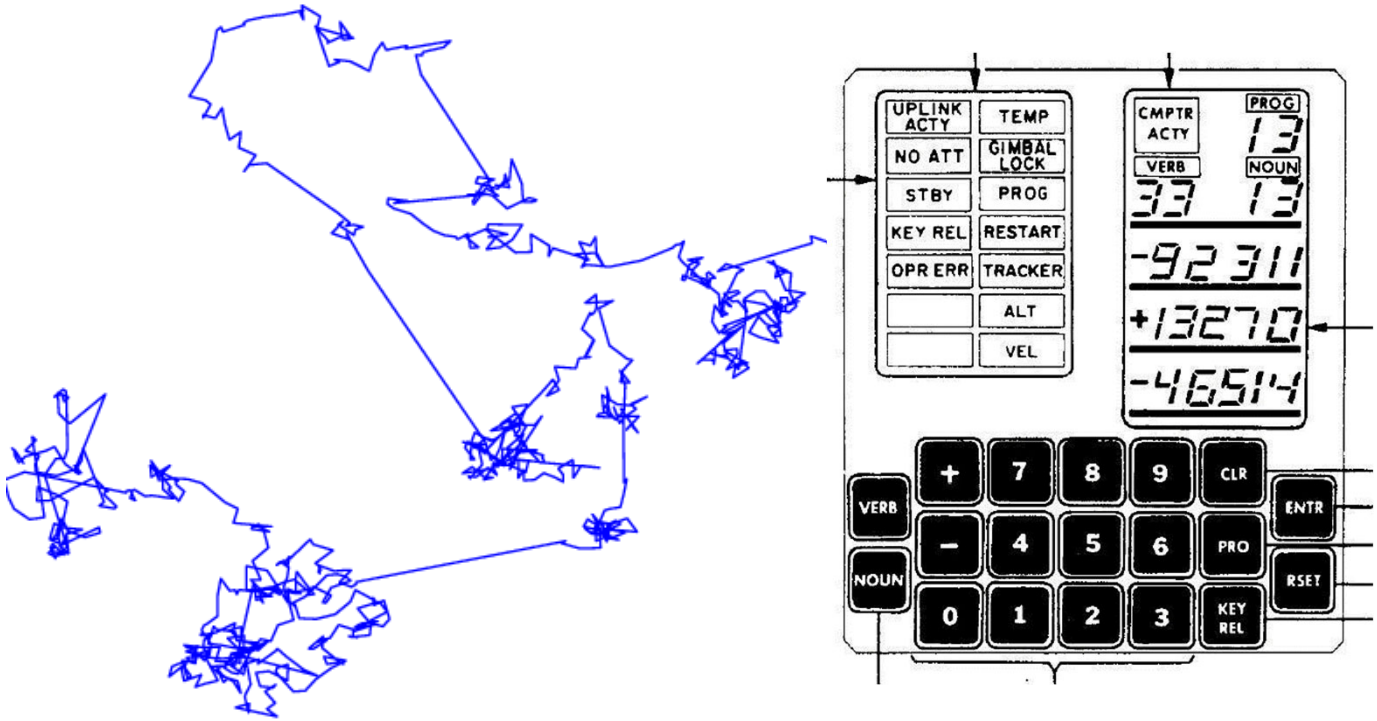
Introduction to path planning

TMR4585

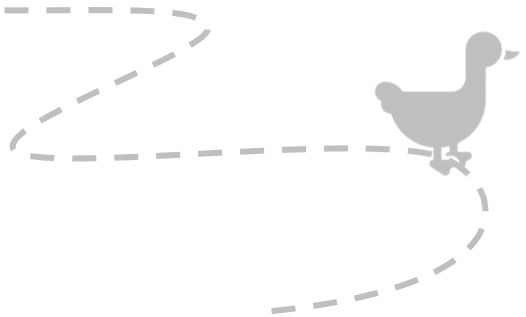
Dr. Trygve Olav Fossum, AURLab, IMT, NTNU



Agenda

Part 1 Theory

- Terms and definitions
  - Path planning and motion planning
- Graph-based approaches
  - Djikstra's algorithm, A* (A-star)
- Genetic and random algorithms
- Other notable elements

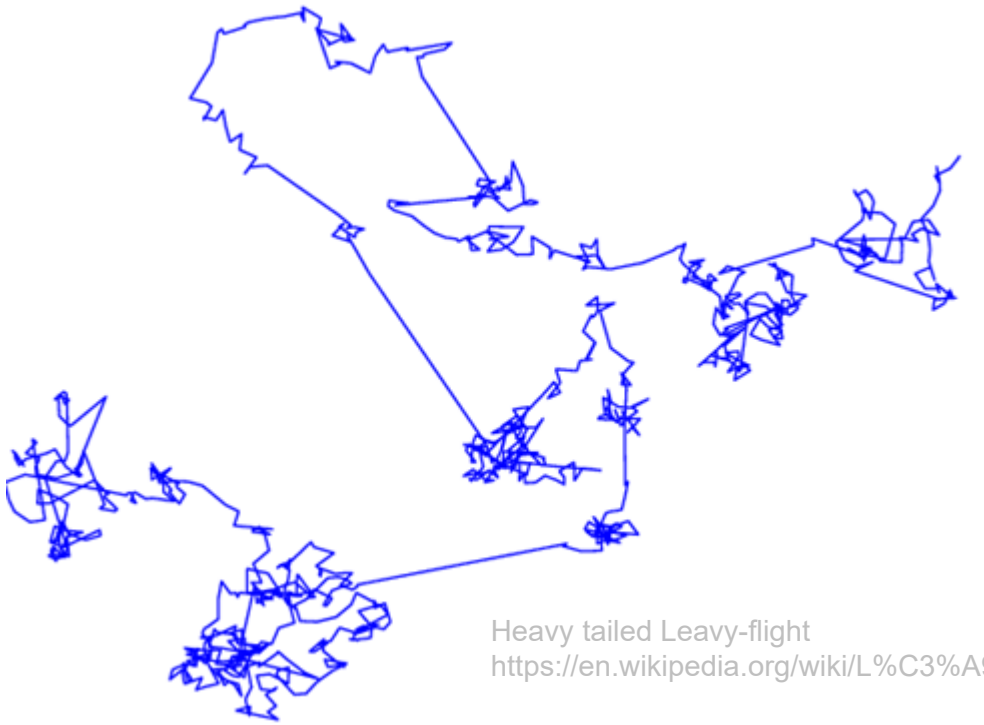


Part 2 Marine applications

- Autonomy as sea and obstacle avoidance
- Informative path planning and adaptive sampling

Part 1 Theory

Basic theory about path planning



Heavy tailed Leavy-flight
https://en.wikipedia.org/wiki/L%C3%A9vy_flight

Terms and definitions

- Path planning (in its purest form) [spatial reasoning]
  - What is the shortest path from A to B?
  - Needs a map representation to use for planning/calculation (path planning can only be applied when a map of the environment is known.)
- Motion planning and control
  - How can I move a physical robot from here to there?
  - Find a sequence of valid configurations that moves the robot from the source to destination.
  - Needs to know physical constraints and forces
  - Equations of motion (e.g. Fossen 6 DOF model for AUV)
  - Maneuvering theory (rigid body kinetics/dynamics)
- Dependence: the planned motion should follow a path and the current motion can affect the path planning. Think GPS (PP) and driving a car (MP).

"Path planning" might be used more often to simply describe the problem of finding a desired path from one state (or sub-set of states) to another. Whereas "motion planning" might be used to describe the same problem but more specifically the actual commanded motions that the robot uses to track the series of desired states along the path.

Terms and definitions

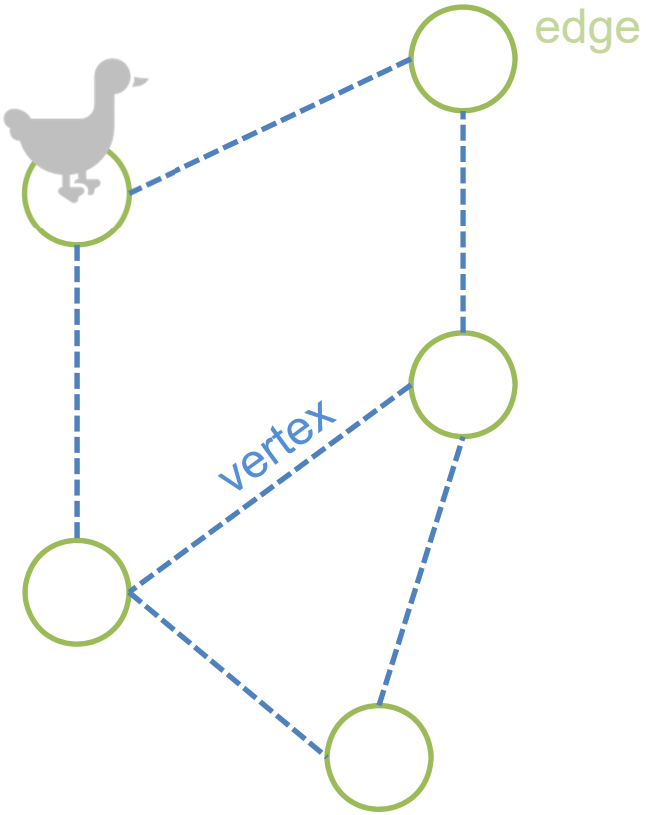
- Path planning vs. motion planning

The **path planning** is somewhat trivial. There's only one path: the rope. The **motion planning** on the other hand is not that easy.

