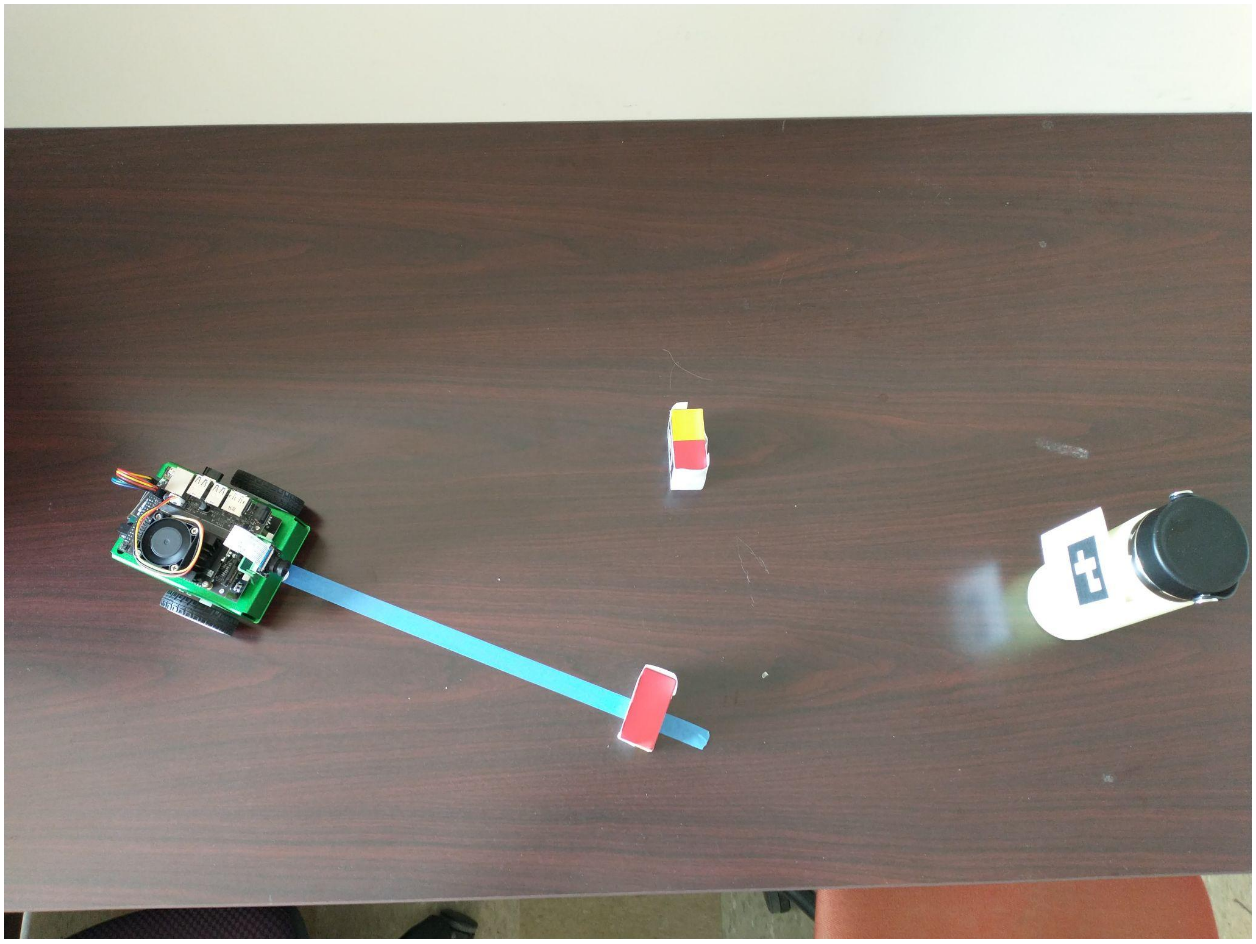
