ASU Racing Team
Formula AI and Shell AI
AI and Robotics Workshop

Name: Waleed Khalid Mohamed Waleed Alzamil
CUMULATIVE GPA: 3.4
ID: 2002011
Year: Junior (Electrical and Computer Engineering)
My Resume: My resume - Google Drive
GitHub: WaleedAlzamil80 (Waleed Alzamil) (github.com)
LinkedIn: Waleed Alzamil | LinkedIn
Kaggle: waleed alzamil | Novice | Kaggle
Personal e-mail: waleedalzamil80@gmail.com

ASU Racing Team
Formula AI and Shell AI
AI and Robotics Workshop

Technical Mission

This online test is aiming to test your ability to search and self-study in technical topics that you may know or don't know about, feel free to get help from the internet, doctors, teaching assistants, textbooks, or whatever source you fnd available to you (but you can't work on this mission with others that are also planning to attend this workshop)

Please read all the following questions carefully and provide your answers in a single pdf fle, aid your answers with fgures, graphs, or sketches if needed.

Please **don't be intimidated** if you fnd the questions hard, we don't expect you to solve most of them; the goal is to expose you to some of the topics that will be explained in greater depth in the workshop and measure your research skills, so do your best and (more importantly) try to learn along the way.

Submit your answers to this google form:
https://docs.google.com/forms/d/e/1FAIpQLSfym0hBhvQhWUyVDOtzhIU1o5WH97fRgW5ALBrdoMnroVSVw/viewform

The deadline for submission is: 03/08/2022

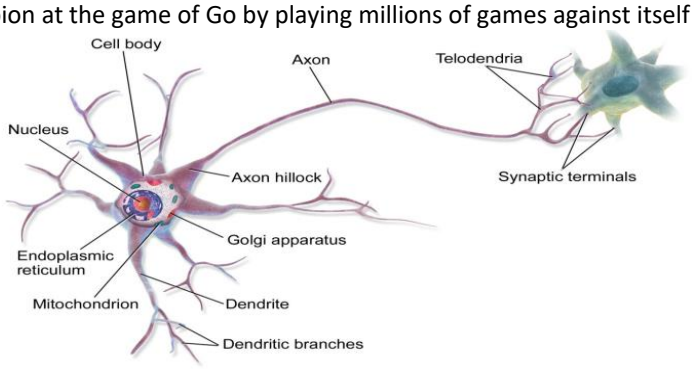
If you have any inquiries about the question below, send them to alsawahkareem@gmail.com

Questions

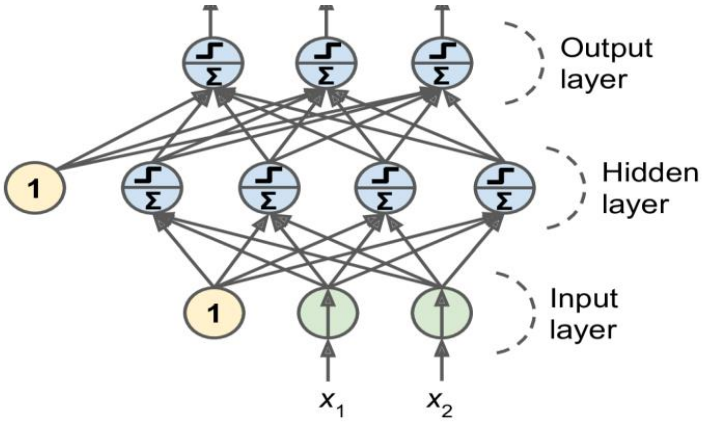
Perception

1- What is a neural network? What are its types and applications?
ANS:

Look at the brain's architecture for inspiration on how to build an intelligent machine. This is the logic that sparked artificial neural networks (ANNs):an ANN is a Machine Learning model inspired by the networks of biological neurons found in our brains. ANNs are at the very core of Deep Learning. They are versatile, powerful, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks, such as classifying billions of images (e.g., Google Images), powering speech recognition services (e.g., Apple's Siri), recommending the best videos to watch to hundreds of millions of users every day (e.g., YouTube), or learning to beat the world champion at the game of Go by playing millions of games against itself (DeepMind's AlphaZero).



ASU Racing Team
Formula AI and Shell AI
AI and Robotics Workshop



A neural network is a **software solution that leverages machine learning (ML) algorithms to 'mimic' the operations of a human brain**. Neural networks process data more efficiently and feature improved pattern recognition and problem-solving capabilities when compared to traditional computers.

