Real-Time Dangerous Marine Animals Monitoring to Alert Scuba Divers Using Computer Vision

R. PERUMALRAJA, M. SARASWATHI, S. SUJA PRIYA, R. SWEATHA AND

Department of Information Technology
Velammal College of Engineering and Technology
Madurai, India.

ABSTRACT

The ocean is filling up with all kinds of venomous and terrifying creatures. Humans in the ocean should be easy prey to deadly marine animals causing injuries and fatal attacks. To deal with this, we have proposed a pattern recognition algorithm with a one-stage object detection model "RetinaNet" to identify deadly marine organisms using high-resolution cameras in drone-captured video-based monitoring along the coast. The deadly organisms are tracked using the Kalman filter algorithm that reduces the overlay effect of detected species in aerial view video samples. Then, they are recognised and localized using RetinaNet where its architecture encompasses several subnetworks that classify marine animals according to their nature either into dangerous class or non-dangerous class and calculates proximity between the swimmer and dangerous marine animal. If the dangerous marine animals are found approaching the swimmers then the drone would alert the human through constant vigilance.

Keywords — *RetinaNet, Computer Vision, Deep neural network, Kalman filter*

INTRODUCTION

The goal of the project is to propose a pattern recognition model that has a high accuracy rate, handles class imbalance, works well with a small training dataset, calculates proximity and provides real-time tracking by processing video input frame by frame. This work proved to be more accurate by detecting objects present on the sea surface, near the surface, and in the air column under the drone. The RetinaNet architecture encompasses several subnetworks that classify marine species according to their object class, then track if they fall into the "dangerous species" class and calculate proximity between detected dangerous class species and swimmer(s) nearby.

Training with less training dataset, RetinaNet provided more accuracy even with larger testing data. Maintaining an optimal value of γ in the focal loss function gives greater accuracy to classification [4]. Detection of a still image as input to an object tracker requires less computational effort and provides more accuracy than videos which prove to be computationally expensive and have low detection accuracy [4] thus the drone monitoring video is processed as individual frames (images). These images are then passed through the 107 convolutional layers for feature extraction and then merged by FPN. Finally, this sub-network performs classification.

The use of the Feature Pyramid Network (FPN) to detect objects proved to be more efficient than other algorithms. With the development of powerful GPUs, processing power has increased. This increases the computational speed and quick real-time processing [4]. Also, the processing of drone-captured object images on a cloud platform provides real-time and has less overhead on drones. The detection results from the FPN network stand out with an accuracy of 97.2% [8]. A State-of-the-art CNN Detector method for marine video along with an object tracker for detection and classification gives greater precision and recall both above and underwater, even though a very small dataset (only hundreds of images) is used to train it [4].

RELATED WORK

A. *RetinaNet*

The novelty of RetinaNet lies in the focal loss that handles class imbalance problems and scale invariance. The RetinaNet model consists of three parts: (i) network, which is used to extract image features; (ii) FPN performs feature processing; and (iii) classification of sub-network, which collectively gives the final object detection result. More the CNN layers, the abstraction degree of feature extraction also increases

The RetinaNet model uses various backbone networks as its underlying source, say, VGGNet, YOLO, ResNet, EfficientNet, etc. However, in our model the backbone network we use is ResNet. The reason that we opt for ResNet is because of its accurate object localization compared to other network models. With ResNet, we also incorporate the idea of Feature Pyramid Networks (FPN) which guarantees thorough detection and feature extraction from it. Fundamentally, ResNet is used as the base and FPN is built on top of it which highly helps the model for feature extraction. The type of datasets we fed was images and video clippings. The FPN is built in such a way that the still images or image frames from regular instances of video streams are laterally connected and have semantically strong features. With this rich semantics, it builds every single image scale without any compromise in representational speed or accuracy. Also, it has overcome the class imbalance problems by applying the Focal Loss function which greatly negotiates the occurrence of higher negative background classes and fewer positive foreground classes. The focal loss function we used will focus on training more hard negative classes than the easy positives. By this, we will attain a smooth class balance in the model. Being a one-stage detector model, the trained samples are tested and classified at a stretch thereby promoting the performance rate of the model.

