PERCEPTION

Instructions

- Parts of the module highlighted in green are “checkpoints” and trainees are required to update the following task sheet once every checkpoint is completed. https://docs.google.com/spreadsheets/d/1jKye-ZOJ6FYTVYJORTY_L2CJqsRVhbJ60ml7lrBhXc8/edit?usp=sharing
- Documentation is a must for every checkpoint and trainees are required to create a google doc sheet where they document their learnings, errors encountered and doubts. This google doc must be made accessible to “anyone with link” and the link for the same must be updated on the task sheet.
  ~~Feel free~~ you are **encouraged** to ask doubts to JDEs if you feel stuck or want to understand a topic better
- Keep in mind, performance in modules is how the team will judge your abilities and effort to assign subsystems once the time comes
- Have a fun learning experience!

Module flow:

Initially, a lot of theory will be covered using resources mentioned. Do stick through them with diligence. This is followed by our perception pipeline and the major assignment. Checkpoints pertain to the stuff written before that particular checkpoint.

Welcome to the first step in a driverless car- Perception!
Ideally, all of you should be familiar with what exactly perception means and what its goals are. If you seem to have forgotten-

Perception means to perceive the environment around us. A driverless car does that using its wide array of sensors that include:

- Cameras
- LiDAR
- RADAR
- SONAR

IITB Racing DV, as of now has the entire perception pipeline based on Cameras and LiDAR. Primarily, cameras are better off suited in

detecting and classifying objects while a LiDAR is predominantly used to estimate depth (distance of the object from the car).

Our team focuses on participating in Formula Student UK/Germany/Netherlands etc and these competitions have their own specific problem statements.
In general, our race car is required to navigate through a racetrack defined by cones on either side (yellow cones on the left and blue cones on the right). Since it is a race, speed is of utmost importance and thus latency(delay in computation) reduction is a key factor.

Keeping the problem statement in mind, the entire job of the subsystem boils down to:

- **Detecting** cones (identifying which of the objects in an image are cones and locating where exactly those particular cones are located within an image)
- **Classifying** cones (categorizing them into their respective color and size- blue/yellow/small orange/big orange/unknown)
- **Estimating depth** of the detected and classified cones (we know where the cones are within the image but we do not know where they are in the real world. Estimating the relative positions of the cones wrt our car is the final task of perception and this is done by finding the distance between the car and the cone(depth) and the angle at which the cone lies wrt our cars' heading)

Now that we have understood the crux of what is *required,* let's move on to understanding *how* we achieve what's required. This is where "Machine learning" "Deep learning" "Computer vision" and other jargon steps in.

Fundamental theory

1. Python (fundamentals):
   Our entire codebase is currently on python (and we aim to switch to c++ a few years down the line since python is slower than c++) and this requires a firm grasp on python, its functionalities, data structures and working in general. Those confident in the same (since it has already been done in the software module) can skip learning python fundamentals and directly move on to the next task.
   Resources:

- https://www.youtube.com/watch?v=_uQrJ0TkZlc watch till 3:50:47
- https://www.w3schools.com/python/numpy/default.asp : Numpy
- https://www.w3schools.com/python/pandas/default.asp : Pandas
- https://www.w3schools.com/python/matplotlib_intro.asp : Matplotlib
- You are free to use any other resource if you wish to learn python fundamentals

Checkpoint 1!

2. OpenCV and Image Processing:

A vital and powerful package in computer vision and used in almost every part of our code. This part of the module is highly important and must be done well.
Resources:
https://www.youtube.com/playlist?list=PLAwxTw4SYaPn-unAWtRMleY4peSe4OzIY
Video 67 to 78
Video 82 to 85
Video 100 to 110
Video 114 to 123

https://www.youtube.com/playlist?list=PL2zRqk16wsdqXEMpHrc4Qnb5rA1Cylrhx
Video 1 to 5
https://www.youtube.com/playlist?list=PLcQ8H2FtZMugV6cbWfdaIgbkGZjKgOYAx
Full playlist

Checkpoint 2!