However. In a maze the **path planning** is hard and **motion planning** is easy

Graph-based approaches

G=(V,E)



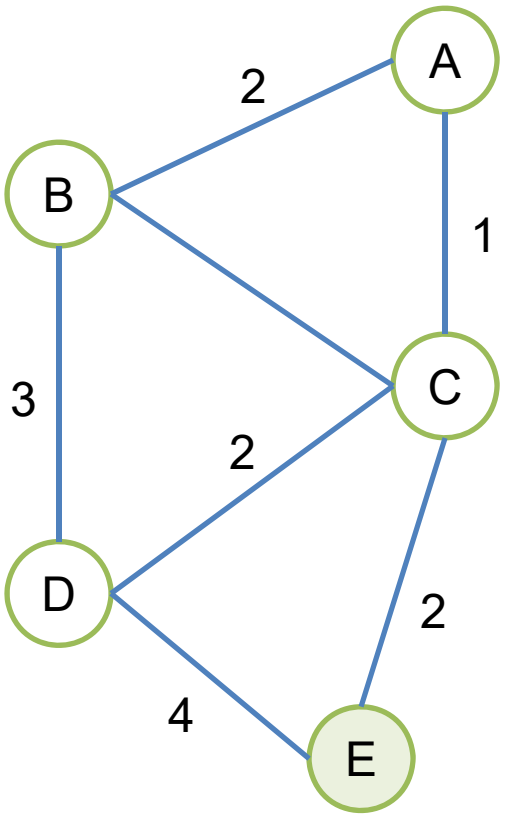
Djikstra's algorithm

A classic algorithm for finding the length of the shortest path in a weighted graph G=(V,E)

Inventor was Edsger Wybe Dijkstra (1930 –2002)

Dutch computer scientist from Netherlands

Djikstra's algorithm



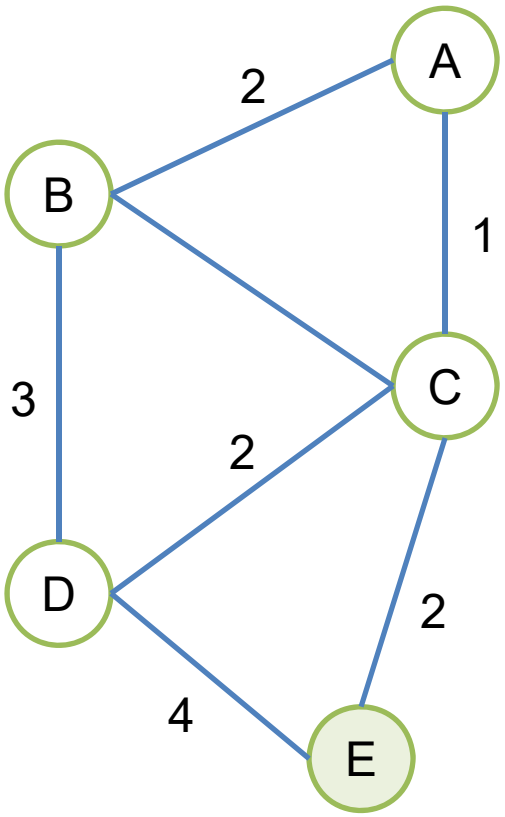
Main idea: Explore paths (vertices) with low cost first.

Keep track of visited nodes.

Keep track of shortest path from origin to edge.

Extension: Use a priority queue to select the paths with low cost.

Djikstra's algorithm



Lemma 1 - Optimal Substructure:

The sub-path of any shortest path is itself a shortest path.

Lemma 2 - Triangle inequality: If $\delta(u,v)$ is the shortest path length between u and v, then $\delta(u,v) \leq \delta(u,x) + \delta(x,v)$

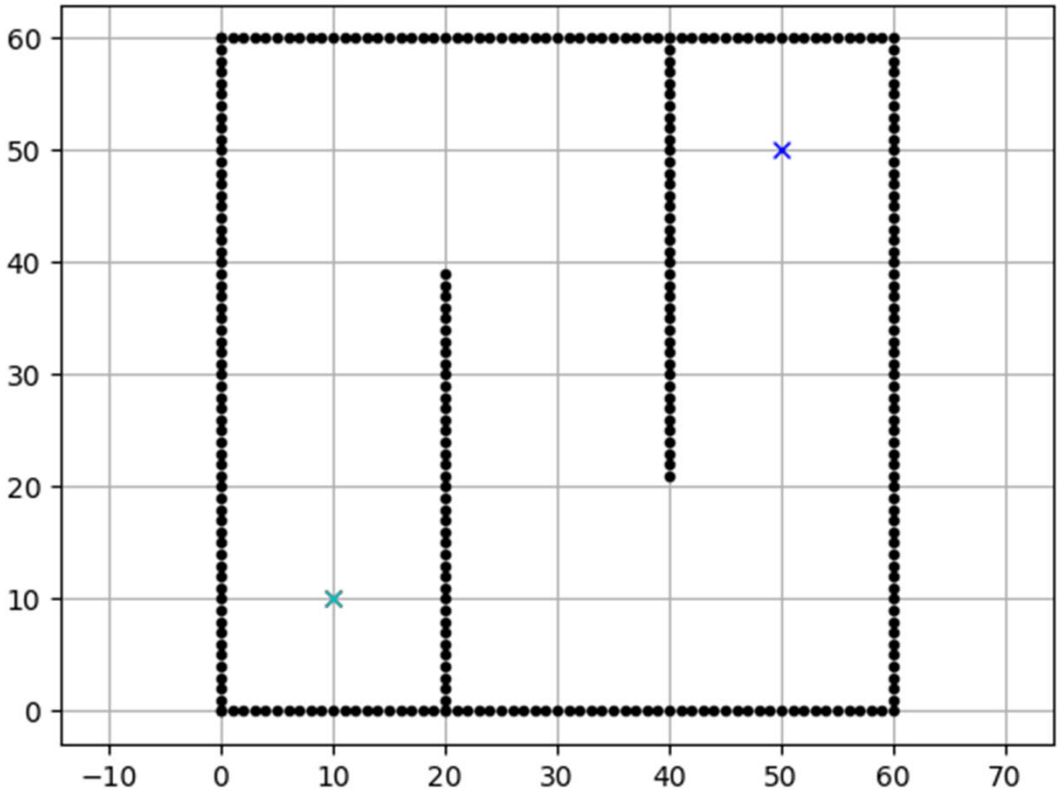
Djikstra's algorithm

```
dist[s] ←o                                  (distance to source vertex is zero)
for all v ∈ V–{s}
     do dist[v] ←∞                          (set all other distances to infinity)
S←∅                                         (S, the set of visited vertices is initially empty)
Q←V                                         (Q, the queue initially contains all vertices)
while Q ≠∅                                  (while the queue is not empty)
do  u ← mindistance(Q,dist)                 (select the element of Q with the min. distance)
    S←S∪{u}                                 (add u to list of visited vertices)
     for all v ∈ neighbors[u]
          do if  dist[v] > dist[u] + w(u, v)        (if new shortest path found)
                then     d[v] ←d[u] + w(u, v)      (set new value of shortest path)
                                                   (if desired, add traceback code)
return dist
```

Source: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf

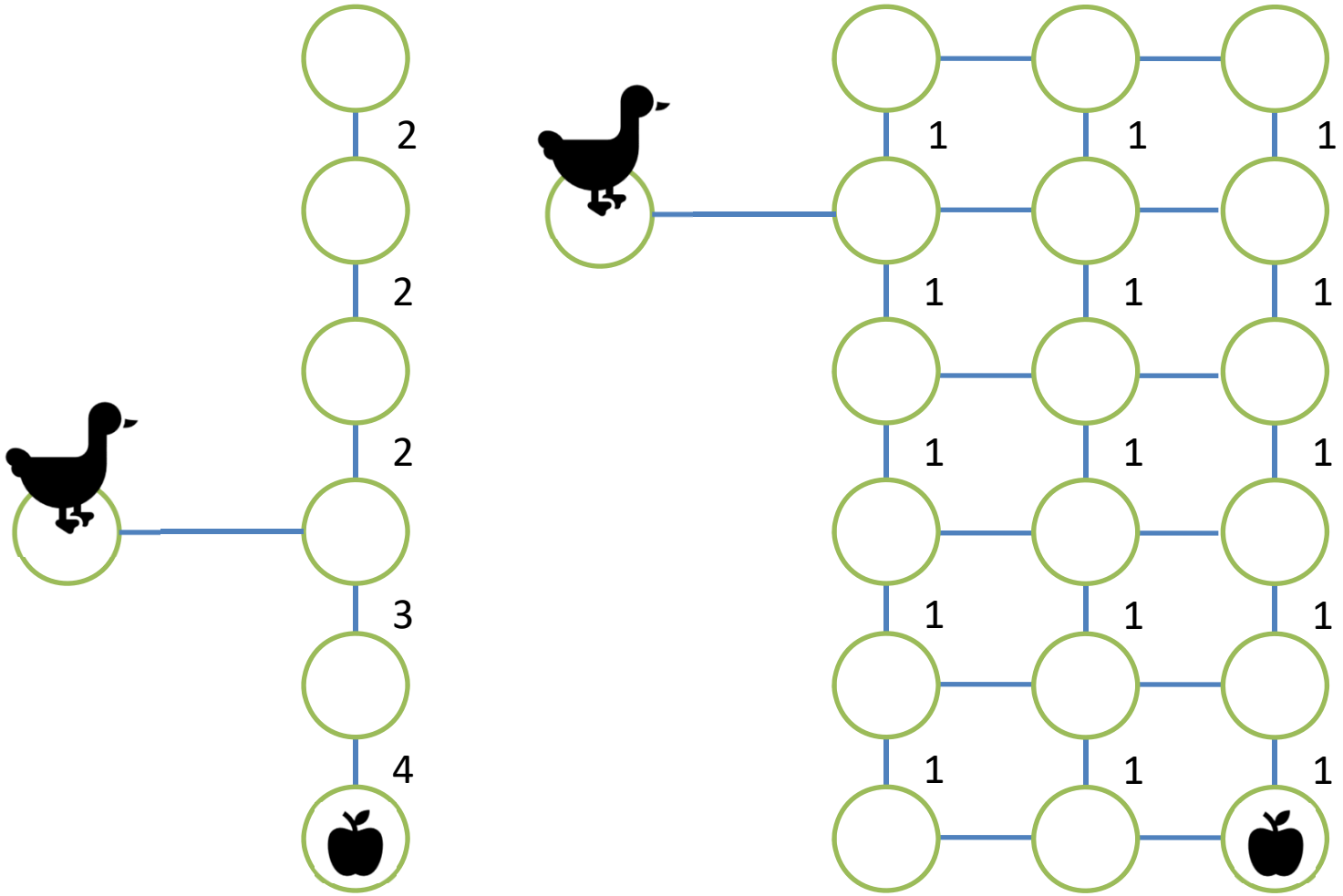
See a calculation example: LINK

Djikstra's algorithm

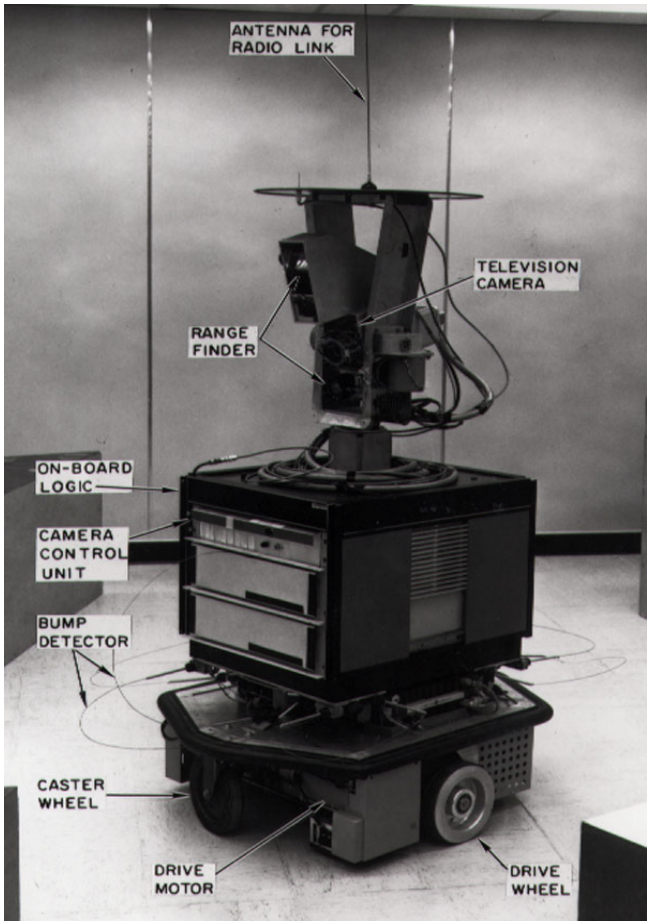


Source: https://atsushisakai.github.io/PythonRobotics

Djikstra's algorithm - problem



A* (A-star) algorithm



An extension to solve weaknesses with Djikstra's.

