Software design document for TAICHI

# Introduction

TAICHI operation is composed of two phases:

* Phase I, comparison between benchmark result
  + One AUV is doing lawnmower.
  + One AUV is doing SPDE-adaptive sampling.
* Phase II, data sharing enabling multi-agent platform.
  + Two AUVs are running simultaneously together with data sharing.

# Software structure

### Mission launcher:

* It launches the mission.
* It handles the communication between each module.
* Attributes
  + Total\_waypoints
  + Counter\_waypoint\_adaptive
  + Counter\_waypoint\_nonadaptive
  + Counter\_waypoint\_data\_assimilation
* Methods
  + Launch the mission()
    - Set the starting waypoint to the first waypoint to the transect trajectory.
    - While True:
      * Get data.
      * Check popup().
      * If pre-run mode is finished, then do two-step planning first.
      * If not popup and If it arrives at the set waypoint.
        + If pre-run mode:

Set next waypoint from designed trajectory.

Update field.

Counter\_waypoint\_pre\_run += 1

* + - * + If not pre-run mode:

If num<MAX:

Set next waypoint from Myopic3D planner.

Else:

Popup and mission complete.

Update field.

Counter\_waypoint\_adaptive += 1

Run myopic3D planner to get pioneer waypoint.

### Setup module:

* It sets up the module with proper parameters.

### Data module:

* It handles the data during the operation.
* Attributes
  + AUV positions.
  + Salinity measurements.
* Methods
  + Save data.
  + Data assimilation.

### AUV module:

* AUV module provides key API for handling AUV maneuvers and data acquisition from DUNE.
* Attributes
  + Speed.
  + Vehicle positions.
  + Salinity measurement.
  + Auv\_handler
    - Set waypoint.
    - Popup.
  + Popup state
* Methods
  + Send\_SMS\_mission\_complete()

### Non-Adaptive Path-planner module (Lawnmower):

* It computes the pre-programmed trajectory given some criteria.

### Non-Adaptive Path-planner module (Transect):

* It computes the transect trajectory based on some vertices.

### Adaptive Path-planner module (Myopic3D):

* It computes the next waypoint based on the updated field.
* It requires
  + Updated mean field.
  + Updated diagonal terms from the Covariance matrix.
  + Previous waypoint index.
  + Current waypoint index.
  + Next waypoint index.
  + Visited waypoint indices.
* Attributes
  + Waypoints.
  + Neighbor hash table.
* Methods
  + Find\_all\_neighbours(current waypoint index).
    - Return all possible neighbor indices.
  + Get\_candidate\_waypoint\_indices(previous waypoint index, current waypoint index)
    - Return all possible candidate indices.
  + Get\_EIBV\_for\_candidates(candidate waypoint indices)
    - Return EIBV for all candidate waypoint indices.

### Kernel module: (SPDE)

* It provides a modular way to assimilate measurements from CTD sensor.
* It is handled by Martin.
* Methods
  + Update(salinity, indices).
    - Update the conditional field using data.
  + postprocessing().
    - Reset Q.
  + candidate().
    - Return post variance term.
  + Mvar().
    - Return MCMC diagonal Sigma terms.
  + Mu.
    - Return conditional mean field.
  + threshold.
    - Return updated threshold.

# Workflow

Mission is launched by missionlauncher.py.

Start: It will start along the pre-programmed transect path.