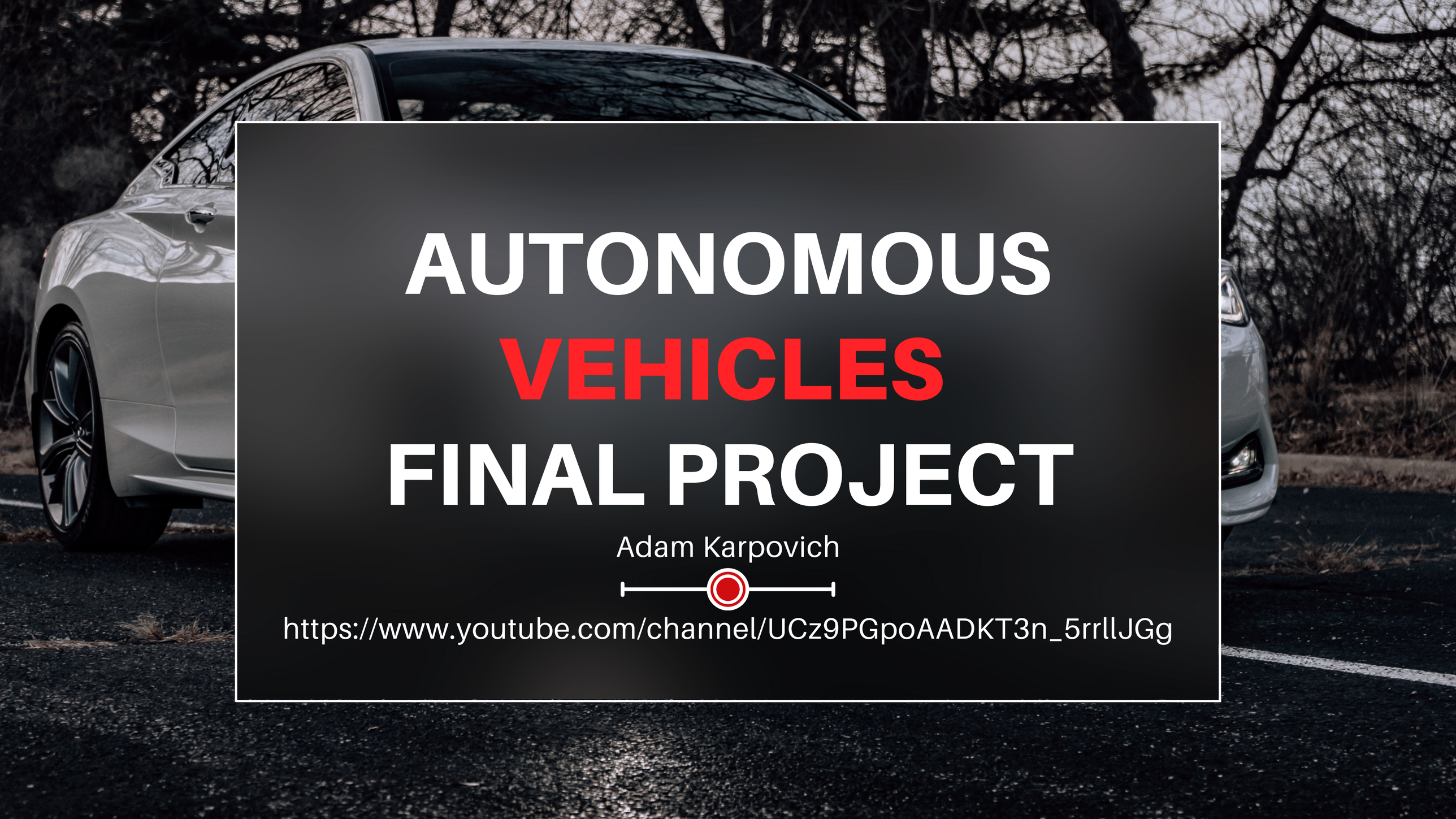


AUTONOMOUS VEHICLES FINAL PROJECT

Adam Karpovich

https://www.youtube.com/channel/UCz9PGpoAADKT3n_5rrllJGg

PROJECT NAME:

Object Detection/ Obstacle Avoidance using ROS and Deep Learning.

1. This project involves autonomous cars with object detection and obstacle avoidance capabilities using ROS and deep learning.