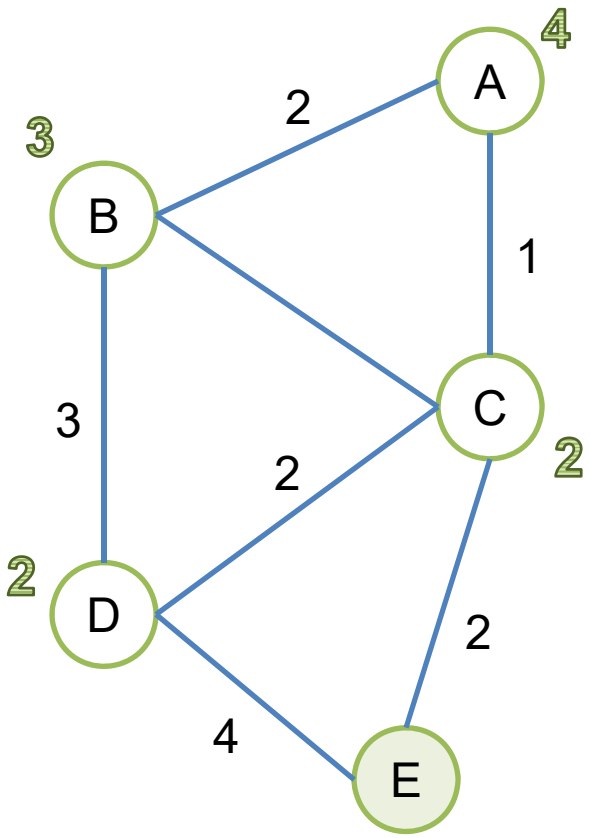
Invented by Peter Hart, Nils Nilsson and Bertram Raphael from the Stanford Research Institute.

First published the algorithm in 1968

https://en.wikipedia.org/wiki/A*_search_algorithm

Source: https://en.wikipedia.org/wiki/Shakey_the_robot

A* (A-star) algorithm

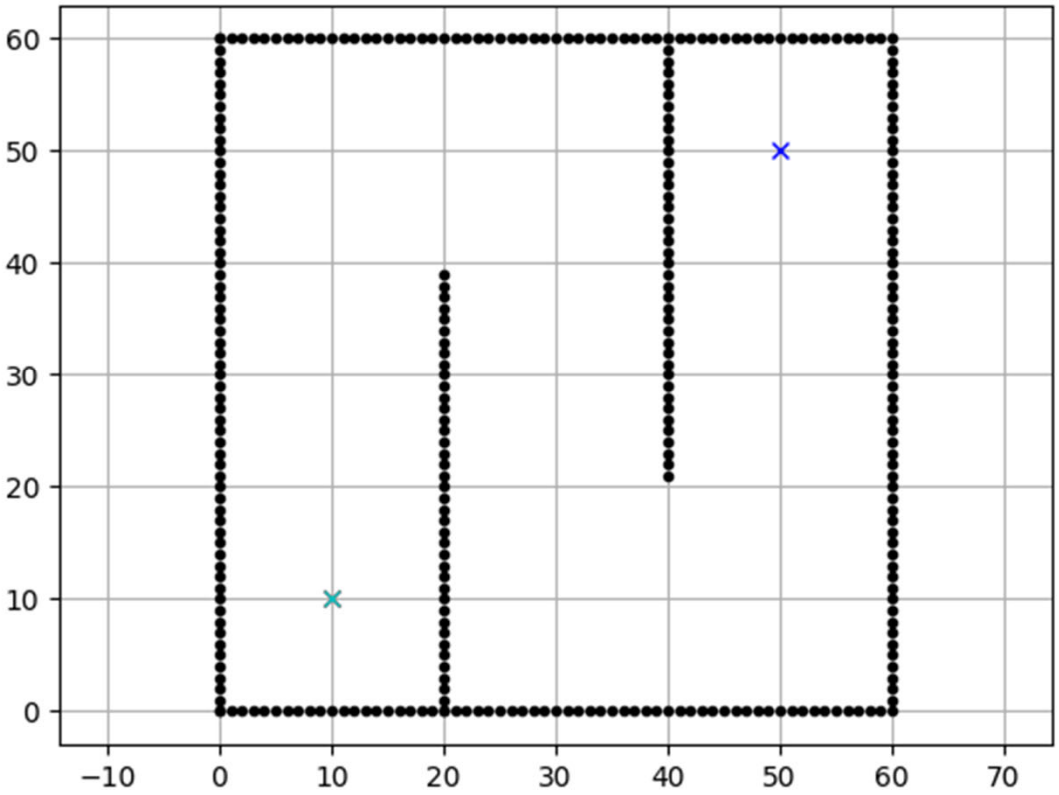


Main idea: Make a combined distance metric (heuristic) that encodes path/vertex distance and distance to target.

$$combined\ dist_{A^*} = dist_{vertex}[v_i] + dist_{tar}[v_i, t_i]$$

See a calculation example: LINK

A* (A-star) algorithm

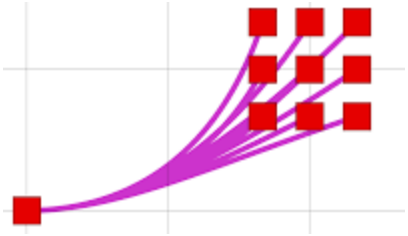


Source: https://atsushisakai.github.io/PythonRobotics

Other graph-based methods

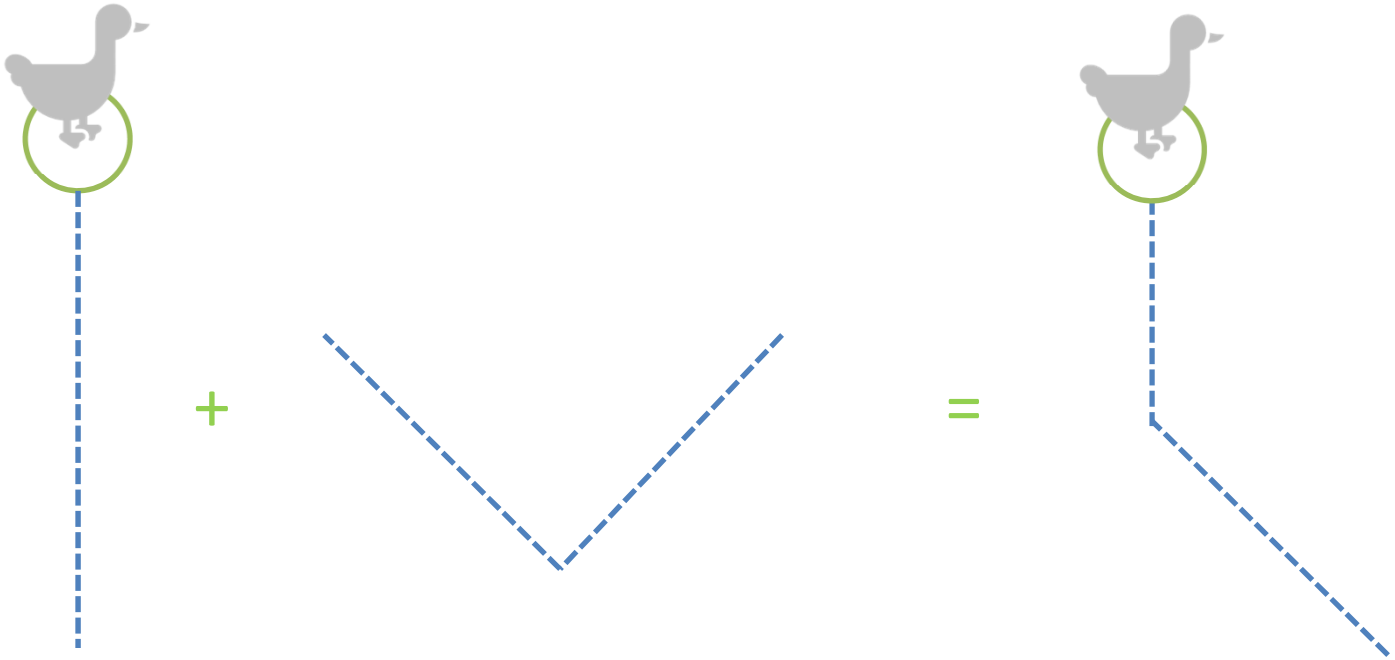
- D* (D-star) (dynamic A*) modern adaption and extension of A* that starts at the goal node and uses back-pointers; dynamically changes vertex weights/costs.

- We can also mention here, state lattice planning (used for example in obstacle avoidance)



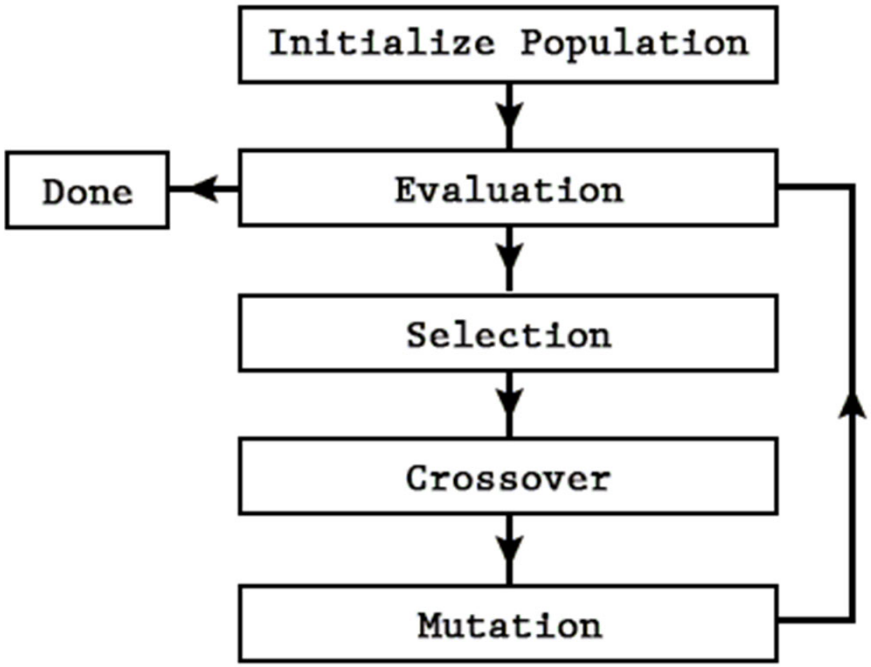
- Read more here: LINK

Genetic and random algorithms



Genetic algorithms

Main idea: Generate a solution, evaluate, combine/mutate, add some randomness, and iterate as long as you have resources. Sometimes called “evolutionary algorithms”.