Types: 1- **feed forward neural network**, often called multilayer perceptron (MLP)
2-**A Recurrent Neural Network (RNN)**
3-**Deep Feed Forward Neural Networks (DFF)**
4- **Convolution Neural Network(CNN)**
5- **Recurrent Neural Networks (RNN)**
6- **General Regression Neural Network (GRNN)**

Applications: 1-**Computer Vision.**
2-**Nature Language Processing.**
3-**Pattern Recognition.** **(And more)**

Source: Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.

2- Write Python code that compiles and trains a deep neural network on the Fashion MNIST dataset. Assume and tune your parameters and hyper-parameters.
ANS: you can find all the details from here.

3- Write Python code that compiles and trains a deep neural network for the ImageNet dataset. Assume and tune your parameters and hyper-parameters.
ANS: you can find all the details from here.

4- Read the LaserNet paper used for 3D-object detection. What sensors does it require? What is the main working principle of this technique? Do you see any way to improve it?

- Object detection is a critical task in the automation industry.
- Industrial controls engineers and software developers need to know when an object or target has arrived at a particular location.
- The seven most common types of object sensing technologies include electro-mechanical, pneumatic, capacitive and photoelectric.

Types:

1. Electro-Mechanical, 2. Pneumatic, 3. Magnetic, 4. Inductive
5. Capacitive, 6. Photoelectric, 7. Ultrasonic

There are two types of 3D object detection methods: region proposal based and single shot methods. The region proposal bases methods work by proposing several possible regions containing objects, and then extract per region features. These features are then used to attempt object detection. These methods could be divided into:

1. Multi-view based: Used multiple data sources like LiDAR front view, Bird's Eye View (BEV), and camera images.
2. Segmentation-based: uses semantic segmentation techniques to remove most background points, and then generate an amount of high-quality proposals on foreground points.
3. frustum-based methods uses existing 2D object detectors to generate 2D candidate regions of objects and then extract a 3D frustum proposal for each 2D candidate region.

Single shot methods directly predict class probabilities and regress 3D bounding boxes of objects using a single-stage network. They do not require region proposal generation and post-processing. They generally run at higher speed than Region proposals methods. These methods could be divided into:

1. BEV-based Methods: These methods mainly take bird-eye view representation as their input. The point cloud is used to create a 2D image representation as if taken from a bird's view. Normal 2D convolutional layers are then used to attempt object detection.
2. Discretization-based Methods: These methods transform a point cloud into a standard discrete representation, then use CNN to predict object categories and 3D boxes. Basically, the point cloud is transformed into a set of features that are then fed into a CNN network to attempt object detection.
3. Point-based Method: These methods take raw point clouds as inputs directly.

5- What is the diference between stereo-camera and mono-camera? State mathematically how coordinates can be transformed from the 2D pixel domain of the two images of the stereo-camera to 3D world coordinates.

ANS: **Stereo images provide a different image to each eye**, whereas mono images show the same image to both eyes. In real life, obviously we see a slightly different perspective from each eye.

The pose of the object is then estimated in each of the sequential images, which provides a Rotation matrix and Translation vector.
Using: $X_{cam} = X_{obj}R + T$, I hoped in obtaining the object coordinates, knowing the camera image coordinate as well as the pose (This did however not work because the Rotation matrix and Translation vector is from the camera perspective).
Someone can think:
I know that the Rotation from the camera perspective will be the same as from the object perspective and thus for the rotation portion, I can make use of the output provided.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

1) I can use Gaussian Elimination for find X,Y,Z,W and then points will be X/W , Y/W , Z/W as homogeneous system.

2) I can use the OpenCV documentation approach:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$
$$y' = y/z$$
$$u = f_x * x' + c_x$$
$$v = f_y * y' + c_y$$

as I know u, v, R , t , I can compute X,Y,Z.

However, both methods end up in different results that are not correct.

What can we do now?

If you got extrinsic parameters, then you got everything. That means that you can have Holography from the extrinsic (also called Camera Pose). Pose is a 3x4 matrix, holography is a 3x3 matrix, **H** defined as

```
H = K*[r1, r2, t],          // eqn 8.1, Hartley and Zisserman
```

with **K** being the camera intrinsic matrix, **r1** and **r2** being the first two columns of the rotation matrix, **R**; **t** is the translation vector.

Then normalize dividing everything by **t3**.

What happens to column **r3**, don't we use it? No, because it is redundant as it is the cross-product of the 2 first columns of pose.

