***SD Plan***

Functional Requirements:

* Ability to accept an RGB image as input and process it in real-time
* Ability to detect and classify cones in the image as either blue (left boundary), yellow (right boundary), or orange (start line)
* Ability to output the bounding boxes and class labels for each detected cone in the image
* Ability to handle different scales and orientations of cones in the image
* Ability to work in a variety of lighting conditions and image quality levels

Non-Functional Requirements:

* Accuracy: The object detector must achieve a high level of accuracy in detecting cones in the image, with a low false positive rate.
* Speed: The object detector must operate in real-time, with a processing time of less than 100 milliseconds per image.
* Robustness: The object detector must be robust to variations in lighting conditions, image quality, and cone orientation.
* Generalizability: The object detector must be able to generalize to new images and perform well on unseen data.
* Compatibility: The object detector must be compatible with the other components of the autonomous racing vehicle's perception system.
* Usability: The object detector must be easy to use and integrate into the autonomous racing vehicle's perception system.

Aims:

1. To develop an object detection model that accurately detects cones in RGB images.
2. To evaluate the accuracy and speed of the object detector

Objectives:

1. To gather and preprocess a publicly available dataset containing cones of the Formula Student design (FSOCO)
2. To train a state-of-the-art object detection algorithm on the FSOCO dataset
3. To fine-tune the model to achieve high accuracy in detecting cones in RGB images.
4. To evaluate the model's performance on a validation dataset and compare it to the performance of other existing object detection models.
5. To implement the trained model on the autonomous racing vehicle's perception system and test it in real-world scenarios.
6. To ensure that the object detector is fast enough to perform real-time object detection during dynamic events in the FS-AI competition.

**STEPS to take during the project:**

Collect the dataset: Acquire the FSOCO dataset or any other publicly available dataset that contains cones of the Formula Student design in RGB format. Ensure the dataset has a sufficient number of examples of each type of cone (blue, yellow, and orange) to train a robust model.

Annotate the dataset: Annotate the images in the dataset with bounding boxes and class labels for the cones. This will be the ground truth for training and evaluating the object detection model.

Split the dataset into training, validation, and testing sets: Split the annotated dataset into three parts, with approximately 70% for training, 15% for validation, and 15% for testing. The validation set will be used to evaluate the model during training and adjust the model parameters, while the testing set will be used to evaluate the final model performance.

Choose the object detection model: Choose an appropriate object detection model, such as YOLO or Faster R-CNN, for the task. Evaluate the selected model on a few examples from the dataset to verify its suitability for the task.

Train the model: Train the object detection model on the annotated images using a suitable loss function and optimizer. Monitor the accuracy and loss on the validation set to determine when to stop training and adjust the model parameters, if necessary.

Evaluate the model: Evaluate the model on the testing set to measure its accuracy in detecting cones in the image and its speed of classification.

**Modules we will need in python (in CMD type “pip install <MODULE NAME>”):**

Numpy: A library for numerical computing with support for arrays and matrices.

OpenCV: A computer vision library that provides functions for image processing and computer vision tasks, including image loading and resizing, object detection and recognition, and more.

TensorFlow or PyTorch: A deep learning framework for building and training neural networks, including object detection models.

Matplotlib: A library for plotting and visualization that can be used for visualizing the output of the object detector.

Scikit-learn: A machine learning library that provides tools for data preprocessing, such as splitting the dataset into training, validation, and testing sets.

ImageAI: A library that provides a high-level API for training and deploying object detection models, including pre-trained models.

os: A standard library for file and directory manipulation.

argparse: A library for parsing command-line arguments in Python scripts.