Genetic algorithms in practice

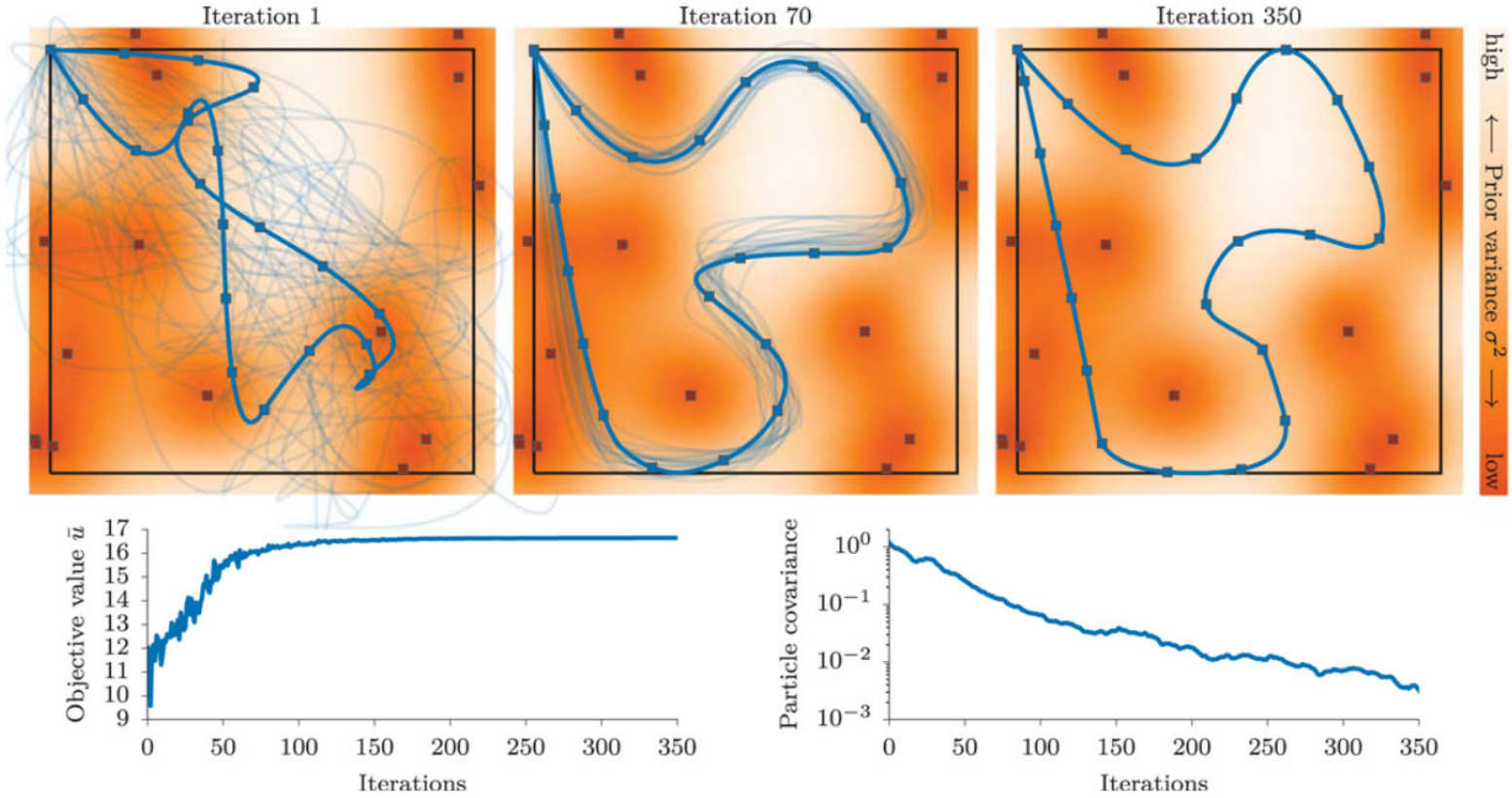
REGULAR ARTICLE

WILEY

Adaptive continuous-space informative path planning for online environmental monitoring

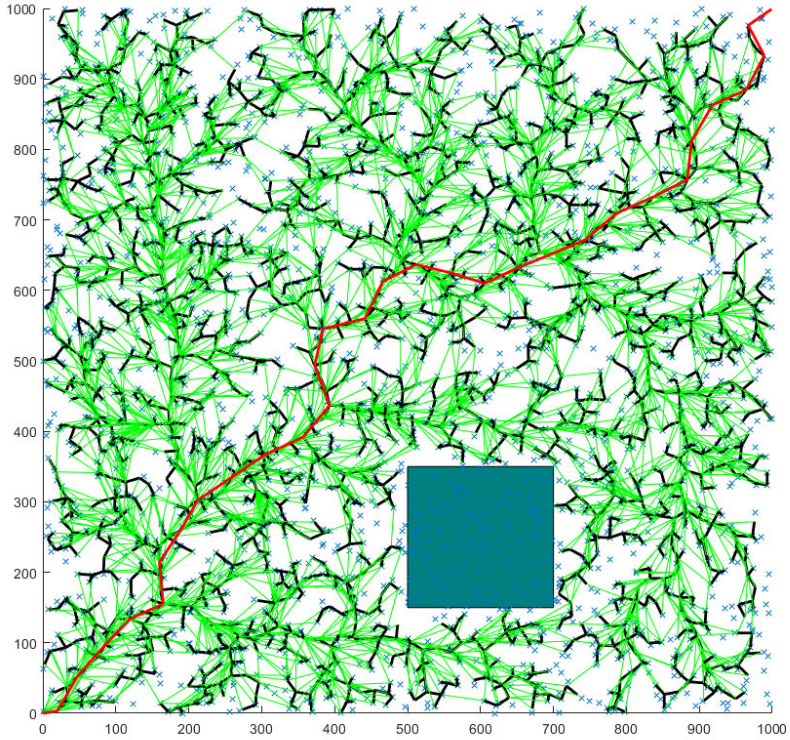
Gregory Hitz[1] | Enric Galceran[1] | Marie-Ève Garneau[2] | François Pomerleau[3] | Roland Siegwart[1]

Hitz, G., Siegwart, R., Galceran, E., Garneau, M.-ève, & Pomerleau, F. (2017). Adaptive continuous-space informative path planning for online environmental monitoring. (November 2015), 1427–1449. https://doi.org/10.1002/rob.21722



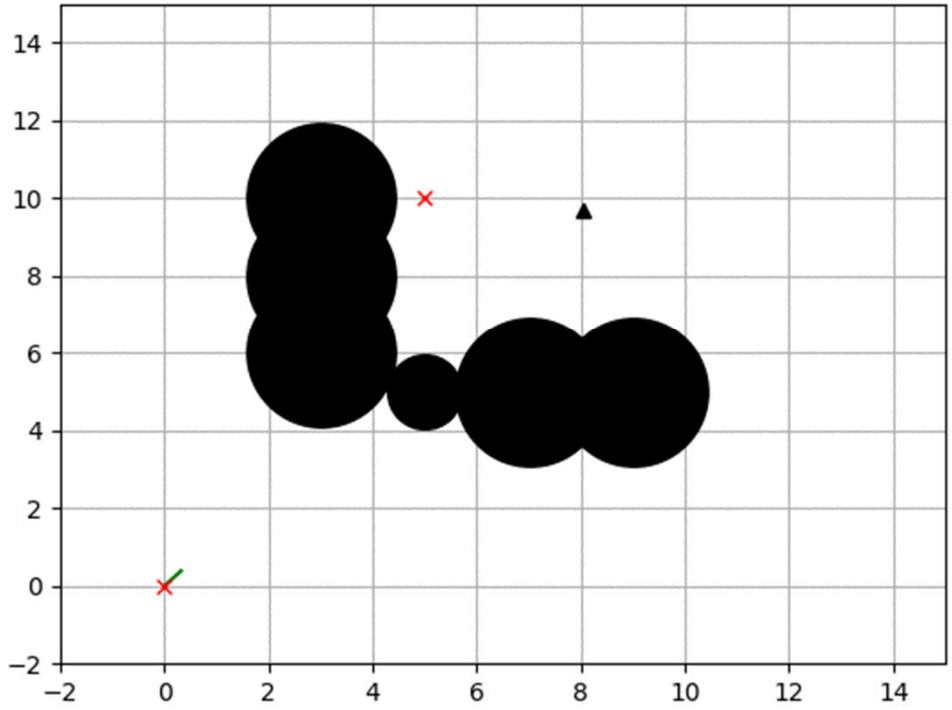
Random based algorithms

Main idea: Use a random search to iteratively explore or exploit results that can be used to reason about the fitness of a path.



