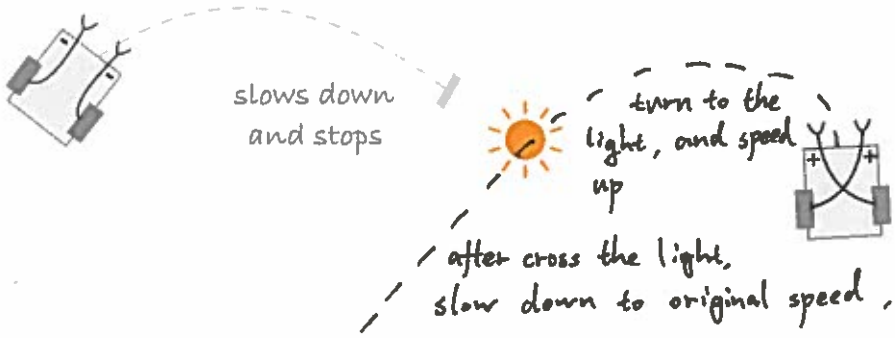
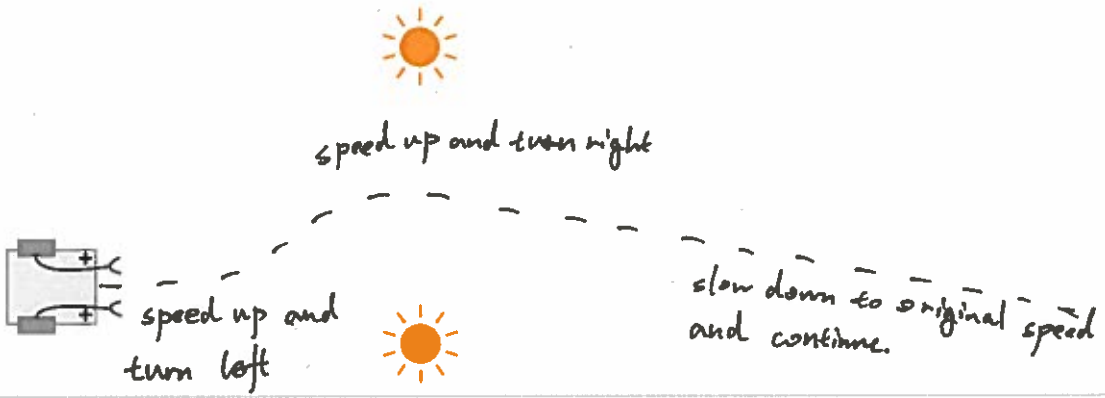
Question 1: Braitenberg Vehicles _____/2points

Consider the following three scenarios with Braitenberg vehicles. Sketch how you expect the vehicles to move using dashed lines (see example) and briefly explain. Note that the exact relation between the sensors and motors might result in different behaviors (e.g. the vehicle in the example might or might not stop); you just need to illustrate one valid behavior.

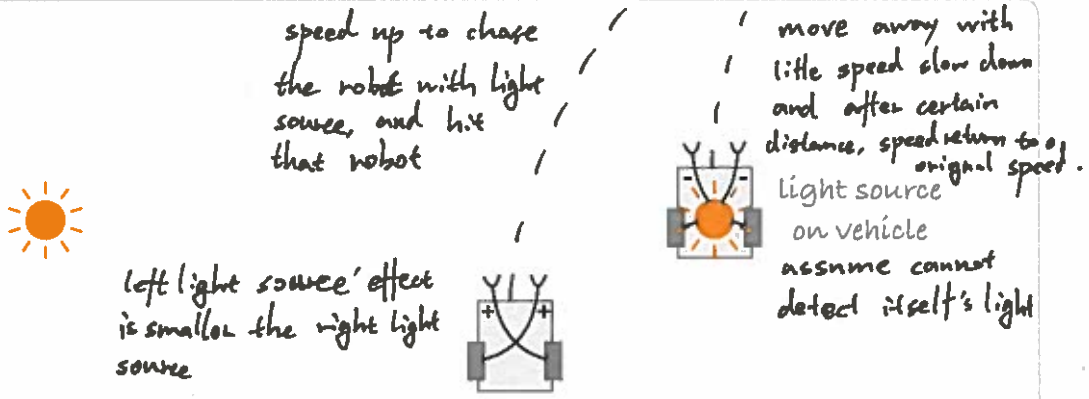
(a) (0.4 pt)



(b) (0.8 pt)



(c) (0.8 pt)



Question 2: RANSAC Algorithm _____/2.5points

The RANSAC (RANdom SAmple Concensus) algorithm is commonly used in robotics for solving perception problems. One such problem is finding a large plane (floor, table, walls) in the robot's point cloud. A point cloud is a list of 3-dimensional points (x, y, z) that are known to be occupied.

