CMPSC488 Section 2
Github: https://github.com/joeoakes/CMPSC488FA24Sec2Team2
Testing Status Page (Spreadsheet):
https://docs.google.com/spreadsheets/d/1gX3ksFSNGmzlHVL34X6IZp0Qe264xLXucDIeHmrW3fk/edit?usp=sharing

Testing Environment and Framework

We employ the use of Pytest to run our unit, integration, and hardware tests. Tests are written for all Python modules and are setup in the following way:

Directory Structure

the `__init__.py` tells python to treat all files in this directory as python packages allowing us to import into our test modules.

```
├── src/ # main source code
│ ├── module1.py
│ ├── module2.py
│ ├── __init__.py
│
├── tests/ # tests folder
│ ├── test_module1.py
│ ├── test_module2.py
│ ├── __init__.py
│
├── requirements.txt
├── README.md```
```

Example Unit Test

```
def test_detect_object_success(mock_torch_hub, mock_model):
        """Test detect_object with a successful detection."""
```

```
        model_path = "../src/yolo/model"
        weights_path = "../src/yolo/weights"
        image_path = "../Dataset/images"
        result = detect_object(model_path, weights_path, image_path)

        assert result == (100, 50, 200, 150, "MockedObject")
        mock_torch_hub.assert_called_once_with(model_path,
weights_path,
        source="local")
        mock_model.return_value.assert_called_once_with(image_path)
```

this is an example unit test for evaluating the performance of the YOLO module that's used in object recognition. The parameters in the function signature are special pytest objects called fixtures that are used to create common objects to be tested in the function below. These are marked with `@pytest.fixture` decorators. The function above runs tests on a torch hub and model that are fixtures generated to perform this test among others.

Continuous Integration

The uniform and centralized testing directory structure makes it simple to run tests. Locally, this can be done by running pytest tests (tests being the name of the directory that holds all our tests) in the parent directory of the project. This is done automatically when commits are made to main through Github Actions using a YAML script.

YAML Test Script

(link:https://github.com/joeoakes/CMPSC488FA24Sec2Team2/actions/runs/12002110016/job/33453581141)

Test Status Page
(link:
https://docs.google.com/spreadsheets/d/1gX3ksFSNGmzlHVL34X6IZp0Qe264xLXucDIeHmrW3fk/edit?usp=sharing)

1. installs the correct version of python
2. downloads all dependencies (or pulls from cache)

3. runs pytest on the parent directory (triggering all tests)
4. returns the test results and cleans up.

Our Testing Status Page Features the names of each test function, a brief description of the test case, the creator of the test and its status (as well as some comments).

Test Statistics

Using SQL, these side data displays automatically display some general statistics about our testing status, namely:

- Total number of working, pending, and not working tests
- Number of failures in each testing category (only shows categories with at least one failure)

Test Log

TestCategory	TestCase	FunctionName	Tester
Navigation	placing robot on grid	test_place_robot()	Aaron Feinberg
Navigation	robot matches initial position in instantiation	test_robot_initial_position()	Aaron Feinberg
Navigation	ensure nearby positions are free	test_valid_nearby_positions()	Aaron Feinberg
Navigation	ensure next cell is best (from hueristic)	test_next_best_cell()	Aaron Feinberg
Navigation	check valid and invalid spaces based on grid	test_bounds_checking()	Aaron Feinberg
Kinect	getting depth data from kinect sensor	test_get_depth_data()	Aaron Feinberg
Kinect	displaying image from kinect	test_display_depth_image()	Aaron Feinberg
Kinect	getting distance for a single pixel from bounding box	test_get_distance_at_pixel()	Aaron Feinberg
Kinect	gets a region defined by a border box	test_get_sub_region()	Aaron Feinberg
Kinect	drops pixel values less than 255 in bounding box (unrelated values)	test_remove_less_then_255()	Aaron Feinberg
Kinect	calculates avg depth value for all pixels in bounded region	test_get_avg_depth_in_region()	Aaron Feinberg
Kinect	creates a bounding box on the frame	test_draw_border	Aaron Feinberg
Movement	moves all motors forward	move_forward()	Clay Blockinger
Movement	moves all motors Backward	move_backward()	Clay Blockinger
Movement	moves the motors in a way to shift left	move_left()	Clay Blockinger
Movement	moves the motors in a way to shift right	move_right()	Clay Blockinger
Movement	Spins clockwise	rotate_clockwise()	Clay Blockinger
Movement	Spins counterclockwise	rotate_counterclockwise()	Clay Blockinger
Movement	shield gose down	deploy_shield()	Clay Blockinger
Movement	shield gose up	retract_shield()	Clay Blockinger
Movement	laser points to 90	set_laser_arm_position(90)	Clay Blockinger
Movement	laser points to 0	set_laser_arm_position(0)	Clay Blockinger
Movement	laser points to 180	set_laser_arm_position(180)	Clay Blockinger
Movement	all stop no movement	stop_all()	Clay Blockinger
Beacons	Mock data for beacons with positions and distances.	mock_beacon_data()	Mansib Ahmed
Beacons	Test RSSI to distance conversion.	test_rssi_to_distance()	Mansib Ahmed
Beacons	Test the trilaterate function.	test_trilaterate()	Mansib Ahmed
Beacons	Getting distances from beacons	test_get_distances()	Mansib Ahmed
Beacons	test populating grid with beacon data	test_print_grid()	Mansib Ahmed
YOLO	Fixture to create a mocked model with necessary behavior.	mock_model()	Mansib Ahmed
YOLO	mock torch hub	mock_torch_hub()	Mansib Ahmed
YOLO	Test detect object with a successful detection.	test_detect_object_success()	Mansib Ahmed
YOLO	Test detect object when no detection meets the confidence threshold.	test_detect_object_no_detection()	Mansib Ahmed
YOLO	Test detect object when an invalid model path is provided.	test_detect_object_invalid_model_path()	Mansib Ahmed
CoreS3	Set the class attributes to default values	test_imu_initialization()	Joe Porrino
CoreS3	Assert that the imu get data function can read the mock serial data	test_get_imu_data_successful_read(mock_serial)	Joe Porrino

Statistics

Statistics	
Status	**Count**
Working	38
Pending	5

Failures Per Test Category	
TestCategory	**Count**