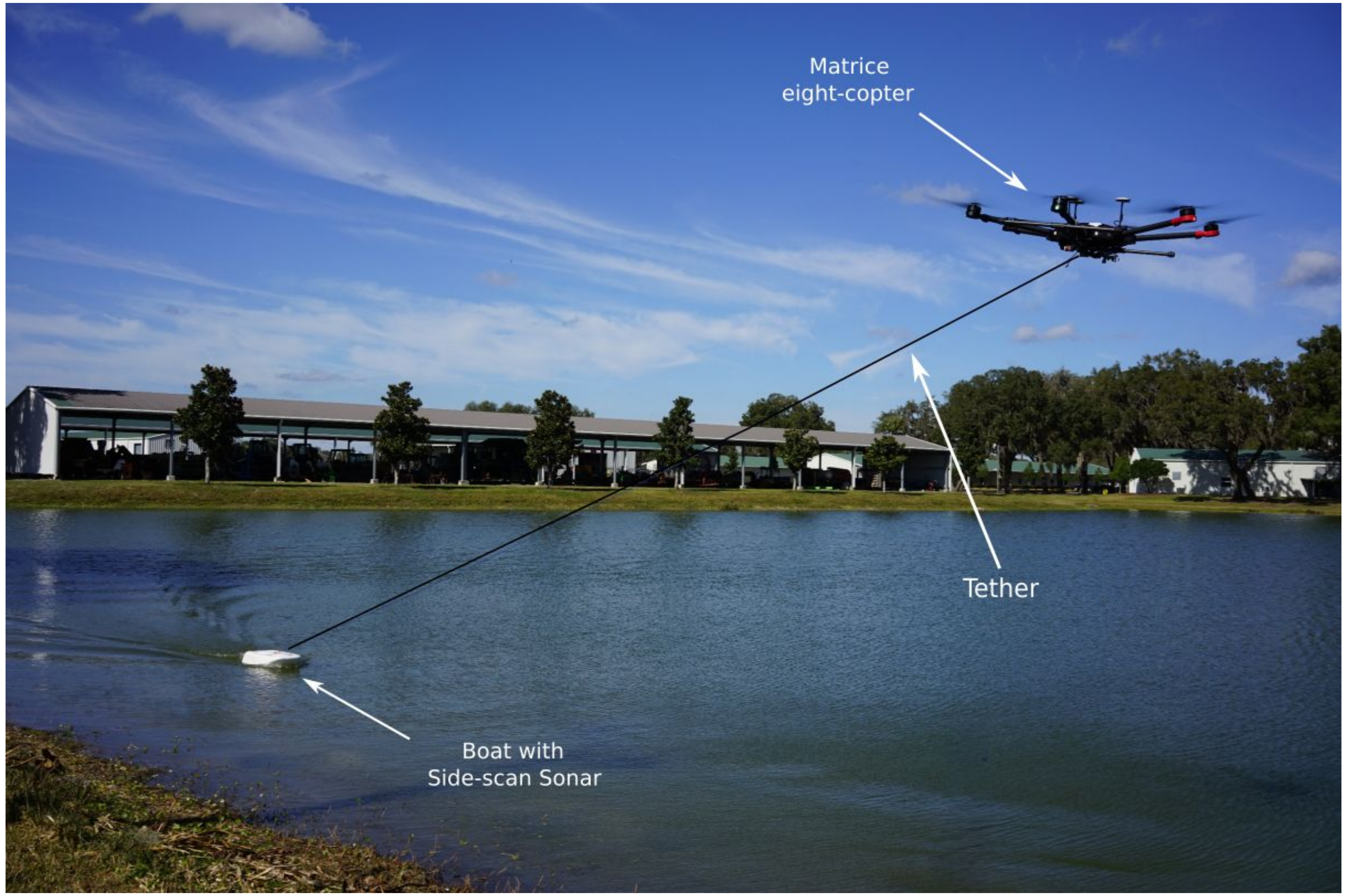
Survey of marine life



[3]

The Bathy-drone

- Autonomous **drone towing a tethered boat** equipped with a sonar
- Can be flown to the survey location
- No propulsion system on boat
- Can traveling at speeds of 0-15 mph
- The boat houses a low-cost commercial off-the-shelf recreational fish-finder and a downscan sonar



ACTIVE PERCEPTION AND ROBOT INTELLIGENCE LAB

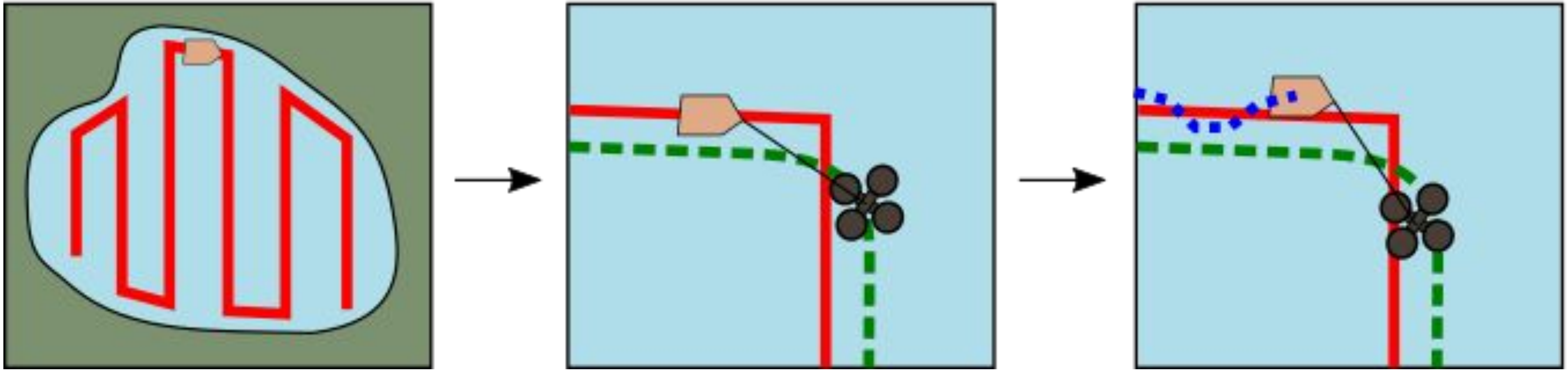
The Bathy-drone

Objective

- develop a G&C system that can handle the tethered dynamics such that the onboard sensor's field-of-view fully **covers the region of interest**

- Need to solve the following three components:

Boat Path Planning | Boat Trajectory Planning | Trajectory Tracking Control



SPARSE POINT CLOUD GENERATION AND AUTOMATIC OBJECT DETECTION USING BATHY-DRONE

The algorithm consists of two stages:

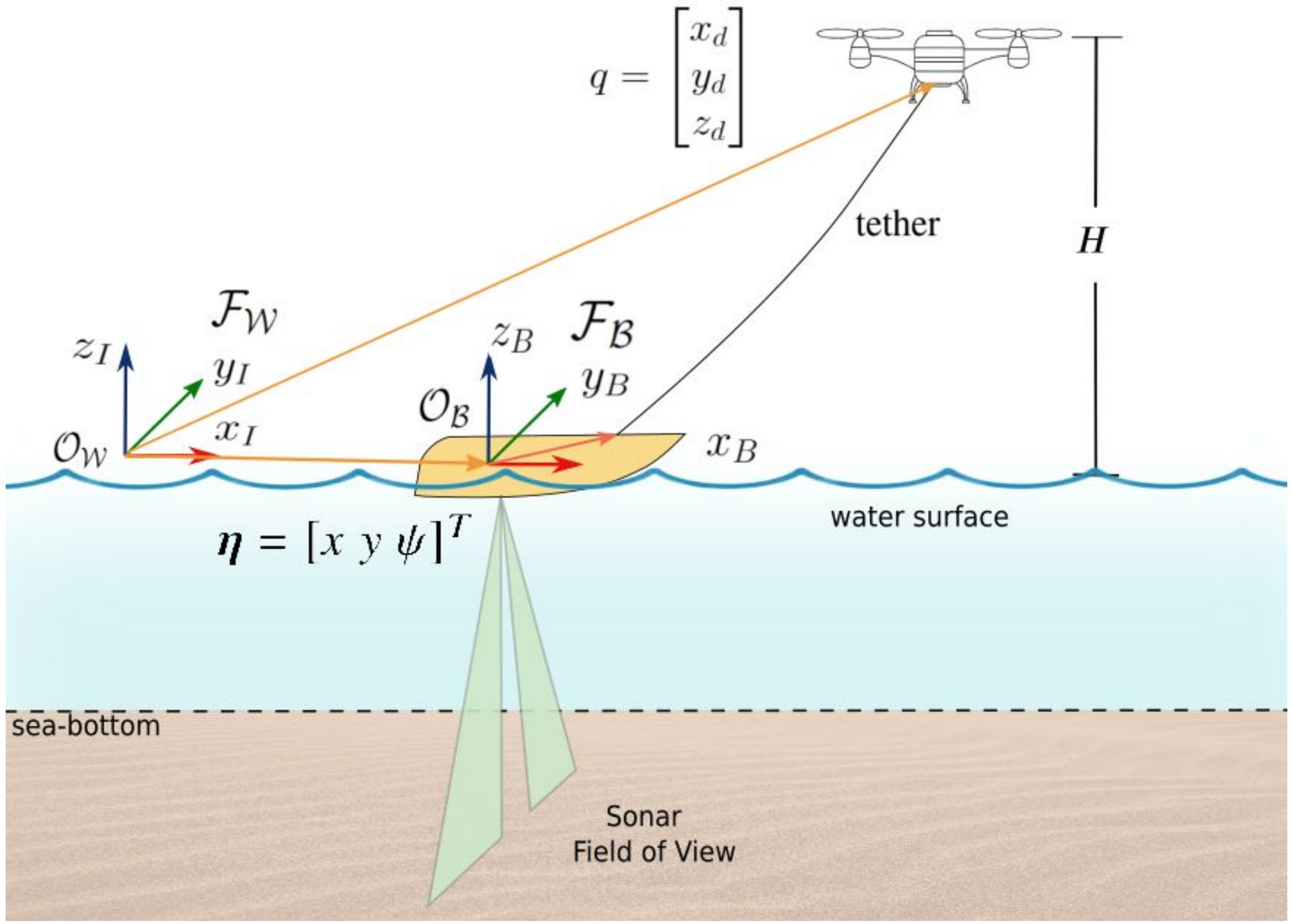
(1) Dynamic model
(2) Tracking Control
(3) Trajectory Planning
(4) Path Planning

Dynamic Model

Dynamics Model of Bathydrone

ACTIVE PERCEPTION AND ROBOT INTELLIGENCE LAB

- The hydrodynamics model of the boat getting pulled by the tether can be derived by with a FBD
- Tension force is calculated by the pose difference between boat and drone



Mass matrices — Coriolis matrices — Drag matrix — Tension Force of tether

$$\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\boldsymbol{M}_{A}\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_{A}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} = \boldsymbol{\tau}$$

Dynamics Model of Bathydrone

Mass matrices — Coriolis matrices — Drag matrix — Tension Force of tether

$$\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\boldsymbol{M}_{A}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{A}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{hydrodynamic forces}} = \boldsymbol{\tau}$$

Adding the rigid-body and hydrodynamic forces matrices:

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix}$$

$$C(v) = \begin{bmatrix} 0 & 0 & -mx_g r - mv + Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & mu - X_{\dot{u}} u \\ mx_g r + mv - Y_{\dot{v}} v - Y_{\dot{r}} r & -mu + X_{\dot{u}} u & 0 \end{bmatrix}$$

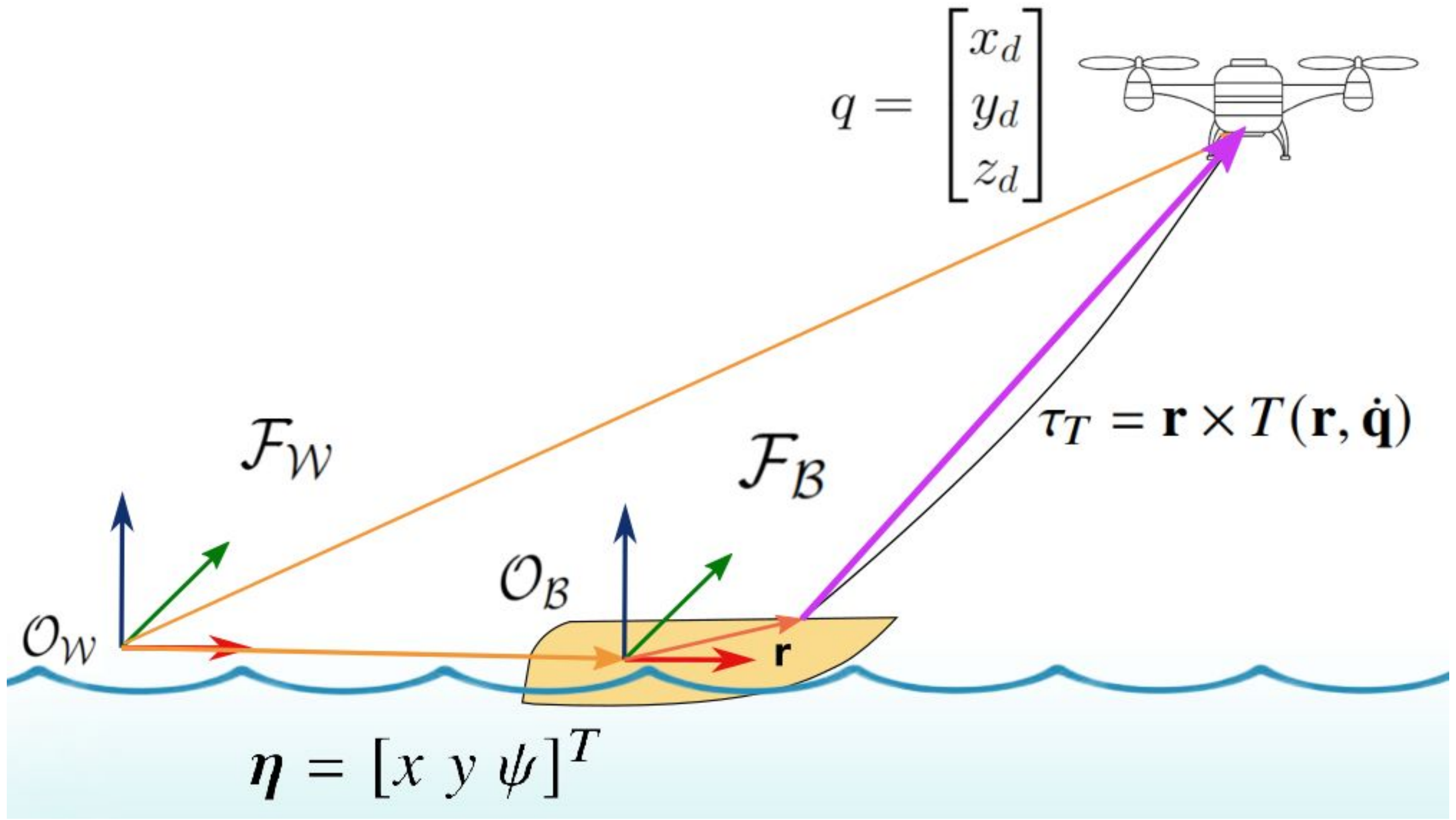
$$D(v) = \begin{bmatrix} -X_{|u|u}|u| & 0 & 0 \\ 0 & -Y_{|v|v}|v| - Y_{|r|v}|r_\psi| & -Y_{|v|r}|v| - Y_{|r|r}|r_\psi| \\ 0 & -N_{|v|v}|v| - N_{|r|v}|r_\psi| & -N_{|v|r}|v| - N_{|r|r}|r_\psi| \end{bmatrix}$$