Now that you have holography, project the points. Your 2d points are x, y. Add them a z=1, so they are now 3d. Project them as follows:

```
p        = [x y 1];
projection = H * p;          //project
projnorm   = projection / p(z);    //normalize
```

However,

projecting 2D image coordinates into 3D "camera space" inherently requires making up the z coordinates, as this information is totally lost in the image. One solution is to assign a dummy value (z = 1) to each of the 2D image space points before projection as answered by Jav_Rock.

```
p        = [x y 1];
projection = H * p;              //project
projnorm   = projection / p(z);    //normalize
```

One interesting alternative to this dummy solution is to train a model to predict the depth of each point prior to reprojection into 3D camera-space. I tried this method and had a high degree of success using a Pytorch CNN trained on 3D bounding boxes from the KITTI dataset.

State Estimation

6- What are the different approaches to odometry sources and localization in self-driving cars?

Ans: There are many different techniques to help an autonomous vehicle locate itself.

- **Odometry** —This first technique, odometry, uses **a starting position** and **a wheel displacement calculation** to estimate a position at a time t. This technique is generally very inaccurate and leads to an accumulation of errors due to measurement inaccuracies, wheel slip, ...

- **Kalman filter**—The previous article evoked this technique to estimate the state of the vehicles around us. We can also implement this to **define the state of our own vehicle**.

- **Particle Filter—**The Bayesian filters can also have a variant called particle filters. This technique compares the observations of our sensors with the environmental map. **We then create particles around areas where the observations are similar to the map.**

- **SLAM** —A very popular technique if we also want to estimate the map exists. It is called SLAM (Simultaneous Localization and Mapping). In this technique, we estimate **our position** but **also the position of landmarks**. A traffic light can be a landmark

- **Inertial Measurement Unit (IMU)** is a sensor capable of defining the **movement of the vehicle** along the yaw, pitch, roll axis.

This sensor calculates **acceleration** along the X, Y, Z axes, **orientation**, **inclination**, and **altitude**.

- **Global Positioning System (GPS)** or NAVSTAR are the US system for positioning. In Europe, we talk about Galileo; in Russia, GLONASS. The term Global Navigation Satellite System (GNSS) is a very common satellite positioning system today that can use many of these subsystems to increase accuracy.

7- What is known by SLAM? What are its different types?

ANS: SLAM stands for simultaneous localization and mapping (sometimes called synchronized localization and mapping). It is the process of mapping an area whilst keeping track of the location of the device within that area. This is what makes mobile mapping possible. This allows map construction of large areas in much shorter spaces of time as areas can be measured using mobile robots, drones or vehicles. SLAM systems simplify data collection and can be used in outdoor or indoor environments.

SLAM —A very popular technique if we also want to estimate the map exists. It is called SLAM (Simultaneous Localization and Mapping). In this technique, we estimate **our position** but **also the position of landmarks**. A traffic light can be a landmark

Types:

- Graph SLAM
- Occupancy Grid SLAM
- DP-SLAM
- Parallel Tracking and Mapping (PTAM)
- LSD-SLAM (available as open-source)
- S-PTAM (available as open-source)
- ORB-SLAM (available as open-source)
- ORB-SLAM2 (available as open-source)

We have also:

1-Visual SLAM, also known as vSLAM, calculates the position and orientation of a device with respect to its surroundings while mapping the environment at the same time, using only visual inputs from a camera.
Feature-based visual SLAM typically tracks points of interest through successive camera frames to triangulate the 3D position of the camera, this information is then used to build a 3D map.

2-A LiDAR-based SLAM system uses a laser sensor to generate a 3D map of its environment. LiDAR (Light Detection and Ranging) measures the distance to an object (for example, a wall or chair leg) by illuminating the object using an active laser “pulse”.
LiDAR is both a fast and accurate approach and can be used in a wide range of environments and conditions. The laser sensor point cloud generated from this method is highly accurate and is ideal for mapping in construction. These high precision distance measurements can be used for a whole host of other applications too.

8- Describe briefly the ICP algorithm used for localization using LIDAR point clouds.
Write a pseudo-code for the algorithm.

Ans: you can find a file called "Evaluation_of_the_ICP_Algorithm_in_3D_Point_Cloud" on my GitHub repository. And there you can find the code.

9- What are the types or variants of the Kalman filter? What are the uses and applications?

