ME 598: Final Group Project

Autonomous Object Sorting using a Mobile Robot

For this project, students will use navigation and localization to locate objects and move them to specified locations within a *Robotics Playground* simulated robot environment. An initial simulation file with preconfigured environment and arena settings will be provided and must be used as-is for the "available robot hardware" to satisfy the project objectives (sensors may not be modified and no additional sensors may be added).

Arena Layout and Robot Objective

Figure 1 illustrates the robot within the virtual arena with two colored objects. The arena walls are each 5 meters in length and the arena origin is defined at its center [0 0]. The robot will always begin at position [-1.5 -1.5] oriented in the positive x-direction, and there will always be two moveable objects present, one each at [0 1.5] and [0 -1.5], respectively. Each object will either be *red*, *green*, or *blue*, though *a maximum of one object of each color* will be present. Though the possible colors and starting locations are known, which object is which color may vary. Notice that there are six distinct initial possibilities for the object starting conditions: red-green, red-blue, green-red, green-blue, blue-red, blue-green.

The objective is for the robot to autonomously seek the objects and to sort them by color into their designated zones (red object to red zone, etc). After all objects are sorted, the robot must return to the arena's origin at [0 0] and come to a complete stop, as illustrated in Figure 2. During runtime, the robot should avoid collisions with the arena walls (this includes collisions by the front "plow").

Your group's project simulation will be tested in multiple trials, with various distinct initial conditions. Project grading will *partially* be assessed based on simulation performance in these trials.

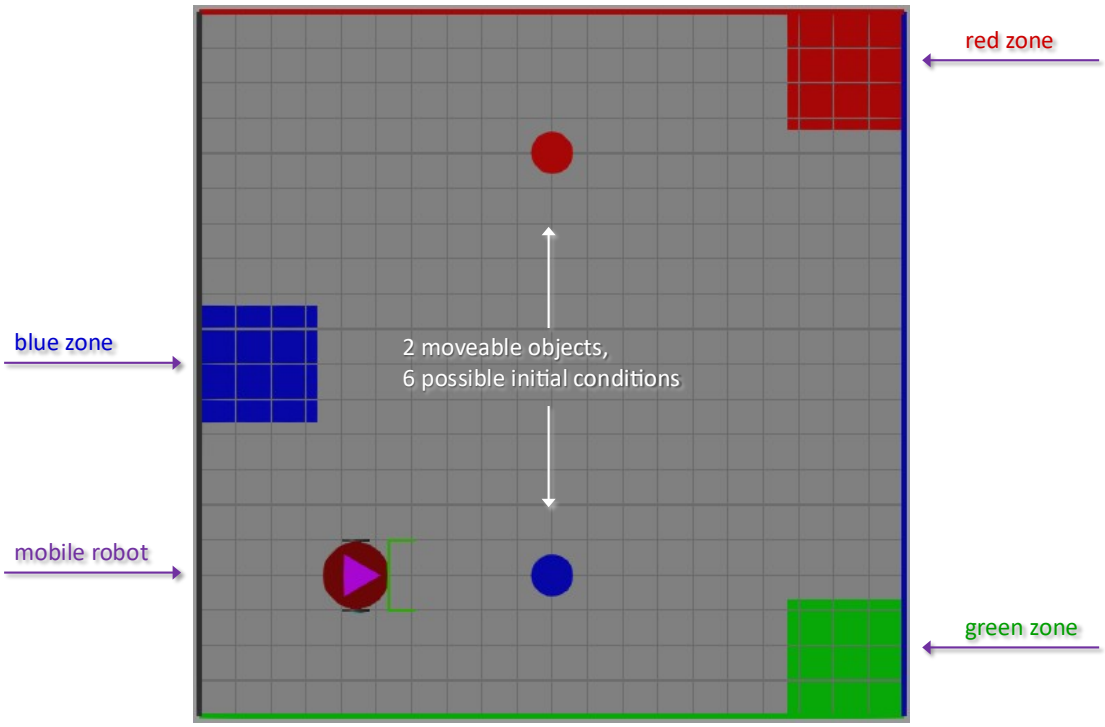

Figure 1 – Overhead view of robot within simulated environment.

