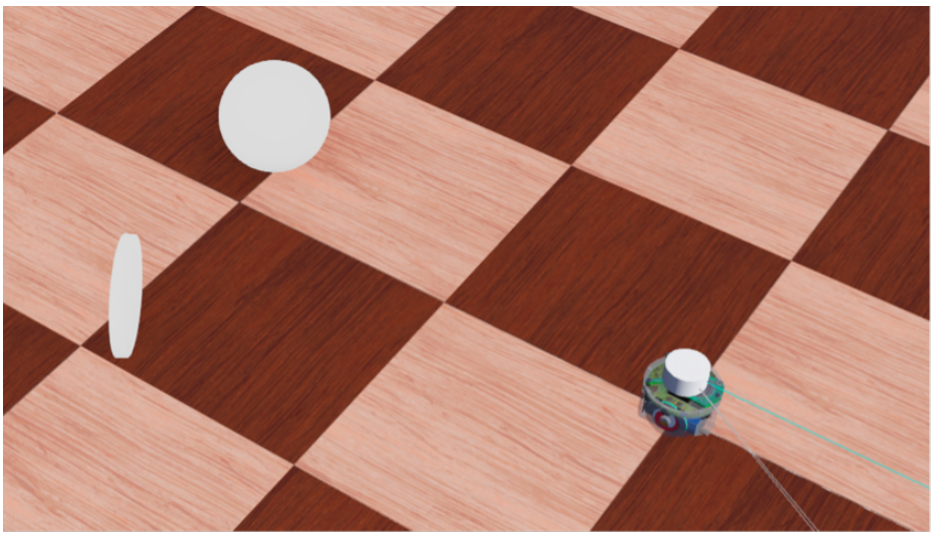
CS3630 Project 1: Multi-Modal Sensing

Released Jan 15, 2025
Due Jan 29, 2025



1 Introduction

In this project, we introduce the Webots simulator, which you will then use to program an e-puck robot to monitor an arena using its onboard camera and LiDAR. The robot needs to detect a random object (a sphere or a disk) with a known diameter that is generated within the field of view of the camera. The lab consists of two parts:

- **Part 1: Camera Only**: Determine the distance and relative angle to the object solely using the camera. Use the provided code, which detects circles in an image and outputs their center and radius, as well as the lecture notes from `L7_Camera.ppt` on Canvas to accomplish this.

- **Part 2: Camera and LiDAR**: Determine the distance to the object and identify its shape (sphere or disk) using both the camera and LiDAR. The LiDAR data consists of an array of 360 distance measurements around the robot, covering angles from 0° to 359°.

2 Webots Simulator

We will be using the Webots simulator for this project and future assignments. Follow these steps to get started:

2.1 Installation

Install Webots following the installation procedure (this includes documentation for Linux, Windows, and Mac).

2.2 World File

You have been provided with a world file and controllers for the e-puck robot. You will need to find the distance and heading to an object relative to the robot. The code structure and respective functions are described in the next section.

To test out your scripts in the Webots simulator, open the world file named `Project1.wbt` and run the simulation by clicking the *Play* button from the top menu. You can reset the simulation by clicking the *Reset* button.

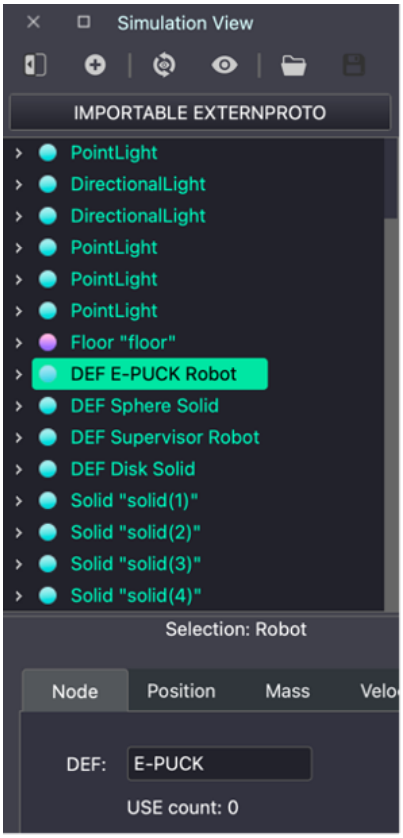
3 Controller Code

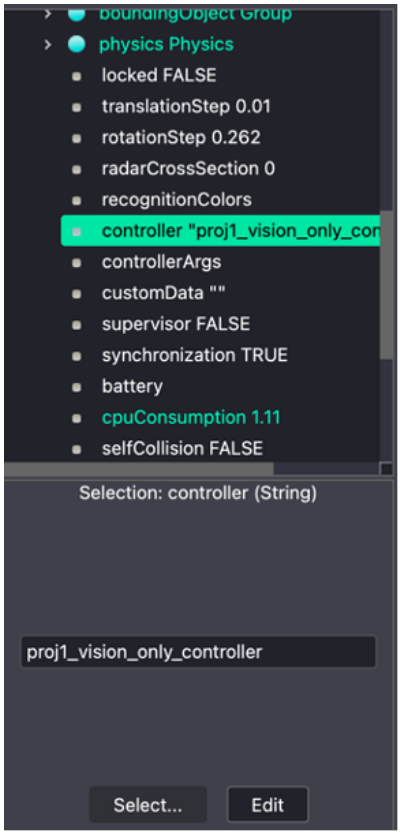
You can change the code for controlling the e-puck robot by clicking the drop-down menu for 'E-PUCK' from the left scene tree view. Then, click the 'controller' field and 'Select...' button to switch to a different controller.