(a) (1.5 pt) Below is a sketch of the RANSAC algorithm applied to finding a plane in a point cloud. Fill in the blanks of the algorithm. Then answer the following questions about the algorithm.

```
Input: point_cloud, K, N
best_model := undefine , best_model_consensus_set_count = 0.
while iterations < K
    maybe_inliers := N randomly selected points from the point_cloud
    maybe_model := plane fit onto maybe_inliers
    consensus_set := maybe_inliers

    for every point in point_cloud not in maybe_inliers
        if point is approximately fit maybe_model
            add point to consensus_set

    consensus_set_count := number of point in consensus_set

    if best_model is undefine or consensus_set_count > best_model.consensus_set_count.
        best_model := maybe_model
        best_model_consensus_set_count := consensus_set_count.
    increment iterations

return best_model
```

(b) (0.2 pt) What is the minimum value for **N**? Why?

In RANSAC Algorithm, typicall N = minimal sample size to fit a model
model is a plane.
3 point can define a plane
} ⇒ N = 3

(c) (0.8 pt) Assume 70% of the points in the point cloud correspond to a table plane. What should **K** be to ensure that the algorithm finds the correct plane with 95% probability?

Knowing $N=3$, $w=0.7$, $p=0.95$, and $(1-w^N)^k = 1-p$

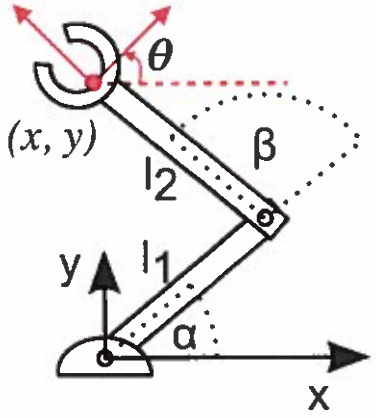
$$\Rightarrow k = \frac{\log(1-p)}{\log(1-w^N)} = \frac{\log(1-0.95)}{\log(1-0.7^3)} = \frac{\log 0.05}{\log 0.657} = 7.13$$

⇒ when $k = 8$, can ensure that the algorithm find the correct plane with 95% probability

Question 3: Simple Arm Kinematics _____/1.5points

Consider a planar two link robot arm with two motors that control α and β as shown below. The length of the arm links are l_1 and l_2. The pose of the robot's end-effector is specified by its position (x, y) and orientation θ.

(a) (1 pt) Derive the forward kinematics; i.e. express x, y and θ in terms of other known entities. If you define any intermediate variables, be sure to illustrate them on the picture.



$$\theta = \alpha + \beta - 90^\circ$$

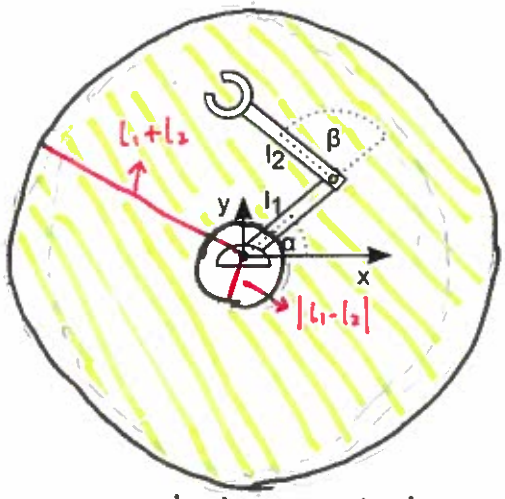
$$x = \cos(\alpha+\beta) l_2 + \cos\alpha\, l_1$$

$$y = \sin(\alpha+\beta) l_2 + \sin\alpha\, l_1$$

(b) (0.5 pt) Visually illustrate and describe the complete set of points (x, y) that this robot can reach. Assume all joints can rotate 360 degrees.

when $\beta = 0^\circ$, robot can reach the farest place. in this case.

$$\begin{cases} x = \cos\alpha\, l_2 + \cos\alpha\, l_1 \\ y = \sin\alpha\, l_2 + \sin\alpha\, l_1 \end{cases} \Rightarrow x^2 + y^2 = (l_2 + l_1)^2$$



so, the complete set of points (x, y) is.