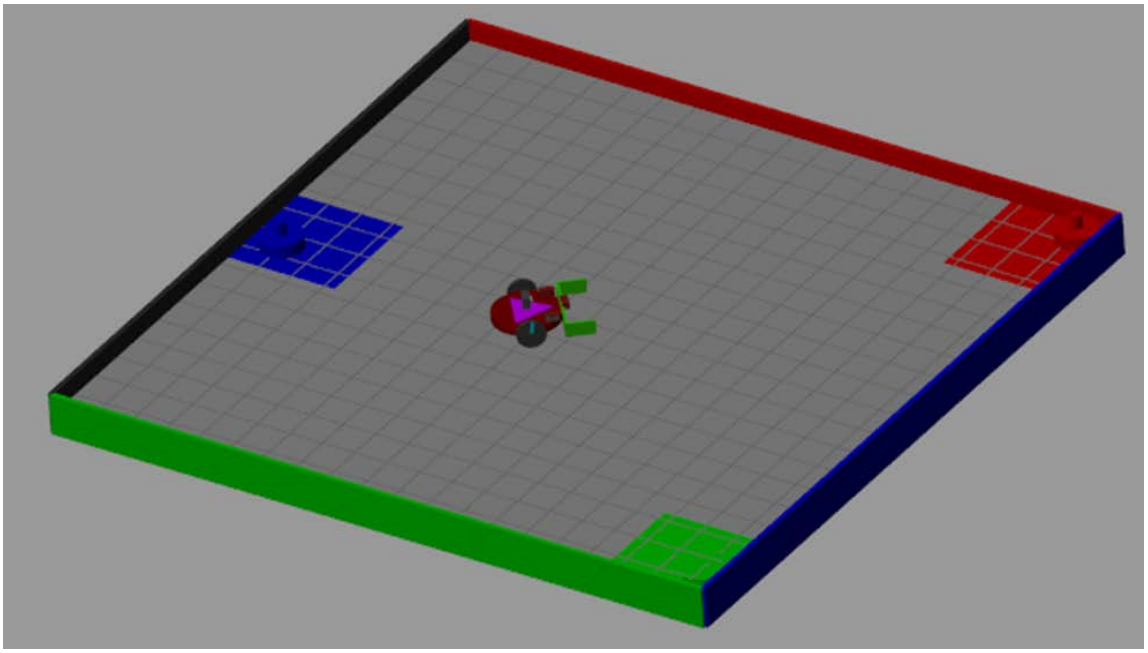

Figure 2 - Successful simulation completion.

Robot Attributes

The robot is configured with a forward-facing object detector (similar to a color-detecting camera), three distance sensors, and odometry capabilities. Groups may use any of the provided measurement signals within their simulation but *may not modify the sensor parameters.* Additionally, groups *may not* use any additional sensors from the *Robotics Playground* (what you see is what you get). Figure 3 illustrates the available robot sensors. Each sensor's configuration may be accessed by *right-clicking* the respective block, and navigating to *Mask → Look Under Mask.* Notice that sensor parameter configurations have been purposefully locked and disabled to prevent modification.