3. Machine learning:

Steps in the building block of perception- ML.
As a “driverless” car, our computers are required to function like the human brain to identify and classify objects around the car (in our case- cones). What goes behind making a computer mimic a human brain is a truckload of math full of linear algebra

and calculus. It is essential that we understand what this entire process entails (atleast at a surface level, without diving too deep into the math).

Resources: (Go through all of these in the order specified)

- Overview of ML: https://www.geeksforgeeks.org/introduction-machine-learning/
- https://youtube.com/playlist?list=PLEiEAq2VkUUI73199L-Aym2MnKjBxJ-4X
  videos: 1,9,11,12

The above will give you a basic idea of what machine learning is all about. Following provides a deep dive into the algorithms:

Checkpoint 3!

4. Deep learning:

   A subset of machine learning especially used by us for object recognition and classification. Again a highly important part of the module and trainees are required to understand concepts explained in the following resources thoroughly to develop a strong base for work to be in in the perception subsystem.

   Resources:
   https://www.youtube.com/playlist?list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU except videos 26 to 29

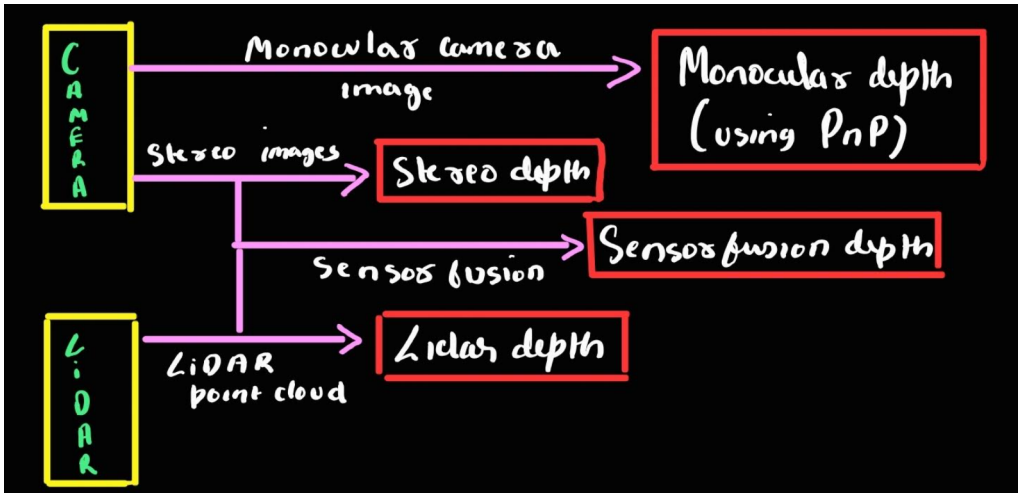
   Checkpoint 4!

Our Pipelines

Now that you are well equipped with the mathematical, software and theoretical fundamentals, we get back to addressing the question of *how* our team achieves the task of **detecting, classifying and estimating depth** of cones.
Our pipeline (a flow of how things work) is divided into 4 major channels

