Milestone Two
Environmental Perception

ARTICLE 1: Objective

This milestone consists of two parts:

Part 1:

- The first part of this task required:
  - Add gaussian noise to the Odometry readings with variance.

$$X = half\ of\ car's\ wheel\ base$$
$$Y = half\ of\ car's\ wheel\ track$$
$$Yaw = half\ of\ car's\ wheel\ track$$

  - Apply one of the filtering methods, for example “Kalman Filter, Extended Kalman Filter”.
    - You must obtain values that are close to the original readings without noise.
    - You should then use the filtered readings to run a closed loop controller that can follow the same 4 tracks.

Part 2:

- The second part “Perception” involves using the car’s camera module:
  - To identify the required objects (human, car, cone).
  - In this milestone, you must add to the scene multiple objects from the required objects.
    - You don’t have to move the car in this part of the milestone, just keep the car stationary.
    - You must be able to detect a human being walking in front of the car and draw a bounding box around it with (label, and accuracy) that indicates the object, you must do the same for the vehicle and the cone separately.

ARTICLE 2: Requirements

Part I:

- Record 2 videos for each track you already have from the previous milestone.
    - One with an open loop controller using the noisy data you created,
    - One with a closed loop controller that has the data you filtered as feedback.
- Export the actual data, the data after adding the noises, the ground truth data in a CSV file.
- Plot the position of the car in X and Y in case of (open loop controller with noised data – closed loop controller with filtered data)
- You are required to finish the path as fast as possible.
- Calculate the RMS and plot it.

Part II:

- You are required to insert (Human, cone and car) your own scene.
- Using the camera model in the car scene:
    - Insert a human being that acts as pedestrian passing in front of the camera and detecting it while drawing a boundary box around the object showing the centroid with an annotation showing the (name and accuracy) of the object “in this case a human detected with accuracy 0.9”.
- Using the same previous setup, you are required to detect a car standing in front of your Car (Camera) and draw a bounding box around it shows the centroid, label and accuracy.
- This time you are required to detect a cone in front of a camera.
- You can start testing with a single object in front of the camera then after verifying the effectiveness of your module, you can increase the number of objects of the same kind in front of your camera and try to identify them.

ARTICLE 3: Submission criteria:

You are required to:

- Record a video of your car performing the previously given tasks, as mentioned 2 videos for each track.
- A CSV file that contains the ground truth values of the sensors in comparison to the filtered ones.
- Plots of the position of the car with respect to X and Y axis using the noised sensory data, filtered sensory data and the ground truth data.
- Minimize the time used by the car to finish the track as accurately as possible.
- For the perception part, you are required to submit a screenshot from the scene showing the view of the camera, and the detected object with the boundary box around it, showing the Centroid, label and accuracy.
- If you used one or more of MATLAB's tools in the submission you will be awarded 5% bonus points from overall score.

Notes

- To accomplish this milestone, you need to have some knowledge about the following topics:
    - Data Filters like Kalman, Extended Kalman
    - Object detection using famous algorithms for example: YOLO, TensorFlow, R-CNN, RetinaNet, Etc.
    - Types of closed loop controllers like (PID, Pure Pursuit, Stanly, etc.)
- You are not required to use any of the mentioned algorithms or techniques, but they are mentioned as an eye opener so you can look for a suitable method for your task.
- Codes will be subjected to a plagiarism checker to validate authenticity.
- No late submissions will be accepted.

Simulation

Mounted Sensors:

- Odometer sensor.
- Mono-Camera with resolution 960x480.
- Velocity sensor.
- Velodyne-32 lidar

Topics:

```
/SteeringAngle #subscriber
#  Type: std_msgs/Float64
#  info: Used to set the steering angle
#  uint: Degree
/cmd_vel #subscriber
# Type: std_msgs/Float64
# info: Used to set the force on gas pedal range from 0~1 (0 for no pressure
on gas pedal and 1 for full pressure on gas pedal).
/odom # publisher
# type: nav_msgs/Odometry
# info: Used to publish the position and velocity of the vehicle
# uint: (position is in meter realtive to the world) (orinetation is quatrion
and you must transoform it into euler) (Velocity is in meter/sec)
/Imu
# type: sensor_msgs/Imu
# info: Used to publish the linear acceleration and angular velocity and
angular position of the vehicle
# uint: (linear acceleration is in m/s^2 realtive to the world) (angular
velocity is rad/sec)
/velodyne_points
# type: sensor_msgs/PointCloud2
# info: Used to publish Lidar points cloud
# uint: point distance is in meter
/brakes
# type: std_msgs/Float64
# info: Used to brake apply brake pressure from 0.0 to 1.0
# unit: No unit
/startSimulation
# type: std_msgs/Bool
# info: Used to start the simulator if the the message is "TRUE"
# unit: No unit
/pauseSimulation
# type: std_msgs/Bool
# info: Used to pause the simulator if the the message is "TRUE"
# unit: No unit
/stopSimulation
# type: std_msgs/Bool
# info: Used to stop the simulator if the the message is "TRUE"
# unit: No unit
/image
# type: sensor_msgs/Image
# info: Used to publish the image from the camera in the simulator
# unit: No unit
```

Note:

The simulator is available on EVER Academy at Udemy Platform in Autonomous Track Milestone Two.