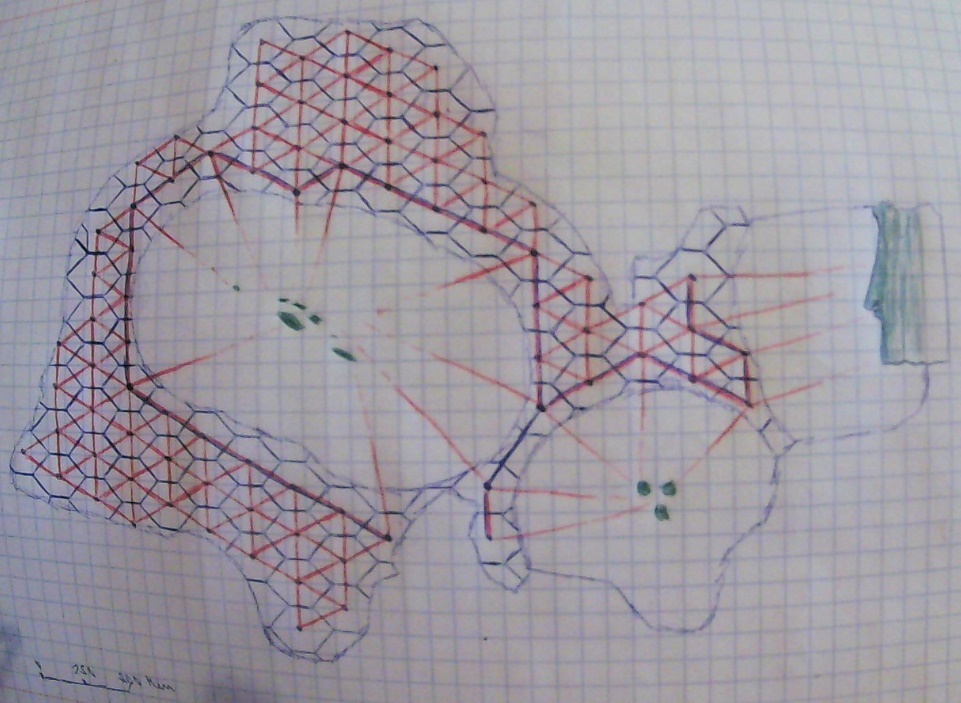
**Project SeaNet**

**Autonomous maritime patrol of Portugal’s Exclusive Economic Zone and proposed Extended Continental Shelf**

**Abstract:** The purpose of this project is to produce a (prototype demonstrator) tool for the Portuguese Navy to use in deploying Unmanned Surface Vehicles (USV) to patrol the Portuguese maritime territory. Due to the current ambitions of Portugal to expand their maritime assets (Extended Continental Shelf currently pending UN approval) new Navy assets will be required to patrol this greater area. To achieve this goal a fleet of Unmanned Surface Vehicles (USV) can be deployed, potentially minimizing costs, operating in complete autonomy under the watchful eye of the Portuguese Navy.

**Raw ideas and options:**

* Use graph to represent positions in the territory to be patrolled by USVs.
* Voronoi diagram\visibility graph to produce graphs. Voronoi diagram will be used to create areas around vertices in the ECS zone. For the set of Points (vertices) of the Voronoi diagram create lattice of equidistant points in ECS (honeycomb pattern with vertices in center of each hexagon). Visibility graph will be used to create a graph within the EEZ (islands and continent as obstacles) so USV can navigate these zones and move to base efficiently. Border of honeycomb (ECS graph) with EEZ can be incorporated as obstacle in Visibility graph to connect the ECS graph with the EEZ graphs.



Some edges are docks\bases (for maintenance) the USV must go to if required. Consider also mobile edges (or other mechanism) for refuel ships and USV carriers (vertex only visible to USVs that need to see it or infinite adjacent edge weights) (quite possibly carriers are in charge of patrolling an area and so no USV is deployed by a stationary base there unless required (maintenance\refuel)).

Graph is weighted, by distance of edge and additionally weather/sea conditions, but not directed (all edges are bi-directional, each direction with possibly different costs due to sea conditions at destination vertex). (Graph will have cycles!)

USV travels from A to B by taking the minimum length\energy path (Shortest path algorithm), with refinement (if there is a possible direct path (no obstacles) from vertex x\_i to x\_k, use it).

* Executes search pattern around vertices area (if vertex is to be patrolled).
* Vertices the USV must patrol may have a priority (vertices in priority queue) possibly depending on external factors (user defined, shortest path or random).
* USV is deployed from base, given a set of vertices to patrol (with possible priority queue), finds shortest path (to first vertex in queue and then to the next and so on, if applicable) to reach and traverse all vertices and then cycles through them (solve traveling salesman?) until further orders (new set of vertices, authorized refuel time, authorized maintenance time, etc.)
* For search pattern research currently used search patterns theory.
* Lévy–walk search
* Project Wild Weasel algorithm but with polygons instead of circles. (https://www.researchgate.net/publication/357810703\_Missile\_Evasive\_Actions\_for\_Increased\_Survivability)
* Develop with Digital Twin and C2 (Command and Control) capabilities.
* Each USV can be an object where methods are actions on or from it (move, stop, patrol, display warning (maintenance), etc.). Each write their status on a SQL database (location, path, destination, status, etc.) to be used in the visualizations, where it is also possible to command and control de USV (stop\cancel task, go to\patrol location(s), go to base/carrier) (execute USV object methods).
* USV failure rate comes from a probability distribution, namely the probability of failure in a given time interval (t days after maintenance). This can be an Exp or Poisson distribution (research required to find appropriate distribution). Two types of failure:
* Warning: USV can travel to receive maintenance (this can be a simple maintenance warning or non-critical component failure).
* Alert: USV can’t travel, catastrophic failure detected (critical component failure).
* Edges traversed by USV can affect its failure rate (maintenance\failure rate).
* Most times the USV will just alert that it’s time for maintenance and do it autonomously.
* Time in maintenance depends on type of failure (or just standard maintenance time).
* USV may be able to travel to base autonomously or in case of total failure require rescue, this increases maintenance time (downtime).
* Capability to update USV graph (add\delete locations). (Maybe just comment on the possibility (Future Work) as this might compromise safety of USV fleet)

**Plan:**

* Get weather and sea data (may need to compute sea state from other measurements).
  + Short analysis of average\extreme sea conditions to comment on USV requirements (in **R**)
* Compute graphs on map (appropriate data structure for efficient edge weight update)
* Build cost function (weight) to associate with each edge and implement
* Create USV object and SQL database
  + Each USV object has (creates) a SQL table to update (location, status, etc.)
  + Implement USV object methods
    - Route computation (shortest path algorithms)
    - Move
    - Patrol vertices
    - Search pattern
    - Warnings and simulated malfunctions
    - Etc.
* Implement visualization (GUI)
  + Monitoring the USVs requires only information to be read from the DB
  + C2 requires appropriate tools on the GUI, these execute USV object methods
    - Event-Driven Programming.
* Use Digital Twin to analyze appropriate number of USVs to patrol the territory, go further or comment on the possibility due to the existence of the Digital twin (Art gallery problem?).