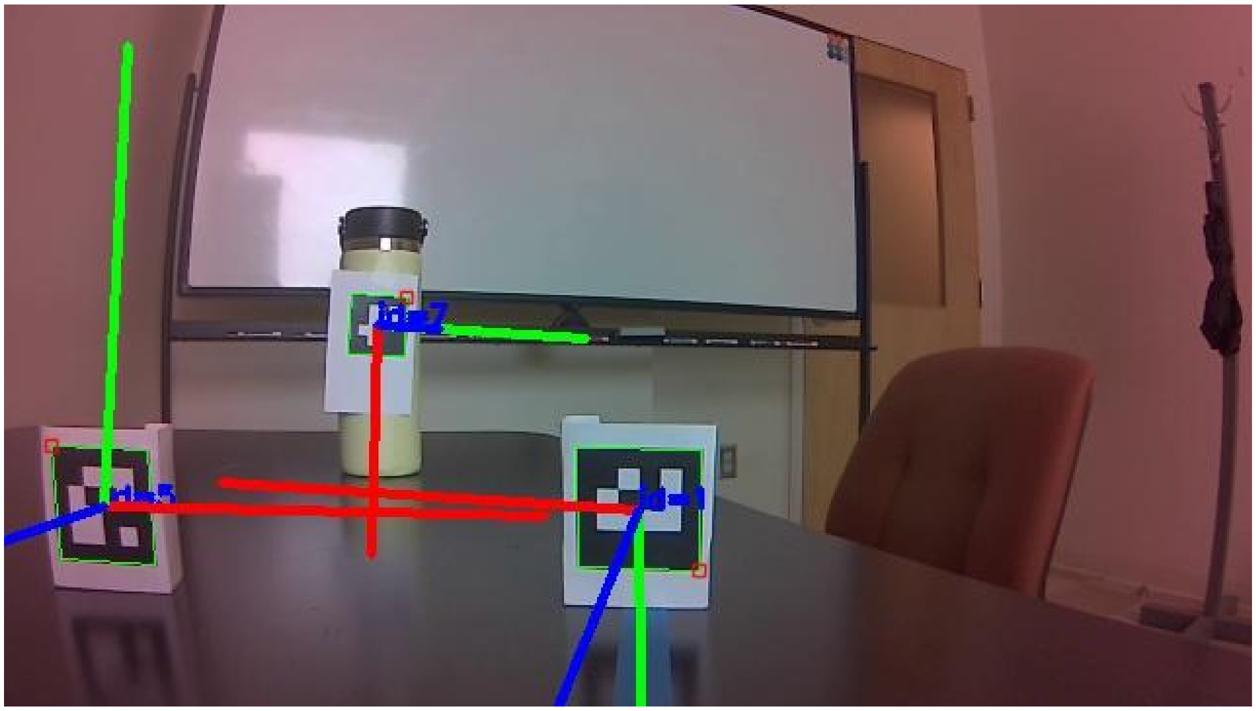
ECE498: Oct6th exam