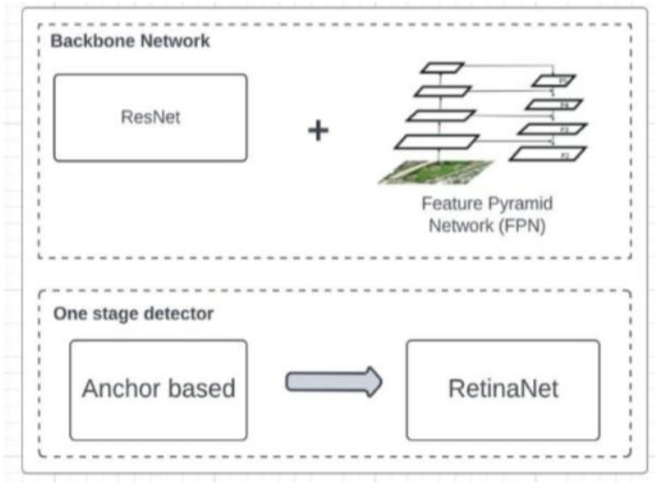


Figure 1.1 Underlying object detection networks used in our experiment

B. *Kalman filter and Hungarian Algorithm*

The Kalman filter algorithm is used for multiple target tracking. With this algorithm, we can track the motions of multiple targets grounded on certain values, in particular the centroid of an object and the tracking window. The algorithm works by taking *kth* frame and $k+1th$ frame as inputs and applying three phases (1) *Motion model* a vector matrix of horizontal and vertical centroid, width and height of the tracking window and speed of respective coordinates (2) *Feature Matching* (3) *Model update.* The algorithm first takes a frame as input and applies all phases; if the frame is found to be the *kth* frame, the Kalman filter initiates from the beginning and finds for feature matching. If it is a $k+1th$ frame it then goes for an update. The model is being tracked and adjusted empirically. With this, we can detect the trajectory motions of the marine animals and the divers/swimmers. The Hungarian algorithm combinedly works to find the optimal matching pairs for every timeframe with the updated locations of all marine animals.

PROPOSED SYSTEM

❖ SYSTEM OVERVIEW DIAGRAM

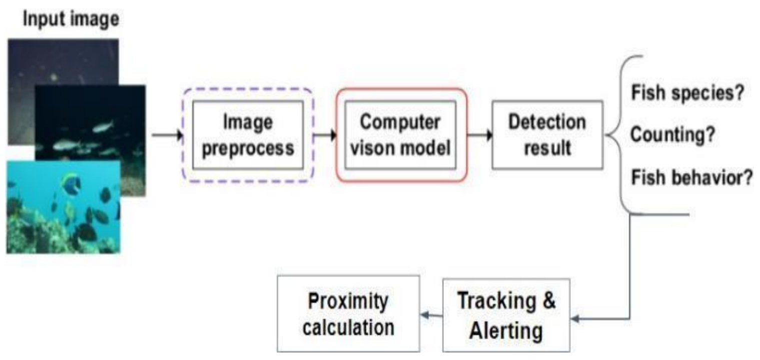


Figure 2.1 System diagram of the proposed model

Various marine image files are preprocessed by identifying the marine species in them and labelling their position values as "Class_Identifier, x, y, width, height" in a text file. Class_Identifier is used to distinctly identify different species for the model to train. Preprocessing can be done manually or by automation tools. The RetinaNet model is loaded with these train and test data input for processing. The detected classes/objects along with their position in the test input sample are used to find the distance between any 2 labelled objects(fish or swimmer). Thus, swimmers in dangerous zones can be alerted quickly if such dangerous animals approach their way.