2. A camera and sensors are used to provide data, which is processed using OpenCV and ROS modules to generate movement commands for the car.

3. The project provides a opportunity to learn about robotics and ROS with potential real-world applications.

An autonomous car that can navigate a simple environment while avoiding obstacles.

Hardware requirements:

- Microcontroller board (Raspberry Pi for example)
- Camera module for object detection
- Ultrasonic or LiDAR sensor for obstacle avoidance
- Optional: Bluetooth or WiFi module for remote control

Software requirements:

- ROS (Robot Operating System) framework for building the software
- OpenCV library for image processing and object detection
- TensorFlow or PyTorch for training a deep learning model for object detection
- Python programming language for writing the code

On the next page is a small demo of the code that could serve as a starting point

The script assumes that you have a robot with motors, a camera module, and an ultrasonic or LiDAR sensor connected to your computer, and that you have installed the necessary libraries/Docker and drivers. If you don't have the required hardware and software, the script may not work as intended.

```python

def scan_callback(scan_msg):
    # detect obstacles within a certain range
    if min(scan_msg.ranges) < 0.5:
        # stop the robot if an obstacle is detected
        vel_msg.linear.x = 0
        vel_pub.publish(vel_msg)
    else:
        # move the robot based on keyboard input
        key = readchar.readkey()
        if key == 'w':
            vel_msg.linear.x = 0.5
            vel_msg.angular.z = 0
        elif key == 'a':
            vel_msg.linear.x = 0
            vel_msg.angular.z = 0.5
        elif key == 's':
            vel_msg.linear.x = -0.5
            vel_msg.angular.z = 0
        elif key == 'd':
            vel_msg.linear.x = 0
            vel_msg.angular.z = -0.5
        else:
            vel_msg.linear.x = 0
            vel_msg.angular.z = 0
        vel_pub.publish(vel_msg)

if __name__ == '_main_':
    rospy.init_node('simple_robot')
    vel_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
    scan_sub = rospy.Subscriber('/scan', LaserScan, scan_callback)
    vel_msg = Twist()

    while not rospy.is_shutdown():
        # wait for new laser scan messages
        time.sleep(0.01)

```

requirements.txt

```
roslibpy
python-geometry-msgs
sensor_msgs.msg
readchar
```

if you want to run this script -
run: python <filename>.py

if you want to run this script without installing all the modules on a machine with docker installed on it
here is the Dockerfile:
and how to run the container

Dockerfile

```dockerfile
# Set the base image
FROM python:3.9-slim-buster

# Set the working directory
WORKDIR /app

# Copy the requirements file to the working directory
COPY requirements.txt .

# Install the required packages
RUN pip install --no-cache-dir -r requirements.txt

# Copy the autocarPython.py file to the working directory
COPY autocarPython.py .

# Set the command to run the autocarPython.py script
CMD ["python", "autocarPython.py"]

```

docker build -t autocar .
docker run -it autocar

import the readchar library, which allows us to read keyboard input.

In the scan_callback function, we check for keyboard input using readkey() function from readchar library, and set the robot's velocity based on the pressed key.

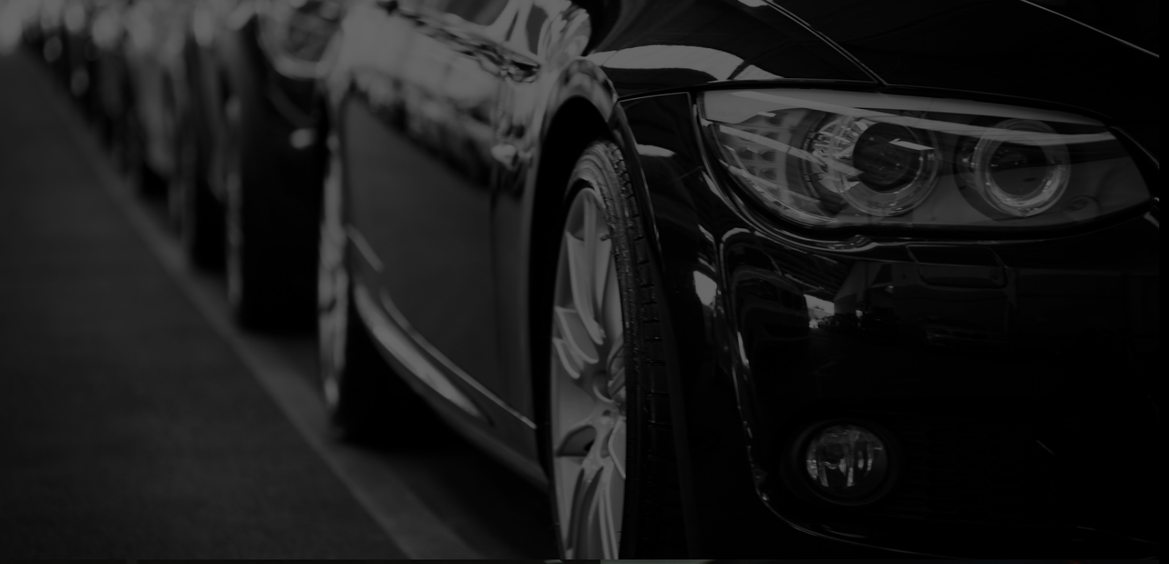
If the 'w' key is pressed, the robot moves forward 0.5 m/s.