Setup everything until the Sep 27 lecture . Print out a few ARUCO markers from the sheet provided aruco_4x4_0-7_50mm.pdf . Put the ARUCO markers on some objects. Here I have put the markers on a bottle (Marker 7), and two wooden blocks (Marker 1) and (Marker 6). Here the bottle is our goal and the rest of the objects are obstacles. Set up the jetbot so that

1. it can see (and detect) all the markers and
2. If the Jetbot moves in a straight line, it will head towards an obstacle, not the goal.





Create a ROS node that

1. subscribes to the /aruco_detections topic to get the pose (position+orientation) of all the markers. The /aruco_detections are of the type aruco_opencv_msgs/msg/ArucoDetection
2. It then plans the shortest path to the bottle using A-star (preferably RRT) algorithm. You can read about RRT here or on wikipedia . An okay RRT* implementation here. Please do not plan in 2D dimensions (x, y). The robot state is at least 3D (x, y, theta) (including orientation). Write the state space, action space, and the state transition function. If you write the state space and action space as discrete, you will have to use A-star. If you write them as continuous, then you will have to use RRT or RRT*. Once you reach closer than 10cm to the goal, the robot must stop.

   HINT 1: From the /aruco_detection you will get the relative pose of the goal, not the robot. Unless you convert the pose of the goal into the pose of the robot, you will not have the current state of the robot.
   One idea to address that is that you can equivalently think of the goal itself moving towards the robot. How is the state transition function of the goal different from the state transition function of the robot? What happens when you rotate the robot by 10 degrees when the goal is 40cm away, how does the goal move relative to the robot?
   HINT 2: Solve the problem in stages, simpler to more complex. Maybe create a tiny simulation first before solving it on the real robot?

3. Publishes desired velocity messages to /jetbot/cmd_vel topic. The cmd_vel are of type [geometry_msgs/Twist Documentation]

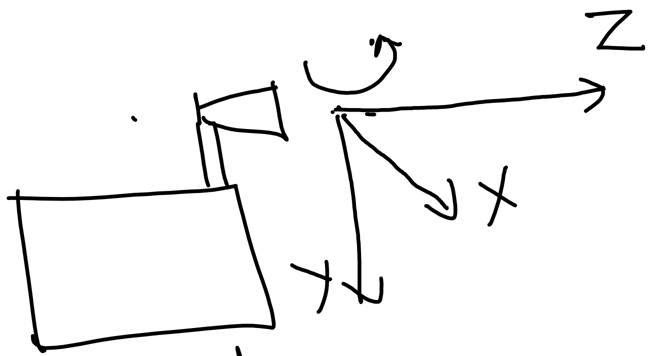
Run the following ROS nodes at the same time:

1. gscam gscam_node from Sept 27 lecture. It publishes camera images.
2. aruco_opencv aruco_tracker_autostart from Sept 27 lecture. It subscribes to camera images and publishes /aruco_detections
3. Your ROS node that you created. It subscribes to /aruco_detections and publishes /jetbot/cmd_vel
4. jetbot_ros motor_waveshare from Sept 25 lecture. It subscribes to /jetbot/cmd_vel and runs the motors at the desired speed.