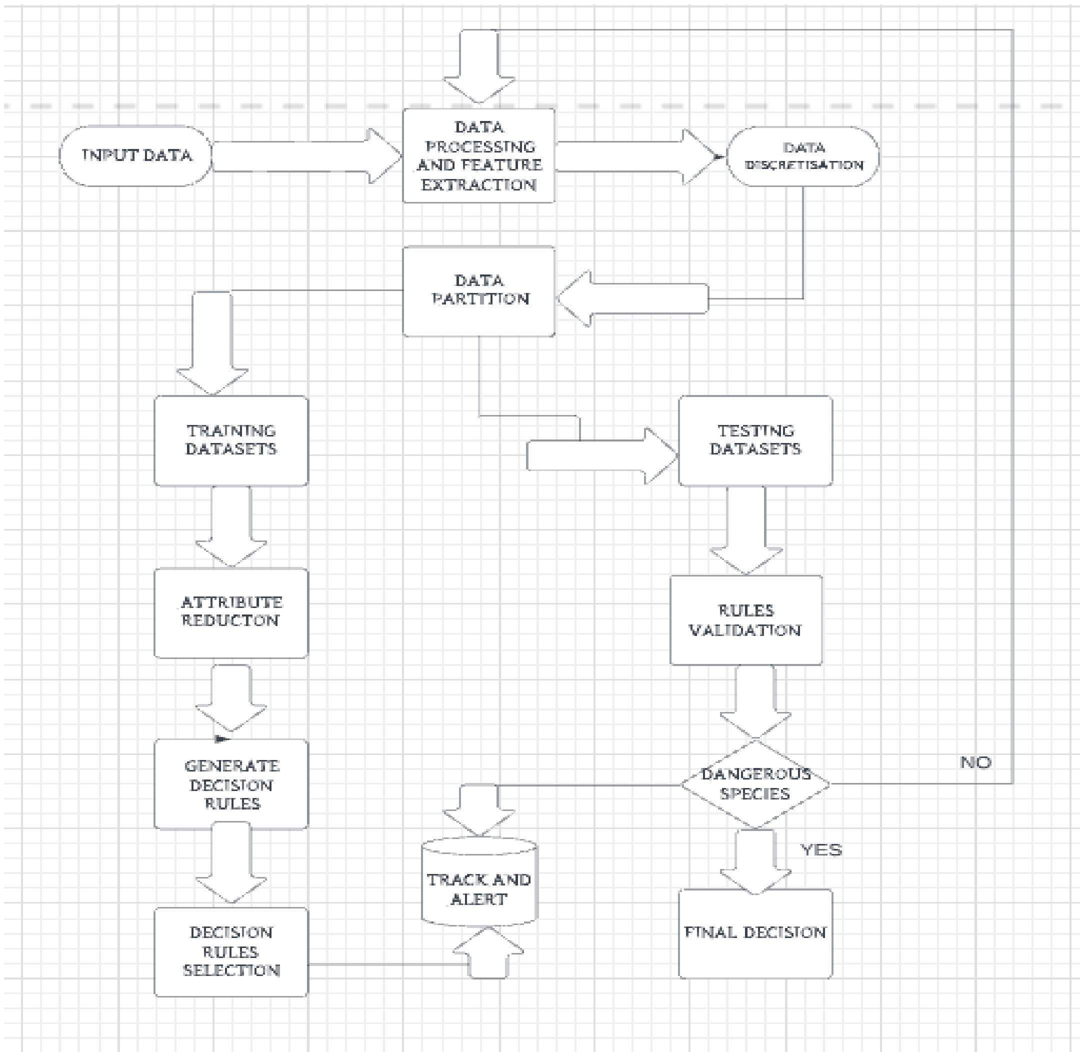


Figure 2.2 Flow chart depicting overall working of the proposed system.

❖ IMPLEMENTATION PHASES

The implementation of the project undergoes several phases. Each phase contributes to the major outcome of the project's model without compromising the best practices.

A. *Training and Detection*

For training the model, we used more than 2600 video clippings (taken as instances) and more than 2000 image datasets from MS COCO and Kaggle. The datasets are augmented by rotation and flipping to yield better results. The marine image datasets are annotated as TXT files that contain annotation information such as [Class_Identifier, X_value, Y_value, Width, and Height] of the class object in that image. With RetinaNet – a one-stage object detector, the marine animals at each frame are detected by dint of FPN and FL (Focal Loss). First, the input video is processed at each frame for object detection using the train dataset information provided to it. The 106 Convolutional layers run at the start which acts as a sub-network that lines up the feature and image correspondingly. The final layer (107th) takes filtered and scaled images put them into a single list and classification is performed.

B. *Classification*

The marine objects are classified based on the Class_Identifier specified in the annotated file that uniquely identifies the marine species. In our case, we must classify the marine bodies that are found harmful and considered dangerous to humans diving along the coast. We classify the dangerous marine animals and non-dangerous marine animals which are put under two main classes dangerous and non-dangerous respectively and the classes are labelled as such.