1. LiDAR only
2. Monocular camera

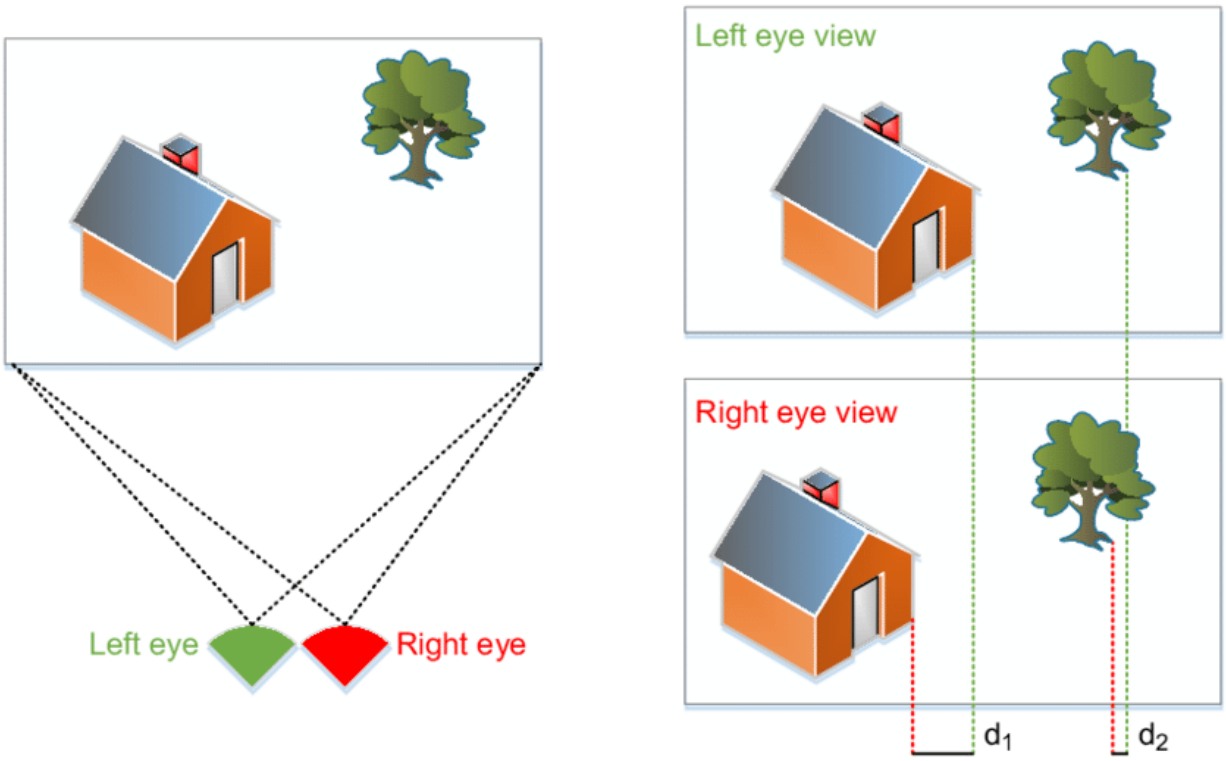
3. Stereo camera
4. Lidar - Camera sensor fusion



Sensor fusion will be covered in other subsystem modules, LiDAR will be taught once you become a perception JDE :) and in depth analysis of the camera pipelines are as given below:

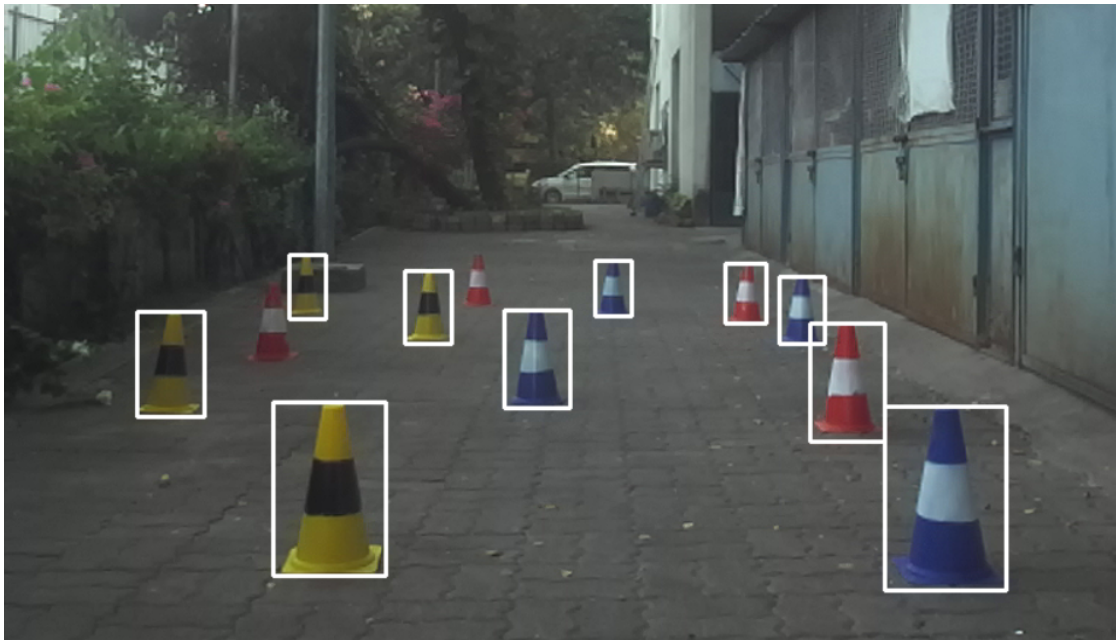
Stereo camera pipeline

A stereo camera is used to take two images of the same scene. These images are called left and right images. The right image and left image, although very similar, are not exactly the same and points in the right image have a slight leftward shift (called **disparity**) compared to similar points in the left image. This disparity is exactly what we use to find depth of any cone.

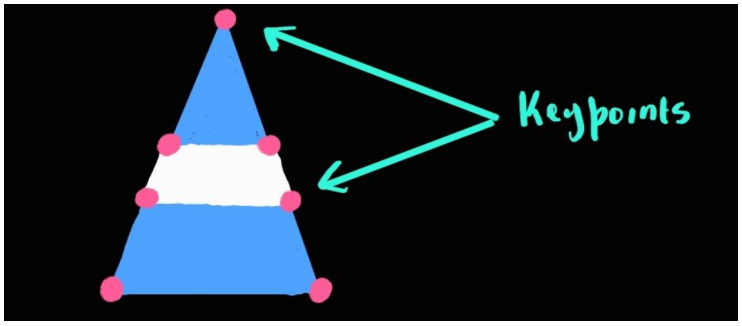


In the above image, d1 is the disparity in the point at the bottom corner of the house and d2 is the disparity in the bottom corner point of the tree.
Depth of any object is inversely proportional to its disparity and the exact relation is given by the formula:
Depth = focal length * baseline / disparity
Where baseline is the distance between the left camera lens and right camera lens.
The entire stereo pipeline is designed to find the disparity of every cone and thus find depth of every cone. The following is the process to do just that:

1. Capture both left and right images
2. On the left image run **YOLOv5** (a state of the art object detection and classification model) to detect and draw bounding boxes around cones seen in the left image. Bounding boxes contain 3 types of information: class of the cone, coordinates of the center and size of the bounding box and confidence.
   Below is a snapshot of us running inference on an image captured in front of the IITB racing lab



3. Now that we have bounding boxes around all the cones in the left image, we run **RektNet** which is a custom model derived from **ResNet** to detect 7 **keypoints** on the cone



These keypoints are the points for which we find disparity later on.

4. These keypoints have been found for all the cones on the left image but need to be found for the right image too. This can be done by running YOLO and RektNet on the right image too, but deep learning modules are usually computationally very heavy and take up a lot of time. Thus, another approach is to "propagate" these keypoints from the left image to the right image. The process for this is left as an exercise which is given formally later in the module.

   After we propagate the keypoints from left image to right image, we now have Bounding boxes (BB) and keypoints on the left image and keypoints on the right image. The problem with these

propagated keypoints is that they are not very accurate on the right image. To resolve this, we find the bounding box of the cones on the right image using the keypoints on the right image. Now we have bounding boxes for both images.

5. To find accurate keypoints for the cones of both the images, we now use **SIFT** (Scale invariant feature transform), an algorithm that finds “defining” points in the image. SIFT gives us keypoints for cones in both the images and also **matches** corresponding keypoints in both the images. Snapshots from running our code to detect and match keypoints in both the left and right images are as shown below:

6. Once SIFT has detected and matched keypoints in both the images, all we are left with is to subtract the x coordinate of two corresponding keypoints and that gives us disparity!
7. Disparity is then used to find depth

Viola! We have found depth by finding disparity. These estimated depths are then compared to true depths measured during testing and relative errors are found. By analyzing our errors, we improve the code and this is an iterative process.

Checkpoint 5!

