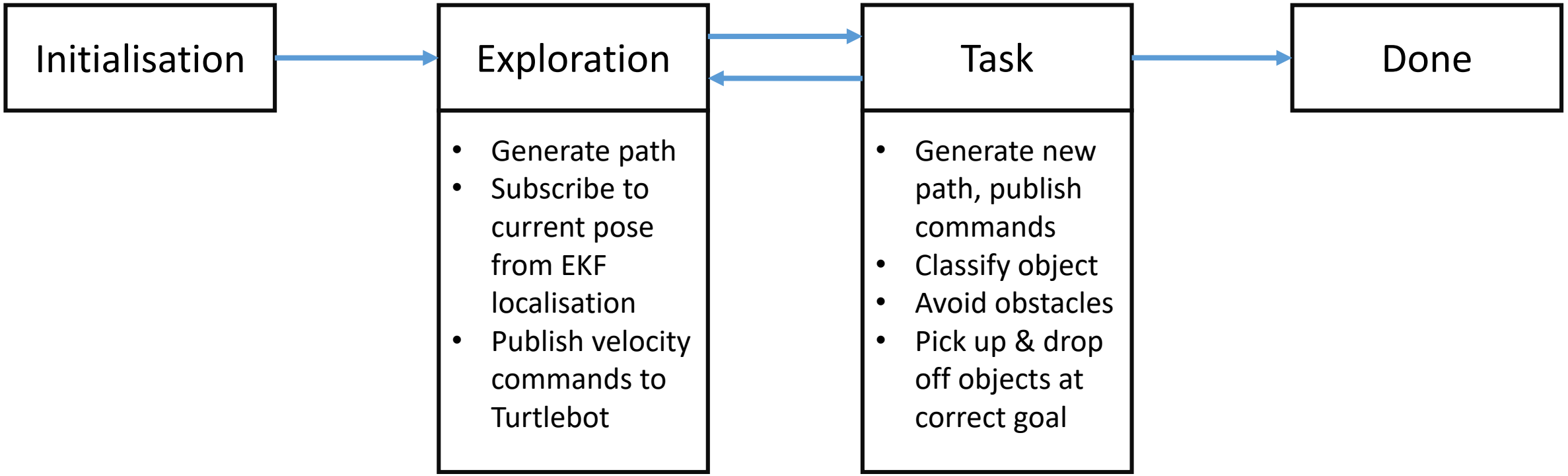
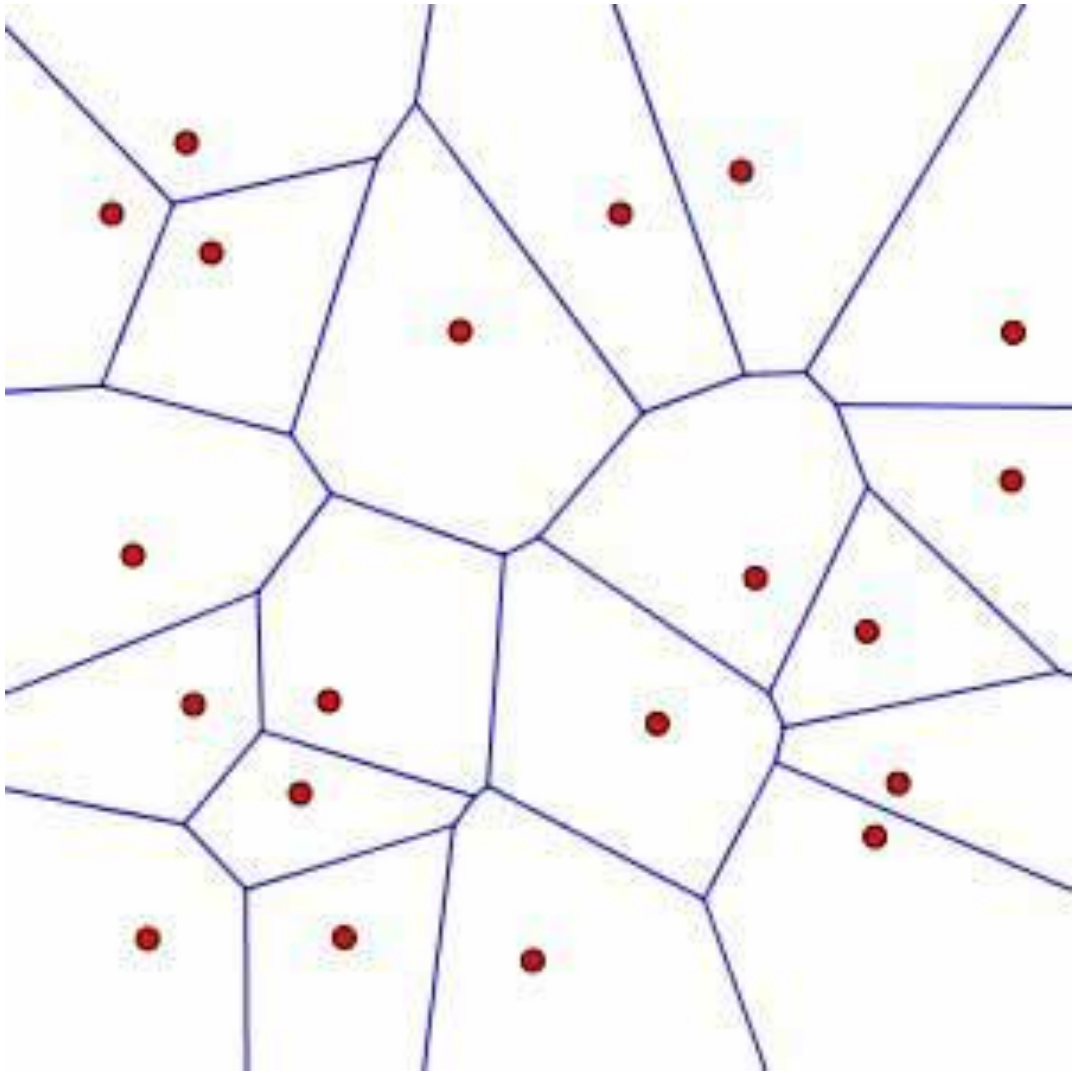
State Machine