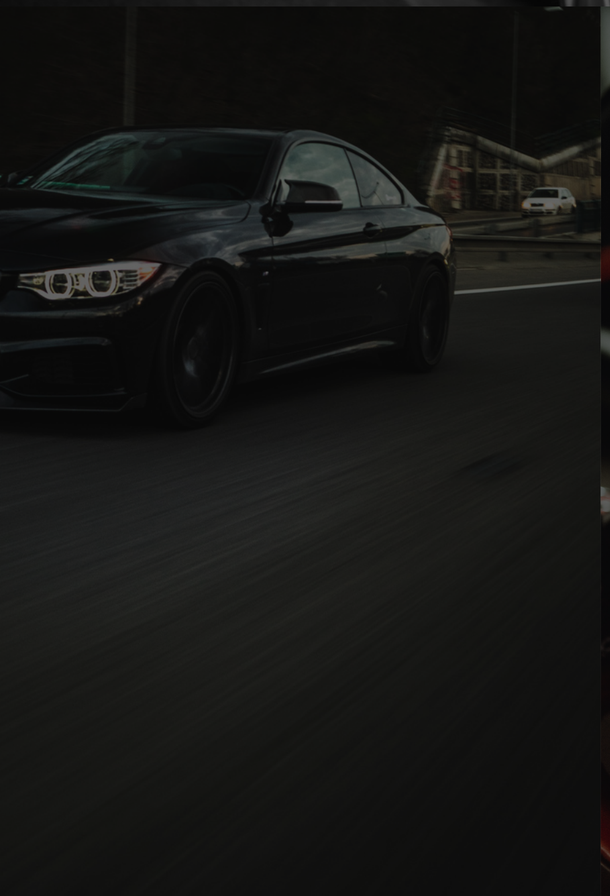
If the 'a' key is pressed, the robot turns left with an angular velocity of 0.5 rad/s and zero linear velocity. Similarly, the 'd' key makes it turn right.

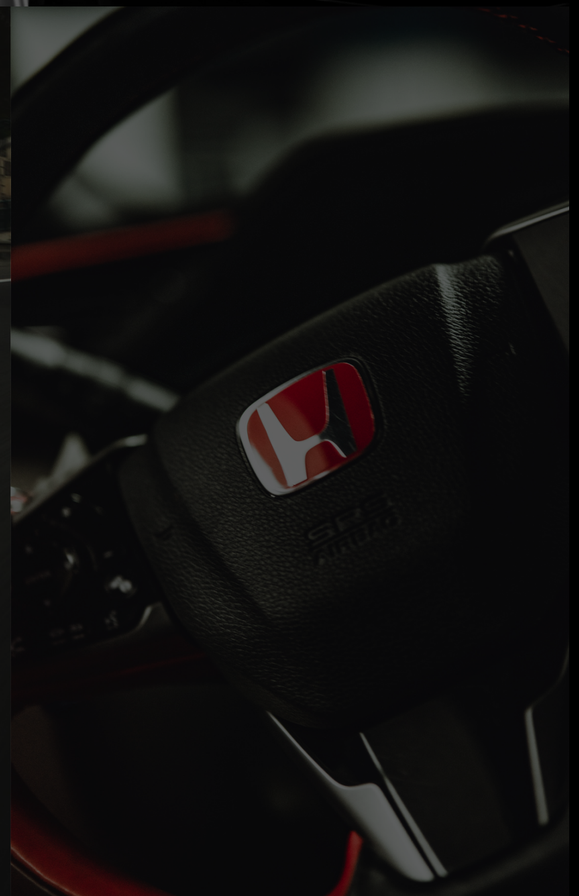
The 's' key makes the robot move backward.