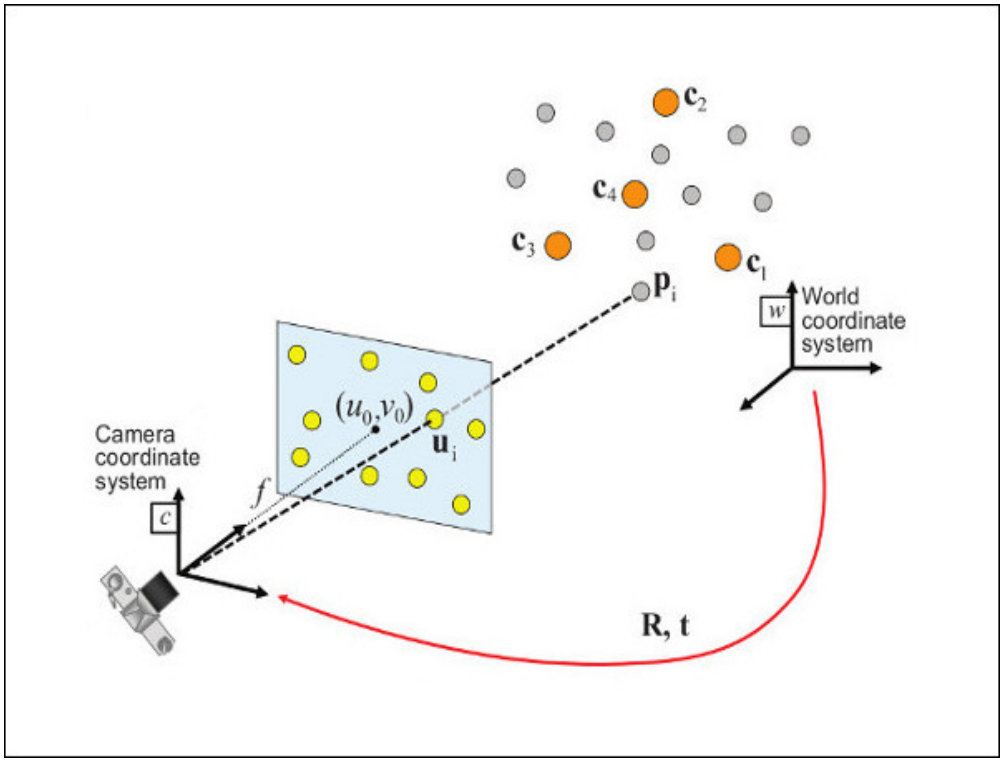
Assignment

Refer to the yellow highlighted process in step 4 of the stereo camera pipeline.
You are given two (one for left image and one for right image) 2D arrays which store values in the range 0 to 255 specifying the intensity of pixels. A keypoint with coordinates (Xo, Yo) in the left image is given to you. (Note: coordinates in an image and indice values in an array differ slightly. You have to find how they differ and what the relation is)
Find the coordinates of the same keypoint on the right image array by writing a pseudo code (just the logic)
The most rudimentary method to do this Patch regression.
You are necessarily required to use patch regression for the assignment and can also explain any other innovative/creative methods for keypoint propagation if you wish.

Checkpoint 6!

Monocular camera pipeline

We have learnt how to get depth from the stereo pipeline. Stereo uses two images whereas monocular uses a single (left) image to find depth and we do this using the PnP (Perspective n Point) algorithm (recall this was also present in the recruitment assignment)



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}\,\mathbf{\Pi}\,{}^{c}\mathbf{T}_{w} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

u and v are the coordinates of a point in the image frame. The matrix A denotes the camera intrinsic matrix(which needs to be calibrated before using the algorithm). Fx and Fy are the focal length of the camera in two perpendicular directions(for us, Fx = Fy). Cx and Cy are basically the coordinates of the center point of the image.
T matrix is what we call the extrinsic parameter matrix or simply known as the pose matrix.
Our aim in the PnP algorithm is to find this pose matrix so that the depth of that cone can be found from the camera.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = {}^{c}\mathbf{T}_{\mathbf{w}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Checkpoint 7!