$$\begin{cases} x^2 + y^2 \le (l_2+l_1)^2 \\ x^2 + y^2 \ge (l_2-l_1)^2 \end{cases}$$

when $\beta = 180^\circ$, robot can reach the nearest place.

$$\therefore \begin{cases} x = -\cos\alpha\, l_2 + \cos\alpha\, l_1 \\ y = -\sin\alpha\, l_2 + \sin\alpha\, l_1 \end{cases} \Rightarrow x^2 + y^2 = (l_2 - l_1)^2.$$

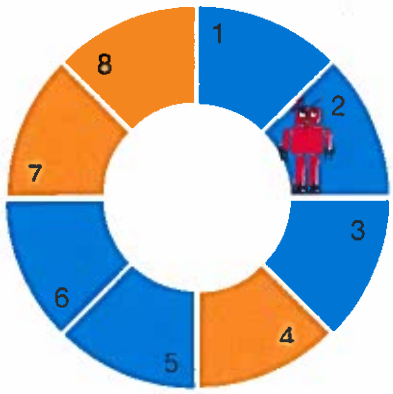
Question 4: Bayes Filters _____/2.5points

Consider the following simplified robot localization problem. A robot moves in a circular corridor discretized into eight segments. Each segment is either orange or blue. The robot has a noisy sensor that can detect the color of the segment that it is in. The robot can move clockwise (CW) or counterclockwise (CCW) in one-step or two-step leaps, but its motors are also noisy.

Sensor model: If the segment is blue, the robot sees the segment as blue with 95% probability and mistakes it as orange with 5% probability. If the segment is orange, the robot sees it as orange with 85% probability and mistakes it for blue the rest of the time.

Action model: If the robot takes a one-step leap in either direction, it reaches the neighboring segment in that direction with 90% probability and stays where it is otherwise. If the robot takes a two-step leap in either direction, it reaches the segment two steps away in that direction with 80% probability, it undershoots (reaching the segment only one step away) with 10% probability, and it overshoots (reaching the segment three steps away) with 10% probability.

(a) (1.5 pt) Since our localization problem has a finite discrete set of states (1-8) we can use a *histogram* filter which represents the probability distribution as a look up table with an entry for each state. Assume that the robot starts off at segment #2 but it has no idea where it is, so it has a uniform probability distribution over all possible states. Iterate the Bayes filter algorithm for the five steps shown in the table below by entering the probability estimate of each state (1-8) after that step.



STEP	P(1)	P(2)	P(3)	P(4)	P(5)	P(6)	P(7)	P(8)
Uniform priors	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
Sensed blue	0.183	0.183	0.183	0.029	0.183	0.183	0.029	0.029
Two-step CCW move	0.167	0.060	0.167	0.167	0.044	0.044	0.167	0.163
Sensed orage	0.018	0.006	0.018	0.307	0.005	0.005	0.307	0.335
One-step CCW move	0.008	0.017	0.278	0.035	0.005	0.276	0.332	0.050
Sensed orage	0.001	0.002	0.036	0.078	0.001	0.036	0.736	0.110

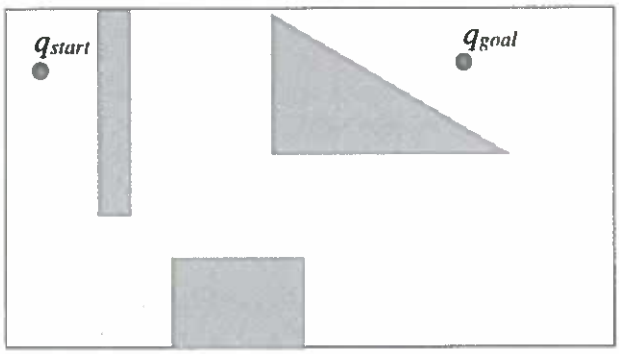
(b) (1 pt) Given uniform priors, what sequence of steps (movements and sensing) should the robot take to localize itself (i.e., have one state with higher probability than others) as fast as possible? Why?