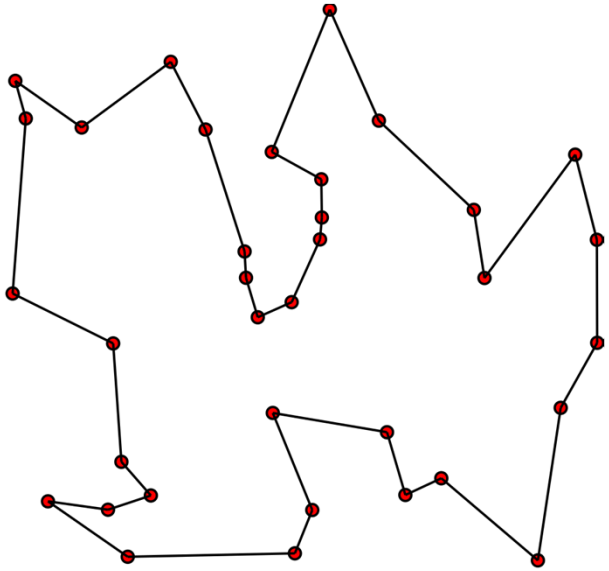
Random based algorithms

Rapidly-exploring random tree (RRT)



Source: https://atsushisakai.github.io/PythonRobotics

Traveling salesman problem

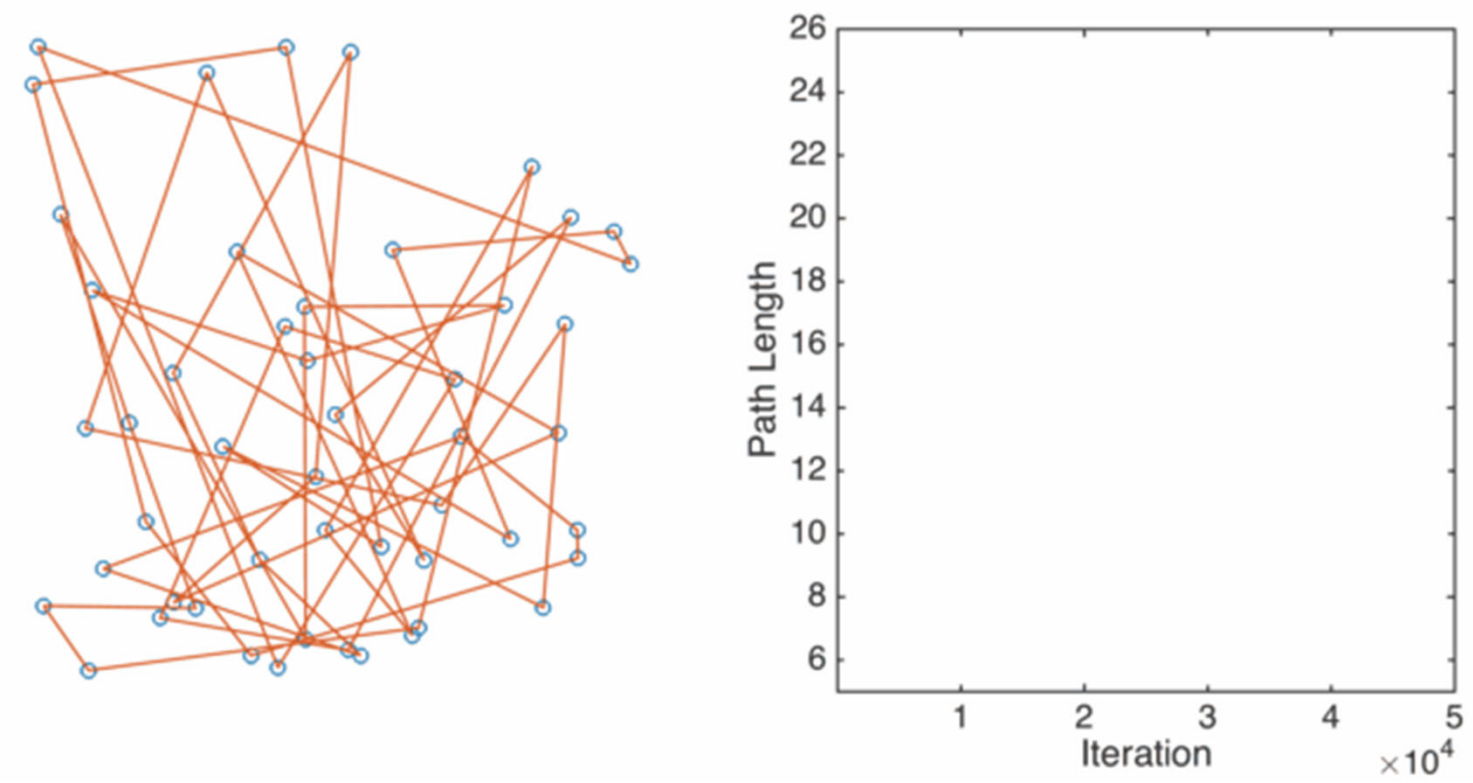


The **travelling salesman problem** (**TSP**) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

Source: https://en.wikipedia.org/wiki/Travelling_salesman_problem

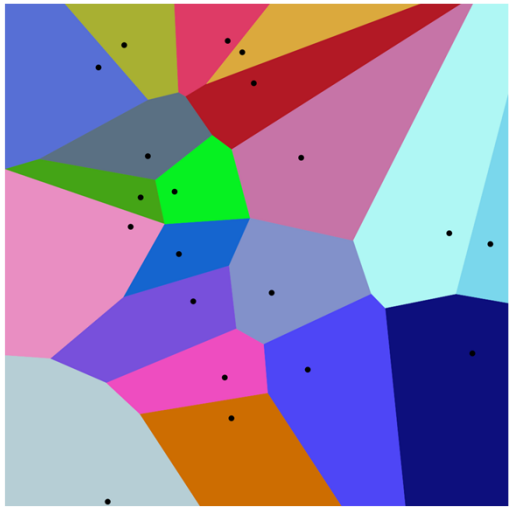
Random based algorithms

Simulated annealing to solve TSP

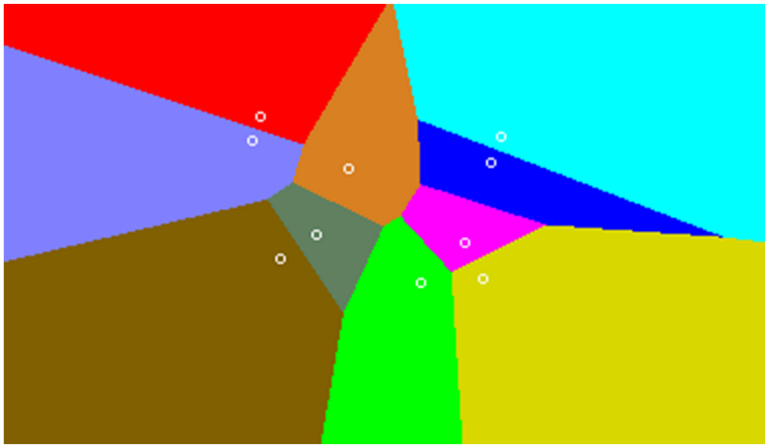


Other elements: Voronoi

Voronoi diagrams: a Voronoi diagram is a partitioning of a plane into regions based on **distances to points in a specific subset of the plane**. That set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points closer to that seed than to any other. These regions are called Voronoi cells.

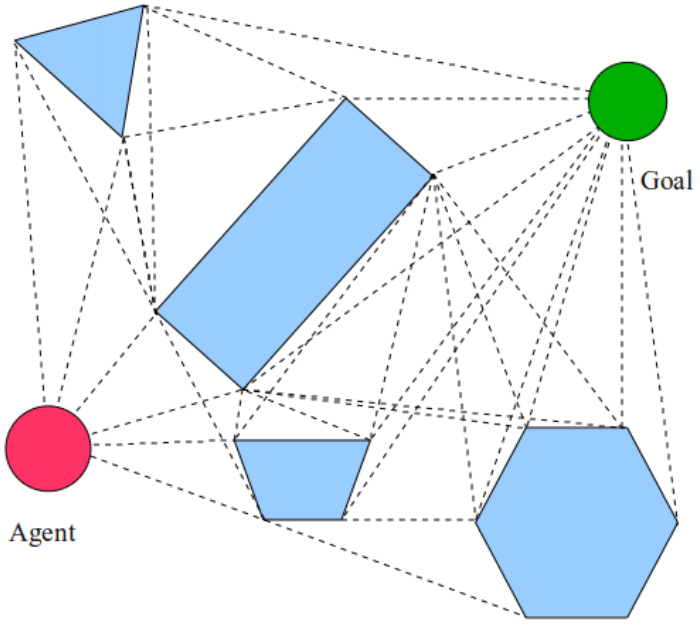


Source: https://en.wikipedia.org/wiki/Voronoi_diagram



Other elements: Visibility graph

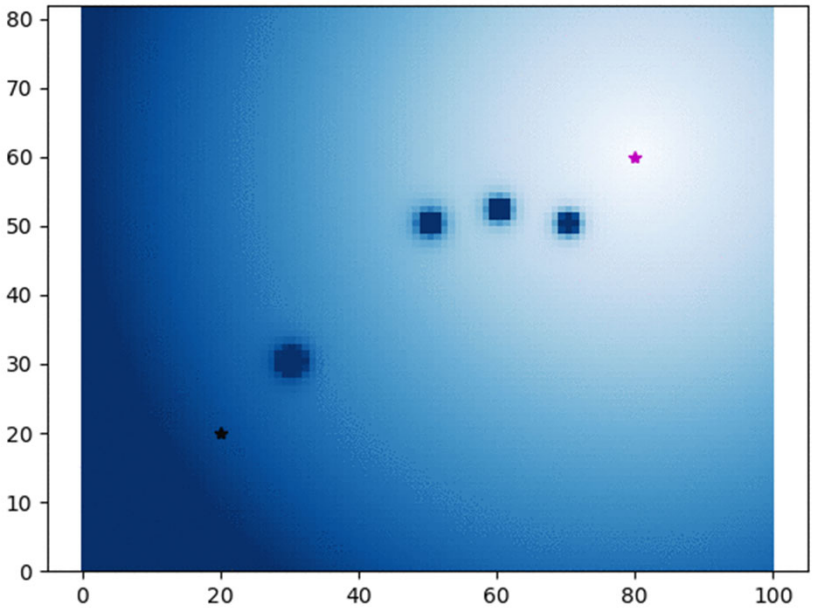
Visibility graph: all non-blocked lines between polygon vertices, start and goal. To find a path search the graph of these lines for the shortest path using e.g. Djikstra's algorithm.



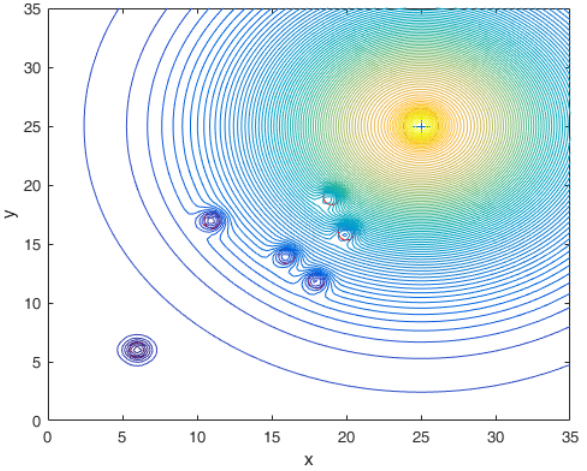
Source:
https://en.wikipedia.org/wiki/Visibility_graph

Other elements: Potential field

Potential field methods: A using a virtual potential field one can *create trajectories based on the gradients*. An obstacle can be given positive potential (source), while a goal may be given negative potential (sink).