RVSS_ws/src/rvss_workshop/scripts/state_machine.py

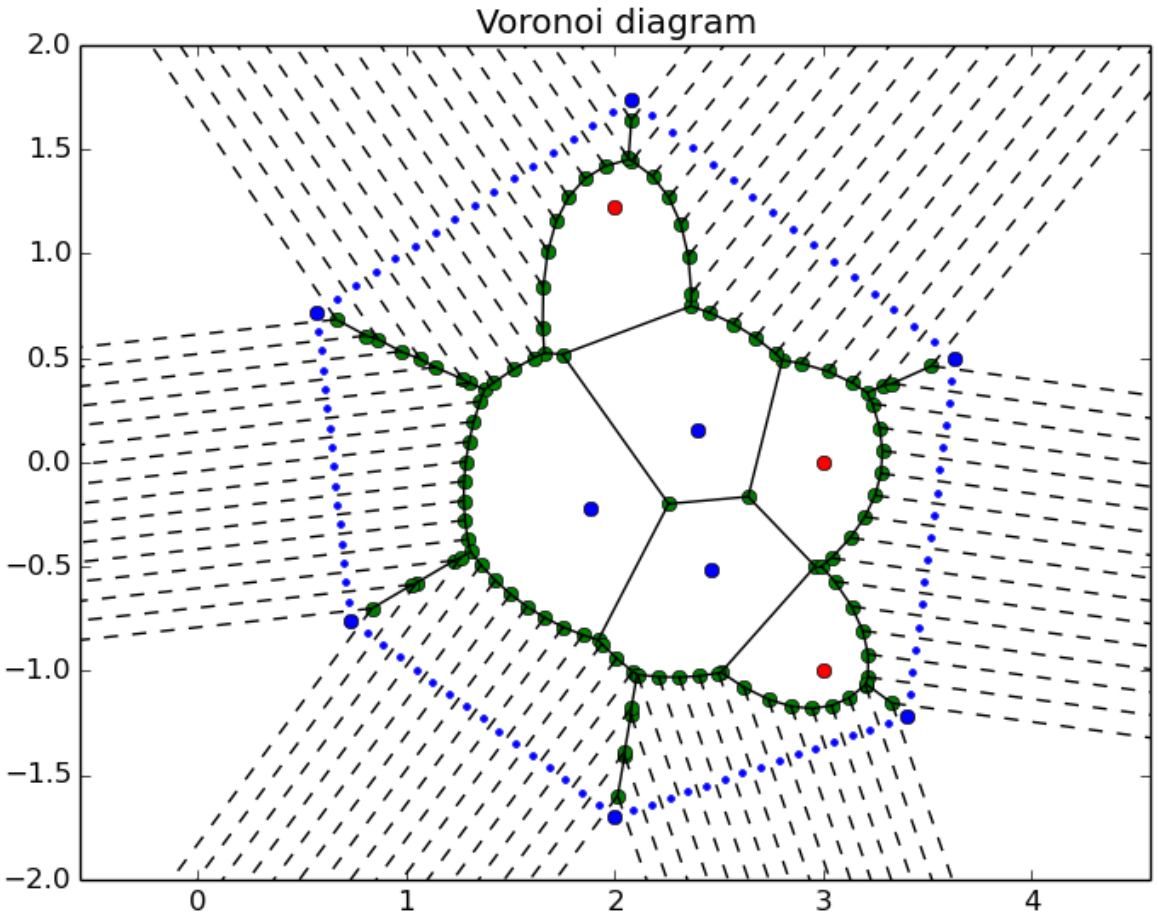


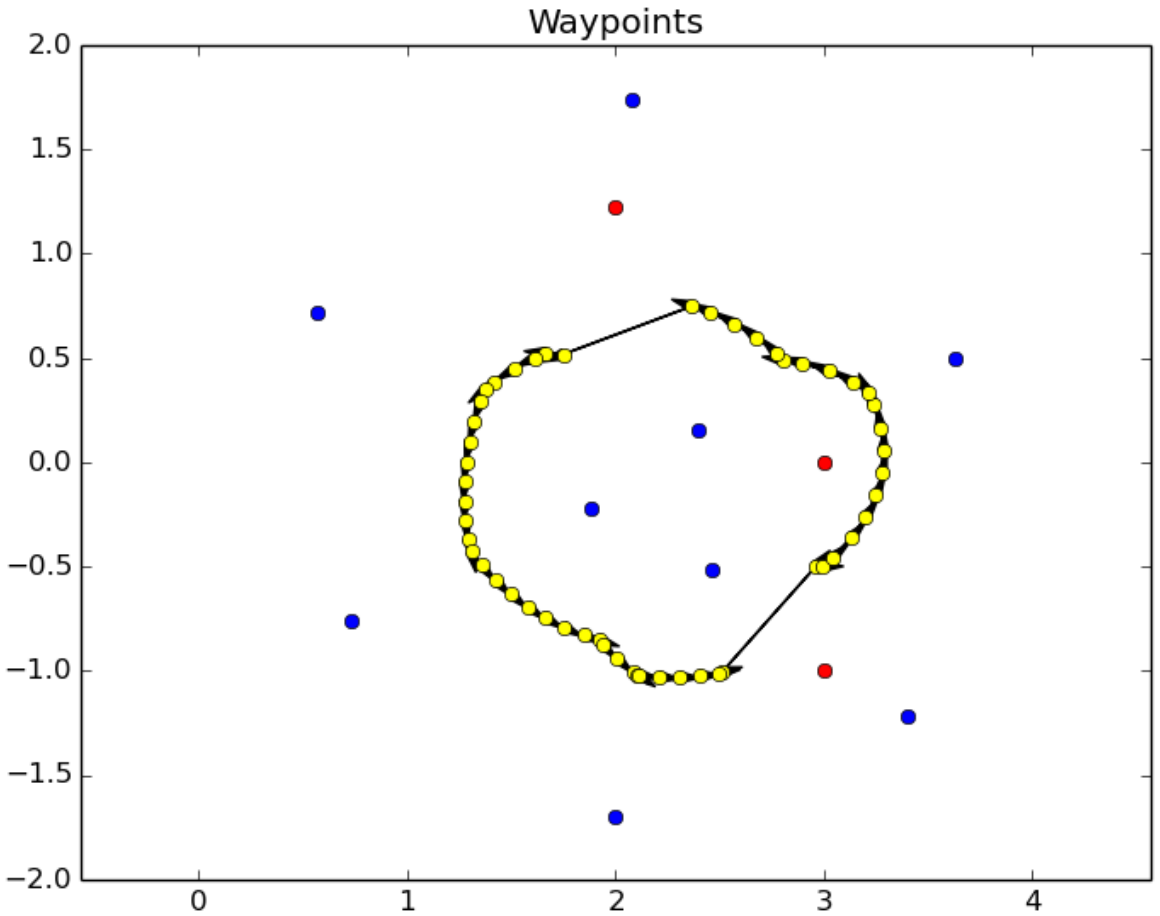
Exploration

- Voronoi diagram – partitioning of plane into regions based on distance to seed
- Each region consists of all points closest to the corresponding seed



