Visual Object Detection:

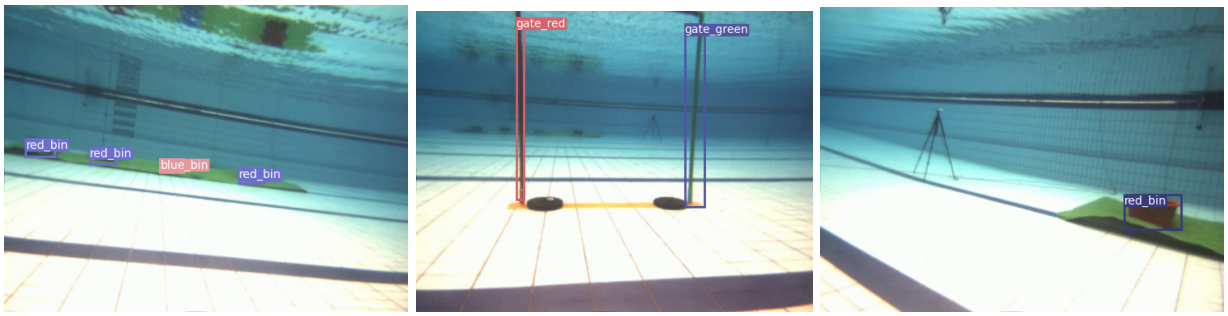
Computer Vision(CV)

1:**OpenCV**: We as a team have been using OpenCV to perform image processing techniques to extract information from camera frames and detect obstacles & tasks at hand in many of our previous competitions. We used bounding box vertices to locate tasks such as gates & bins. These detections also helped in localizing our bot as by applying basic math similarity concept we were able to estimate the position of our bot with respect to task (having the focal length of the camera at hand and the original length and width of task objects and it comparing with the ratio of pixel lengths of edges). The center we calculated from the contours/hough lines made around the task(gates) helped in aligning the bot and pass through the gate.

Neural Network

2:**YOLOv3/v5**:Although traditional image processing methods were successful in detecting objects underwater(with suitable adjustment in HSV values as per lighting) but it comes with certain drawbacks such as failing to detect from a greater distance and unreliable results in different lighting and background. Here we have found Yolov3(You Only Look Once) CNN (a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images.)to be effective. We realized if we can navigate to the object close enough using this neural network-based model, then we can apply suitable thresholding techniques or can use the same model to get the center and pass through the target(gate) or reach the task at hand. For training the CNN, images were collected from previous competitions. Data augmentation was also performed to introduce variation and increase the dataset. We used our own Labeler Toolbox to label the images. 1/5th of the images were reserved as the test set to run the inference and the model(based on open-source neural network framework Darknet) was trained on the rest of the data. Unfortunately, we could not test the model real-time underwater on Nvidia Jetson due to the pandemic, but we still achieved very encouraging results testing it on our local systems(25 fps 87%).

Together with new machine learning approaches, traditional image processing techniques, color correction algorithms, and other fine-tuned approaches, we hope to strengthen the perception system of Makara 4.0

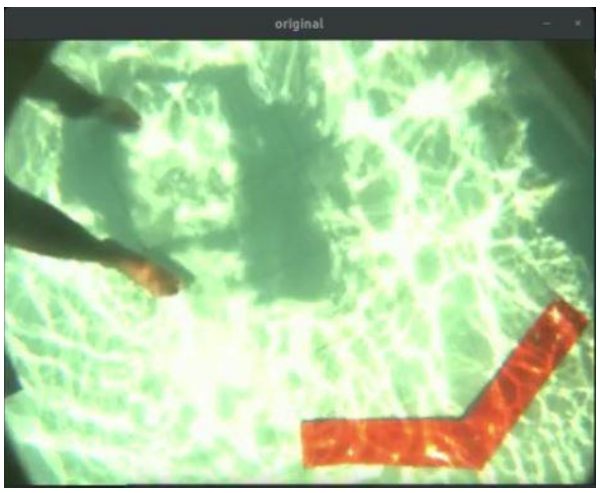


Better Perception Techniques:

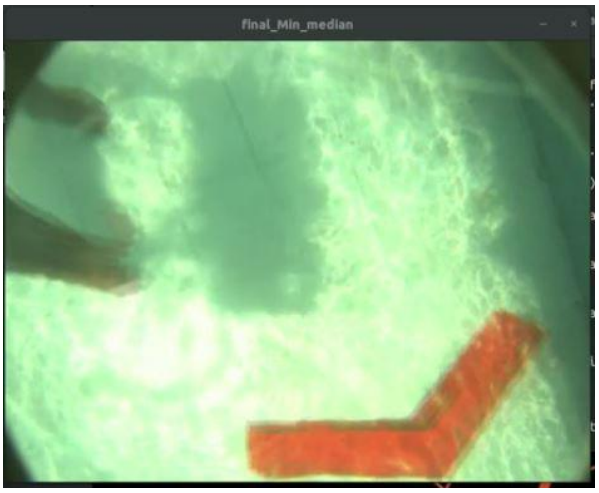
1. Glare Removal

Sometimes the images obtained from the bottom camera are too bright and keep fluctuating due to sunlight reflection(happens when the arena isn't that deep and has a shiny bottom surface). To remove glares from these images we have used an algorithm to make them more visible and clear.

The algorithm keeps track of the most recent 'n' images and finds the median, mean variation of 'n' RGB values respectively for each pixel in the image. The final image obtained has less glare and the lighting is spread equally. As the glares are rapidly moving due to surface water motion, taking its median helps to distribute its intensity and makes the image more clear and visible.