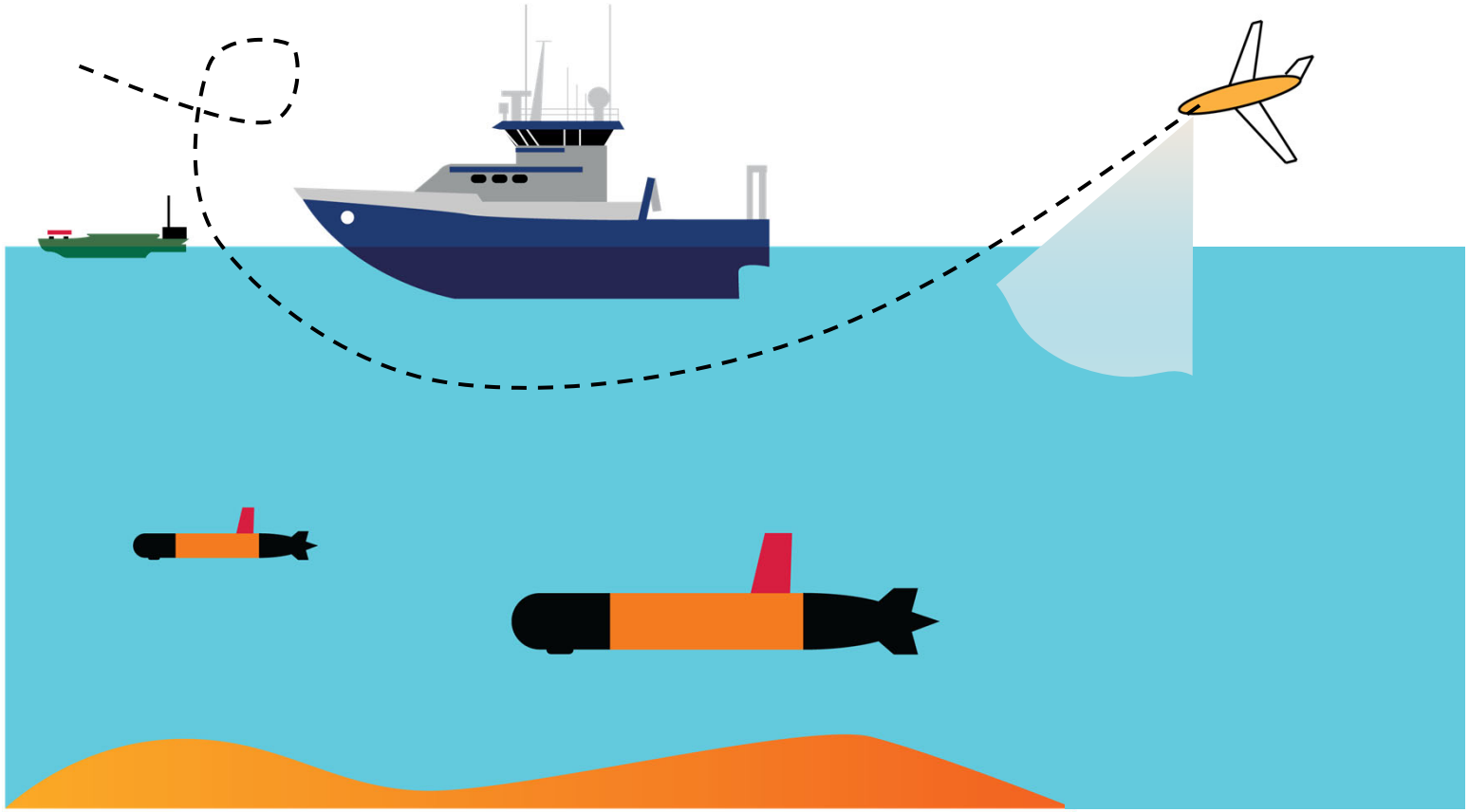


Source:
https://atsushisakai.github.io/PythonRobotics



Part 2 Marine applications

Path planning for marine applications



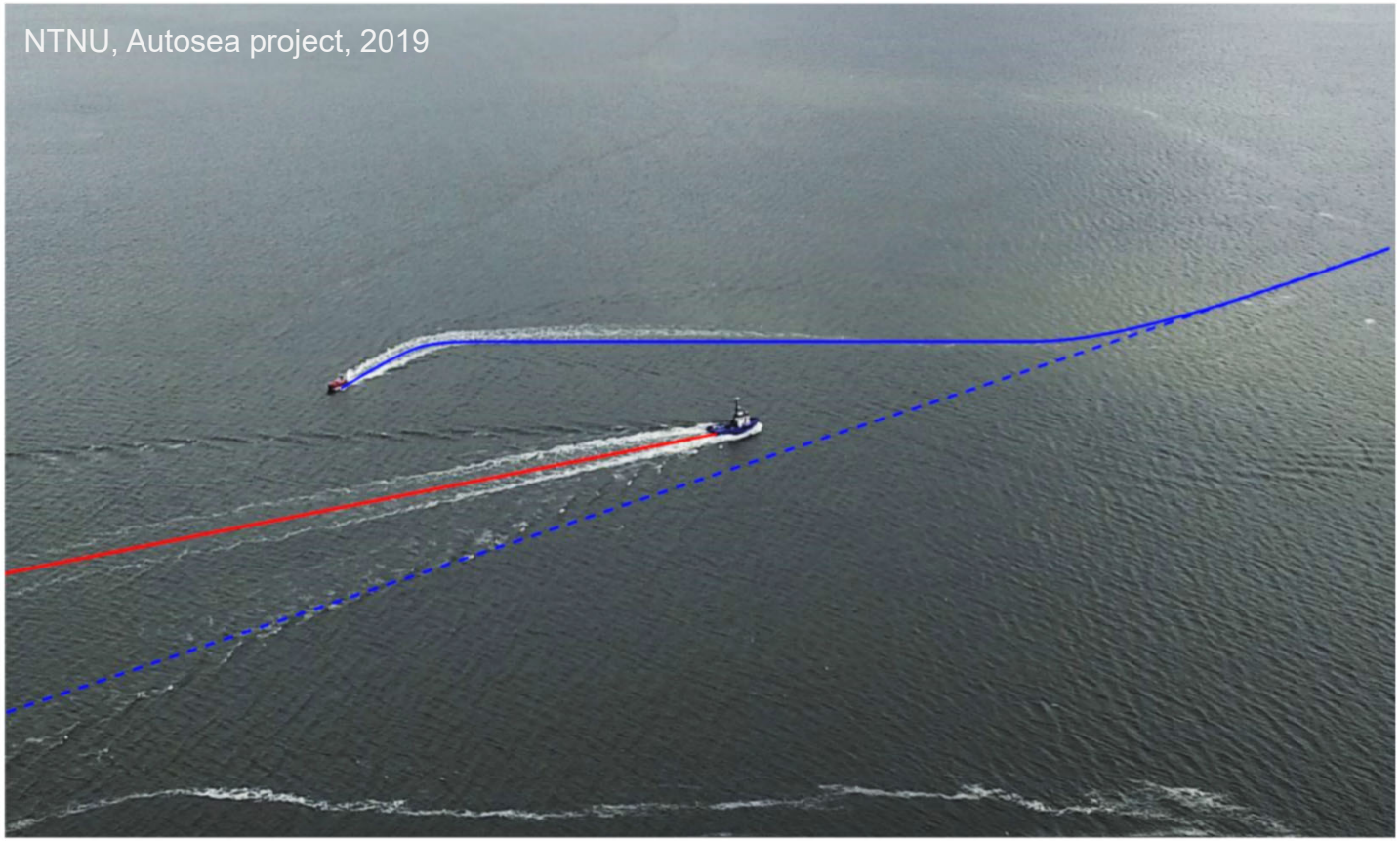
App. 1: Obstacle avoidance

In order to achieve the goal of autonomous ship traffic a few central technologies needs to be in place:

1. Local (on board) navigation and positioning
2. Situational awareness (global/relative positioning)
    1. Detection sensors (radar, AIS, cameras, lidar, acoustics, etc.)
    2. Environment representations (world model) and maps that can assimilate data and be used to make inference.
    3. Autonomy that can use the information from the world model to make decisions.
        - Obstacle avoidance