Exploration





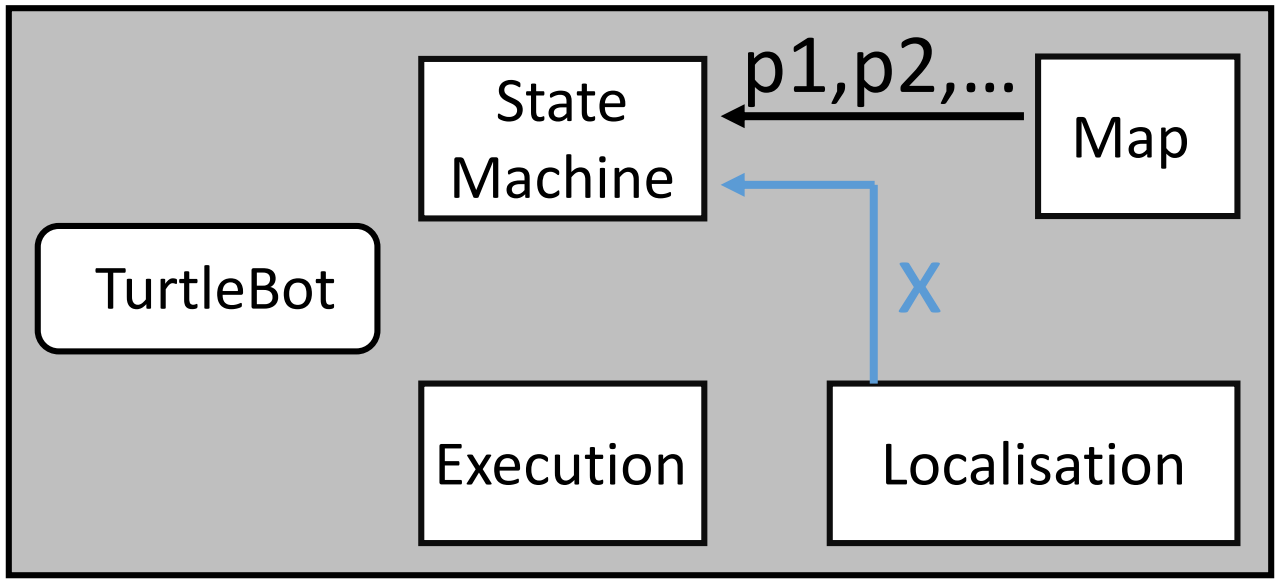
Exploration

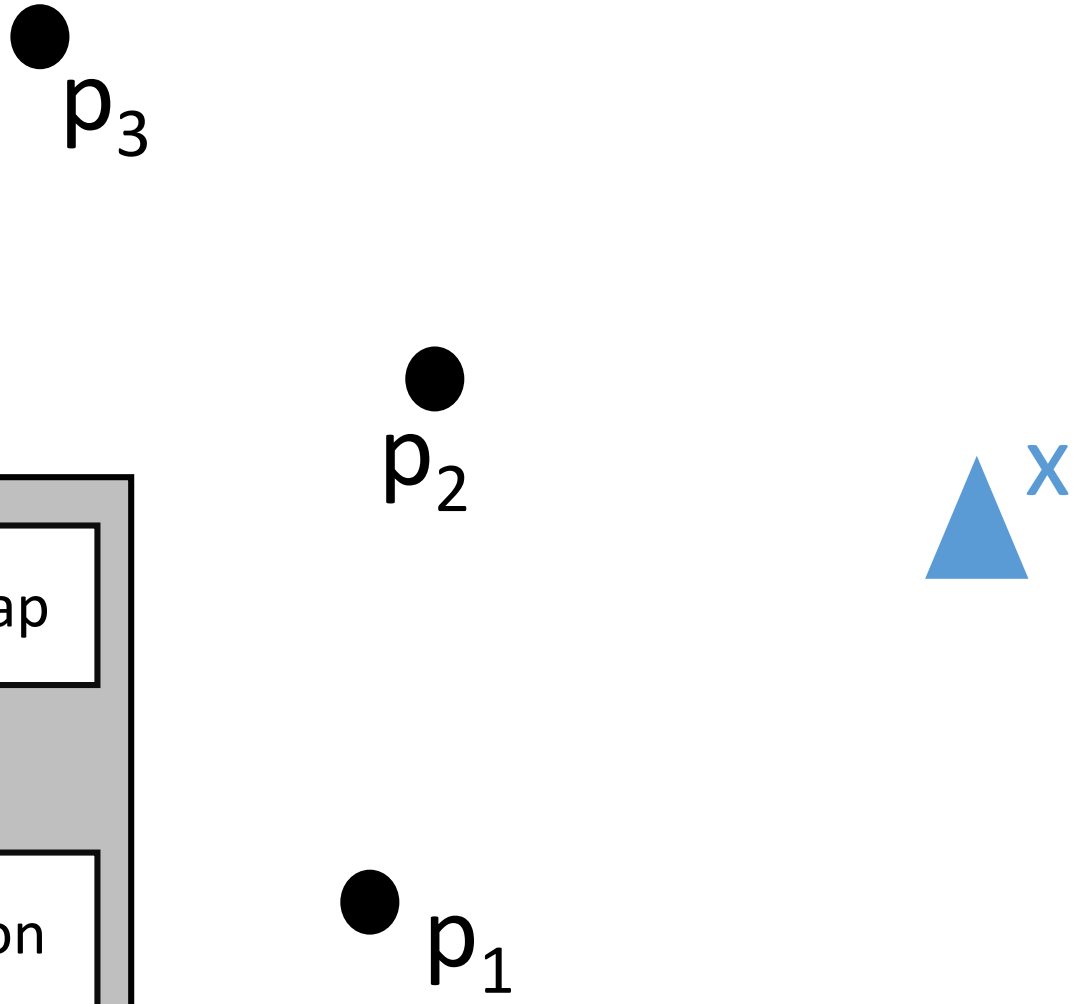
RVSS_ws/src/rvss_workshop/scripts/planning.py

```
def generateLoop(nBisections,minDistance,direction,ptsIn,ptsOut,ptsObs):
    "Generates path from voronoi diagram"
    #Inputs
    #   nBisections - (int) no. times to bisect hexagon sides
    #   minDistance - (float) closest turtlebot centre can be from obstacles
    #   direction   - (string) 'anticlockwise'/'clockwise': direction to move in

    #Outputs