Ans: A wide variety of Kalman filters have now been developed, from Kalman's original formulation, now called the **"simple" Kalman filter, the Kalman–Bucy filter, Schmidt's "extended" filter, the information filter, and a variety of "square-root" filters**

Extended Kalman filter algorithm for state estimation

Linear discrete state-space model with additive noise

1: $x_{k+1} = A_k x_k + B_k u_k + w_k$ ▷ State equation
2: $y_k = C_k x_k + D_k u_k + v_k$ ▷ Output equation

3: Initialization : For $k = 0$, define
4: $\hat{x}_0^+ = \mathbb{E}[x_0]$ ▷ Initialize mean
5: $P_{\tilde{x},0}^+ = [(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$ ▷ Initialize error covariance

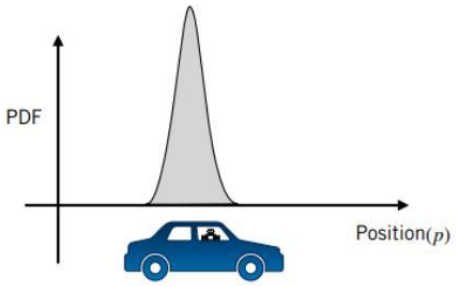
6: **for** $k = 1, 2, ...N$ **do** ▷ Loop till number of samples N
7: $\hat{x}_k^- = A_{k-1}\hat{x}_{k-1}^+ + B_{k-1}u_{k-1}$ ▷ State estimate time update
8: $P_{\tilde{x},k}^- = A_{k-1}P_{\tilde{x},k-1}^+ A_{k-1}^T + P_w$ ▷ Error covariance time update

9: $L_k = P_{\tilde{x},k}^- C_k^T [C_k P_{\tilde{x},k}^- C_k^T + P_v]^{-1}$ ▷ Kalman gain
10: $\tilde{y}_k = y_k - [C_k \hat{x}_k^- + D_k u_k]$ ▷ Innovation

11: $\hat{x}_k^+ = \hat{x}_k^- + L_k \tilde{y}_k$ ▷ State estimate measurement update
12: $P_{\tilde{x},k}^+ = (I - L_k C_k) P_{\tilde{x},k}^-$ ▷ Error covariance measurement update
13: **end for**

And you can find every thing about it's application in a file called "Kalman Filter Applications" on my GitHub repository.

10- Write C++ code that implements the linear Kalman filter for one predict and one update step for the following problem. (Hint: You may use Eigen library)



$$\mathbf{x} = \begin{bmatrix} p \\ \frac{dp}{dt} = \dot{p} \end{bmatrix} \qquad \mathbf{u} = a = \frac{d^2p}{dt^2}$$

Motion/Process Model

$$\mathbf{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

Position Observation

$$y_k = [1 \quad 0]\, \mathbf{x}_k + v_k$$

Noise Densities

$$v_k \sim \mathcal{N}(0,\, 0.05) \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, (0.1)\mathbf{1}_{2\times 2})$$

ANS: you can find everything about it here

Path Planning

11- What are the different path planning algorithms that are used in autonomous mobile robots?

Ans: The Path Planning approaches in mobile robot can be classified into traditional or conventional method and Soft Computing method. The traditional method does not enforce intelligence into the path planning and it includes Graph Searching Techniques, Artificial Potential Field, cell decomposition method, Vector Field method and Road Map method. The soft computing methods introduce intelligence into the path planning and that includes **Genetic Algorithm, Ant colony Optimization, Swarm algorithm, neural networks, and Fuzzy logic**.

12- Write Python code that implements BFS (Breadth First Search) or A* Algorithm - whichever you find yourself comfortable with - algorithm on a 5 x 5 occupancy grid.

Obstacles on (2,0), (2,1), (2,2)

Start on (1,1) - Goal on (4,1)

Assume the grid is a 2D NumPy array, obstacle cells have a value of 1, and free cells have a value of 0.

ANS: you can find everything about it here

Navigation Control

13- What is the diference between kinematic and dynamic modeling of vehicles?

ANS: Kinematics will give you the values of change of objects, while dynamics will provide the reasoning behind the change in the objects. Kinematics and dynamics are two branches of Classical Mechanics that deals with the motion of particles.

From this Introduction we can say that the Kinematic model studies the motion of a robot mechanism regardless of force and torque that cause it. It allows to compute the position and orientation of the robot manipulator's end-effort relative to the base of the manipulator as a function in the joint variable. And we can say that the Dynamic model studies the relation between the applied forces or torques and the resulting motion of an industrial manipulator. The dynamic model of a robot studies the relation between the actuator joint torques and the resulting motion. For more information and details you can find a beautiful compartion between them in a file called "Kinimatic and dynamic modeling of vehciles.pdf" on my GitHub repository.