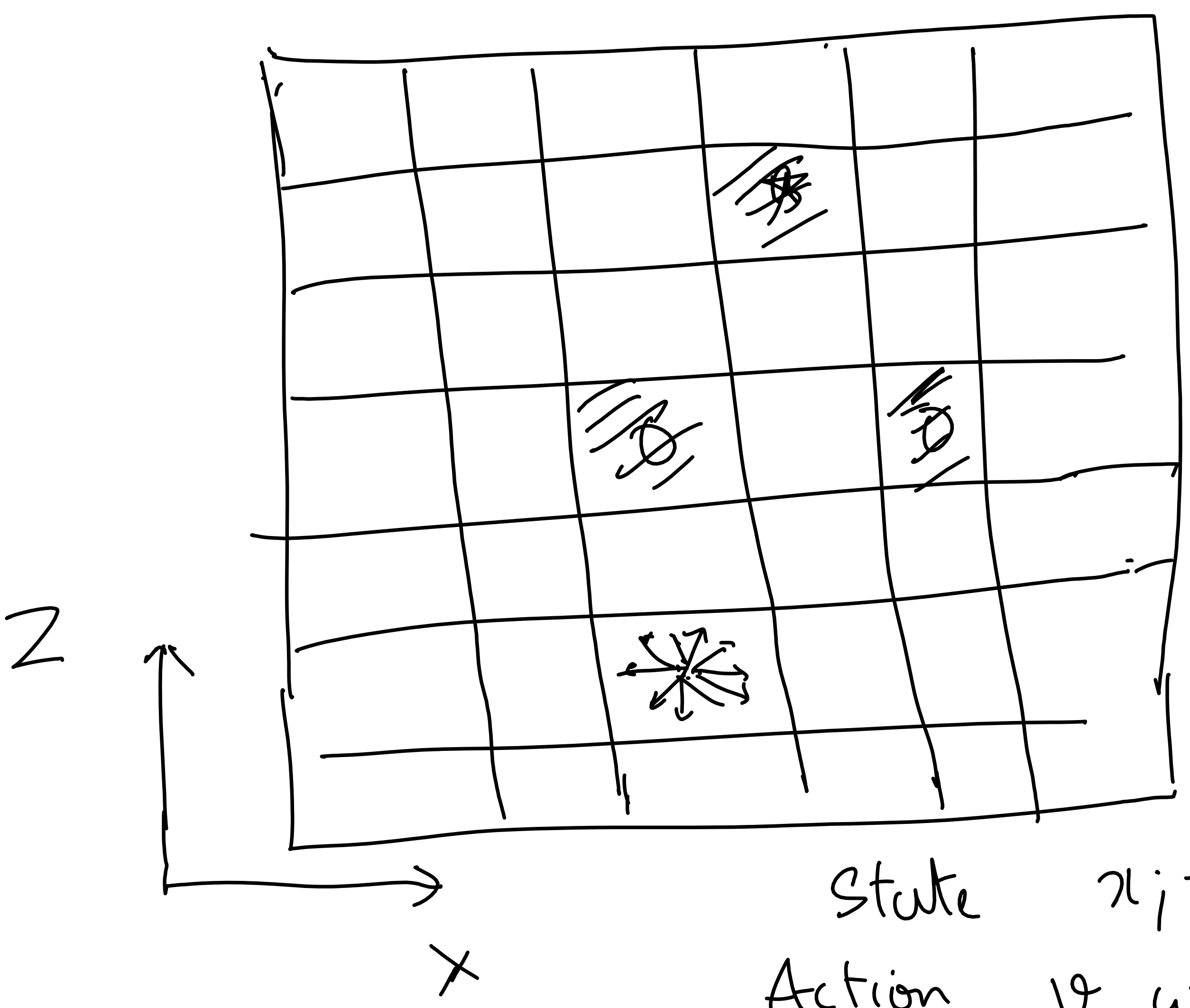
Submissions

Submit your code, a document explaining the steps followed along with reasons and a video showing the robot demo.

You will be graded mostly on the knowledge of path planning algorithms and ROS knowledge you demonstrate, rather than the end demo.



quaternion to rotation around Y-axis



State x, z, θ_y

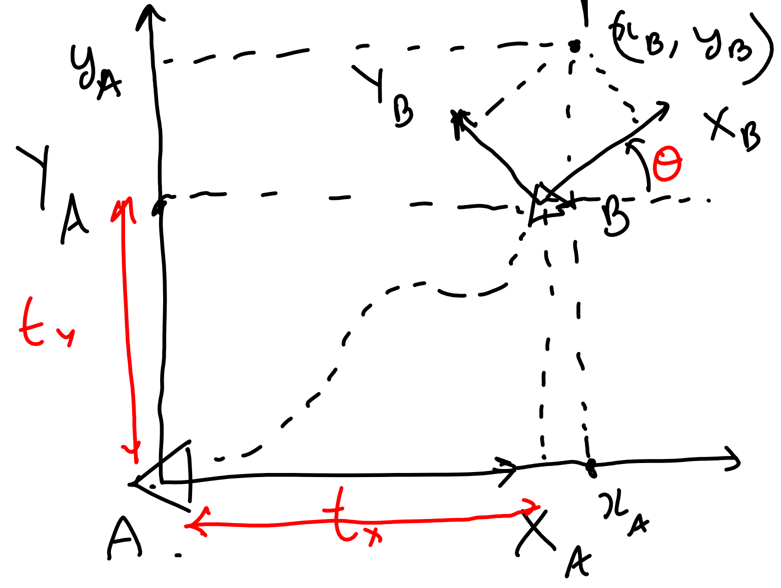
Action v_t, ω_y

$$\begin{aligned}
x_{t+1} &= x_t + v\cos\theta\,\Delta t \\
z_{t+1} &= z_t + v\sin\theta\,\Delta t \\
\theta_{y_{t+1}} &= \underline{\theta_t} + \underline{\omega_y}\,\Delta t
\end{aligned}$$

This is true for the position and orientation of robot

Coordinate Transforms

A coordinate transform from coordinate space A to space B is a function that can convert coordinates in space A to coordinates in space B



$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = {}^{A}f_{B}\left(\begin{bmatrix} x_B \\ y_B \end{bmatrix} \right)$$