Tension Force applied to the Boat

- Applied in a determined application point r
- Tension direction is the position of the drone relative to the boat
- Magnitude measured experimentally

$$\frac{\boldsymbol{T}}{||\boldsymbol{T}||} = \frac{\boldsymbol{q} - \boldsymbol{\eta} + (J(\boldsymbol{\eta}) \cdot \mathbf{r})}{||\boldsymbol{q} - \boldsymbol{\eta} + (J(\boldsymbol{\eta}) \cdot \mathbf{r})||}$$

$$M = \mathbf{r} \times T(\mathbf{r}, \dot{\mathbf{q}})$$



Tension Force Measurement

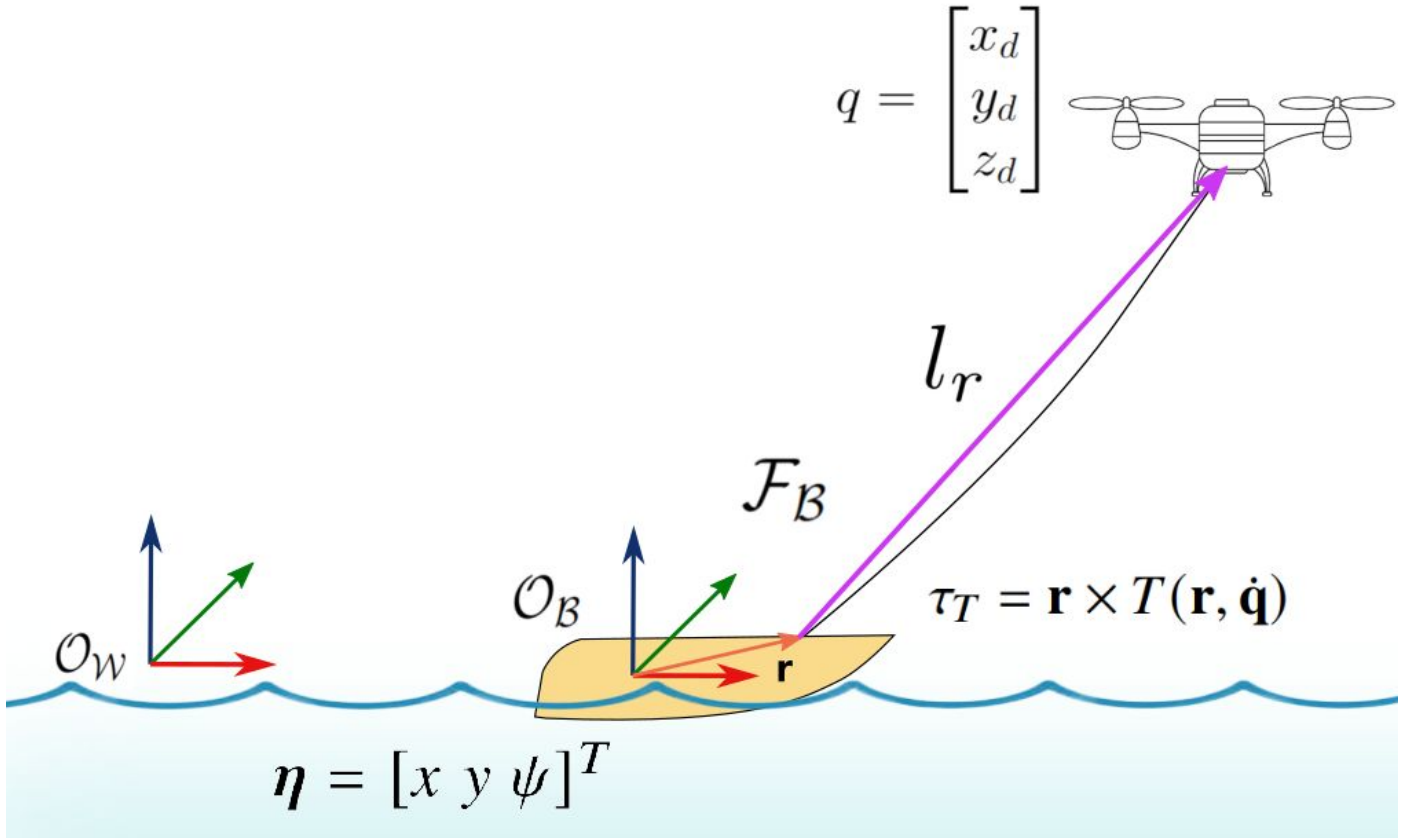
- **Force Gauge** in line with the rope of the boat and record the force as well as boat and drone inertial measurements to make model tension

NOTE: Force sensor not to scale

Tension Constraint

- We define an epsilon ϵ so that $l_r + \epsilon$ means drone is drone is going faster than vessel so we make drone velocity zero

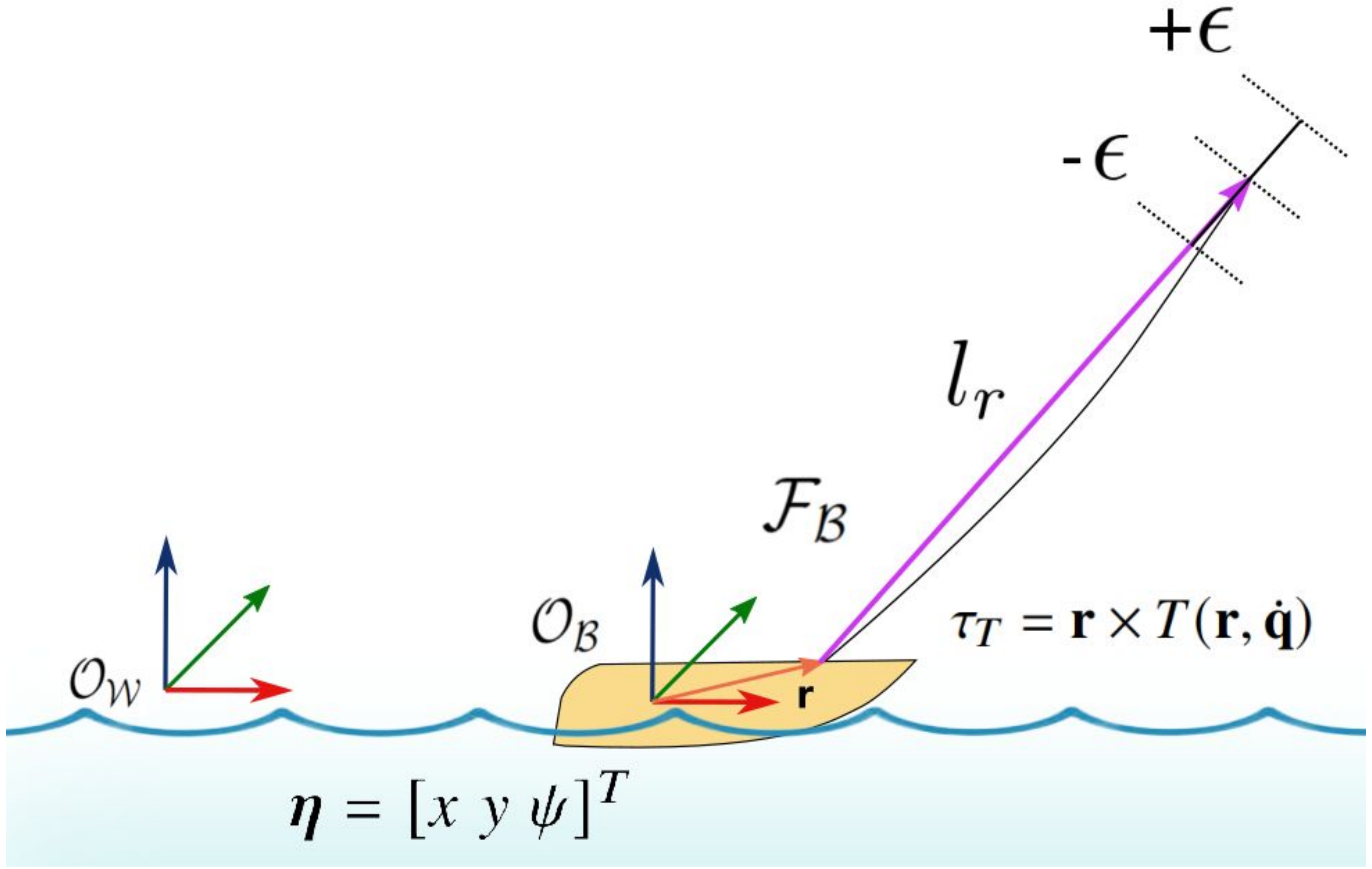
- Additionally, $l_r - \epsilon$ means vessel is going faster than drone so we set the tension force to zero



Tension Constraint

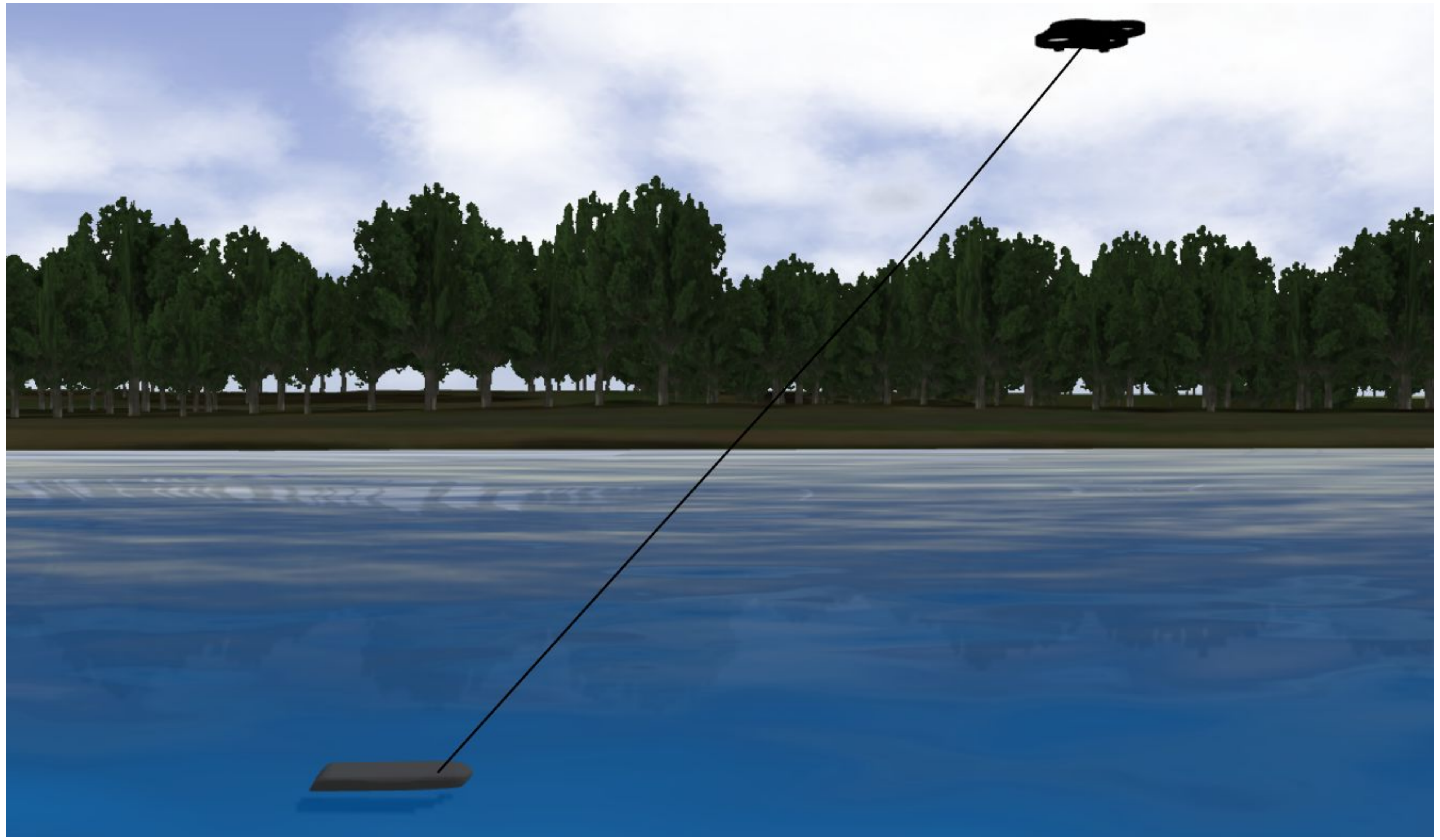
- We define an epsilon ϵ so that $l_r + \epsilon$ means drone is drone is going faster than vessel so we make drone velocity zero