If no key is pressed, the robot's velocity is set to zero.

Finally, we enter a loop where we continuously wait for new laser scan messages to arrive using time.sleep(0.01) call. This time, we reduce the sleep time to 0.01 seconds to allow for faster keyboard input.

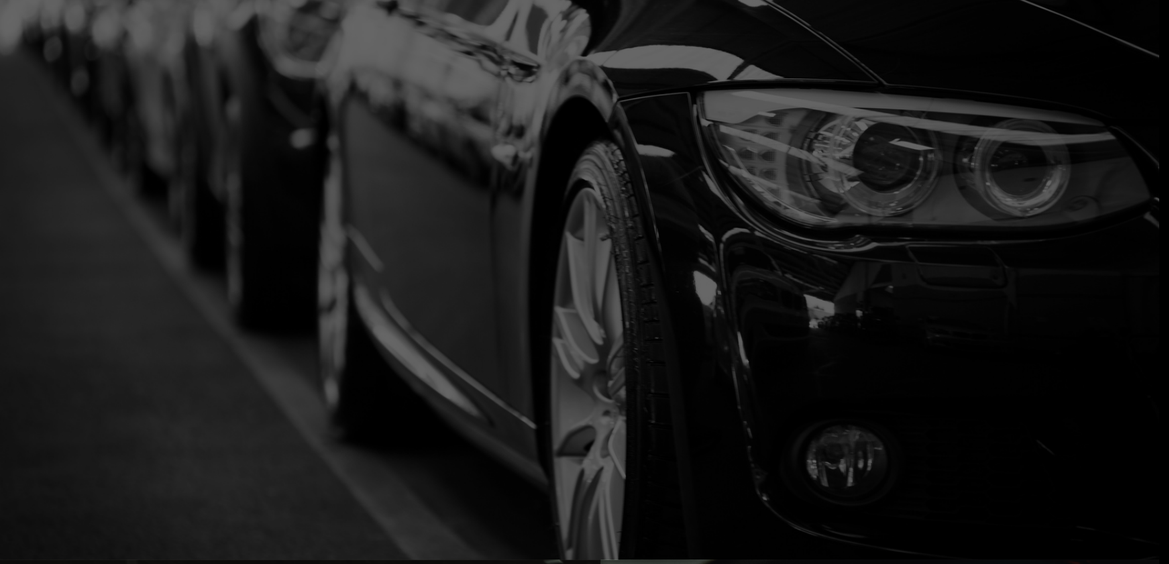


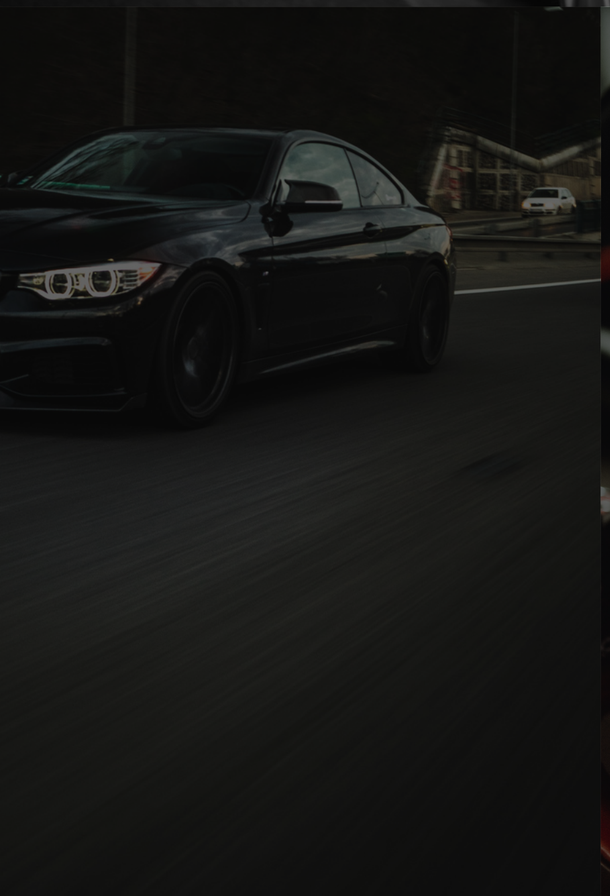


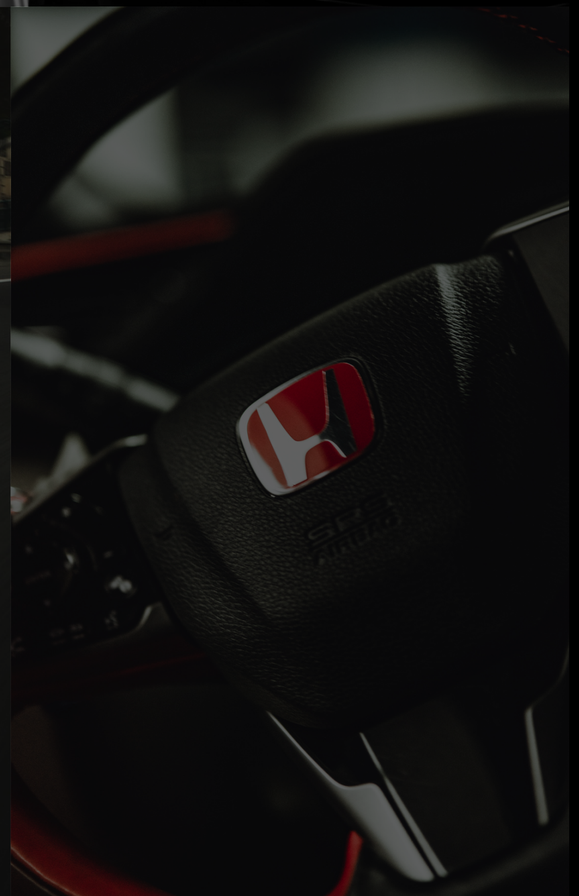


STEPS (1 OUT OF 2):

1. Connect the car to the microcontroller board.
2. Install the camera module and ultrasonic/LiDAR sensor on the car.
3. Install the necessary software libraries and tools on the microcontroller board, including ROS, OpenCV, and TensorFlow/PyTorch.
4. Write a ROS node that subscribes to the camera feed and processes it using OpenCV for object detection. Use the trained deep learning model to detect objects in the camera feed, and output their positions and sizes.
5. Write another ROS node that subscribes to the ultrasonic/LiDAR sensor data and uses it to detect obstacles in the car's path.







STEPS (2 OUT OF 2):

1. Combine the object detection and obstacle avoidance nodes to generate a set of commands for the car's movement. For example, if an object is detected in front of the car, it should stop or turn left or right to avoid it.
2. Test the autonomous car in a simple indoor environment with obstacles such as walls, chairs, and tables. Tune the parameters of the object detection and obstacle avoidance algorithms to optimize the car's performance.
3. Optional: Add a Bluetooth or WiFi module to the microcontroller board to enable remote control of the car from a smartphone, tablet, or laptop.
4. Use "Solutions for Automotive" servises provided by AWS.
5. Use AWS IoT to send telemetry data from the car to the cloud.