    #   orderedPoses   - (array) nPoses x 3, each row = [x,y,theta] in global coords
    #                  - ordered array of poses forming safest path for robot to move in
    #   orderedQuivers - (array) nPoses x 4, each row = [x,y,dX,dY] in global coords
    #                  - [dX dY] is vector to next pose. this variable is best used for
    #                  - plotting

    return orderedPoses,orderedQuivers
```

Exploration

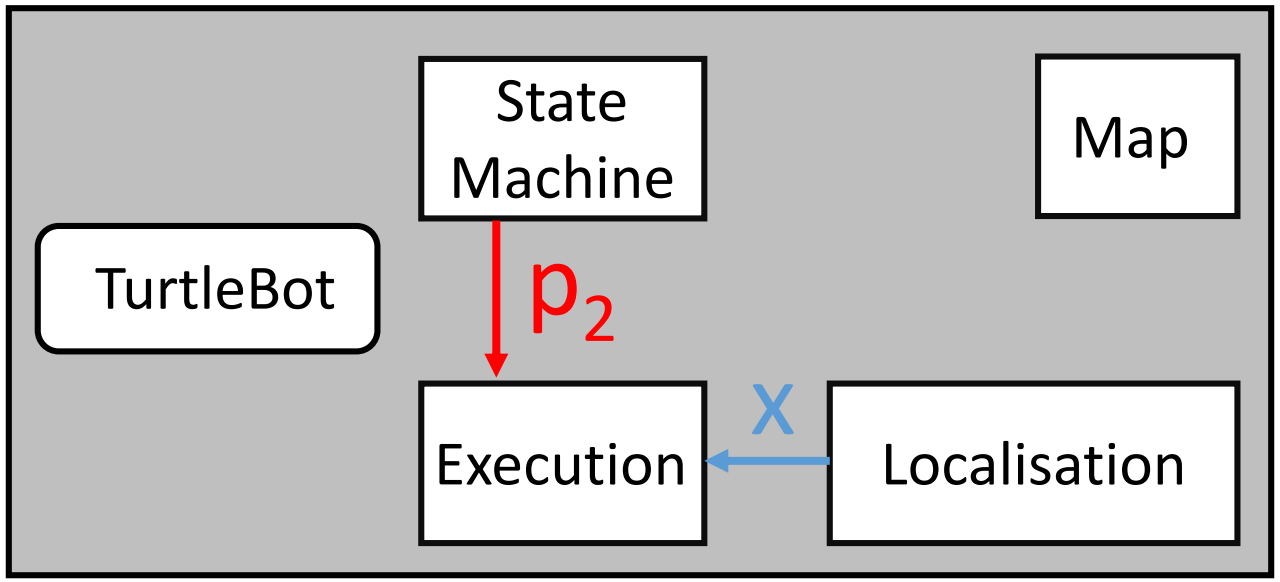
- Find closest point on path from current pose
- Compute range and bearing to next pose