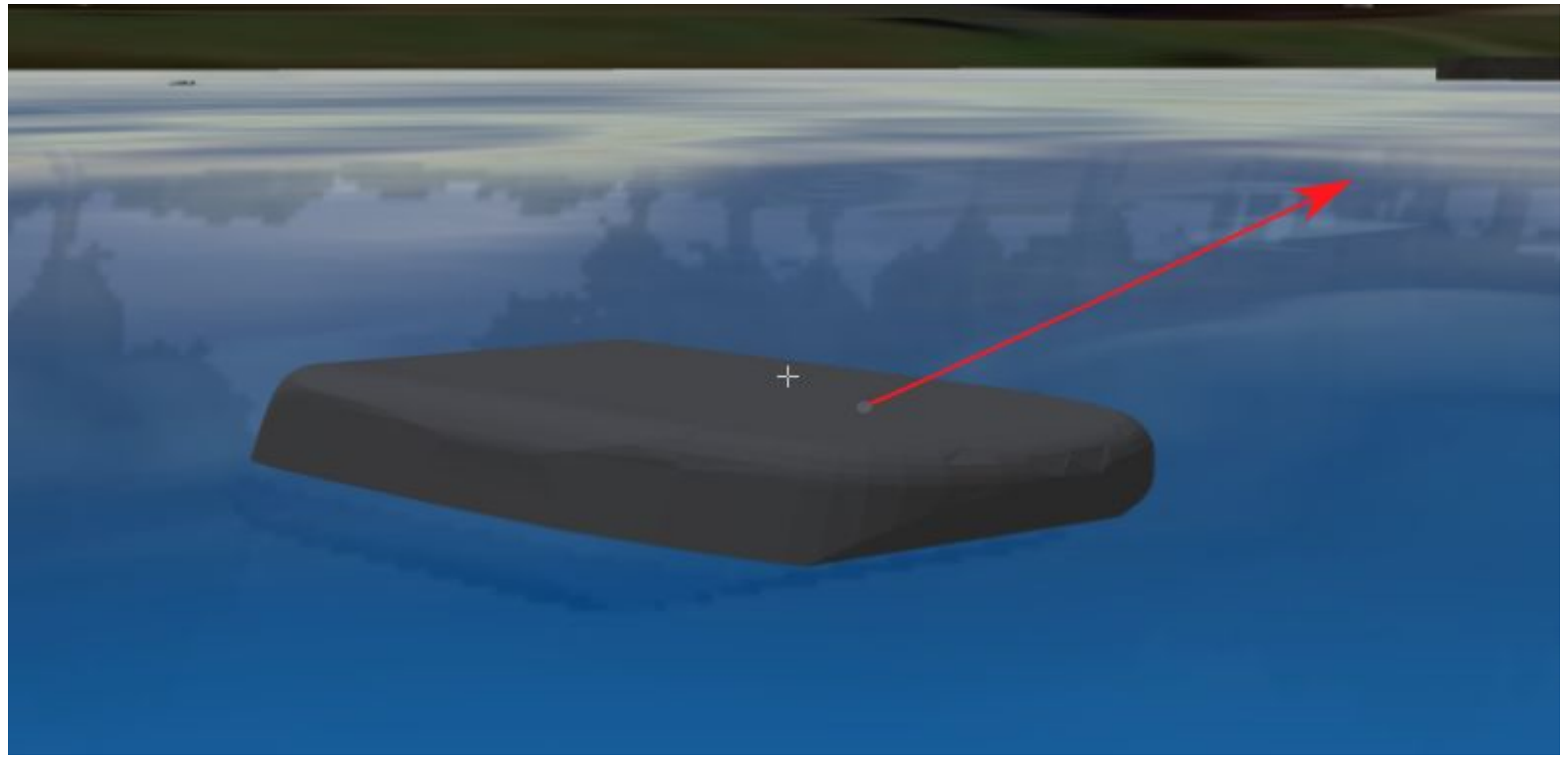
- Additionally, $l_r - \epsilon$ means vessel is going faster than drone so we set the tension force to zero



Gazebo Simulation

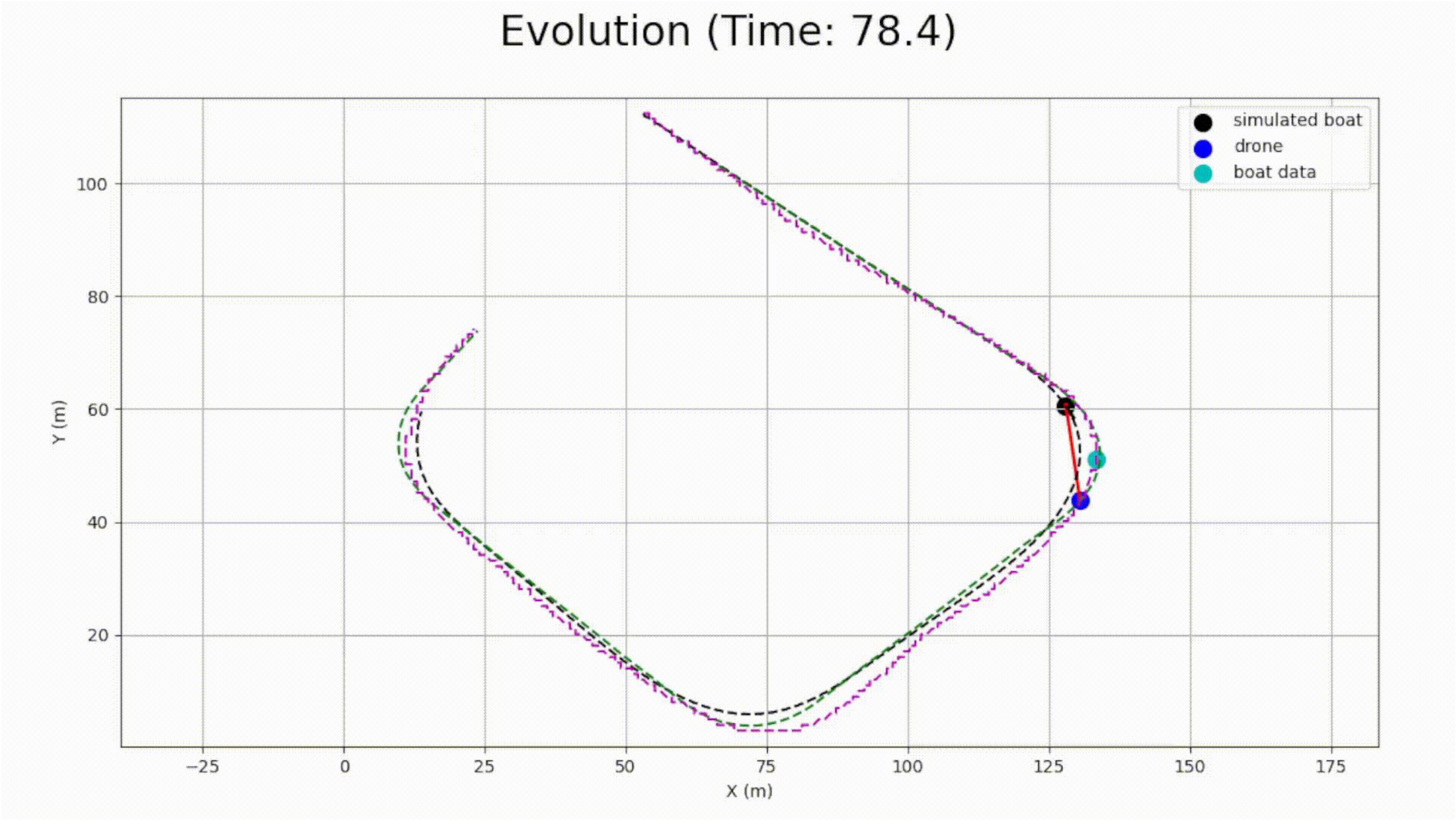
- ROS-based to implement communications in hardware
- Model (6DoF) hydrodynamics physics





Python Simulation

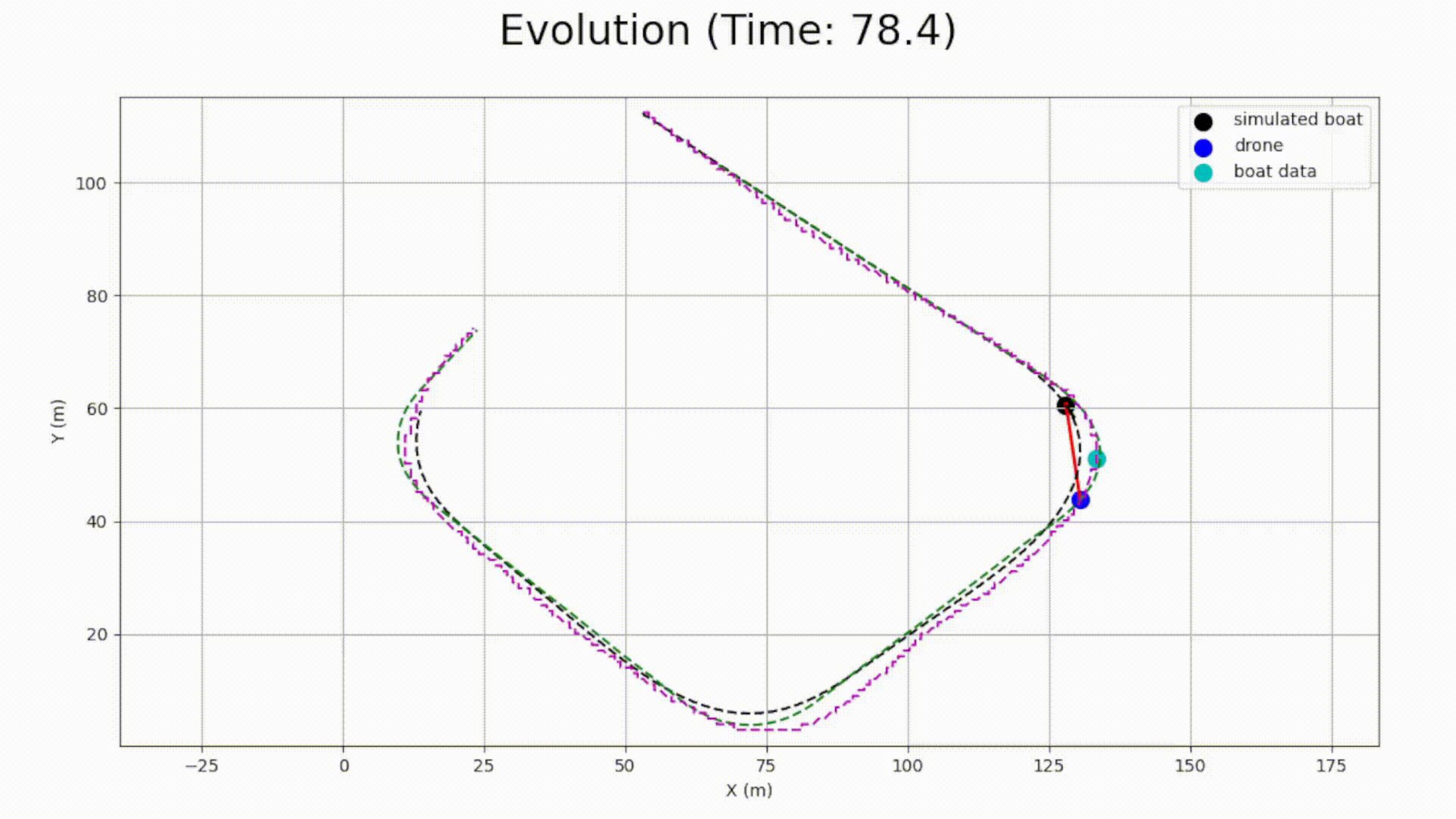
- Faster development and testing
- Model (3DoF) hydrodynamics physics