Should use one sensing right after one movement recursively to localize itself. Because, the movement with increase the uncertainty, and the sensing can narrow down the uncertainty (correct the prediction). So, if have one sensing after multiple movement, the uncertainty might too large to be narrowed down to have a higher probability state (localize)

Question 5: Path Planning _____/2points

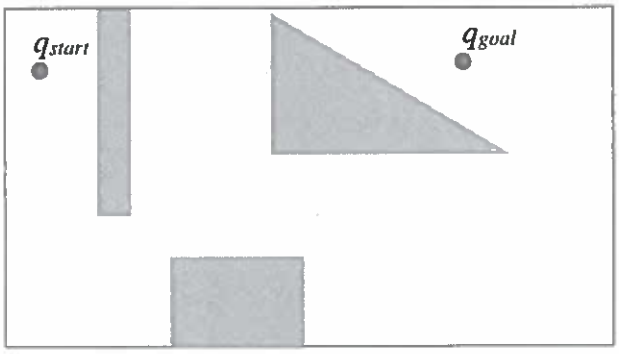
Most path planning problems in robotics can be solved using graph search algorithms, by first turning the map into a graph. On the map given below, apply the following three different methods for converting the given map (where the robot is assumed to be a single point) into a graph. Show your discretization on the map and draw the resulting graph (with nodes and edges) next to it.

(a) (0.5 pt) Occupancy grid with your choice of fixed cell size.

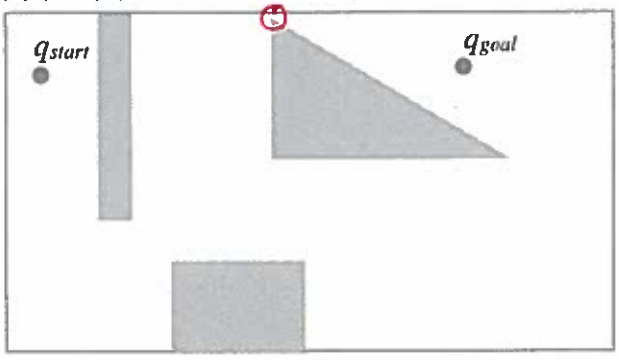


Please see the discretization and resulting graph on next 3 pages.

(b) (0.5 pt) Quadtree with your choice of minimum cell size.



(c) (0.5 pt) Exact cell decomposition.