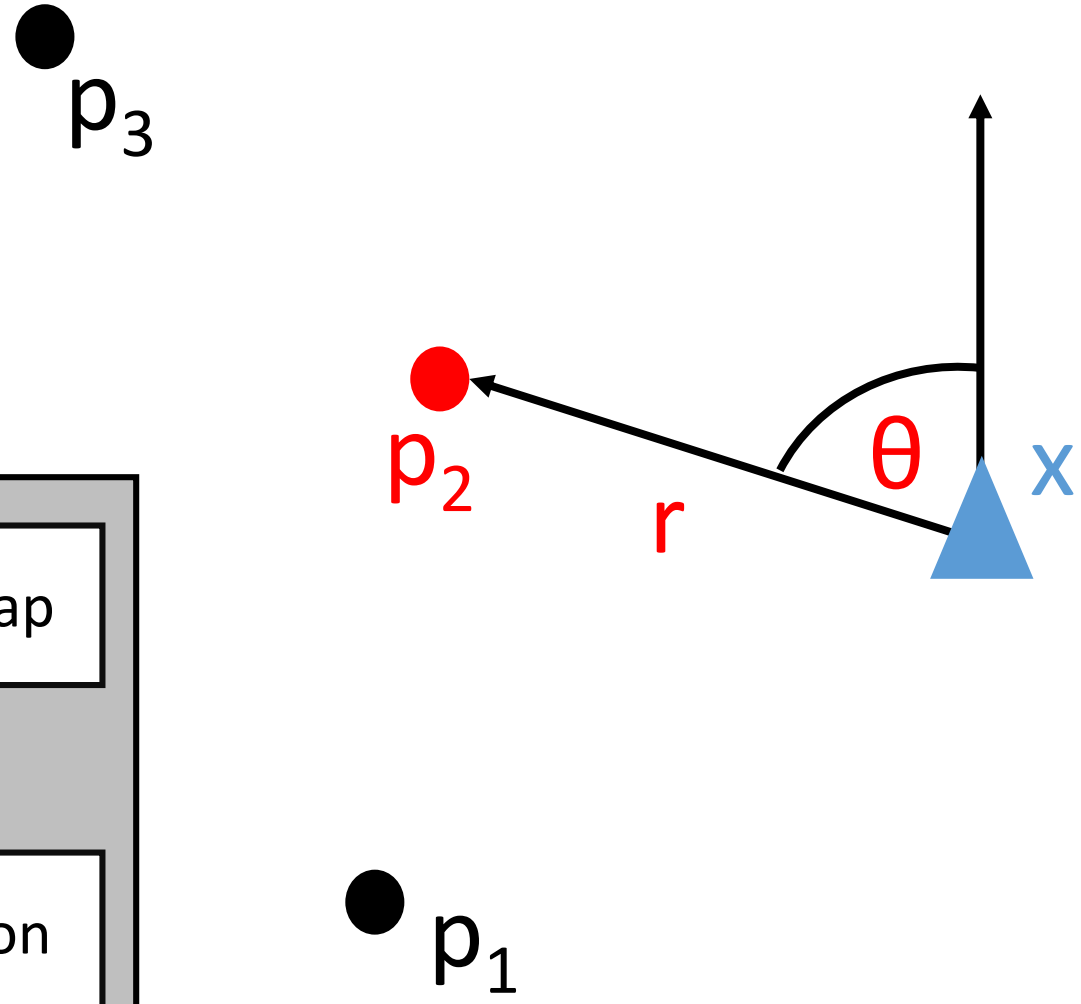




Exploration

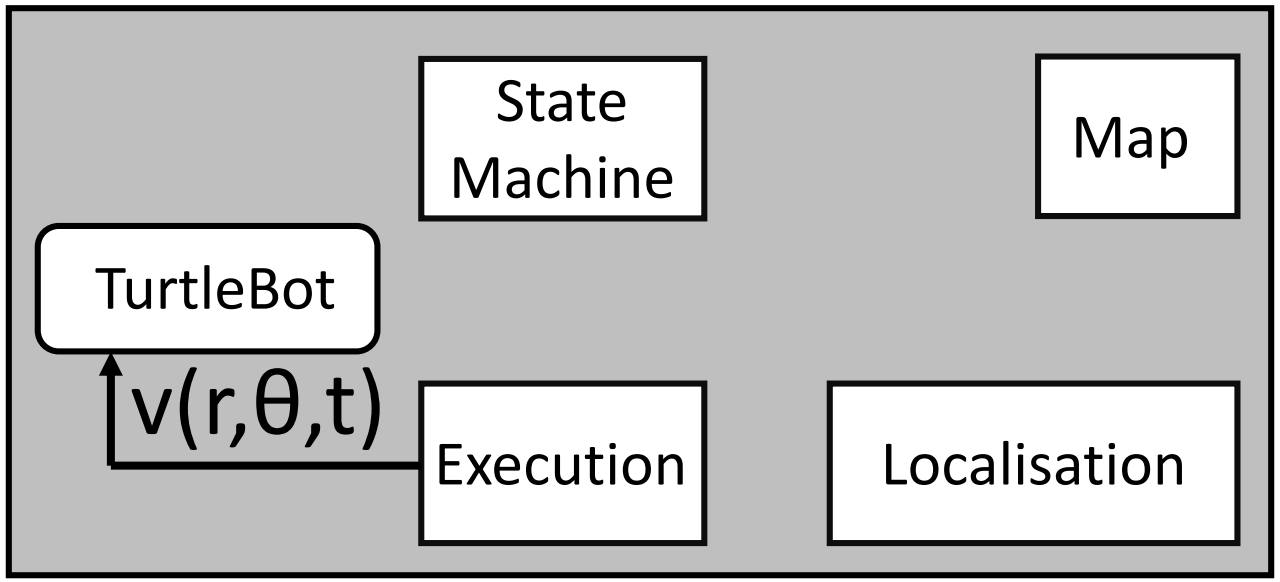
- Find closest point on path from current pose
- Compute range and bearing to next pose