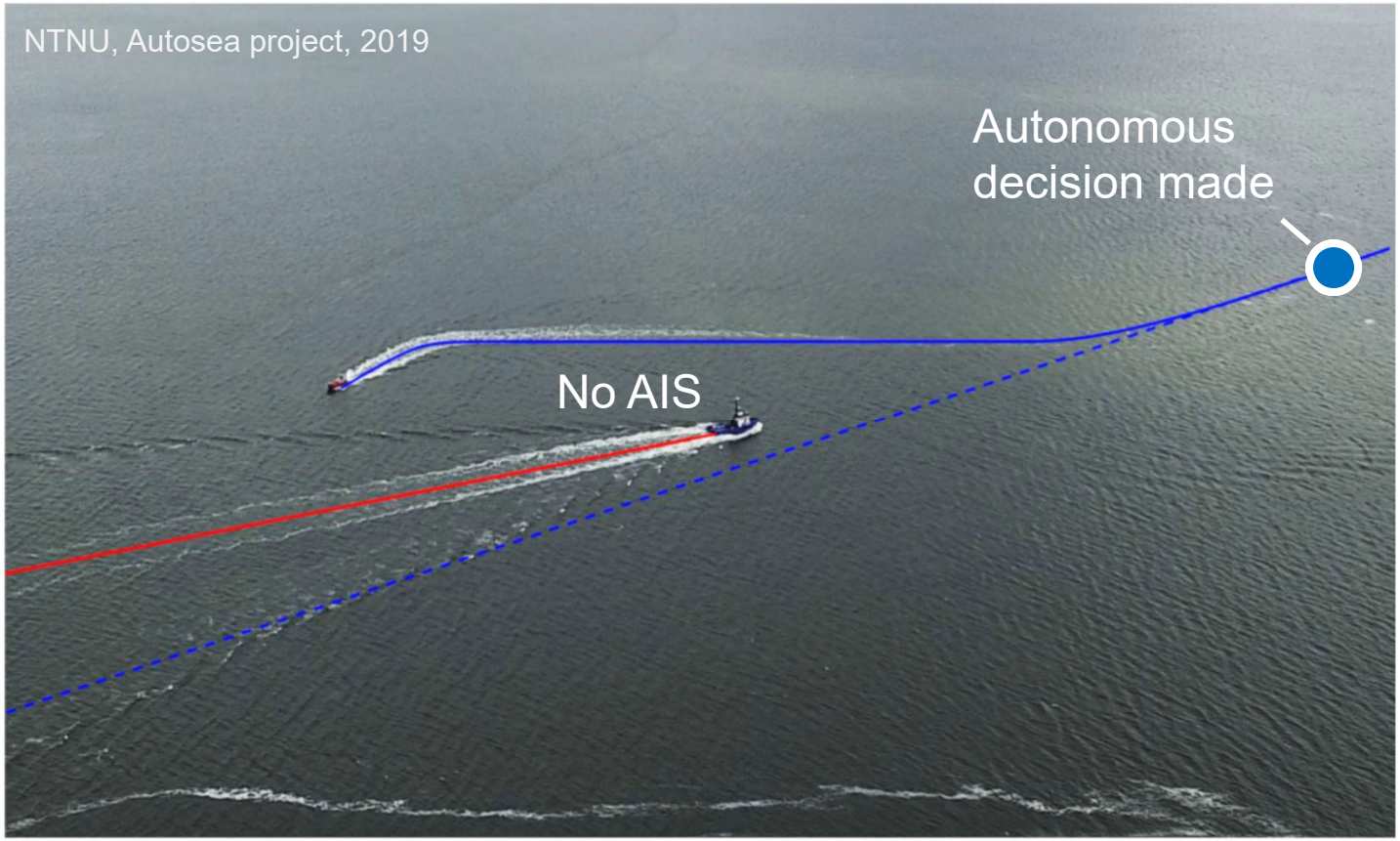
App. 1: Obstacle avoidance

Reactive behavior and motion control laws working together



App. 1: Obstacle avoidance

Reactive behavior and motion control laws working together

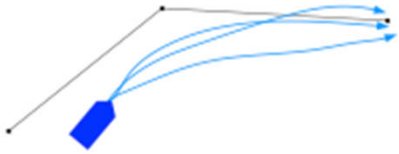


App. 1: Obstacle avoidance

Types of uncertainty that one has to deal with

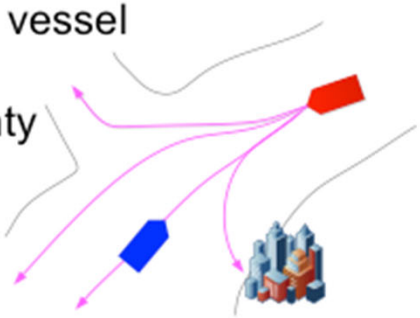
1) Kinematic uncertainty

3) Ownship guidance uncertainty



2) Data association uncertainty

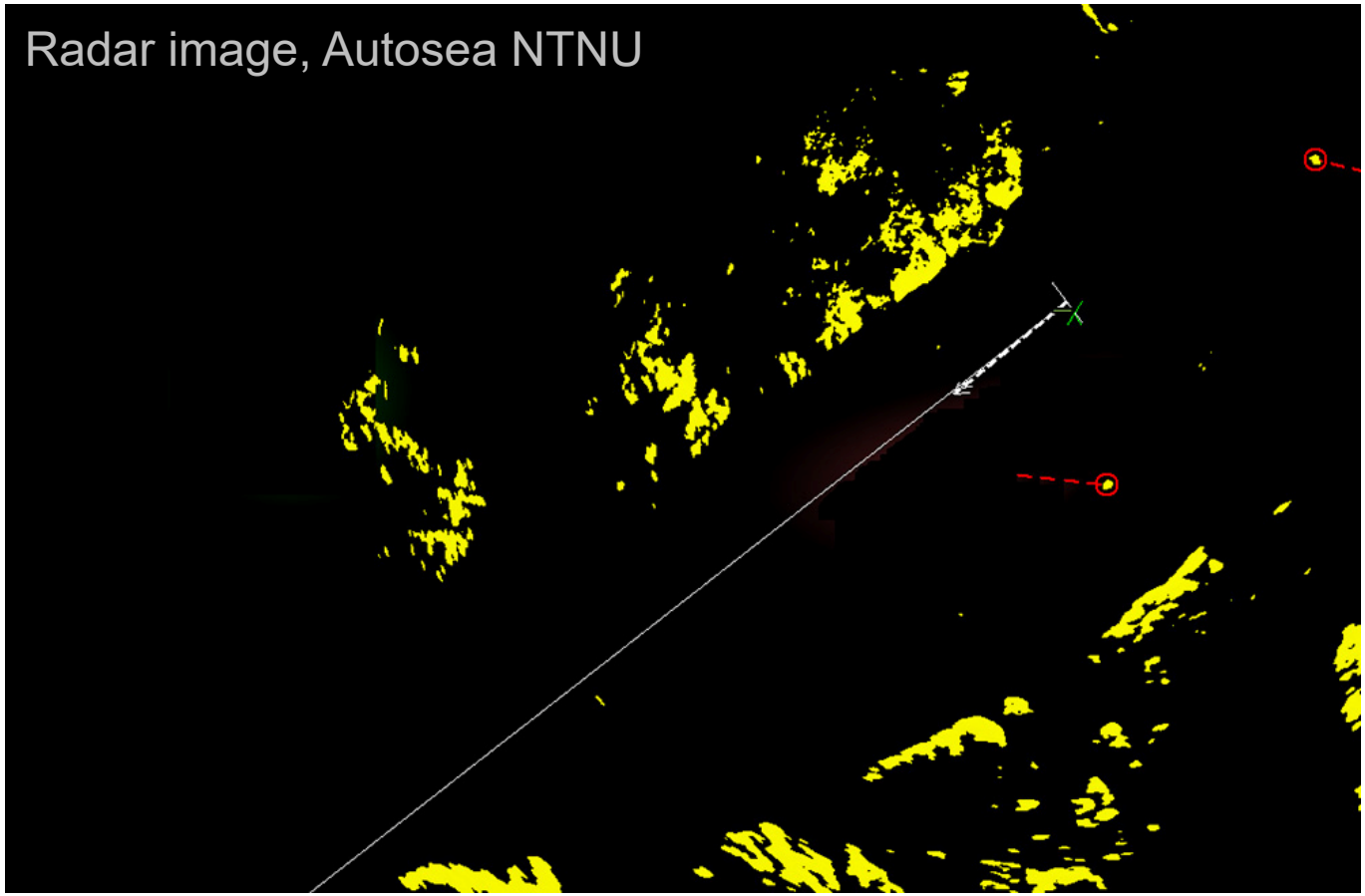
4) Target vessel intention uncertainty



NTNU, Autosea project, 2019

App. 1: Obstacle avoidance

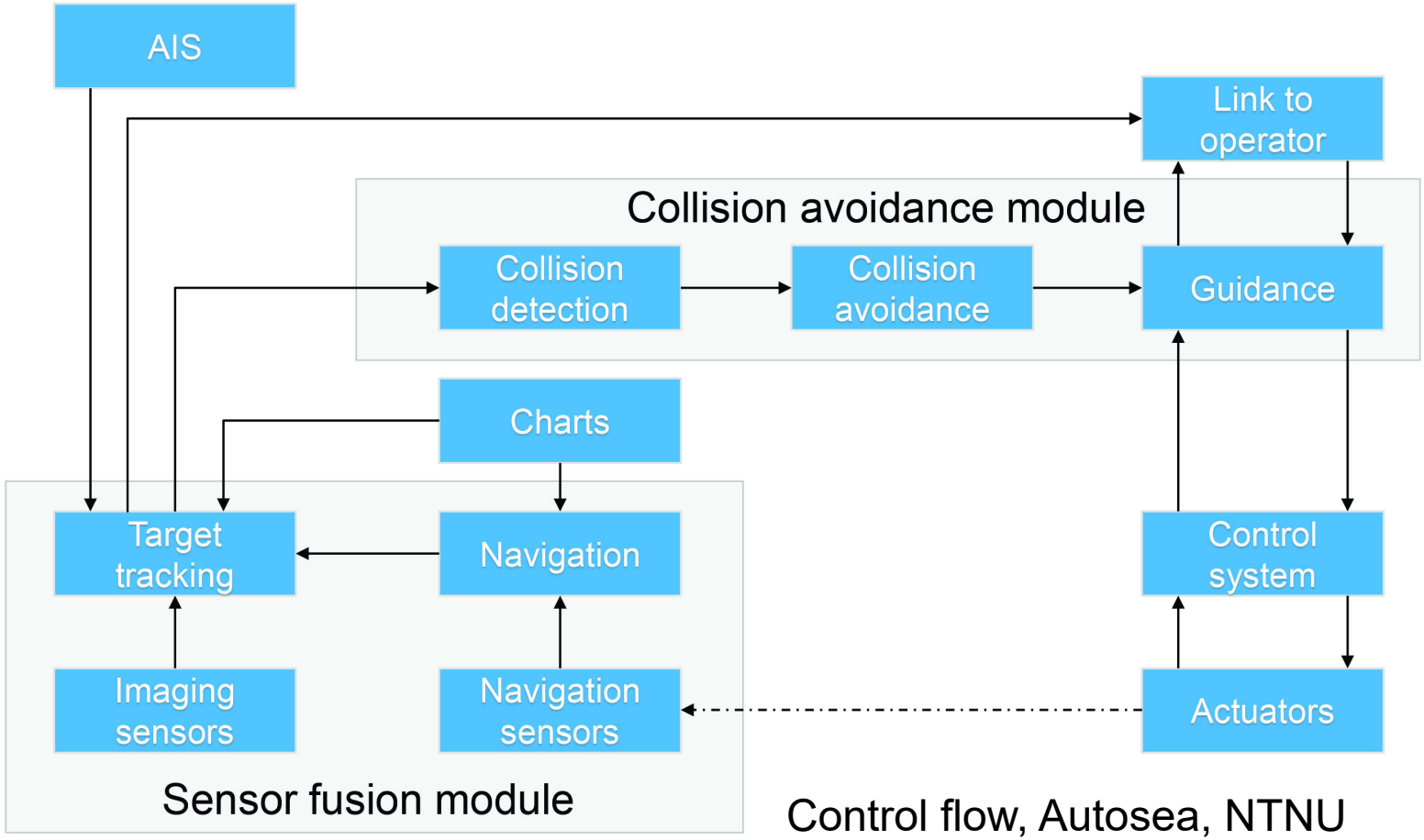
Need to combine several sensors in order to be **robust**



App. 1: Obstacle avoidance

Need to combine several sensors in order to be **robust**

App. 1: Obstacle avoidance



Control flow, Autosea, NTNU

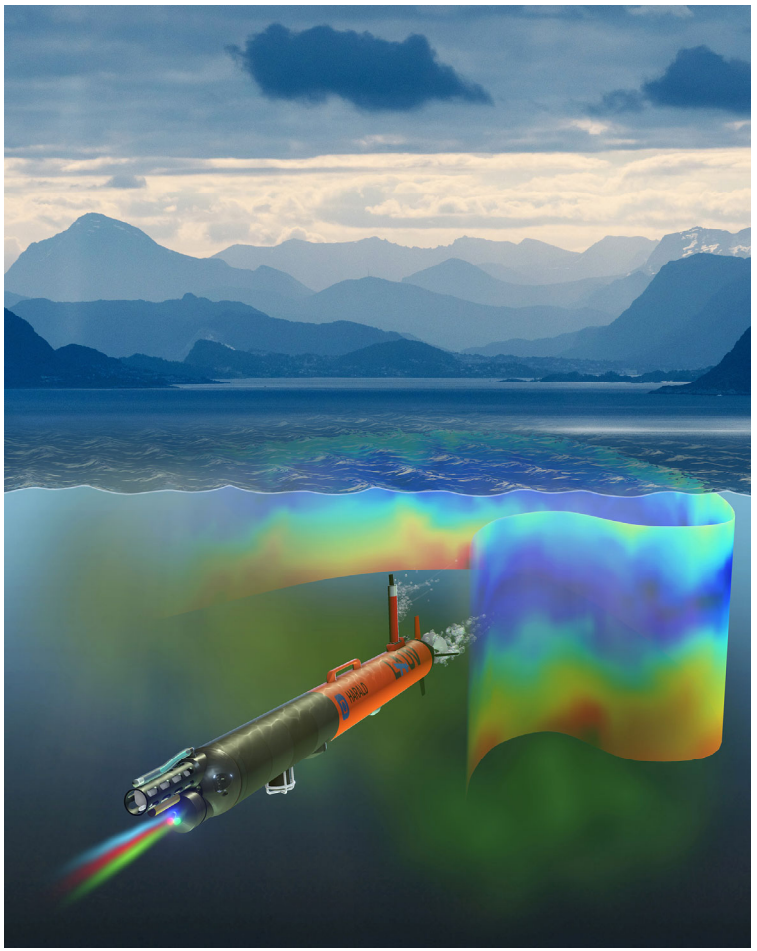
App. 2: Adaptive sampling

In order to effectively explore the ocean (general term is environmental sampling) we need specialized autonomy that can prioritize and select between sampling locations.