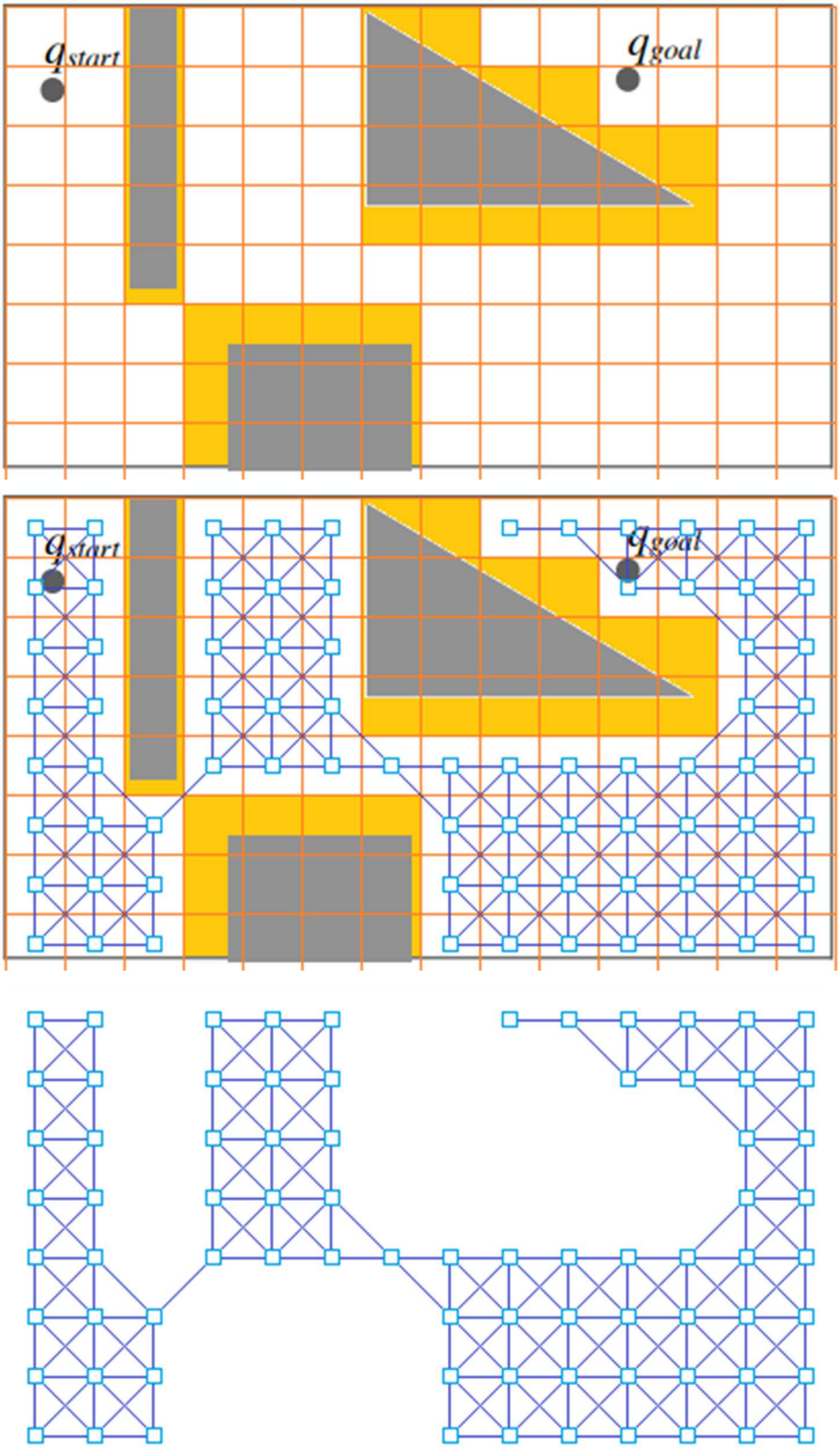


Because the robot is assumed to be a single point, we can say it only 1×1 pixel, and therefore can pass the place where at the top of the triangle obstacles (red point)

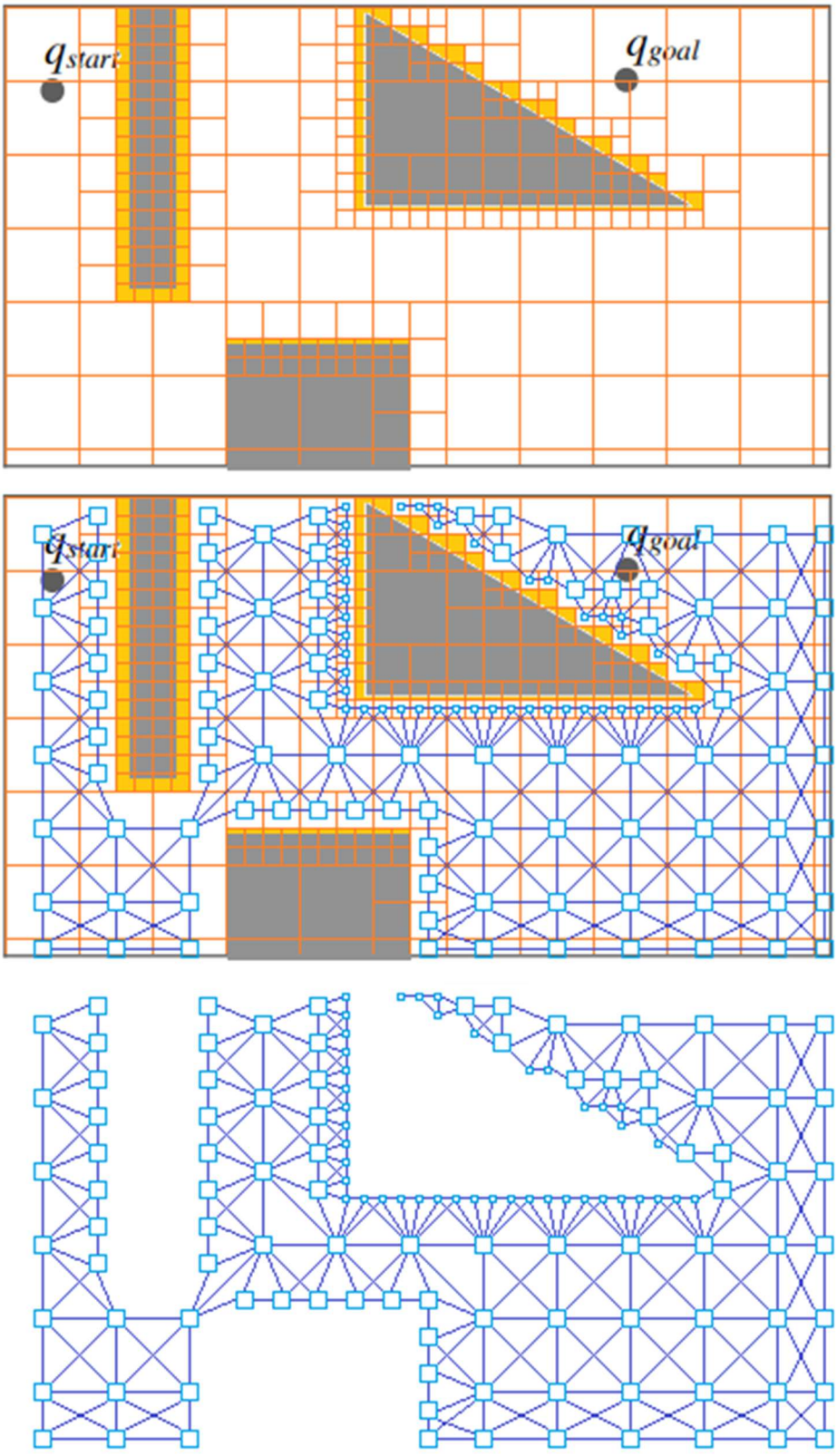
(d) (0.5 pt) Assume you were to use A* search in the resulting graphs to find the shortest path from q_{start} to q_{goal}. Assume the robot moves from one node to a neighboring node moving on a straight line between the centers of corresponding cells. Suggest a heuristic that is admissible for all three graphs.