- **Part 1:** Use `proj1_vision_only_controller`.
- **Part 2:** Use `proj1_vision_lidar_controller`.

4 Requirements Installation

Install the required Python packages using the `requirements.txt` file:

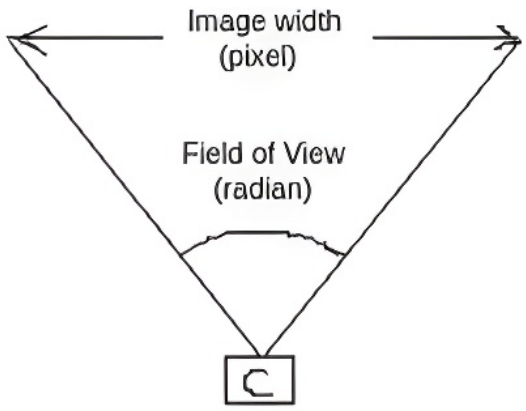





```
pip install -r requirements.txt
```

5 Assignment Preliminaries

The field of view (FOV) is the range of the world (in this project, defined in radians) in the horizontal plane that is visible.

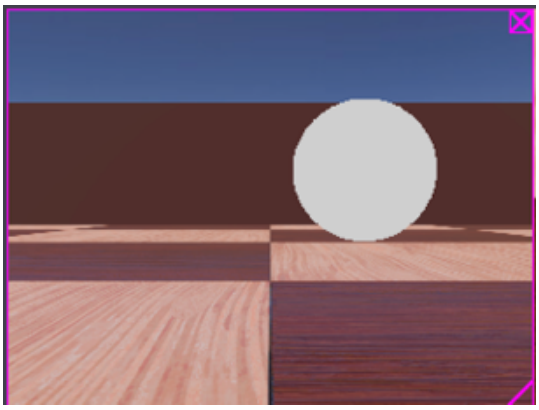


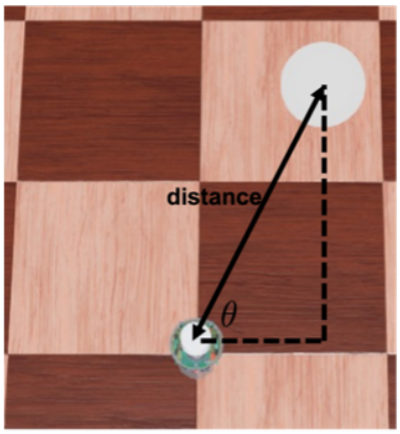
We provide the following formula that can be used to include the field of view in the calculation of focal length (in pixels) for both Parts 1 and 2:

$$\text{focal\_length} = \frac{\text{image\_width}}{2 \cdot \tan\left(\frac{\text{fov}}{2}\right)} \tag{1}$$

Hint: The field of view of the camera of the e-puck robot can be found under: `'E-PUCK/children/EPUCK_CAMERA/fieldOfView'` in the scene tree menu on the left.

6 Part 1: Determine Distance and Relative Angle to an Object

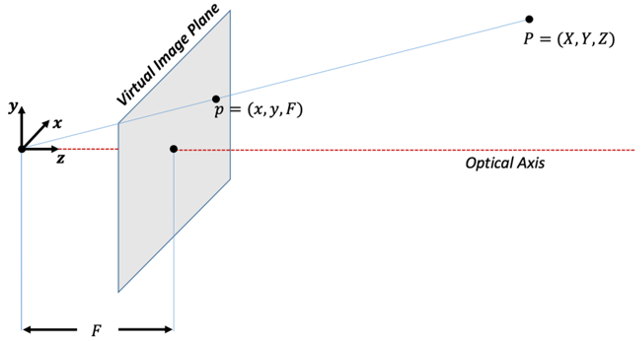




To complete Part 1, you will need to fill out the `camera_only_calculation(image, camera_fov, object_diameter)` function in:

`proj1_vision_only_controller/camera_only.py`

The `camera_only_calculation(image, camera_fov, object_diameter)` function takes in an image, camera field of view, and known object diameter, and returns the distance (from the center of the robot to the center of the object) and relative angle from the robot to the object.



Hint: For a pinhole camera model, the following relationship holds:

$$\frac{\text{focal\_length (pixels)}}{\text{distance\_to\_object}} = \frac{\text{object\_diameter\_in\_image (pixels)}}{\text{object\_diameter}}$$

Hint 2: The camera is not located at the center of the robot. The exact camera position can be found under `'E-PUCK/children/EPUCK_CAMERA/translation'` in the scene tree menu on the left.

