# Report - Perception Task

## Approach used for each task

**Level 1**
When approaching the level 1 of the task I initially didn't know the basics of Computer Vision, therefore I started by researching the openCV library and watched some tutorials where I learned the key fundamentals such as adding filters, draw shapes, warp images, color and shape detection.
Having acquired the foundation I uploaded the two provided frames to the project.

**Level 2 - Detecting Cones**
To detect the cones I created three different masks for each color (red, blue and yellow). To implement each mask I needed to find the correct range values for the Hue, the Saturation and the Value (HSV). I implemented a dynamic trackbar to help determine these appropriate ranges interactively.
Starting from the red cones:

- I found the correct range of values using the trackbar
- I created the red mask using the first frame converted to its HSV version
- Applied filtering to the red mask such as MORPH_OPEN, MORPH_CLOSE and MORPH_DILATE
- Then to strengthen the detection, I also applied shape detection via contours
- Then I've extracted the outlines of the red regions and filtered out noise
- Finally, I drew a box around the detected red cone

The same technique was also applied to the detection of blue and yellow cones, though some adjustments were needed.

Since some of the blue cones are farther away, their detection becomes more challenging. I restricted the area using a rectangle and created a mask based on this area, which made it easier to avoid noise and to decide an appropriate kernel.
For the yellow cones, I've used a similar approach but I applied two different masks on two distinct regions.The masks use the same color range but I implemented two different kernels for filtering. The cones on the right (closer to the camera) used a larger kernel compared to the ones farther away.

**Level 3 - Classify the cones**
During the detection I drew a colored bounding box around the detected cones. For the red cones, I also applied a label on top of the two boxes. To make the output more readable, I added a legend on the top-left corner displaying all three colors and their corresponding cone classes.

**Level 4 - Draw track edges**
While detecting cones, I saved the center of each box (approximately the cone center) into a vector of Points. These points were then used to draw the edges of the track.

For the red cones, I drew a straight line marking the starting line of the track; for the yellow cones, I used the same approach after sorting the centers from left to right with respect to their x-coordinates.
During the visualization of the blue edge, I encountered some complications due to the geometry of the blue cones. They create a narrower curve which makes the sorting applied for the yellow cones insufficient. I found a solution in dividing the centers found in 2 parts using their x-coordinates relative to the mean. For the first group, representing the left cones, I applied a sorting on their x-coordinates while, for the second group, I applied a sorting on their y-axis. Using this approach, I was able to draw the lines separately and then connect them.

**Level 5 - Odometry**
I started with doing some research on the general concept of odometry and the approach I needed to use:

- Find the key features using ORB (Oriented Fast and Rotated Brief)
- match them with their descriptors using the BRUTE FORCE matcher
- find the Essential Matrix which contains information about the transitional vector and the rotational matrix, giving out a hint on how the camera moved between frames

Before applying this technique, I also created a mask to exclude the car. To approximate the car area more accurately, I used three rectangular regions, two for the tires and one for the main body.

**Level Bonus - GUI**
Finally, I implemented a simple GUI to show all the results. Depending on the key pressed, a different visualization is displayed. The only constraint is that the cone detection must be run before showing the track edges, since the cone centers are computed during detection.