$h(n)$ = distance to the node which the cell is contain (closer, when at multiple cells' edge) the goal point

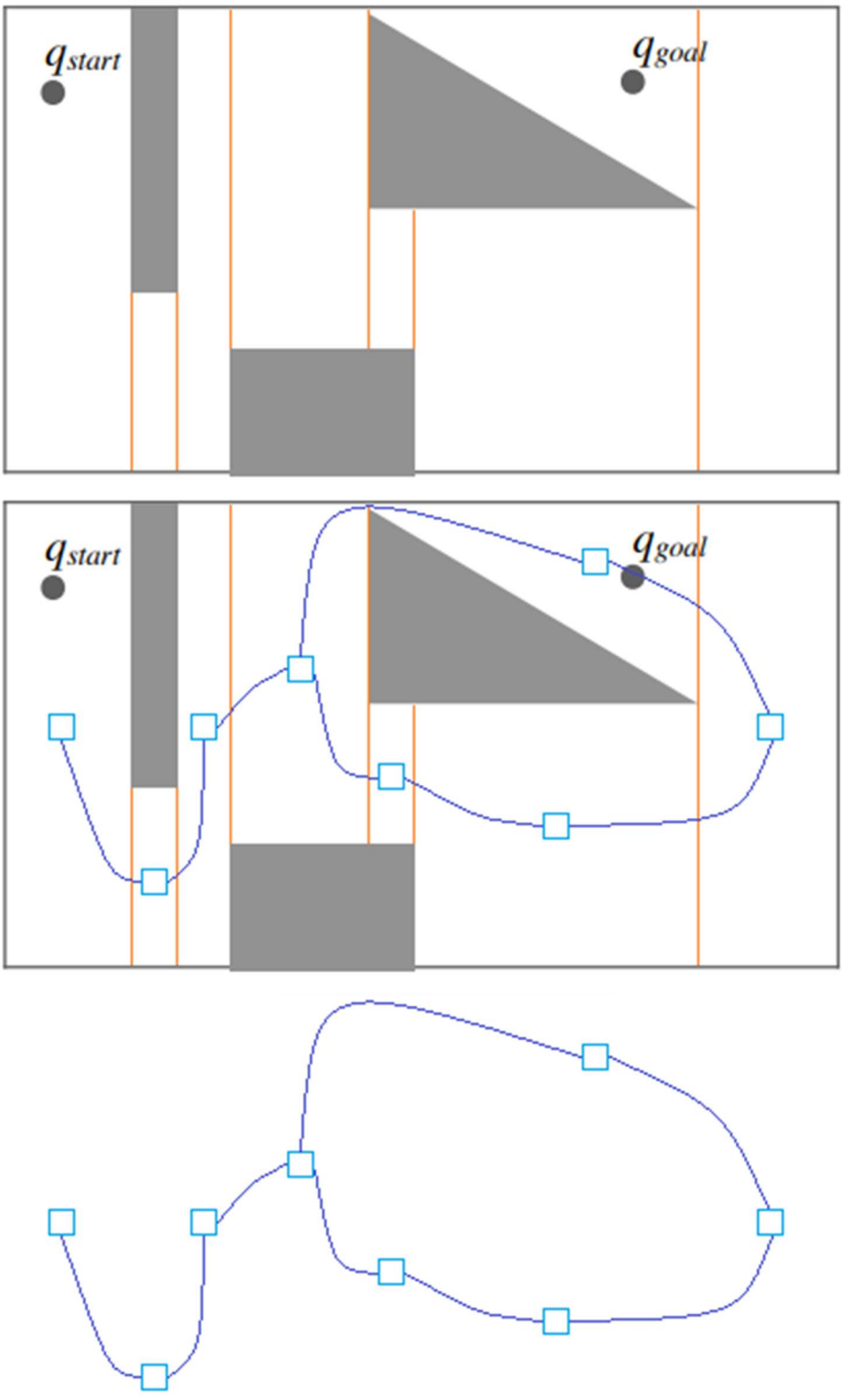
a) Occupancy grid with fixed 30 x 30 pixel cell size



b) Quadtree with minimum cell size is 10 x 10 pixel and large default grid size is 37 x 37 pixel

