LOCATION, DETECTION AND NAVIGATION OF UNMANNED VEHICLES

***Abstract**: Various approaches for sensor orientation were studied to obtain the optimal configuration. An algorithm was developed using Machine Learning to find the location of the vehicle using distance measurements obtained from several sensors placed around it. A Deep Learning algorithm was also developed to detect the objects through a webcam.*

1. Sensor orientation for obtaining most accurate range values

Ultrasonic sensors were used to obtain the distance of the vehicle from the receivers. N (N=12) sensors were used for this purpose such that there were 3 sensors on each side. This gave us a more accurate location of where the vehicle is.

We tried various methods of sensor orientation to figure out the method that had maximum sensors in range at any given point. We decided to make half the sensors long range and the other half short range.

In the first approach, we placed short range sensor between two long range sensors on the all sides. When plotting such a configuration it was found that there were many areas in the plot where very few sensors were in range of the vehicle. This conclusion gave us more insight to obtain a better form of orientation.

In the second approach, we placed the long and short range sensors in alternate fashion. When the following configuration was plotted, we found a much better plot where 4 or more sensors were found to be in range thereby providing us with better range values for location calculation.

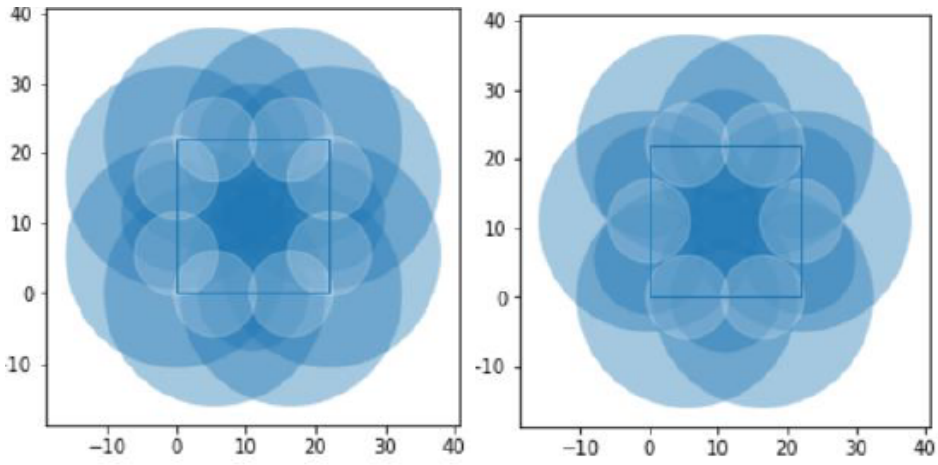


Figure 1: Sensor Orientation

2. Range Estimation from sensor range data

The range measurements obtained from the sensors are corrupted by noise such as multipath reflection, atmospheric condition etc. To estimate the range close to the true value certain estimation algorithms were implemented.

The N - sample mean of the range measurements were taken to minimize the uncertainty. This approach is inefficient in removing the outliers caused by external factors.

Due to this reason, a Machine Learning (ML) model called linear regression was used to improve the accuracy of the range measurements. Linear regression model forms a relationship between the obtained sensor data and the true value.
A dataset of range measurements for every 1.5mts was acquired and plotted vs true value, where the slope of the line is given by:

$$\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

It was found that the model helped us achieve better values at the extreme ranges provided by the ultrasonic sensor.

The results obtained through machine learning are tabulated below:

True Range measurement (cm)	**Measured Range (cm)**	**Range by Linear Regression (cm)**
150	170.57	144.88
300	318.12	298.75
450	466.36	450.57
600	618.5	608.00
750	744.76	739.53
900	894.85	894.24
1050	1047.44	1052.18
1200	1157.91	1177.27
1350	1334.54	1349.55
1500	1485.47	1505.94

3. Position Estimation from the obtained range measurements

There tend to be small deviations from the actual measurements in spite of passing the values through the Linear Regression model. The position of the vehicle was found by two approaches.

In the first approach, we used trilateration. Trilateration is the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles.