14- State different longitudinal and lateral vehicle controllers.

Ans: 1-PI Controller Model. 2- MPC Design. 3- Explicit SS MPC and Implicit MS MPC. 4- Explicit SS MPC and Implicit MS MPC. 5- Combined PI and MPC Controller.

For more information and details you can find them in a file called "Vehicle Lateral and Longitudinal control" in my GitHub repository.

15- Write C++ code that implements the forward kinematics of a kinematic bicycle model. Inputs = [v, δ] (velocity of the vehicle and steering angle) Outputs = [x, y, θ] (The 2D pose of the bicycle, x, y, and yaw)
Assume your step-time for integration and any other missing parameters

ANS: you can find everything about it here

16- Why is modeling an important part of control (Ex: what could the bicycle model be used for in control)? What if there is no model, can you still control a system?

ANS: **Modeling** is a central part of all the activities that lead up to the deployment of good software. We build models to communicate the desired structure and behavior of our system. We build models to visualize and **control** the system's architecture.

Modling helps in:

1-improves process communication. 2-increases control and consistency.

3-aligns operations with business strategy. 4-improves operational efficiencies.

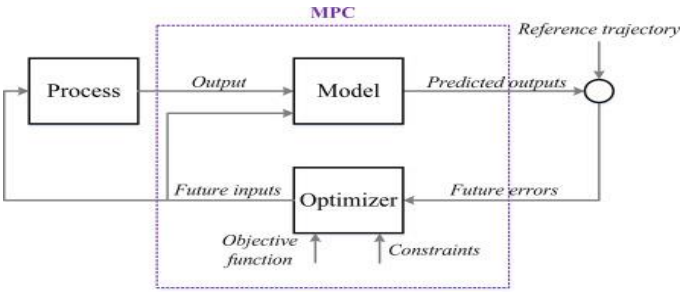
5-helps businesses gain competitive advantage. 6-helps to elucidate difficult processes.

7-facilitates automation. 8-helps preserve knowledge and corporate memory.

9-saves time and money. 10-helps in managing complexity.

NO, the system can't be controlled 100% without some modelling for it.

17- What is the MPC (model predictive control), how does it work? What are the most common issues with it?

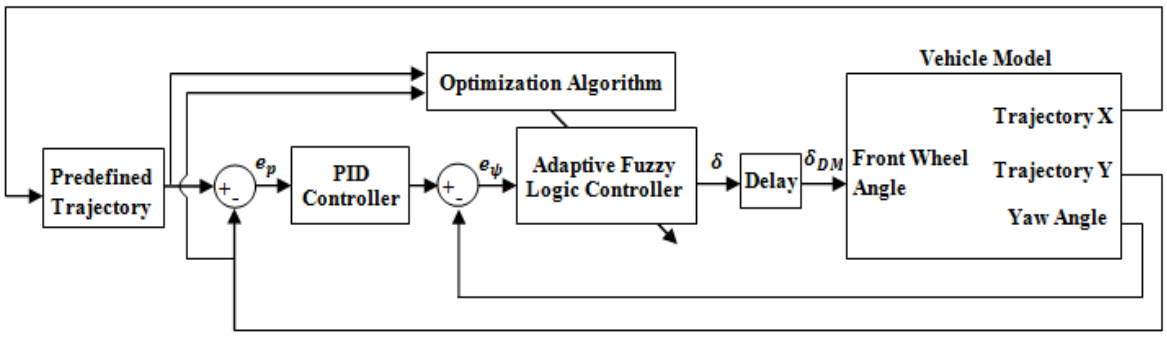


Model predictive control (MPC) is a multivariable control algorithm that makes use of process models and measurements to predict future control actions in the input. As the core of MPC, the process model which could be linear or nonlinear should be able to capture process dynamics of all inputs and outputs, thus allowing reliable prediction calculations. Therefore, the development of process model with sufficient accuracy is the most significant part when it comes to designing MPC. The goal of MPC is to minimize the calculations of the objective function, which is the sum of all the terms that have control requirements and

are therefore weighted according to their significance in the process, over a certain prediction horizon. The control law is a mathematical formula used by the controller based on the computations of the objective function and the process model. For more information and details you can find them in a file called “model predictive control” on my GitHub repository.

Low-Level Control

18- Perform closed loop control on a steering wheel in a self-driving car. Define your controller, the final control element, and the process variable. (Draw the block diagram).



19- What are the hardware and actuators needed for low-level control in self-driving cars?