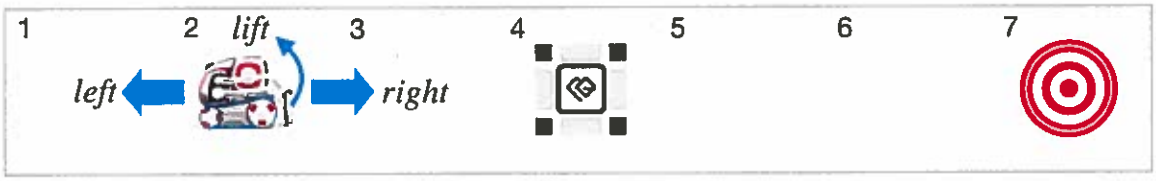


c) Extract cell decomposition



Question 6: MDPs _____/2.5points

Consider a Markov Decision Process (MDP) modeling the Cozmo robot cube carrying task in a simple grid world illustrated below. There is a single cube at square 4. The process terminates after the robot reaches square 7.



Actions: The robot has four actions. It can move *left* or *right* between squares, except it cannot move from square 3 to square 4. When the robot is at square 3 it can try to *lift* the cube. If successful, this action moves the robot to square 4 with the cube in its fork lift. If it fails, the robot ends up at square 5 without the cube. The lift action is available only at square 3 when the cube is at square 4. Assume that the *left* and *right* actions are deterministic and the lift action succeeds 50% of the time.

Rewards: The robot gets a positive reward of +10 when it reaches the target, +4 when it lifts the cube, and +1 every time it moves closer to the target (+2 when it transitions from square 3 to 5 due to a failed *lift* action). The robot gets a negative reward of -2 every time it moves away from the target.

(a) (0.5 pt) How many states does the MDP have considering the cube can be lifted only once?

Total 14 states. At each square, robot can with or without the cube.

(b) (0.75 pt) For the policy π of always moving right and lifting at square 3, compute $V^{\pi}(S_{3/0})$ where $S_{3/0}$ is the state in which the robot is at square 3 without a cube in its forklift. Assume a discount of $\gamma = 1$.

$V^{\pi}(S_{6/0}) = V^{\pi}(S_{6/1}) = 10$ $V^{\pi}(S_{5/0}) = V^{\pi}(S_{5/1}) = 1 + 10 = 11$

$V^{\pi}(S_{4/1}) = 1 + 11 = 12$ $V^{\pi}(S_{3/0}) = 0.5 \cdot (2 + V^{\pi}(S_{5/0})) + 0.5 (5 + V^{\pi}(S_{4/1})) = 15$

(c) (0.75 pt) Perform two iterations of value iteration & compute $V_2(S_{2/0})$, $Q_2(S_{2/0}, right)$, $Q_2(S_{2/0}, left)$. Assume all values are initialized to zero at iteration 0.

$V_0(S_{1/0}) = 0$, $V_0(S_{2/0}) = 0$, $V_0(S_{3/0}) = 0$, $V_0(S_{4/1}) = 0$, $V_0(S_{5/1}) = 0$, $V_0(S_{5/0}) = 0$ $\Rightarrow$

$V_1(S_{1/0}) = 1 \times (1 + V_0(S_{2/0})) = 1$

$V_1(S_{2/0}) = 1 \times (1 + V_0(S_{3/0})) = 1$

$V_1(S_{3/0}) = 0.5 \times (5 + V_0(S_{4/1})) + 0.5 \times (2 + V_0(S_{5/0})) = 3.5$ $\Rightarrow$

$V_1(S_{4/1}) = 1 \times (1 + V_0(S_{5/1})) = 1$

$V_1(S_{5/0}) = 1 \times (1 + V_0(S_{6/0})) = 1$

$Q_2(S_{2/0}, left) = 1 \times (-2 + V_1(S_{1/0})) = -1$

$Q_2(S_{2/0}, right) = 1 \times (1 + V_1(S_{3/0})) = 4.5$

$V_2(S_{2/0}) = \max(-1, 4.5) = 4.5$

(d) (0.5 pt) Consider the policy obtained from value iteration after it converges. Does this policy make the robot come back to try to lift the cube? (Write "yes" or "no") If not, how would you change the MDP to obtain this behavior? Please explain.