11x speed

Python Simulation

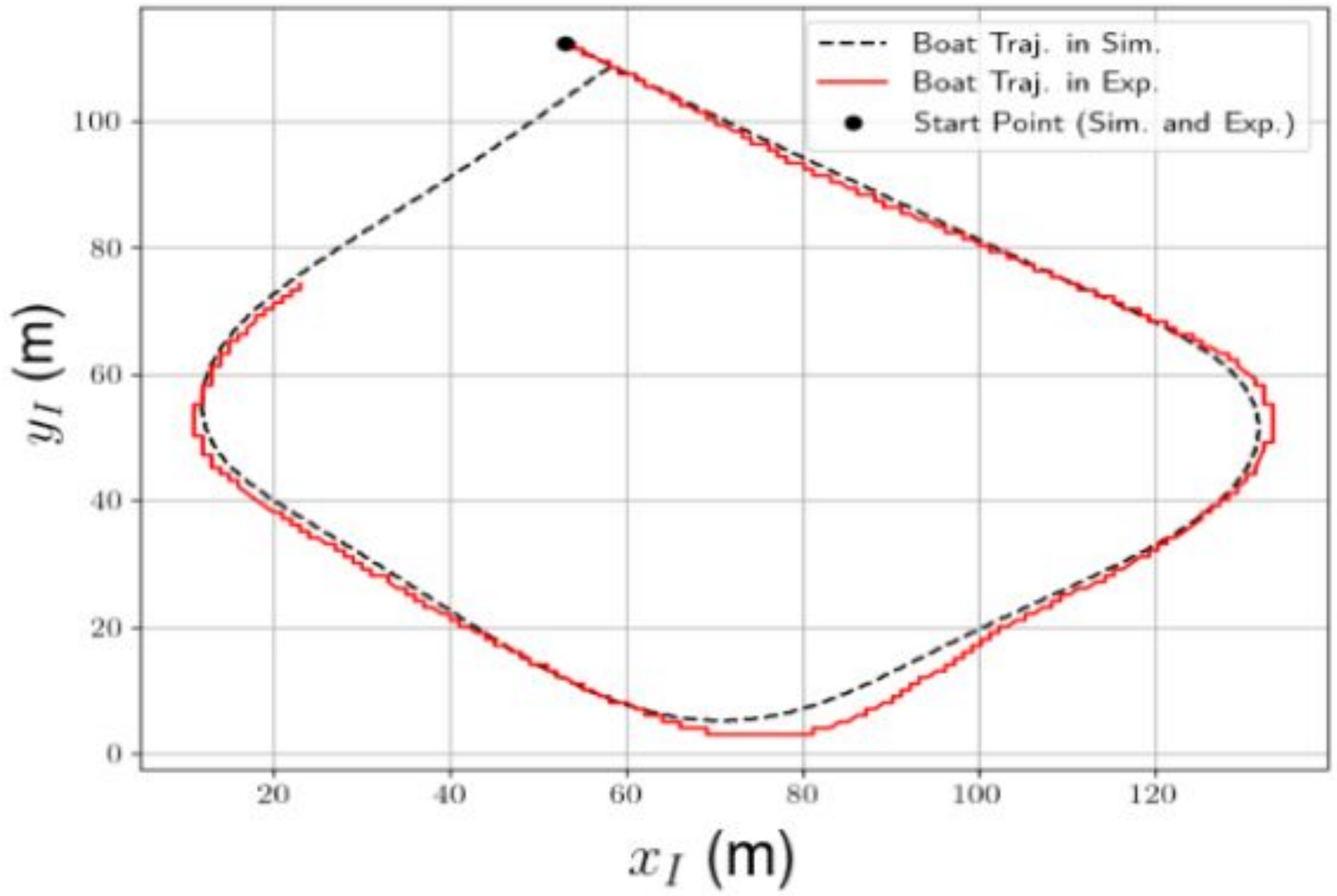
- Faster development and testing
- Model (3DoF) hydrodynamics physics

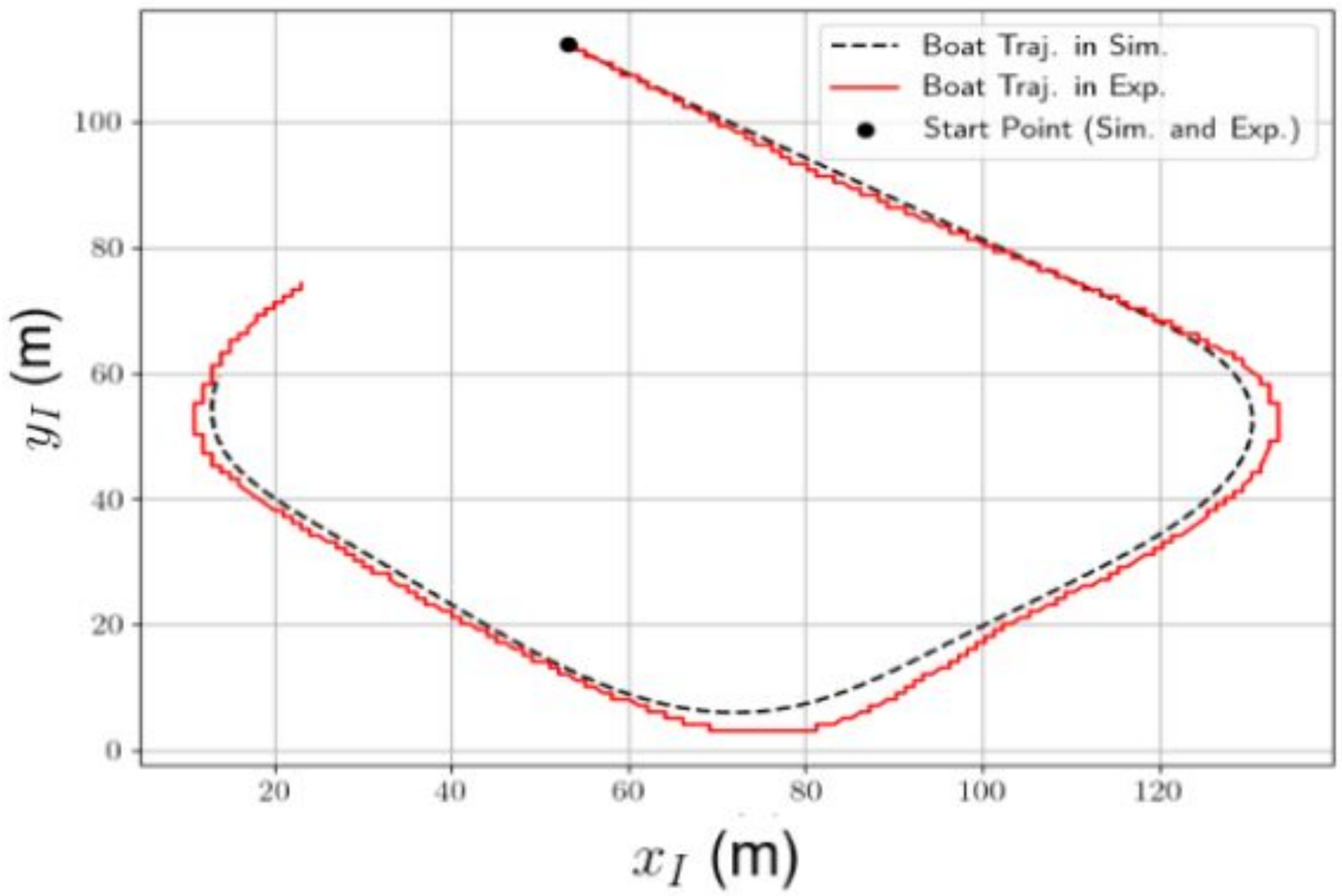


11x speed

Simulation Tuning

- Initial guess from Fossen Book [4]
- Tuned parameters to reduce error with data





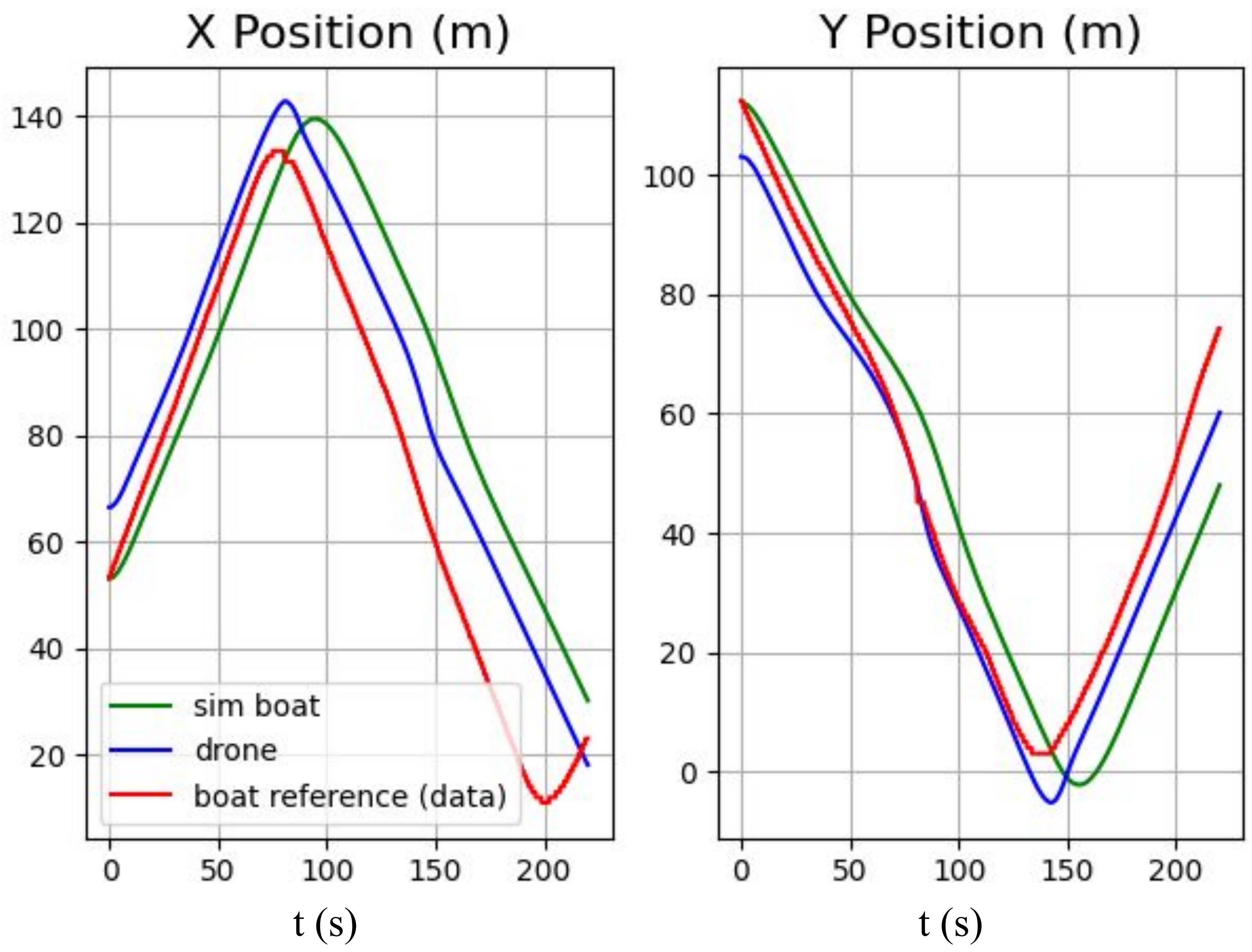
Trajectory Tracking Control

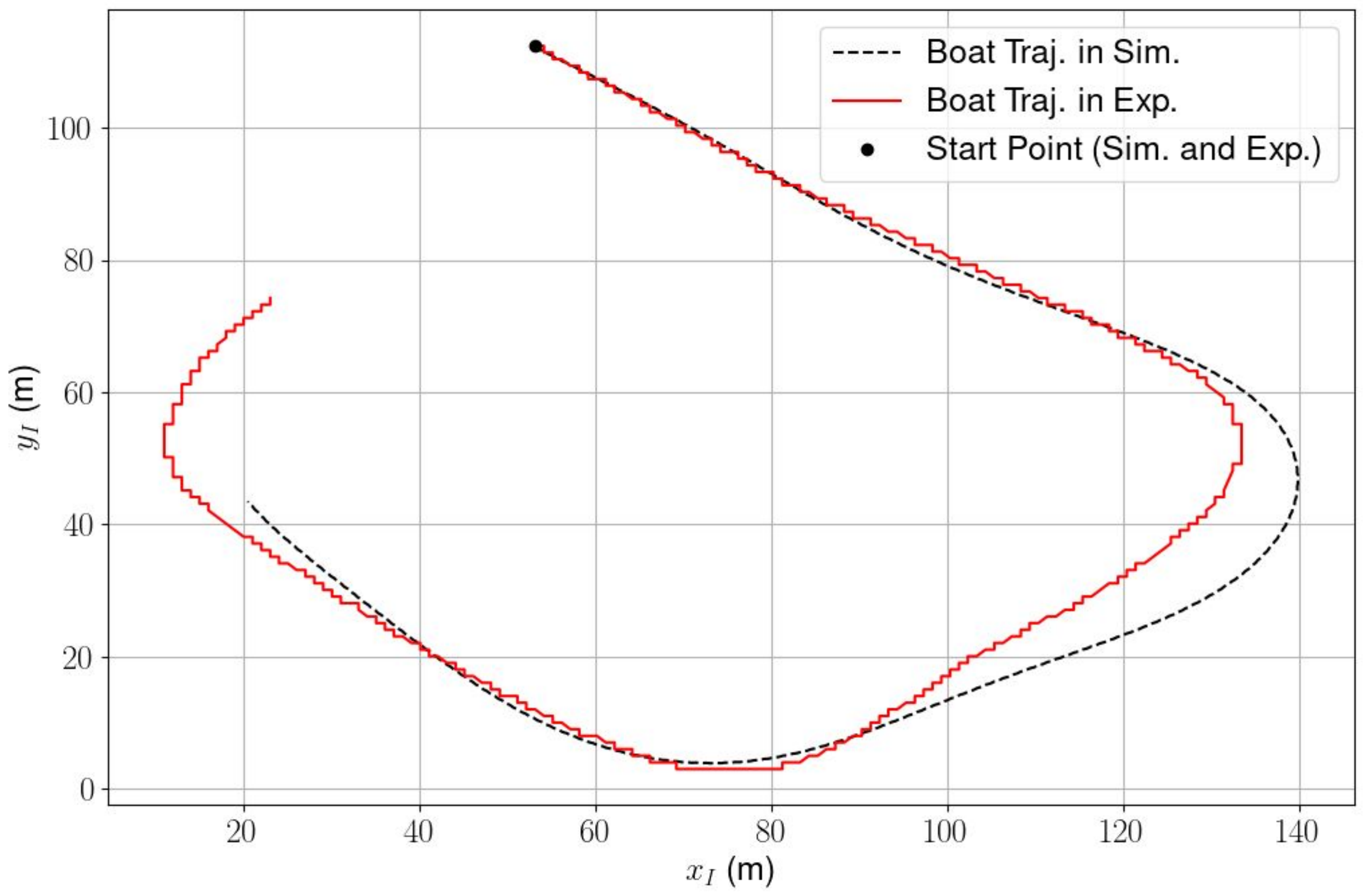
Tracking Controls

- Implemented PID control -> output is drone velocity

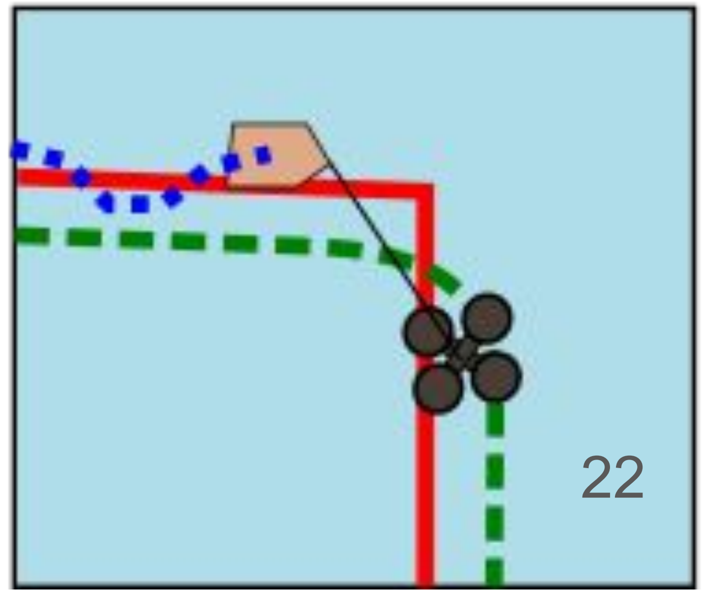
$$v_d = [v_x \; v_y \; 0]^T$$

$$\boldsymbol{v_d} = K_p \boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(t)dt + K_d \frac{\boldsymbol{de}(t)}{dt}$$