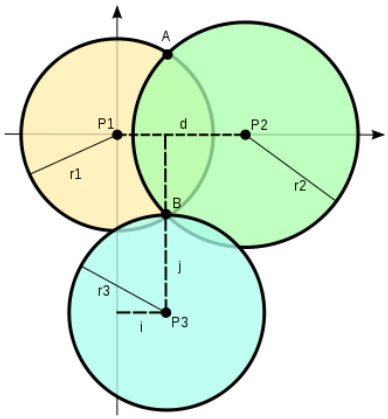


Figure 2: Intersection points from various groups of range measurements

In the second approach, another Machine Learning approach called k-means clustering was used for accurate position estimation. Here, every two range values from the receivers are taken and the two points of intersection are obtained as the two circles provides a maximum of two intersection points. From these two obtained points, we know that the unmanned vehicle is located closer to one of these points.

This can be done without any trigonometry at all. Let the equations of the circles be

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2, \tag{1}$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2. \tag{2}$$

By subtracting the two equations and expanding, we in fact obtain a *linear* equation for x and y; after a little rearranging it becomes

$$-2x(x_1 - x_2) - 2y(y_1 - y_2) = (r_1^2 - r_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2).$$

(If the circles intersect, this is the equation of the line that passes through the intersection points.) This equation can be solved for one of x or y; let's suppose $y_1 - y_2 \neq 0$ so that we can solve for y:

$$y = -\frac{x_1 - x_2}{y_1 - y_2}x + \dots. \tag{3}$$

Substituting this expression for y into (1) or (2) gives a quadratic equation in only x. Then the x-coordinates of the intersection points are the solutions to this; the y-coordinates can be obtained by plugging the x-coordinates into (3).

From these clusters obtained, the center of the largest cluster is taken to be the location of the unmanned vehicle.

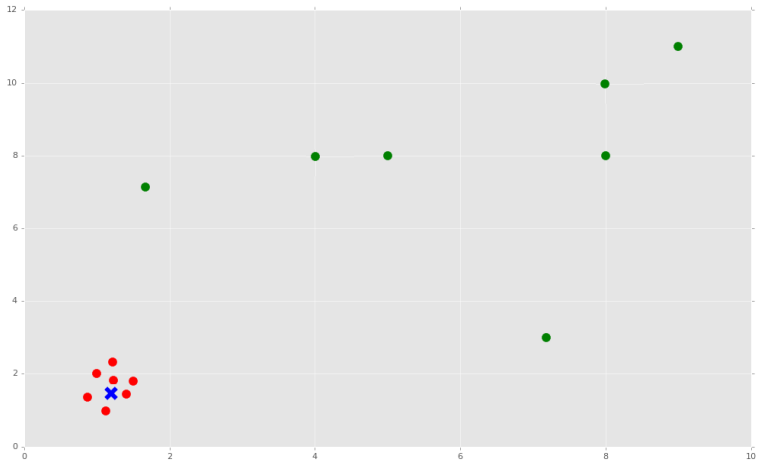


Figure 3: Visualization of k-means clustering

The circles have the ultrasonic receiver locations as their center and the range measurements as their radius. So, from the list of range measurements, various sets of points are obtained. These points are passed through the k-means clustering algorithm and the points obtained are plotted and divided into groups depending on the distance of the point from the center.

4. Object detection for vehicle navigation

Object detection was essential for vehicle navigation as the type of object in front of the unmanned vehicle tells how big a deviation needs to be taken. Thus, to detect the object, image processing using opencv and deep learning using tensorflow was used. A tensorflow object detection API was created to detect the presence of UMV's and other objects.

Methodology:

1. Collected a few images that contained pictures of the unmanned marine vehicle.
2. Annotate/label the images using this program. This process was used to draw bounding regions around the object(s) in the image. The program automatically creates an XML file that describes the object(s) in the pictures.
3. This data was then split into train/test samples.
4. TF Records were generated from these splits.
5. Used a mobilenet configuration file for developing the model
6. Training the model
7. Exported graph from new trained model
8. Used the You only Look once .config along with the created model for more accurate detection of objects.
9. Saved the model as a protobuf file for direct implementation and ran the model on custom objects in real time.

The following was the methodology used on a few collected boat images. We wanted the model to detect any type of common object. So, the pre-trained YOLO model was loaded for building.