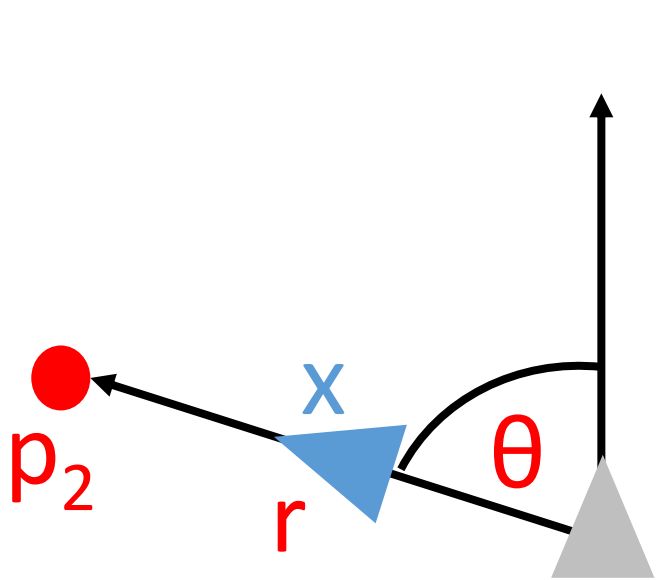




Exploration

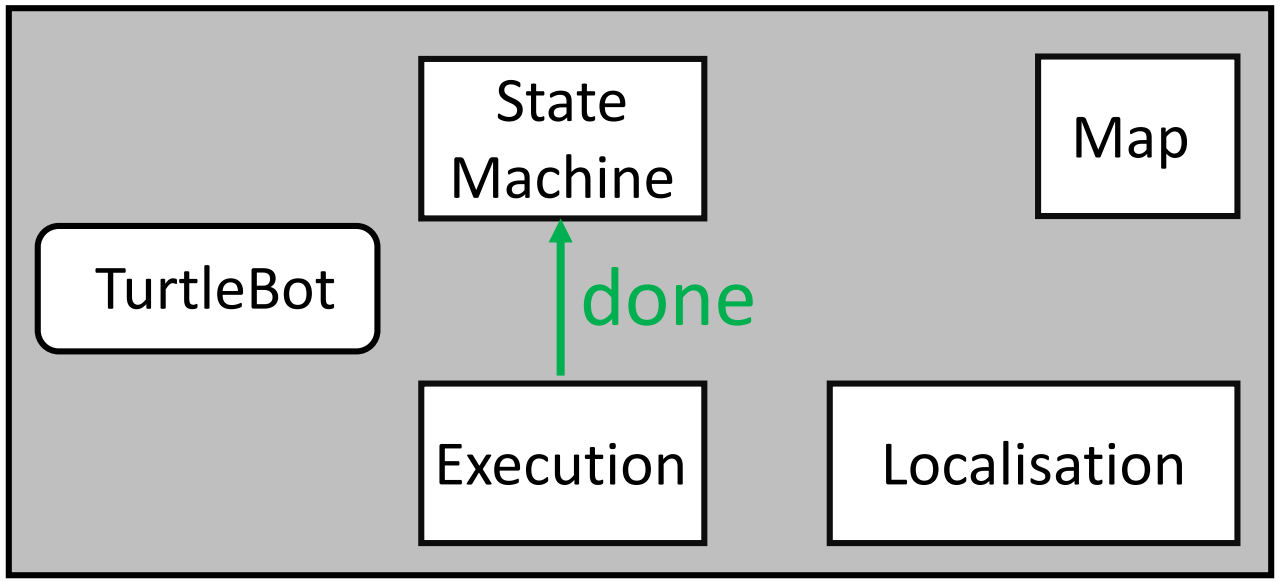
- Perform 1 rotation, 1 translation to desired pose
- Open loop! What are the advantages and disadvantages of this approach?