- **Mean Square Error X:** 422.3
- **Mean Square Error Y:** 230.4

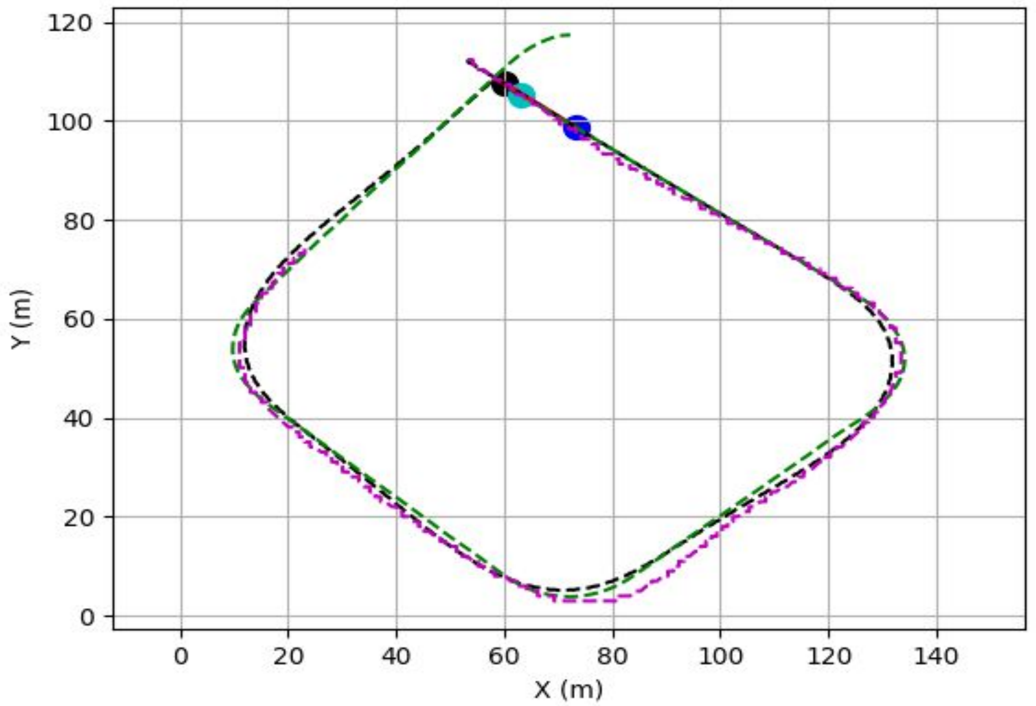


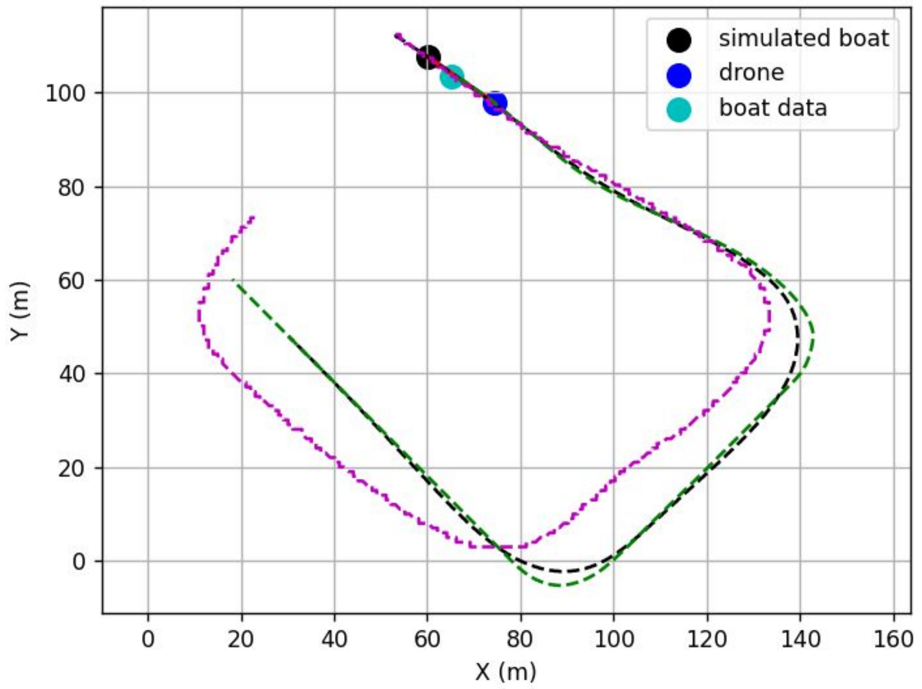
Tracking Controls

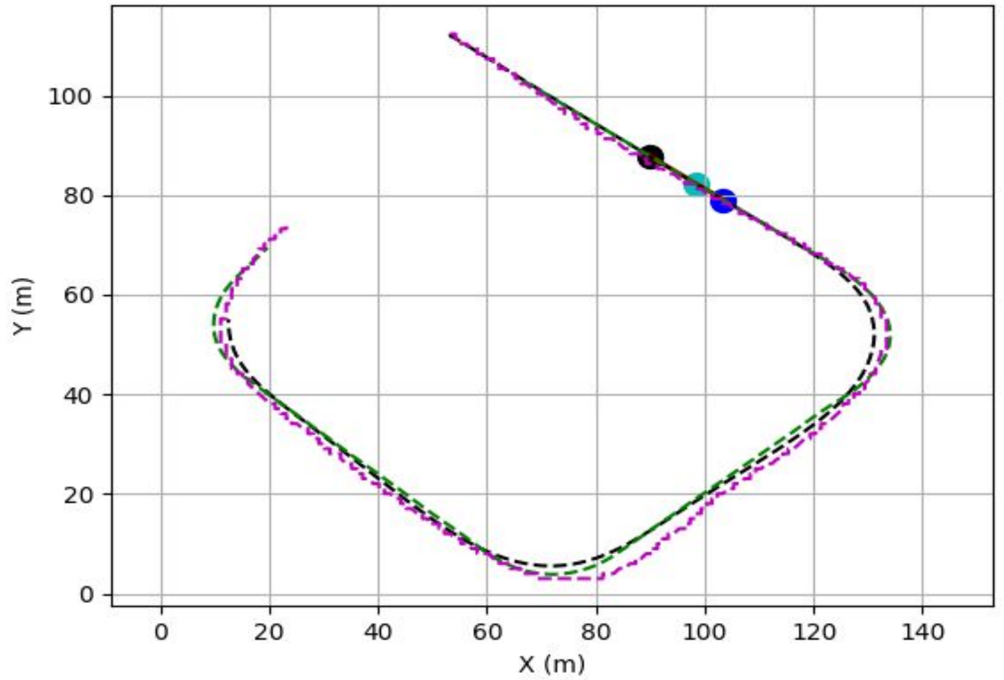
- Implemented PID control -> output is drone velocity