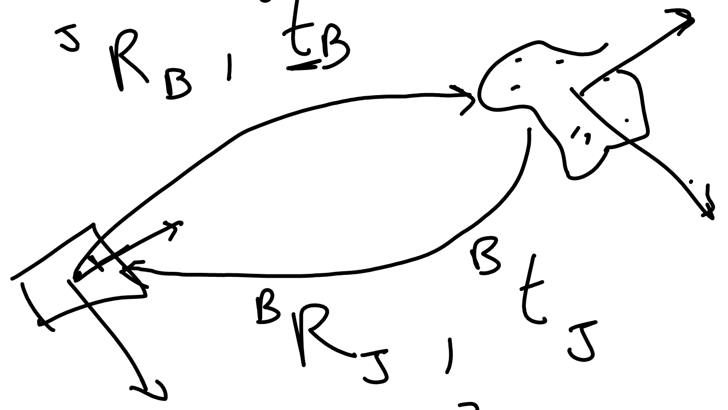
$$\underbrace{\begin{bmatrix} x_A \\ y_A \end{bmatrix}}_{\underline{x}_A} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}}_{{}^{A}R_{B}(\theta)} \underbrace{\begin{bmatrix} x_B \\ y_B \end{bmatrix}}_{\underline{x}_B} + \underbrace{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}_{{}^{A}\underline{t}_{B}}$$

$$\underline{x}_A = {}^A R_B(\theta)\, \underline{x}_B + {}^A\underline{t}_B$$

$$\Rightarrow \quad \underline{x}_A - {}^A\underline{t}_B = {}^A R_B(\theta)\, \underline{x}_B$$

$$\Rightarrow \quad \underline{x}_B = \left[{}^A R_B(\theta)\right]^{-1} \left(\underline{x}_A - {}^A\underline{t}_B\right)$$

$$\left[{}^A R_B(\theta)\right]^{-1} = \left[{}^A R_B(\theta)\right]^{T}$$



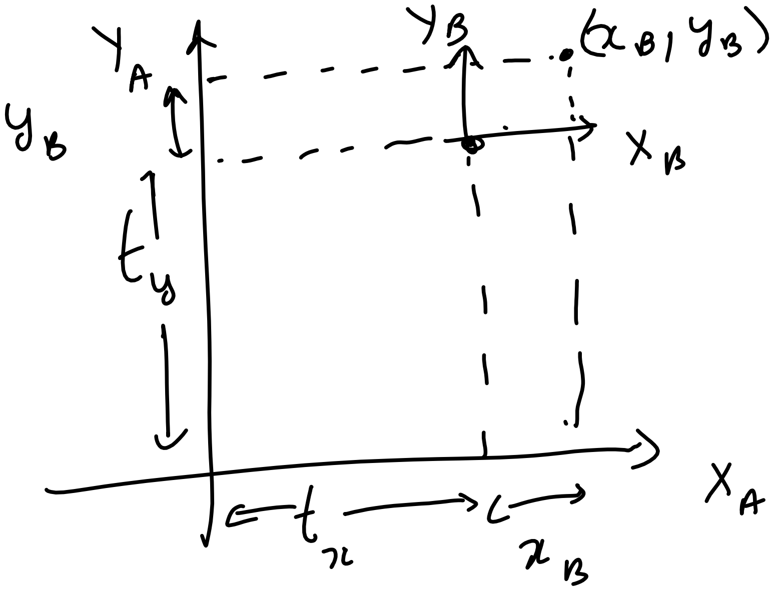
${}^J R_B \,,\; {}^J\underline{t}_B$
${}^B R_J \,,\; {}^B t_J$

$$\underline{x}_B = \underbrace{\left[{}^A R_B(\theta)^T\right]}_{{}^B R_A(\theta)} \underline{x}_A - \underbrace{\left[{}^A R_B(\theta)^T\right] {}^A\underline{t}_B}_{{}^B\underline{t}_A}$$

$${}^B R_J = {}^J R_B^{T}$$

$${}^B t_J = -\,{}^J R_B^{T}\; {}^J\underline{t}_B$$

Translation only



$X_B \parallel X_A$

$$x_A = t_x + x_B$$

$$y_A = t_y + y_B$$

Rotation only