Building the model:

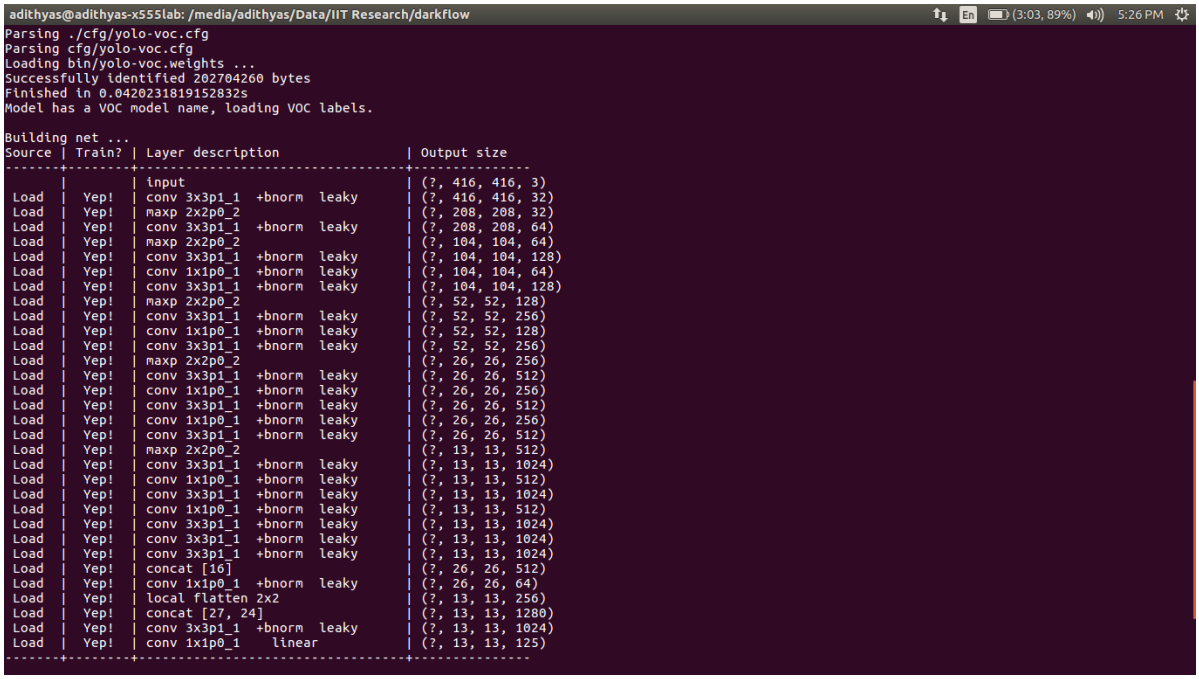


Figure 4: Building the model

The model is built with the yolo.cfg and yolo.weights files and then developing the pb file for storing the model.

The YOLO model used was pre - trained on various common objects such as person, vehicles, animals etc.

Running the model on Test images:

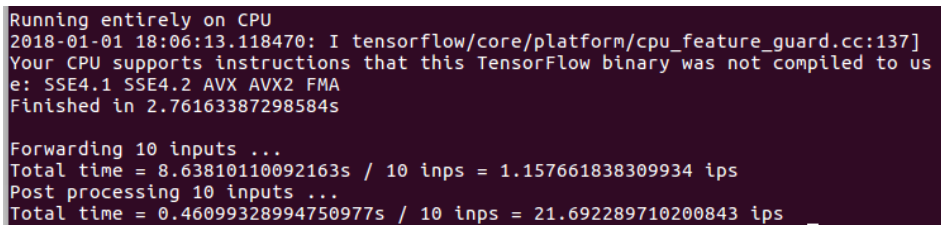
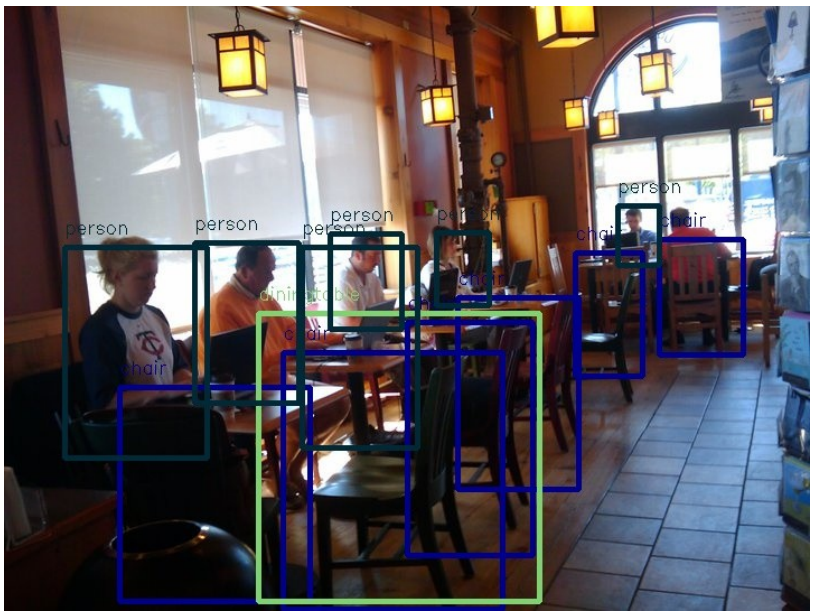
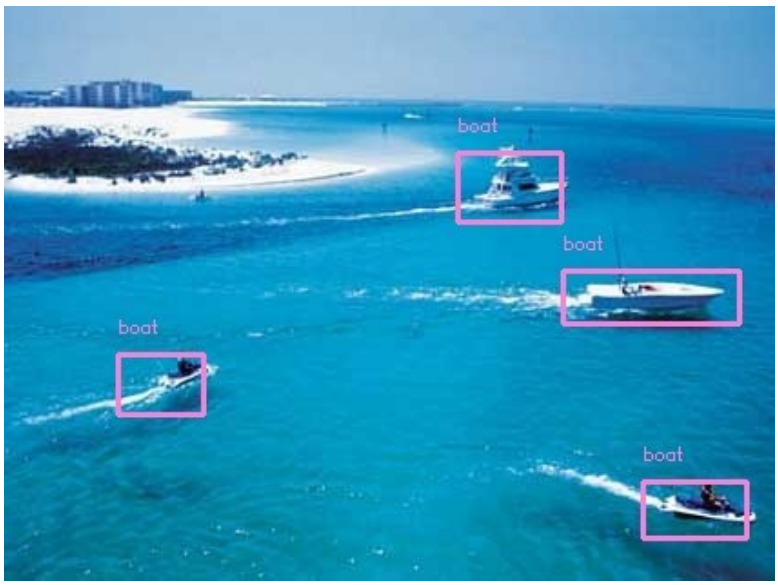


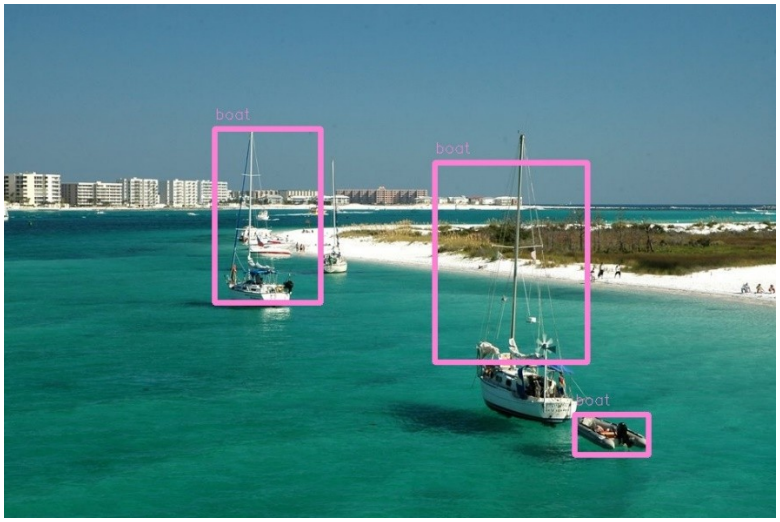
Figure 5: Testing on a set of images

Results:

The following are the results obtained when the images were passed through the pre-trained model







Inferences:

From the results, it can be seen that the common objects are detected successfully. Since we are using ultrasonic sensors for detecting the presence of an obstacle, the object size is fairly large and can be successfully detected.

Future Work

1. Creation of python TCP module to transfer data from LabView to Python.
2. Integration of developed model with hardware components.
3. Improvement of object detection accuracy.
4. Creation of Navigation algorithm for the unmanned marine vehicle to traverse from one point to another without any external aid.

References

1. Tensorflow API for creation of object detection models - https://github.com/tensorflow
2. Darkflow for object detection model - https://github.com/thtrieu
3. F. Pedregosa et al., “Scikit-learn: Machine learning in Python.” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
4. Understanding various approaches and development of plots in Machine Learning - https://pythonprogramming.net
5. Hands-On Machine Learning with Scikit-Learn and TensorFlow - Aurélien Géron
6. Program that creates XML description files for a list of training images - https://github.com/tzutalin