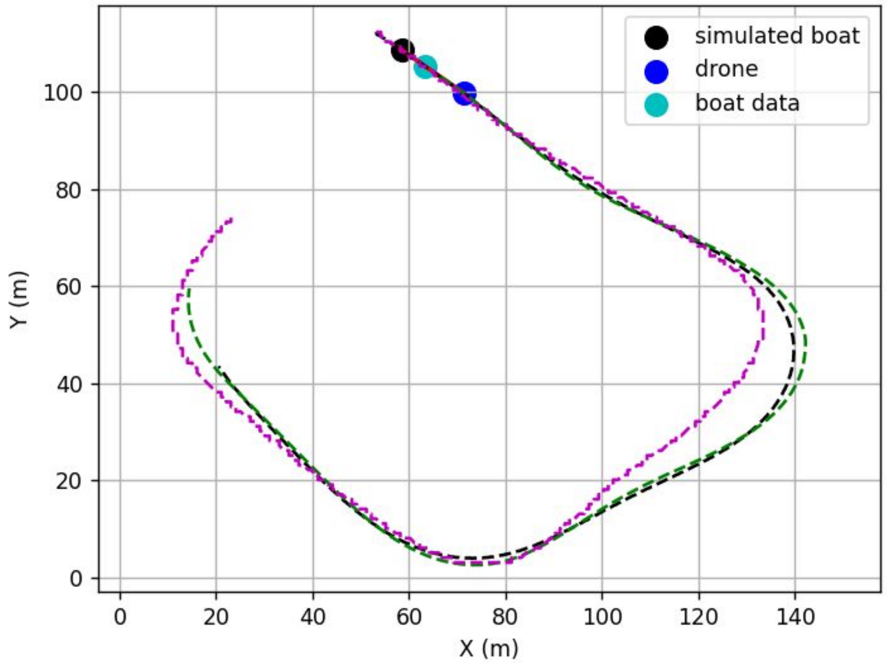
$$v_d = [v_x \; v_y \; 0]^T$$

$$v_d = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

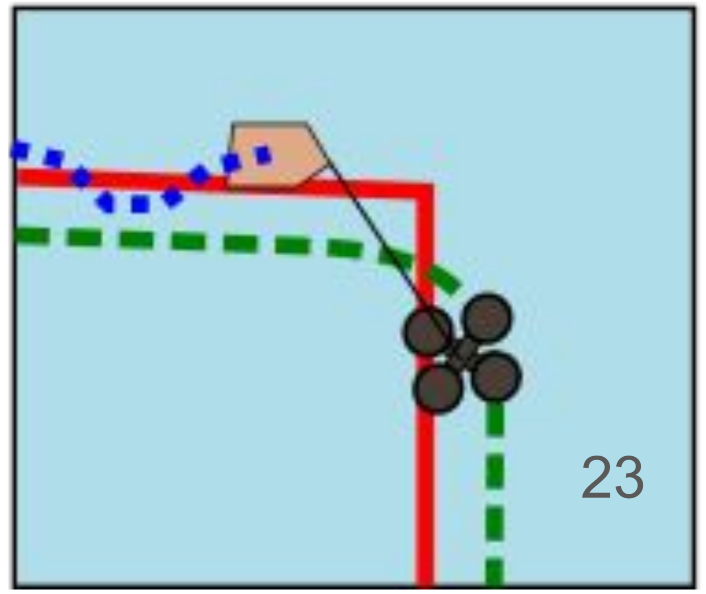








- **Mean Square Error X:** 422.3
- **Mean Square Error Y:** 230.4

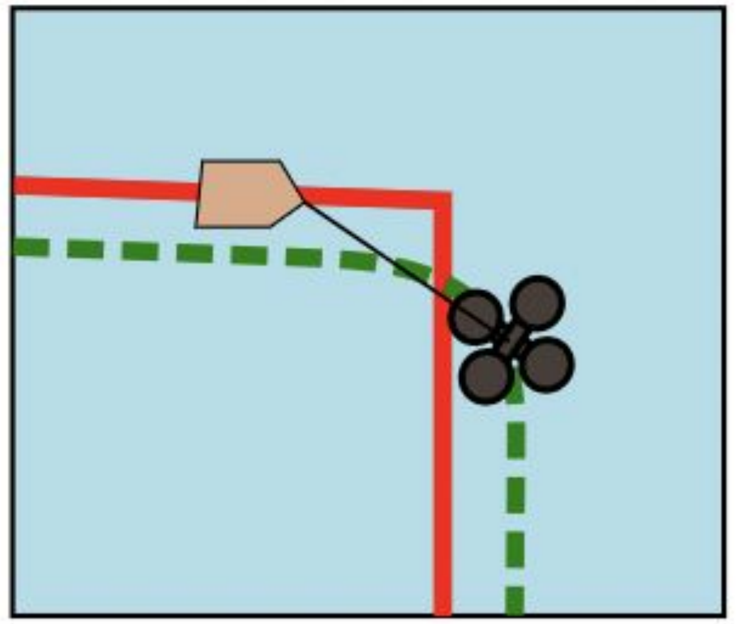


Drone Trajectory Planning

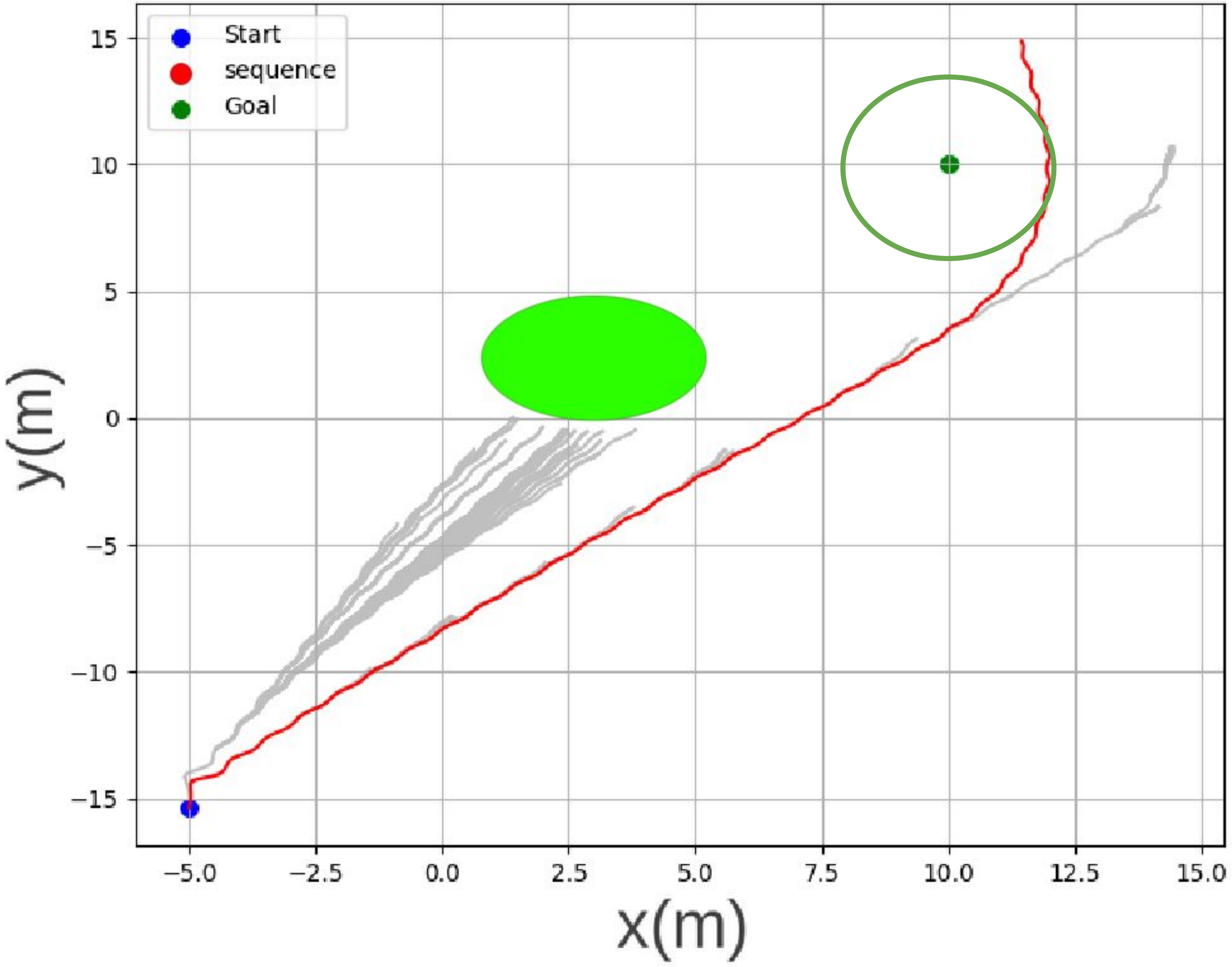
Trajectory Planning

Kinodynamic Rapidly-exploring Random Trees (RRT)

- Samples the state space of the robot and generates trajectories constrained by the dynamics to track the samples
- Chooses the trajectory with the least cost that achieves the goal



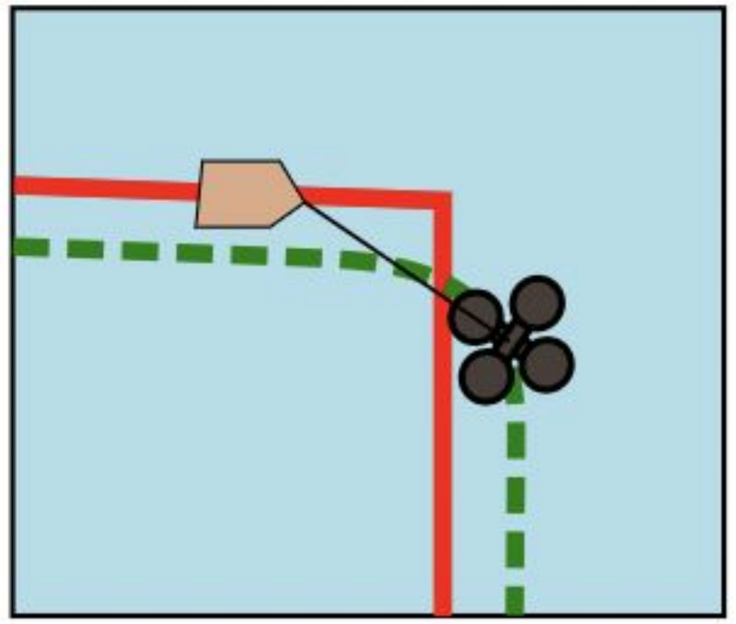
- Allows for real time collision avoidance



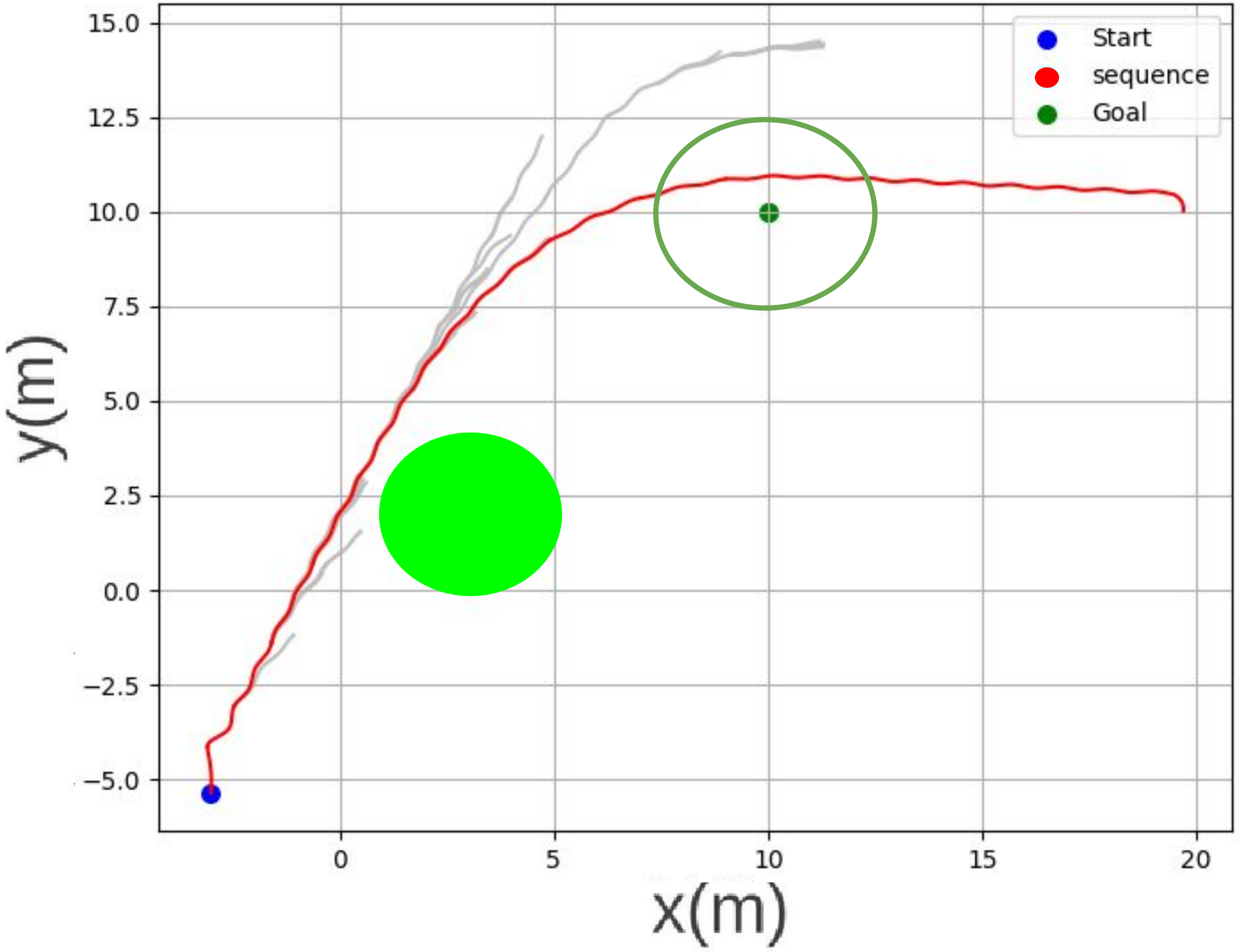
Trajectory Planning

Kinodynamic Rapidly-exploring Random Trees (RRT)

- Samples the state space of the robot and generates trajectories constrained by the dynamics to track the samples
- Chooses the trajectory with the least cost that achieves the goal



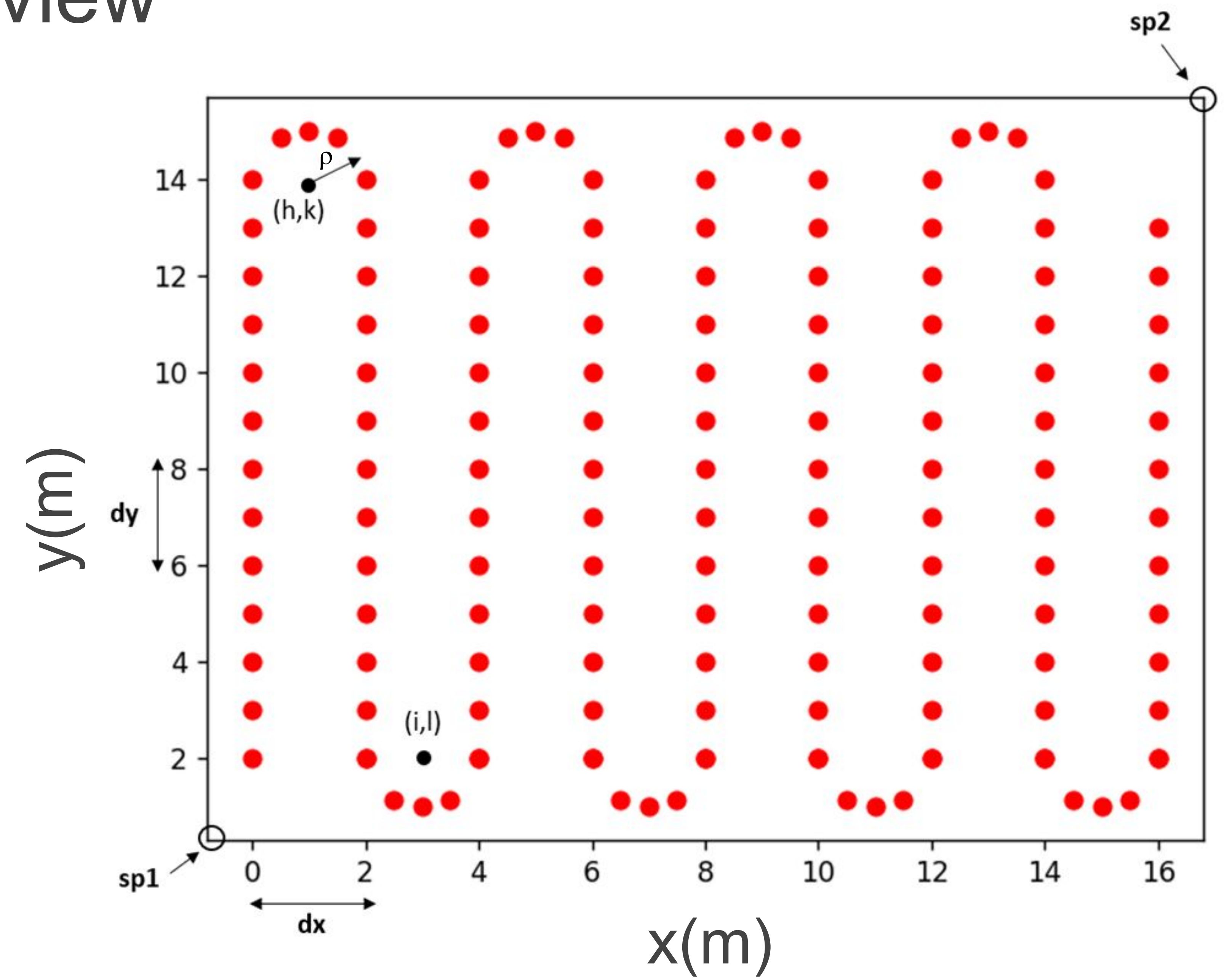
- Allows for real time collision avoidance