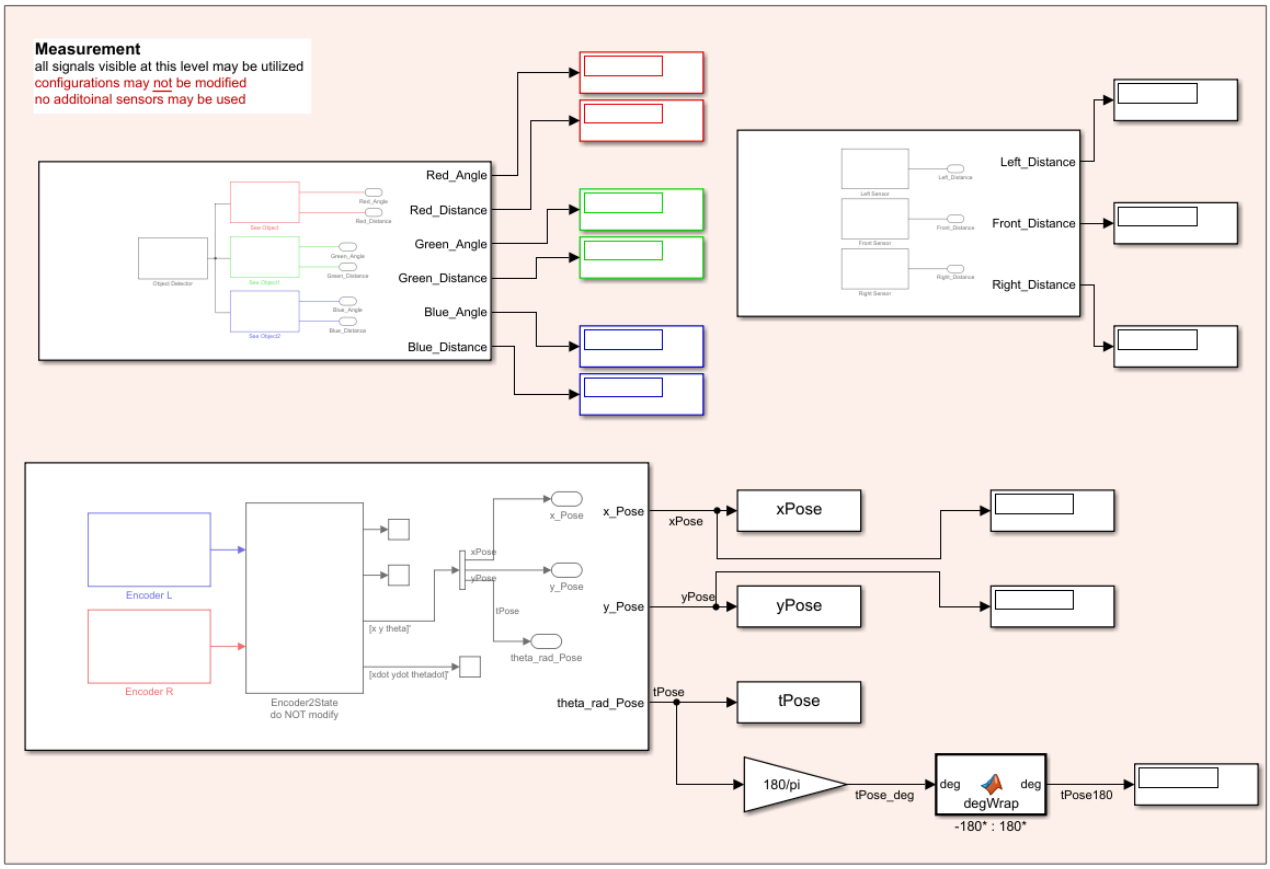

Figure 3 - Provided robot sensor signals.

Specific Tasks and Additional Criteria

Specific tasks that groups must perform for successful project implementation include:

- Create algorithms for recognizing or seeking objects based on color.
- Create algorithms that use arena geometry to determine zone locations.
- Create algorithms to push objects to their designated zone.
- Create algorithms to avoid wall collision.
- Create algorithms to return to the arena origin.

Additional Considerations

Please consider the following additional criteria:

- During operation trials, the robot must operate autonomously with no form of input from users or outside of the simulation.
- The robot *may not* be modified in any way (sensor configuration, additional sensors).
- The beginning order of the objects may be shuffled into one of six possible initial conditions. *Graders may randomly select from these conditions to test your system's performance.*
- An object is within its zone if the *object's center* is within the zone's shaded area.
- Objects must be in their designated zones *at simulation completion* (when the robot returns to the arena center and stops) to be counted.

Suggestions

- Explore utilizing user-defined function blocks within Simulink.
- Explore using Stateflow within your algorithms (see Canvas for some Stateflow resources).
  - This is a **highly recommended approach**.
- Distribute tasks among group members.
  - For example, some members could try to develop an object seek algorithm, while others could work on collision avoidance.
- Budget your time carefully to perform *integration testing.*
  - For example, even if your *seek* and *avoid* algorithms work independently, they may not directly work together.
- **DO NOT ASSUME** that if the system operates correctly under one initial condition, that it will operate under all six possible initial conditions. This will require testing and refinement.

Record and Upload a Video

Record a video of your team's best trial run from the *Mechanics Explorer* window (see *Appendix: Recording Video from the Mechanics Explorer Window*).

Upload this video to your personal *YouTube* account or *Google Drive*, and include a **hyperlink** in your report. Adjust the access privileges so that *anyone with the link* can play back the video (this might *not* be the default setting). The upload must have a title and description following the specification below:

- Title: ME 598A Fall 2025 Final Project, Group [assigned group number]
- Description: Final Project – Autonomous Sorting, Group [assigned group number]: [team member list (last-name alphabetical order)]

Example

- ME 598A Fall 2025 Final Project, Group R9
- Final Project – Autonomous Sorting, Group R9: John Doe, Bob Smith.