E(lift reward) = 2, $R(S_{5/0} \to S_{3/0}) = -4$ $R(S_{3/0} \to S_{5/1}) = 6$, $R(S_{3/0} \to S_{3/0}) = 2$.

So, total reward of moving back and do lift again is 0, while $R(S_{5/0}, right, S_{6/0}) = 1$

So, no come back action in the policy. Reduce reach target and move away penalty can obtain this

Question 7: Reinforcement Learning _____/1points

Consider a Cozmo robot learning to make a person smile. Assume Cozmo has two facial expressions corresponding to two states: *cheerful* and *grumpy*. It also has two sounds it can play: *laugh* and *cry*. Hence the robot has three actions it can take at every step: it can either change its facial expression or play one of the two sounds. The robot has a human facial expression detector which computes a positive or negative reward associated with the detected facial expression in response to robot actions. The robot will use Monte-Carlo control to learn a policy from rollouts.

(a) (0.75 pt) Given the following roll-out of the initial policy π, estimate $Q_\pi(s, a)$ for all state-action pairs. Assume $\gamma = 0.9$.

(b) (0.25 pt) Qualitatively describe the resulting policy.

State	Action	Reward	State	Action	Reward	State	Action	Reward
cheerful	laugh	4	cheerful	change	-1	grumpy	cry	-4

a) $Q_\pi(\text{cheerful}, \text{laugh}) = 4 + 0.9\times4 + 0.9^2\times4 + \cdots = 40$

$Q_\pi(\text{cheerful}, \text{cry}) = 0 + 0.9\, V^*(\text{cheerful}) + 0.9^2 V^*(\text{cheerful}) + \cdots = 36$

($V^*(\text{cheerful}) = Q^*(\text{cheerful}, \text{laugh})$)

$Q_\pi(\text{grumpy}, \text{change}) = 0 + 0.9\, V^*(\text{cheerful}) + 0.9^2 V^*(\text{cheerful}) + \cdots = 36.$

$Q_\pi(\text{grumpy}, \text{cry}) = -4 + 0.9\, V^*(\text{grumpy}) + 0.9^2 V^*(\text{cheerful}) + \cdots = 32.4 - 4 = 28.4$

($V^*(\text{grumpy}) = Q^*(\text{grumpy}, \text{change})$)

$Q_\pi(\text{grumpy}, \text{laugh}) = 0 + 0.9\, V^*(\text{grumpy}) + 0.9^2 V^*(\text{cheerful}) + \cdots = 32.4$

$Q_\pi(\text{cheerful}, \text{change}) = -1 + 0.9\, V^*(\text{grumpy}) + 0.9^2 V^*(\text{cheerful}) + \cdots = 31.4$

b) $\therefore \pi(\text{cheerful}) = \underset{a}{\text{argmax}}\, Q(\text{cheerful}, a) = Q(\text{cheerful}, \text{laugh})$

$\pi(\text{grumpy}) = \underset{a}{\text{argmax}}\, Q(\text{grumpy}, a) = Q(\text{grumpy}, \text{change}).$

That means when in State "cheerful", will choose "laugh" action,

in State "grumpy", will choose "change" action.

Question 8: Robot Perception _____/1points

Object *tracking* is a perception problem in which the robot tries to classify and localize an object continuously over time. For example, an autonomous car needs to track where all the other cars and pedestrians are at a high frame rate. Instead of trying to independently detect the target object in each frame, tracking involves incorporating temporal information, such as detection results in the previous frame.

(a) (0.75 pt) We discussed Bayes filters as a way to localize the robot itself given its sensor and action models. How can we use Bayes filters for tracking external objects? Please explain.

We can still use recursive cycle for Bayes filters for tracking external objects:

- Detect target objects in the frame (also can get the orientation of movement) - sensor
- Don't detect from the frame, but use their motion to predict the location - movement

With the sensing and movement cyde, we can save the comprtation and correct the prediction result from movemt by sensing the target again.

(b) (0.25 pt) In what situations would you expect tracking work significantly better than trying to independently detect the target object in each frame?

When have lots of objects need to track, this will significantly better than detect every objeet in each frame. Such as: track people in High & Density Crowd Scenes.