CS437: Internet of Things
Lab1 Part2

Name: Max, Nishant, Molly
NetID: mtueck2, nsheikh2, tyy2
Late days used: 4
Video Link:
https://drive.google.com/file/d/11e1jdUh1YPVm0O9aLd6tUidAVJgBSs0B/view?usp=sharing

NOTE: coordinates are in terms of (row, col)

Design Considerations
Miscellaneous:

- We had significant problems with the car kit.
    - Wheels would fall off when turning, so we eventually decided to put tape on the wheels to reduce friction. This mitigated the issue to some degree.
    - The car would sometimes turn halfway, sometimes due to dragging along the floor. We couldn't find an easy fix for this.

Advanced Mapping:

- The 100 x 100 numpy array obstacle map translates to 1 cm x 1 cm in space. It considers the car at the center of the y axis (column), where (0,0) and (0,100) indicate left 50 cm and right 50cm of the car, and the x axis shows the 100cm in front of the car.
- Reliability and repeatability of ultrasonic sensors were a main challenge. Any noise could be enlarged and bias the angle of an obstacle due to linear interpolating from the sampled points. The variability between our teammates' ultrasonic sensors could also greatly affect the accuracy of the obstacle map.
- The ultrasonic sweep to the left and right continuously at the rate of about 1 Hz, which should be able to capture dynamic objects moving at a reasonable speed.
- We considered using our camera-based object detection functions to further enhance the ultrasonic map, but we found that the ultrasonic sensor was usually able to detect obstacles without help from the camera.
- We wrote a script to show ultrasonic sensor output in a window using OpenCV, which was more convenient than printing to the console. Since we could "see what the sensor sees", it was easier to tune our polling rate and other factors.

Object Detection:

- *Would hardware acceleration help in image processing? Have the packages mentioned above leveraged it? If not, how could you properly leverage hardware acceleration?*
    - Hardware acceleration for image processing and inference would have yielded better performance compared to what we achieved on the Raspberry Pi's CPU. For example, we could have offloaded work to an external processor like Google's Coral TPU. While it may have been possible to utilize the Raspberry Pi's GPU in a similar fashion (with libraries such as py-videocore6), we decided that this would not be feasible in a short time frame.
        - We considered moving the BGR to RGB conversion to the GPU, but we found that this function didn't appreciably affect execution time.
- *Would multithreading help increase (or hurt) the performance of your program?*
    - Multithreading significantly increased the performance of the program. After some experimentation, we found that the optimal amount of threads was four, which makes sense given that the Raspberry Pi 4B's processor has four cores. Increasing the thread count beyond four led to abysmal performance.
    - We decided to move ultrasound and A* to a separate process, which allowed us to update them more frequently. Previously, the image processing would cause excessive delay. Even though this solution likely cuts into our image processing thread budget, making this change allowed for the car to react to its surroundings (using the ultrasonic sensor) more responsively, while not significantly hurting our ability to detect stop signs with the camera.
- *How would you choose the trade-off between frame rate and detection accuracy?*
    - Frame rate was a limiting factor on the speed of our car. If the car moves too fast, the object detection process will be unable to catch up, causing the car to react to obstacles too late.
    - For detection accuracy, we experimented with confidence thresholds for our obstacles to avoid false positives. Additionally, we achieved better accuracy in well-lit environments, which correlated with camera performance.
    - We settled on the lowest resolution compatible with the camera, 640x480, to lower image processing time. This yielded a higher frame rate at the cost of lower detection accuracy due to a smaller field of view

    - Additionally, since car speed wasn't a specific requirement, we were able to balance the frame rate and detection accuracy by decreasing the speed of the car and the dynamic objects in our obstacle course.
    - We imagine that tradeoffs between frame rate and detection accuracy must be made in the real world, as well. A low frame rate would render the car unable to react quickly to changes in its environment, like someone suddenly stepping onto the road. Weak detection accuracy would lead to both inconvenience and danger from improper detection of people or road signage/structures.

Routing

- A* routing updates immediately after the function receives an obstacle map passed from the previous steps, which includes information from ultrasonic maps with results from OpenCV and TensorFlow detection added.
- The car starts at (49,0) and keeps track of its own location on the grid. While navigating to the goal position, the relative distance to the goal position is calculated and taken into account.
- The A* algorithm can "see" additional space around what the ultrasonic sensor can see - this is to ensure that the car can attempt to navigate even when its "vision" is obscured by closeby objects

Full Self-driving

- Traffic rules:
    - The car must stop at stop signs.
- The car is programmed to drive straight, taking left and right turns as necessary. We felt that this suitably mimicked the behavior of an actual car driving on the streets.
- Stop signs and pedestrians used in our obstacle course are of a standard size, which makes it easier to estimate their proximity with only one camera. As most road signage is of standard size, we felt that this was an acceptable compromise.
    - Initially, we tried to use the ultrasonic sensor's data to get a distance component for obstacles detected by the camera, but this didn't produce consistent results.

Name	**Contribution**

Max Tuecke	Algorithm ideation, environment setup, A* routing, integration testing, document design considerations, demo video
Nishant Sheikh	Algorithm ideation, environment setup, obstacle detection (camera), multiprocess, integration testing, document design considerations, demo video
Molly Yang	Algorithm ideation, environment setup, mapping with ultrasonic sensor, integration testing, document design considerations, demo video