Project Submission

A final report must be submitted via Canvas with associated code. The submission must include:

- A complete detailed report that follows the course report guidelines. This report should address the usual requirements as well as the following.
  - How accurate / repeatable was your system's performance?
  - Discuss various robot trials with different initial conditions. Indicate:
    - XY plots of your robot's motion.
    - Simulated trial durations (seconds).
    - Level of performance (successfully sort one, two, or all three objects?).
  - Provide a detailed explanation of your algorithms and methods. Include figures to facilitate your discussion.
  - A link to your best trial video, as indicated in *Record and Upload a Video* above.
  - A student feedback summary.
    - Explain any difficulties you encountered when completing the project.
    - What did you think of the project? How long did it take you to complete (in hours)?
    - Would you recommend this same project for the future?
- All simulation files.
  - Your system *will be tested* by the professor / TAs before grading is finalized.

Appendix: Recording Video from the Mechanics Explorer Window

1. First, run your simulation fully, to the point of *completion*.
2. Starting from a *stopped and already completed* simulation, switch to the MATLAB window and *right-click* within the animation pane you wish to record. The selected pane border will be highlighted. Click on *Video Creator* in the menu that appears as illustrated in Figure 4 - Selecting the Video Creator option..

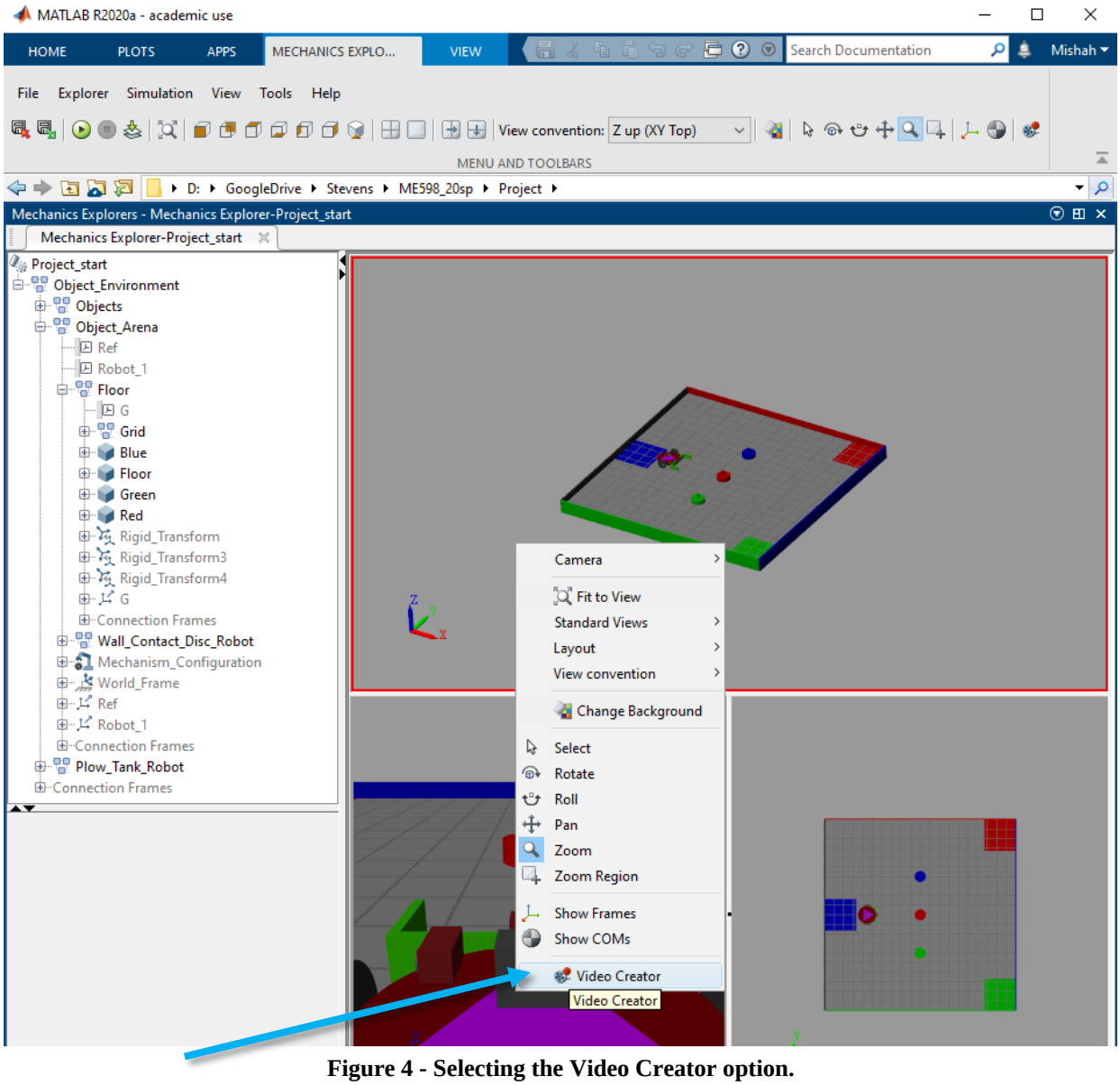

Figure 4 - Selecting the Video Creator option.

3. Click *Create* in the dialog box that follows.

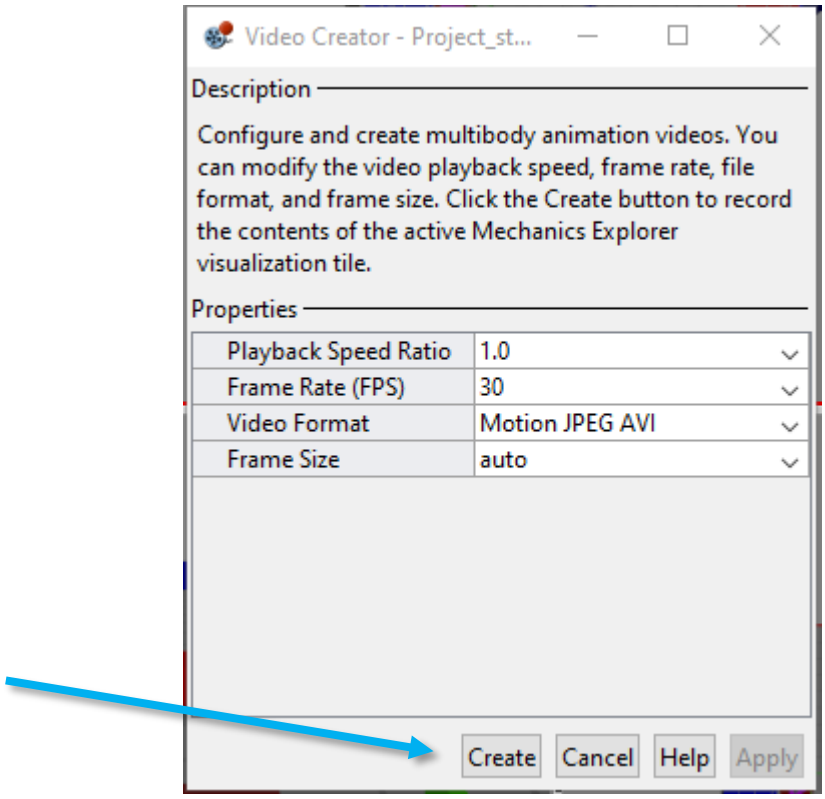

Figure 5 - Clicking Create to create a video.

4. MATLAB will re-run the animation in a separate window and generate a video file.
5. Upload the file per the project guidelines.

Appendix: Recording and Plotting Robot Pose

The provided *Simulink* model includes functionality for logging and plotting the *Robotics Playground* robot's pose.

After running a simulation, *Simulink* will generate the following variables in the global *Workspace*.

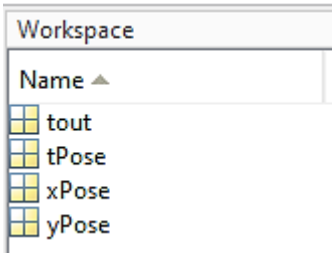

Figure 6 - Odometry workspace variables.

These data can be used to plot the robot's trajectory using Sample Code 1.

```
% Plot XY
figure(1)
plot(xPose,yPose)
title('Overhead XY Plot of Robot Trajectory')
xlabel ('X_I [m]')
ylabel ('Y_I [m]')

% Plot thea vs. time
figure(2)
plot(tout,360/pi*tPose)
title('Robot Orientation \theta vs. Time')
xlabel ('Time [sec]')
ylabel ('\theta [deg]')
```

Sample Code 1 - Sample trajectory plotting code.