Machine learning algorithms in Object Detection/Obstacle Avoidance

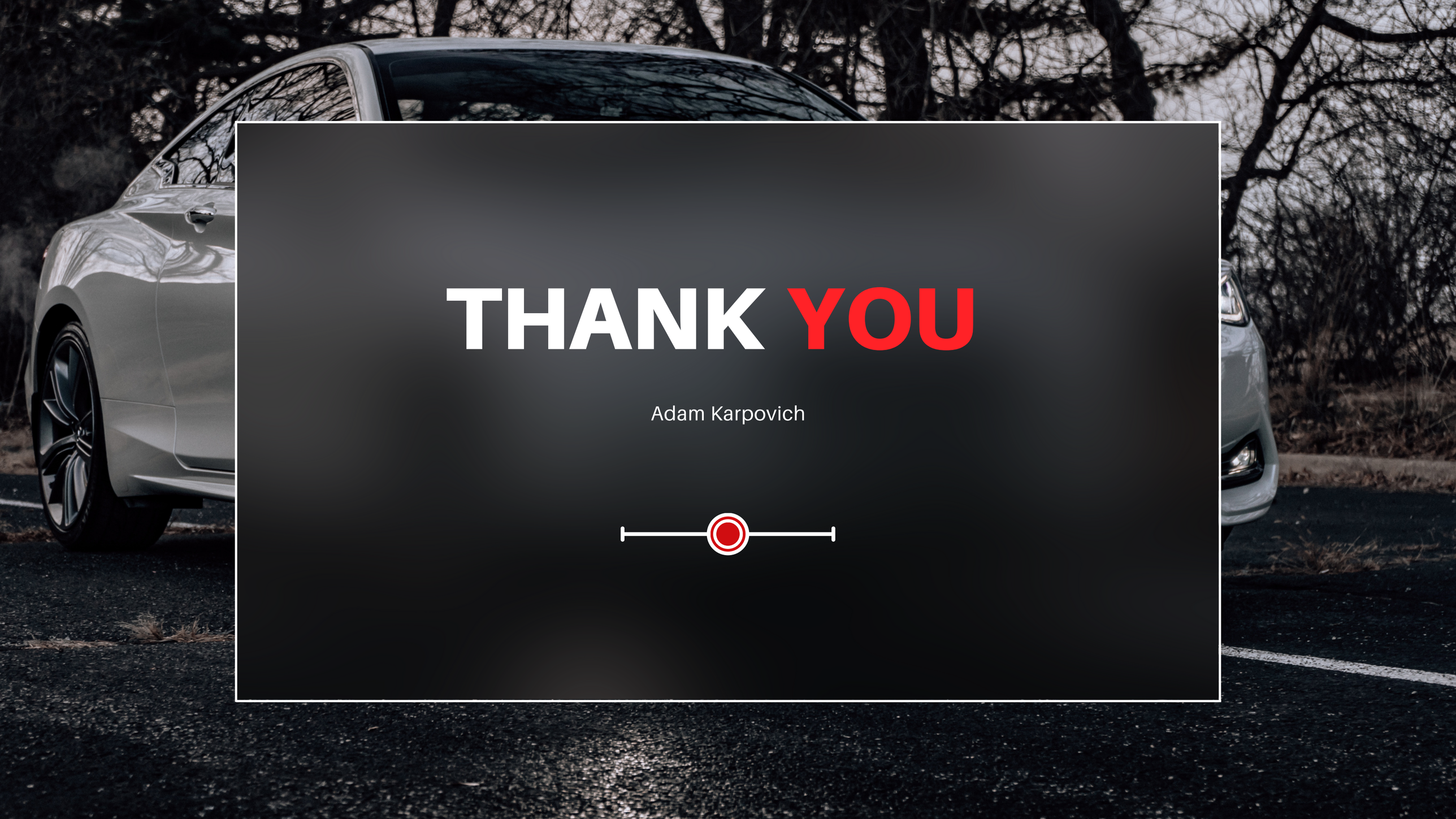
Once you have a dataset, you can train a CNN using a deep learning framework such as TensorFlow or PyTorch. The trained model can then be integrated into the object detection node of your autonomous car and used to detect and classify objects in real-time.

Similarly, you can use machine learning to improve the obstacle avoidance algorithm by training a model to predict the best action to take - turn left, turn right, or stop based on the position and distance of obstacles in the car's path. You can use a supervised learning approach to train the model on a dataset of labeled examples where each input is a set of sensor readings and the corresponding output is the best action to take.

SUMMERY

The project involves building an autonomous car that can navigate a simple indoor environment with obstacles such as walls, chairs, and tables. The car is equipped with sensors such as ultrasonic sensors and a camera to detect objects in its path and an obstacle avoidance algorithm to generate commands for the car's movement.

Machine learning techniques such as CNNs can be used to improve the object detection and obstacle avoidance algorithms. The commands generated by the algorithm are sent to a microcontroller board, which controls the car's movement using motors. The system can be further enhanced by adding a Bluetooth or WiFi module to the microcontroller board to enable remote control of the car from a smartphone, tablet, or laptop. AWS Solutions for Automotive services can be used to enable remote monitoring and management of the car. The system can be optimized by tuning the parameters of the algorithms and the motor control.



THANK YOU

Adam Karpovich