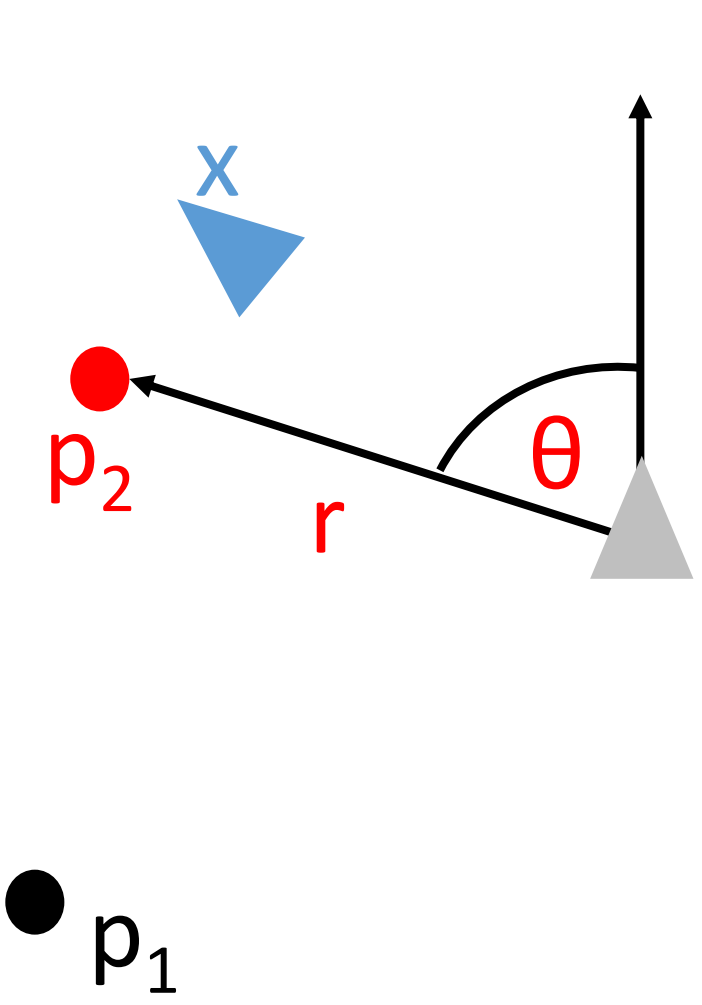




Exploration

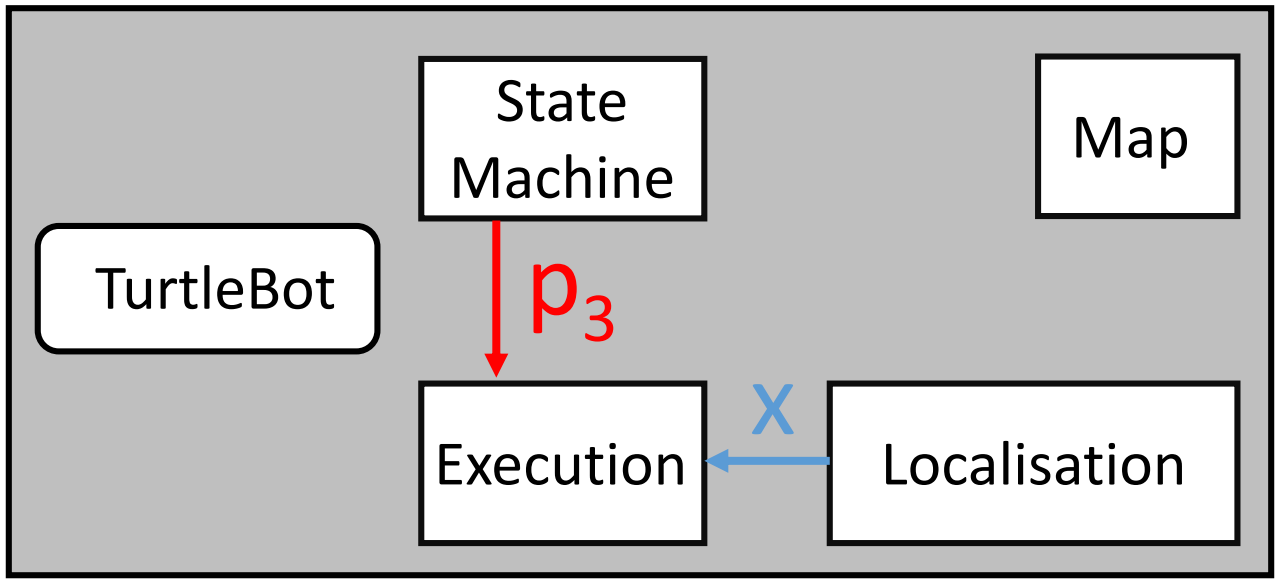
- Perform 1 rotation, 1 translation to desired pose
- Open loop! What are the advantages and disadvantages of this approach?