1.Original



2.Glare removal algorithm applied

2. Color Correction

The Robosub arena watercolor is greenish and unclear and thus making things difficult for the object detection model as they are trained on a different environment.

To change the color and make it more natural blue, we have simply changed the weight of the RGB channel of the image (called **Stretching**) and obviously, the weight of the green channel needs to be reduced. Then to remove the fogginess in the image we used the **Contrast Limited Adaptive Histogram Equalization (CLAHE)** algorithm that helped to increase the image contrast and made images more clear.

1.*Original* *2.Stretched* *3.Stretched+CLAHE*

Model Optimisation:

The perception stack is mostly carried out by object detection methods using deep neural networks. However, there has always been a trade-off between accuracy and real-time fps. We have approached this problem with model optimization using **quantization** by implementing state-of-the-art **quantization-aware training** methods. This approach simply converts all float32 weights and activations to int8 thus drastically reducing model size as well as the inference time. This enables us to use any low-power computing devices and stack multiple models in our pipeline without compromising performance.

Stereo Vision:

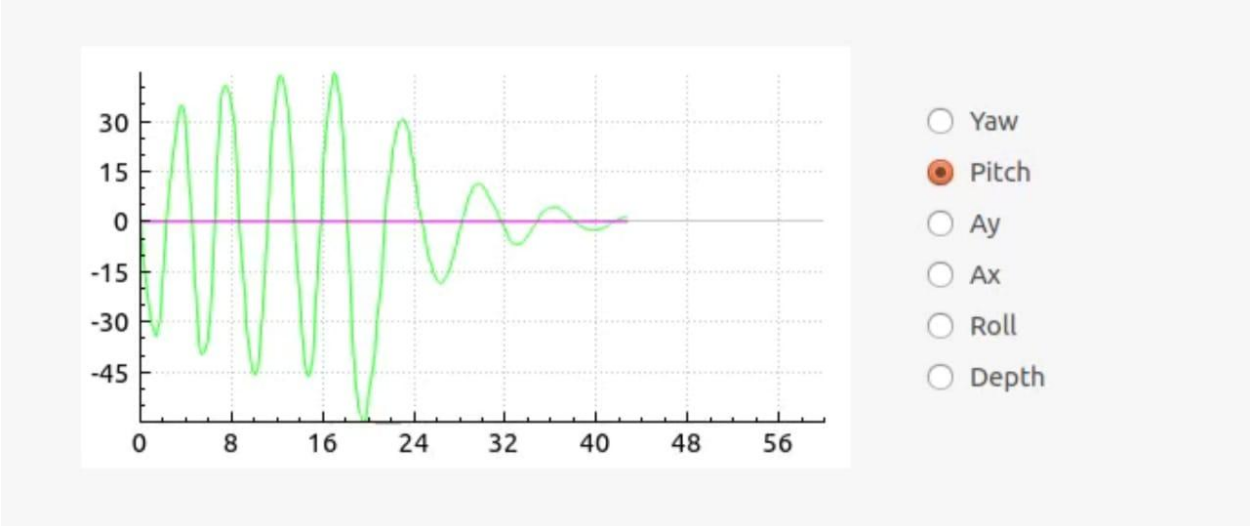
In traditional stereo vision, two cameras, displaced horizontally from one another are used to obtain two differing views on a scene, in a manner similar to human binocular vision. By comparing these two images, the relative depth information can be obtained in the form of a disparity map, which encodes the difference in horizontal coordinates of corresponding image points. Computer vision algorithms are also applied to images for smoothening and rectification of the images. Thus by using stereo vision we can perceive the distance between the object and the camera plane while carrying out various tasks during the mission.

PID Auto-tuning:

The PID controllers used in our AUV have to tune it (that means to obtain the value of Kp, Ki, Kd constants) manually so that it manoeuvres properly.

But this time we tried to automate things and decided to build a tuning algorithm using **Ant Colony Optimization (ACO)**. To tune the constants of each degree of freedom, the algorithm first generates some random values and tries to minimize the error generated from a cost function for each generated value and then determine the best values obtained from this generation and use them as the parents for the next generation. This happens generation over

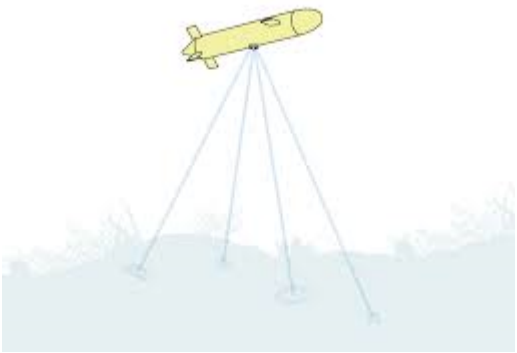
generation till we get an optimal value and every generation is more optimal than its previous one.



(*Results of PID auto tuning on one DOF - pitch*)

Working with DVL and why ?

A **Doppler Velocity Log** (DVL) is an acoustic sensor that estimates velocity relative to the sea bottom. This is achieved by sending a long pulse along with a minimum of three acoustic beams, each pointing in a different direction. Typically, this produces estimates of velocity converted into an XYZ coordinate frame of reference – the DVL's frame of reference. Together with a heading estimate, these velocity estimates may be integrated over the ping interval to estimate a step-by-step change of position – i.e. displacement = velocity × time step.



It is important to ensure that velocity estimates do not have any bias or offsets because this will lead to a growing error in the position estimate. This is where the Doppler Velocity Log becomes a key in the subsea navigation for its accurate estimate of velocity with zero-mean bias. This sensor combined with the **IMU** ensures a proper estimation of obstacles/key places that the vehicle needs to avoid/reach to perform some specific work.

A typical approach we use to generate pathway using DVL :