We must prioritize as we cannot sample the entire domain (too large).

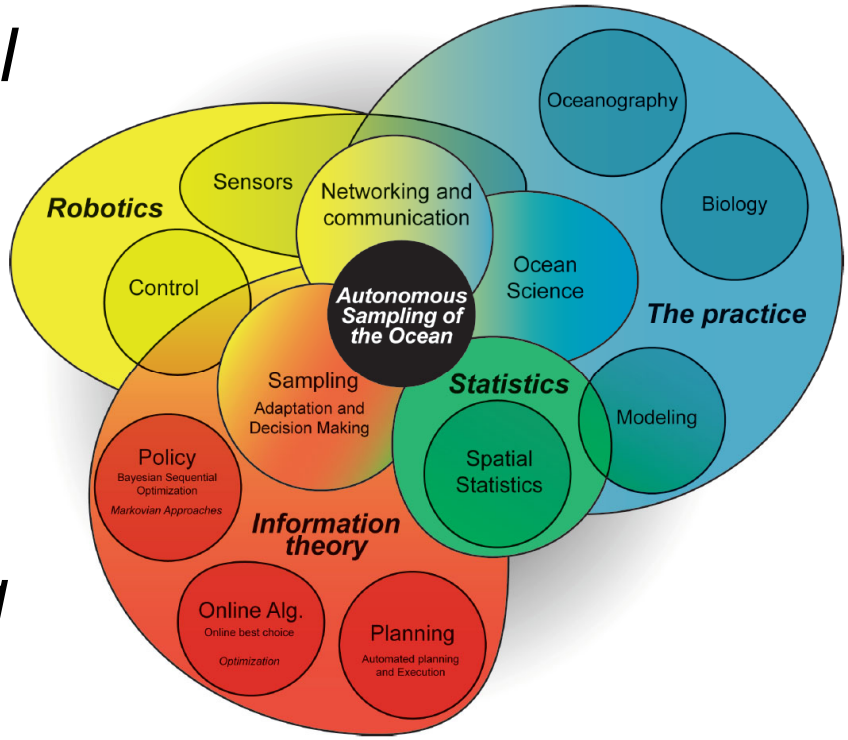
We must adapt as information/features are moving around (dynamic environment) in both space and time (spatiotemporal).

The system (agent) needs to be able to reason about the environment and the value of information.

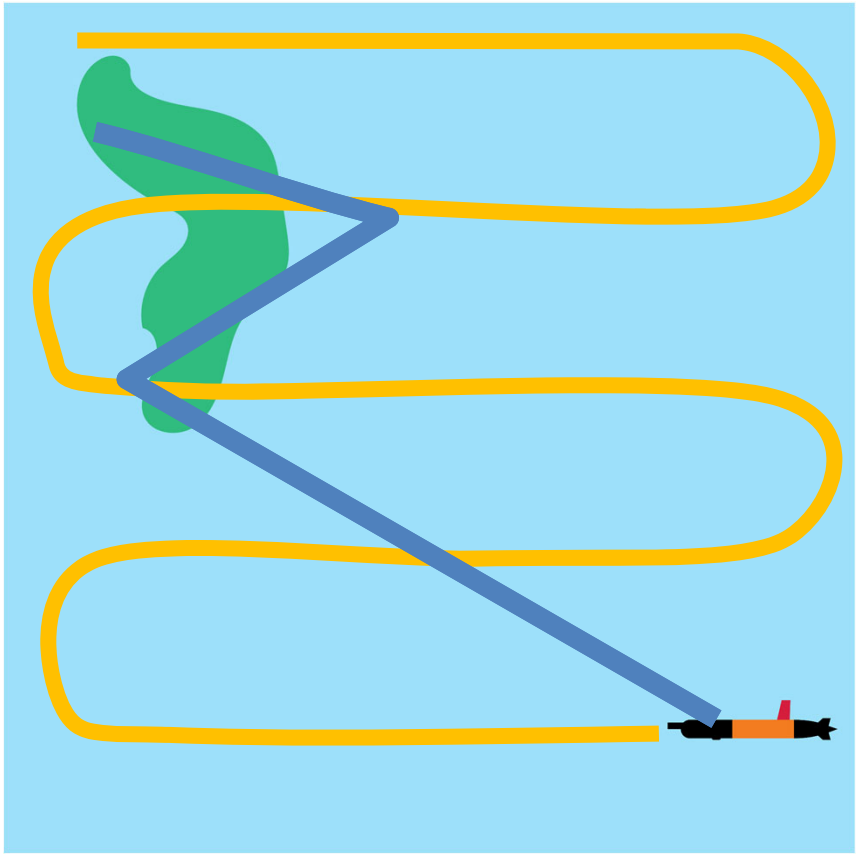


App. 2: Adaptive sampling

The overarching goal is to provide cost effective tools, techniques, and processes for doing ocean based measurements using robotic platforms.

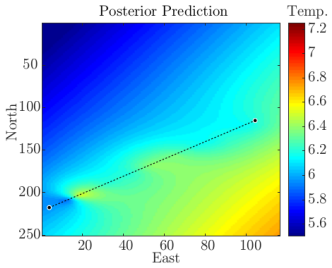


App. 2: Adaptive sampling

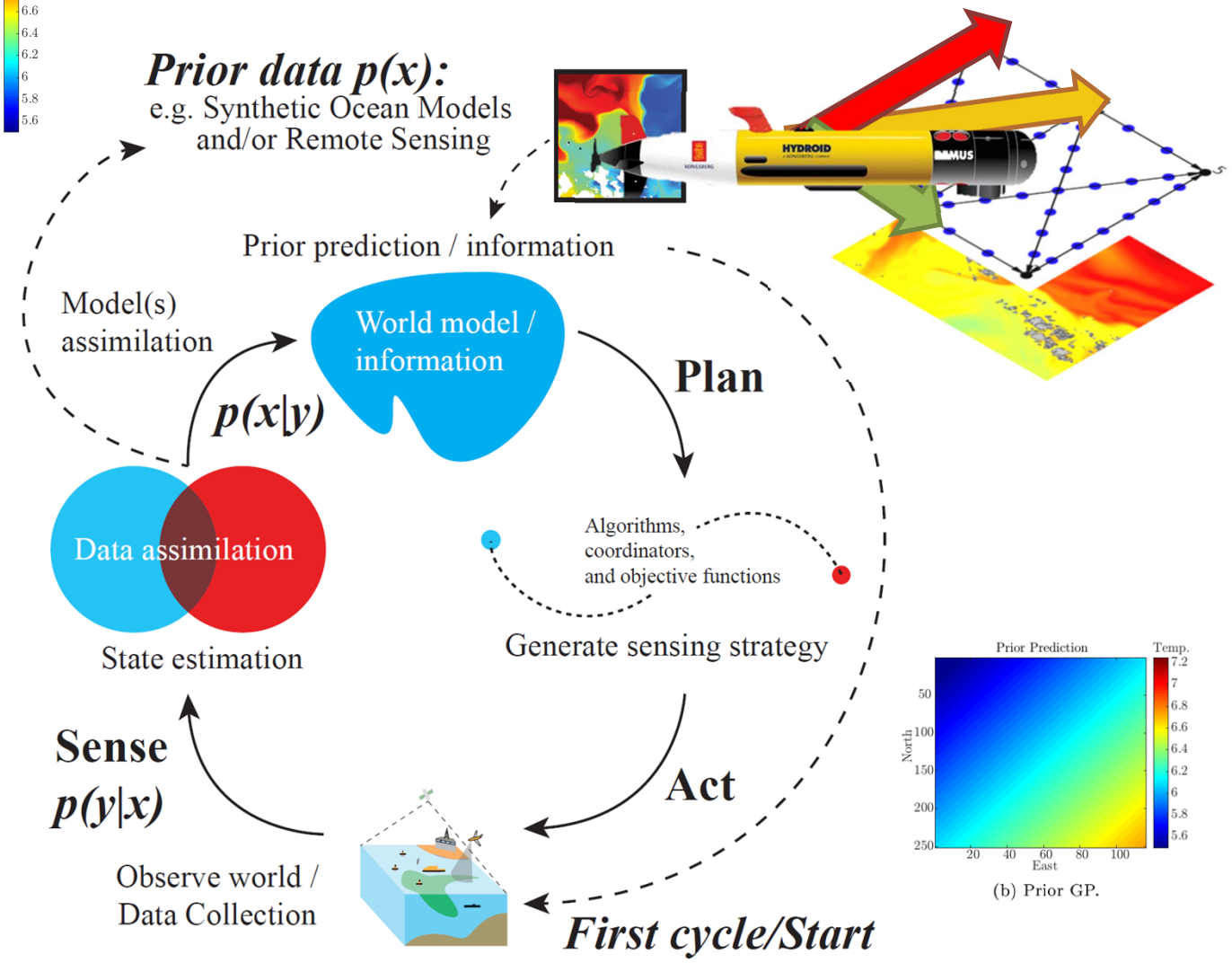


The informative path planning problem:

> ***"Where and when should we measure in order to effectively retrieve relevant and useful information?"***



(c) Posterior GP.



(b) Prior GP.

Notable references

Djikstra (numberphile): https://www.youtube.com/watch?v=GazC3A4OQTE

A* (numberphile): https://www.youtube.com/watch?v=ySN5Wnu88nE

Levy flight (Wikipedia): https://en.wikipedia.org/wiki/L%C3%A9vy_flight

Difference bet. path planning and motion planning (stack overflow): https://robotics.stackexchange.com/questions/8302/what-is-the-difference-between-path-planning-and-motion-planning

Robotic algorithms for python (github): https://atsushisakai.github.io/PythonRobotics

More on graph-based methods and path planning (article): http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.3884&rep=rep1&type=pdf

Autosea project (homepage): https://www.ntnu.edu/autosea