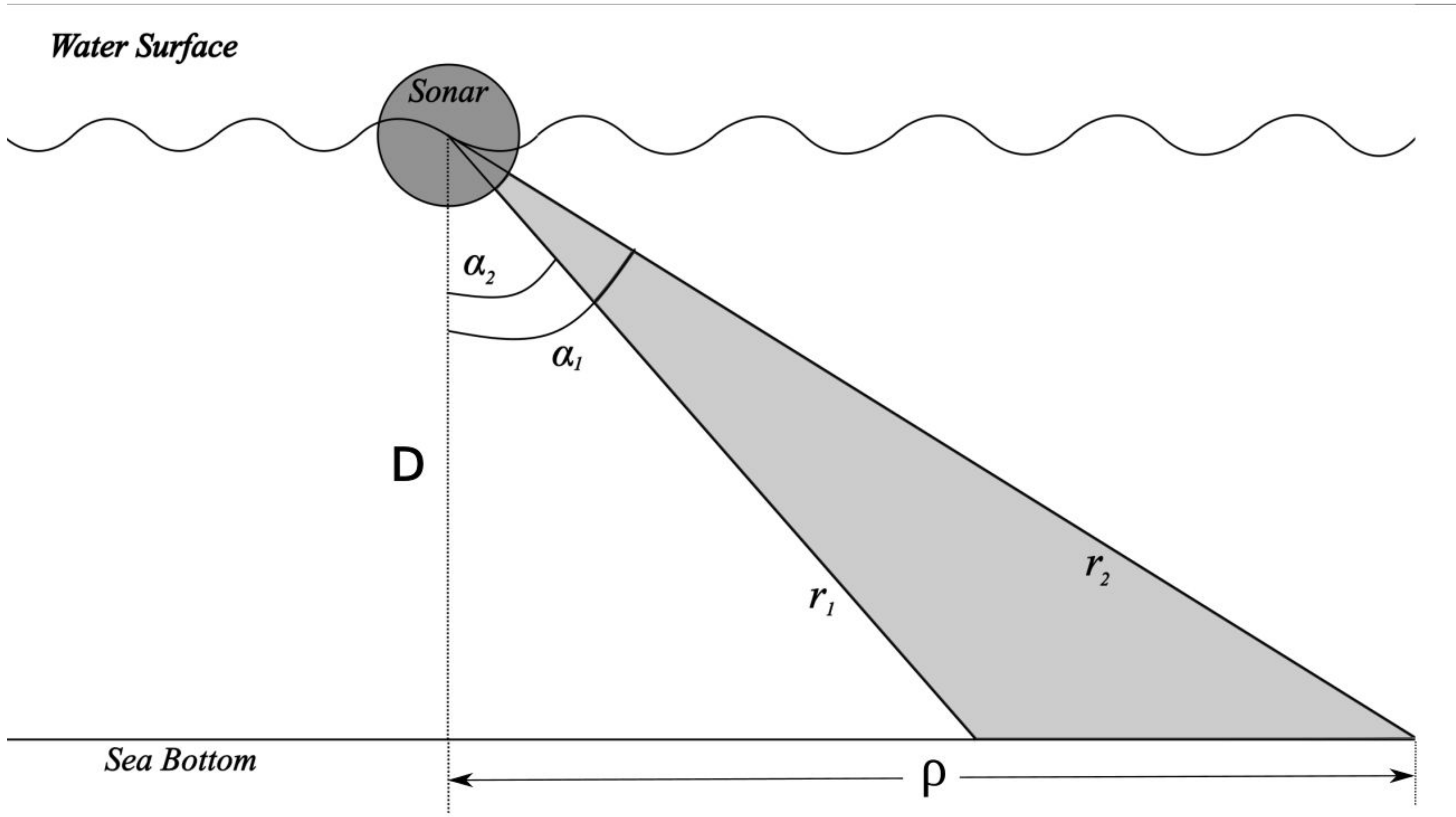
Boat Path Planning

Path Planning Algorithm

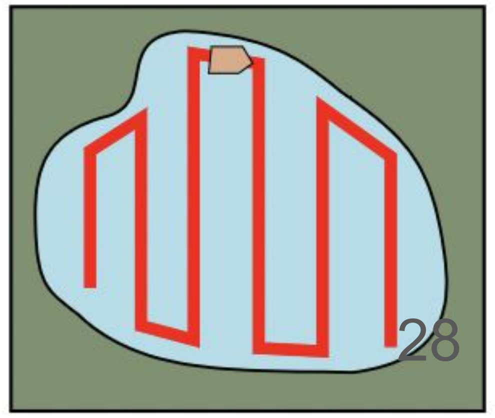
- Generates Boustrophedonic path that covers the maximum area rectangle that is inscribed in a convex polygon
- Parameters are based on sensor field of view



Depth and horizontal distance



$$\rho = D \tan(\alpha_2)$$



Future Work

APRILab
ACTIVE PERCEPTION AND ROBOT INTELLIGENCE LAB

- Implement controller that can better reduce tracking error. Candidates are adaptive, MPC, neural, RL
- Coverage path planning for non-convex polygons and polygons with islands
- Implement in hardware

References

[1] Melo, José and Aníbal Matos. “Survey on advances on terrain based navigation for autonomous underwater vehicles.” Ocean Engineering 139 (2017): 250-264.

[2] K. Mizuno and A. Asada, “Three dimensional mapping of aquatic plants at shallow lakes using 1.8 MHz high-resolution acoustic imaging sonar and image processing technology,” in 2014 IEEE International Ultrasonics Sym-posium, pp. 1384–1387, ISSN: 1051-0117.

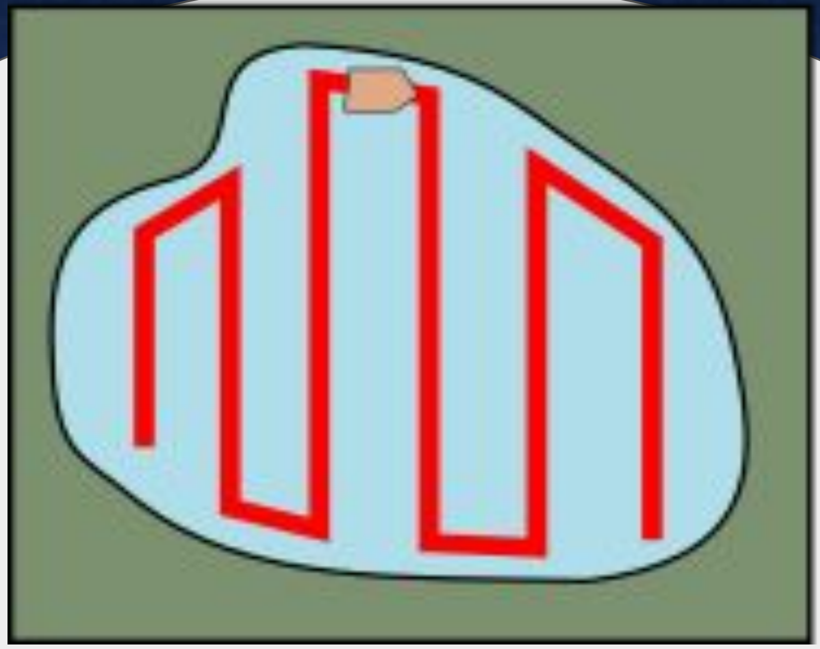
[3] T. Maki, H. Horimoto, T. Ishihara, and K. Kofuji, “Tracking a sea turtle by an AUV with a multibeam imaging sonar: Toward robotic observation of marine life,” vol. 18, no. 3, pp. 597–604.

[4] Fossen, T. I., Handbook of Marine Craft Hydrodynamics and Motion Control, Wiley, 2011.

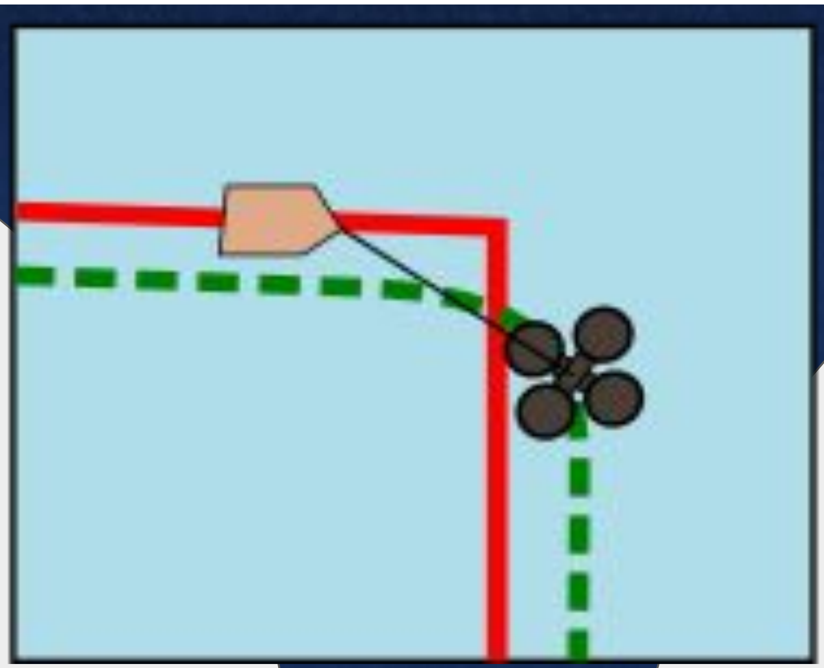
Thank you! Questions?