Hint 3: The relative angle to a point on the left edge of the camera image is $-\frac{\text{fov}}{2}$, and the relative angle to a point on the right edge is $\frac{\text{fov}}{2}$. Additionally, a point on a vertical line crossing the center of the image has a relative angle of 0.

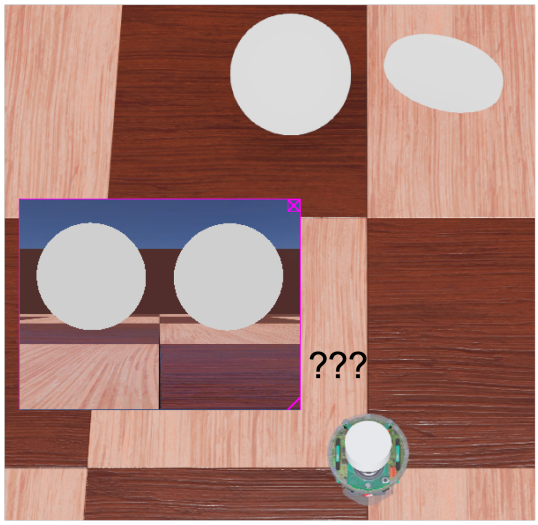
7 Part 2: Detection with Classification Using Vision and LiDAR

To complete Part 2, you will need to fill out the `camera_and_lidar_calculation(image, camera_fov, object_diameter, lidar_data)` function in:

`proj1_vision_lidar_controller/camera_and_lidar.py`

The `camera_and_lidar_calculation(image, camera_fov, object_diameter, lidar_data)` function takes in an image and the point cloud returned from a LiDAR, and returns the distance (from the center of the robot to the center of the object) and a class label (sphere or disk). For simplicity, the height of the LiDAR sensor is the same as the radius of the sphere/disk.

Note : Remember to switch the robot controller according to the instructions in Controller Code section.



Interesting Fact: You can visualize the LiDAR sensor rays by toggling `View/Optional Rendering/Show Lidar Ray Paths`.

8 Provided Functions

To complete the above two parts, you are provided with the `detect_circle.py` file, which contains a function `detect_circle(image)` that detects circles in a given image and returns the pixel values of its centroid and radius. This file (identical copies) can be found in both the `proj1_vision_lidar_controller` and `proj1_vision_only_controller` folders.

9 Local Tests

In both controller folders, there is a *test.py* that you can run to test your code on 5 images that we have provided.

10 Submission Instructions

The files you will submit to Gradescope are:

- `camera_only.py`
- `camera_and_lidar.py`

11 Allowed Libraries

You are allowed to employ the following two libraries. Note that you may not use any libraries outside this list:

- `NumPy`
- `OpenCV`
- `Matplotlib`

12 Rubric for Vision Component

Your solution for Part 1 will be evaluated with 10 test cases. We have provided 5 of these test cases for you to test locally. Each test case is worth 5 points, for a total of 50 points possible.

12.1 Distance Evaluation

For each test case, you will receive a number of points based on how close your detected distance is from the true distance. The difference between your distance and the true distance will be evaluated as a percentage. For example, if the true distance is 0.5 m and you say that the depth is 0.45 m, then the difference is 0.05 m. This is 10% of the true 0.5 m, so the grade would be 2 out of 2.5 for that test case.

Percentage Difference in Distance	Point Count
$\leq 5\%$ of true distance	2.5 points (full credit)
$\leq 10\%$ of true distance	2
$\leq 15\%$ of true distance	1.5
$\leq 20\%$ of true distance	1
$\leq 25\%$ of true distance	0.5
$> 25\%$ of true distance	0

12.2 Angle (Heading) Evaluation

For each test case, you will receive a number of points based on how close your detected angle is from the true angle. The difference between your angle and the true angle will be evaluated as difference in radians. The following table is provided to you in degrees for ease of understanding.

Degrees Difference in Heading	Point Count
≤ 5 degrees	2.5 points (full credit)
≤ 10 degrees	1.875
≤ 15 degrees	1.25
≤ 20 degrees	0.625
> 20 degrees	0

13 Rubric for Vision + LiDAR Component

Similarly, your solution for Part 2 (vision + LiDAR) will be evaluated with 10 test cases. We have provided 5 of these test cases for you to test locally. Each test case will be worth 5 points, for a total of 50 points possible for Part 2.

13.1 Distance Evaluation

For test case, you will receive a number of points based on how close your detected distance is from the true distance. The difference between your distance and the true distance will be evaluated as a percentage of the true distance.

Percentage Difference in Distance	Point Count
$\leq 5\%$ of true distance	2.5 points (full credit)
$\leq 10\%$ of true distance	2
$\leq 15\%$ of true distance	1.5
$\leq 20\%$ of true distance	1
$\leq 25\%$ of true distance	0.5
$> 25\%$ of true distance	0

13.2 Object Shape Detection

For test case, you will receive a number of points based on the correctness of the detected object shape. There is no partial credit for this part.

Object Shape Detection	Point Count
Correct	2.5 points (full credit)
Incorrect	0