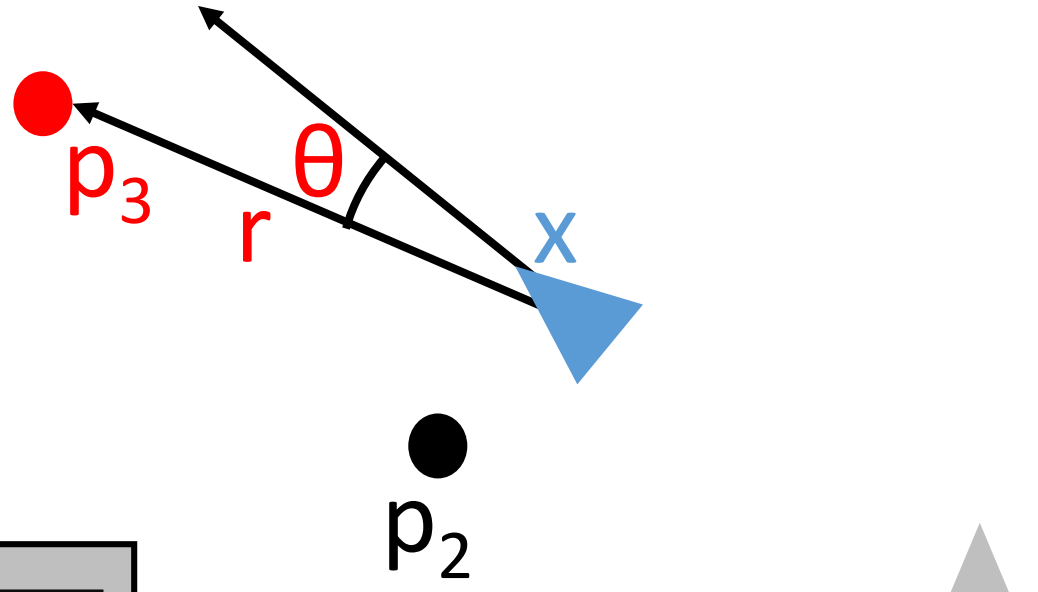




Exploration

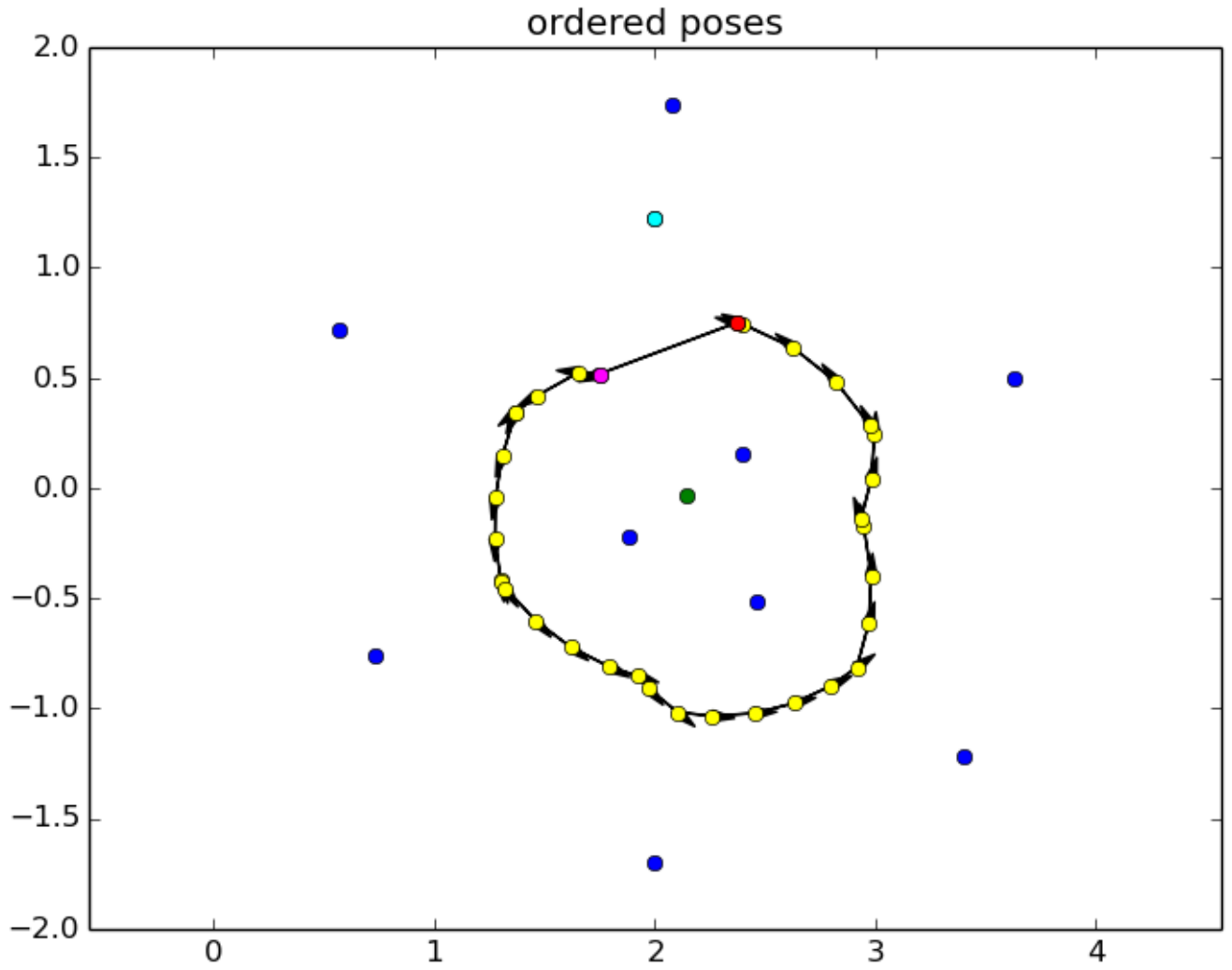
- Get new current pose from localisation
- New target is <u>next</u> waypoint on ordered list





Task Planning

- Leave loop at p1, pick up object at p2
- Return to p1
- Continue traversing loop until reach p3
- Push object to p4
- Return to p3, continue to traverse loop



Task Planning

RVSS_ws/src/rvss_workshop/scripts/planning.py

```
def generateTaskPath(ptGoal,ptsIn,ptsOut,ptsObs,orderedPoses,task,safeDistance):
    "Given a goal point, outputs points necessary to complete pushing task"
    #Inputs
    #   ptGoal - (array), position of object to perform task with
    #   ptsIn  - (array), positions of cylinders forming triangle boundary
    #   ptsOut - (array), positions of cylinders forming hexagon boundary
    #   ptsObs - (array), positions of obstacles
    #   orderedPoses - (array), poses forming safe path
    #   task         - (string), 'inside'/'outside' - where to put object
    #   safeDistance - (float), min distance turtlebot centre can be from any obstacle

    #Outputs
    #   iLeavePt1 - (int), index of closest point on safe path to object
    #   iLeavePt2 - (int), index of point on safe path to leave and drop off object
    #   dropOffPt - (array), point to leave object at

    return iLeavePt1,iLeavePt2,dropOffPt
```