Trajectory Planning and Control of Bathy-drone

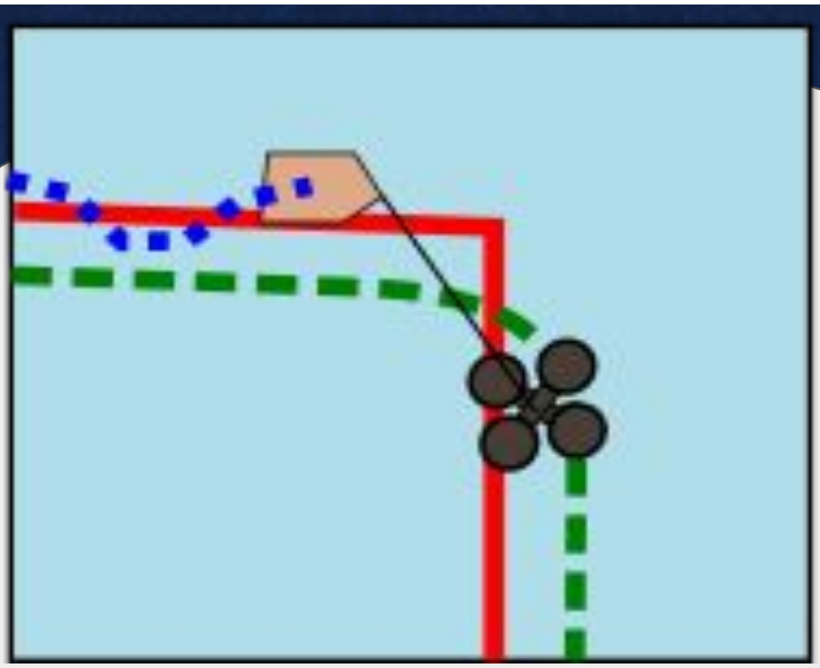
Boat Path Planning

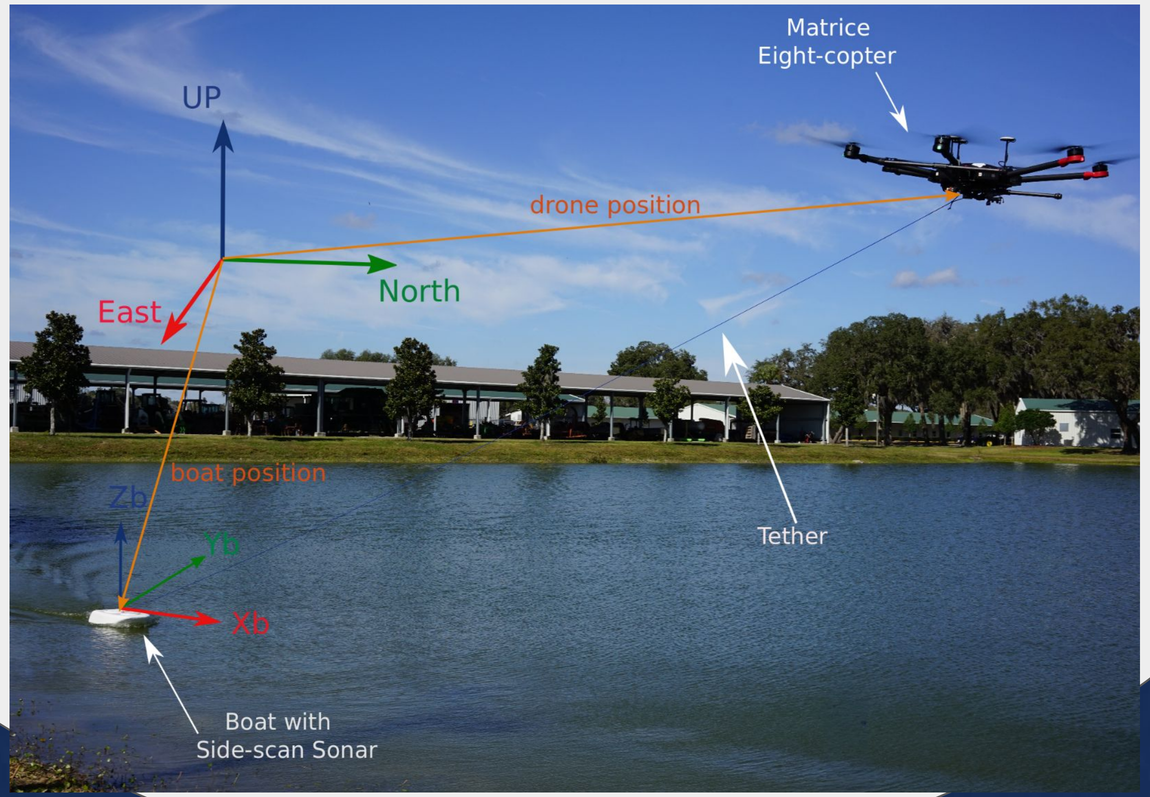


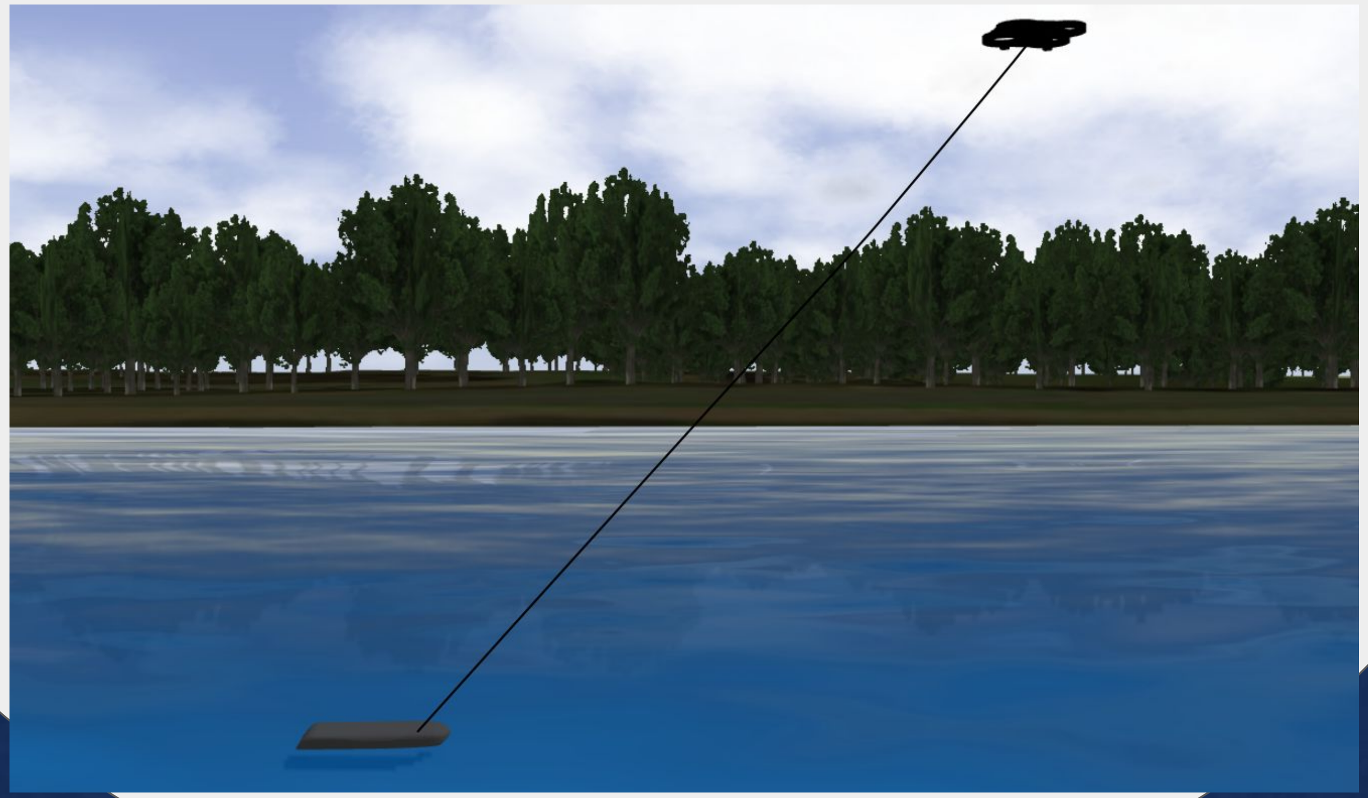
Drone Trajectory Planning



Trajectory Tracking Control







Thank you!
Question?

SCITECH

APRILab
ACTIVE PERCEPTION AND ROBOT INTELLIGENCE LAB

UF UNIVERSITY of FLORIDA

Static Experiment: Steady State

At steady state, the equation **no longer contains the acceleration**, so the unknowns are the linear and quadratic drag coefficients

$$m_\xi \dot{\xi} + d_\xi \xi + g_\xi = \tau_\xi$$

One can solve it inverting the H matrix

$$\lambda = H\Lambda$$

$$\lambda = \begin{bmatrix} \tau_{\xi,1} - g_\xi \\ \tau_{\xi,2} - g_\xi \\ \vdots \\ \tau_{\xi,n} - g_\xi \end{bmatrix} \quad H = \begin{bmatrix} \xi_1 & \xi_1 |\xi_1| \\ \xi_2 & \xi_2 |\xi_2| \\ \vdots & \vdots \\ \xi_n & \xi_n |\xi_n| \end{bmatrix} \quad \Lambda = \begin{bmatrix} d_\xi \\ d_{\xi|\xi|} \end{bmatrix}$$

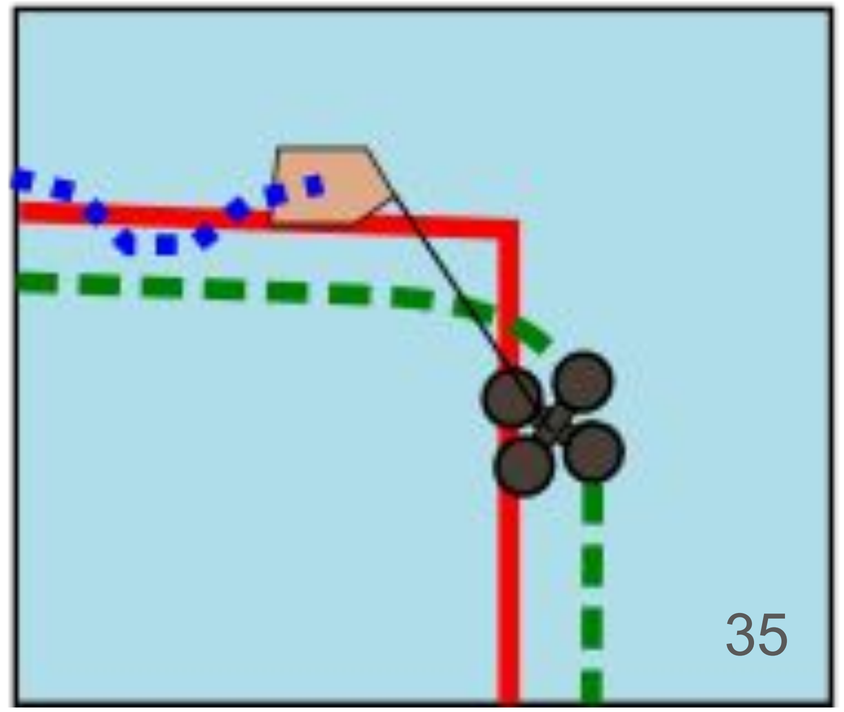
Static Experiment: Steady State

At steady state, the equation **no longer contains the acceleration**, so the unknowns are the linear and quadratic drag coefficients

$$\begin{bmatrix} \tau_{x,1} \\ \tau_{x,2} \\ \vdots \\ \tau_{x,n} \end{bmatrix} = \begin{bmatrix} u_1 & u_1|u_1| \\ u_2 & u_2|u_2| \\ \vdots & \vdots \\ u_n & u_n|u_n| \end{bmatrix} \begin{bmatrix} d_x \\ d_{x|x|} \end{bmatrix} \qquad \begin{bmatrix} \tau_{\omega_z,1} \\ \tau_{\omega_z,2} \\ \vdots \\ \tau_{\omega_z,n} \end{bmatrix} = \begin{bmatrix} \omega_{z,1} & \omega_{z,1}|\omega_{z,1}| \\ \omega_{z,2} & \omega_{z,2}|\omega_{z,2}| \\ \vdots & \vdots \\ \omega_{z,=n} & \omega_{z,n}|\omega_{z,n}| \end{bmatrix} \begin{bmatrix} d_{\omega_z} \\ d_{\omega_z|\omega_z|} \end{bmatrix}$$

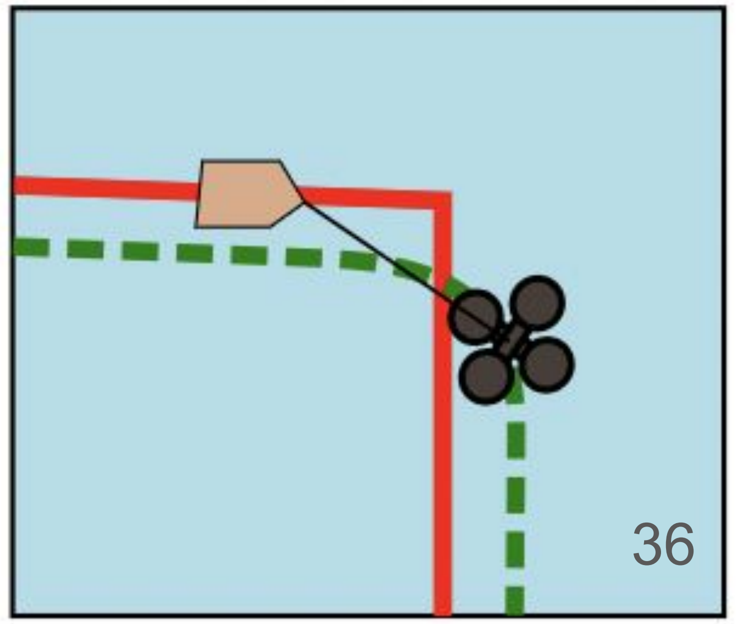
Tracking Controls

- **Goal:** Compute the control input to the system based on the reference signal
- **Inputs:**
  - Trajectory the drone needs to follow
  - State of the system (positions and velocities of the boat)
- **Outputs:**
  - Drone’s velocity at the current time step
- **Approach:**
  - PID control with saturation



Trajectory Planning

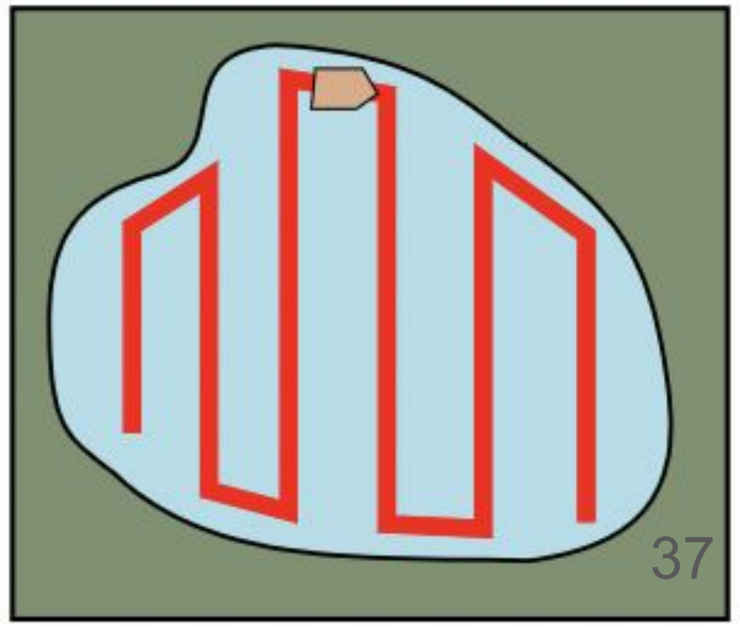
- **Goal:** To compute the drone's trajectory such that the boat can follow the planned path
- **Inputs:**
    - Path planned for the boat
    - Dynamics model of Bathydrone
    - Control law for the drone (PID)
- **Outputs:**
    - Drone's trajectory (or control inputs to the drone)
- **Approach:**
    - Plan the drone's trajectory similar to the boat's path first, and revise the trajectory using the Gazebo simulation by iterations.



Path Planning Algorithm

- **Goal:** To cover the region of interest with sensor field-of-view
- **Inputs:**
  - The geometry of the area of interest (2D polygon), representing the water surface
  - Sensor field-of-view geometry, with respect to the boat configuration
- **Outputs:**
  - A sequence of waypoints for the boat

- **Approach:**
  - Complete coverage path planning algorithms modified to incorporate the given sensor field-of-view geometry and sensor characteristics



Kinematics

Challenge: Weird tethered dynamics

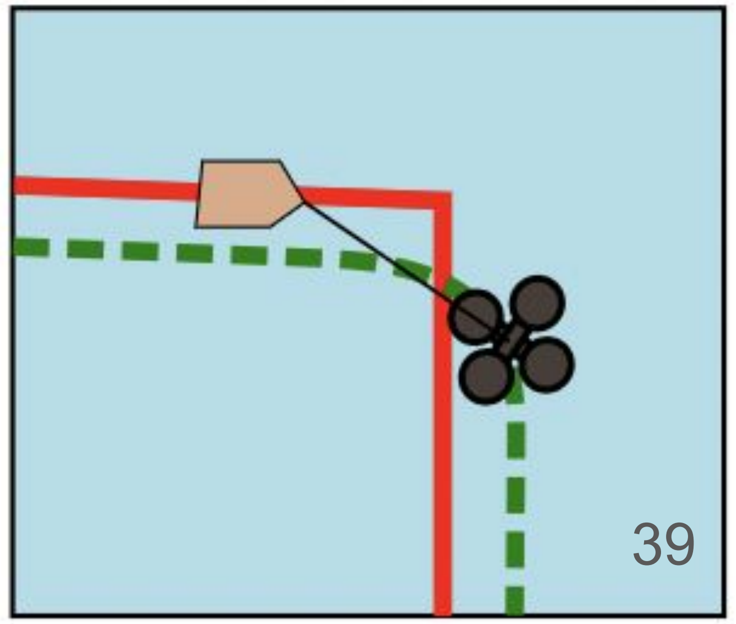
Solution:

- The drone position x, y and orientation θ will be **constrained** to move in a straight line or a **minimum turning radius** ρ, and will not be able to move backwards, which results in the formulation of a Dubins Path,

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\theta}_d \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ K/\rho \end{bmatrix}$$

Trajectory Planning

- **Goal:** To compute the drone's trajectory such that the boat can follow the planned path
- **Inputs:**
  - Path planned for the boat
  - Dynamics model of Bathydrone
  - Control law for the drone (PID)
- **Outputs:**
  - Drone's trajectory (or control inputs to the drone)
- **Approach:**
  - Plan the drone's trajectory similar to the boat's path first, and revise the trajectory using the Gazebo simulation by iterations.



References

[4]
[5]