C. *Tracking and Alerting*

To track the movement of the marine animals, we use the Kalman filter and Hungarian algorithm for MOT. With this, we can track the motions of multiple targets grounded on certain values. Upon tracking, the nearby swimmers are alerted if any dangerous species are encountered. Providing wireless communication between the drone system and the swimmer's tracking device by beeping with light signals. The proximity can be set to a certain radius so that the drone can signal appropriately when a threshold is reached.

D. *Proximity calculation*

The distance between the marine species and the person is calculated using the Rectangle centroid formula. The centroids of these objects are found by using bounding box coordinates of the frame as inputs to the Rectangle centroid *formula*:

$$center\,x = x + ½ * width$$
$$center\,y = y + ½ * height$$

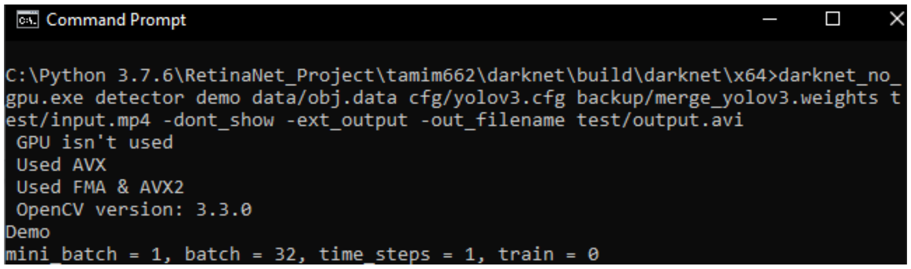
Where x, y, width and height are the objects' bounding box coordinate values in the frame and center_x, center_y are the centroid coordinates.

Finally, proximity is calculated by subtracting the centroid coordinates of both the desirable objects.

E. *Counting*

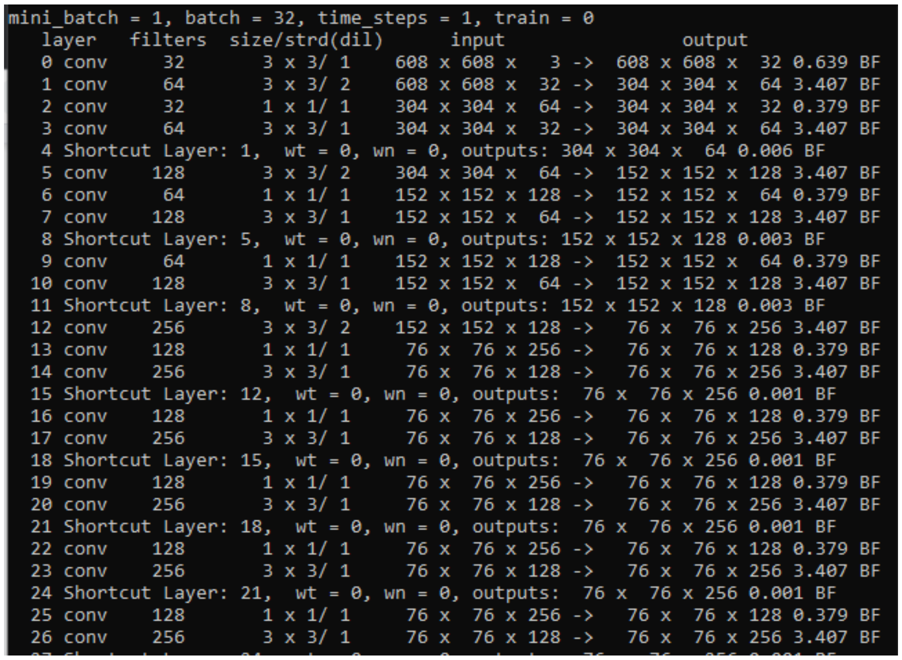
The class labels that contain their names as 'Dangerous' are counted simultaneously while calculating the proximity between swimmers and dangerous marine animals. This help determines any marine area as a Dangerous animal zone, Safe to swim zone or Moderately safe to swim zone.

EXPERIMENTAL RESULTS



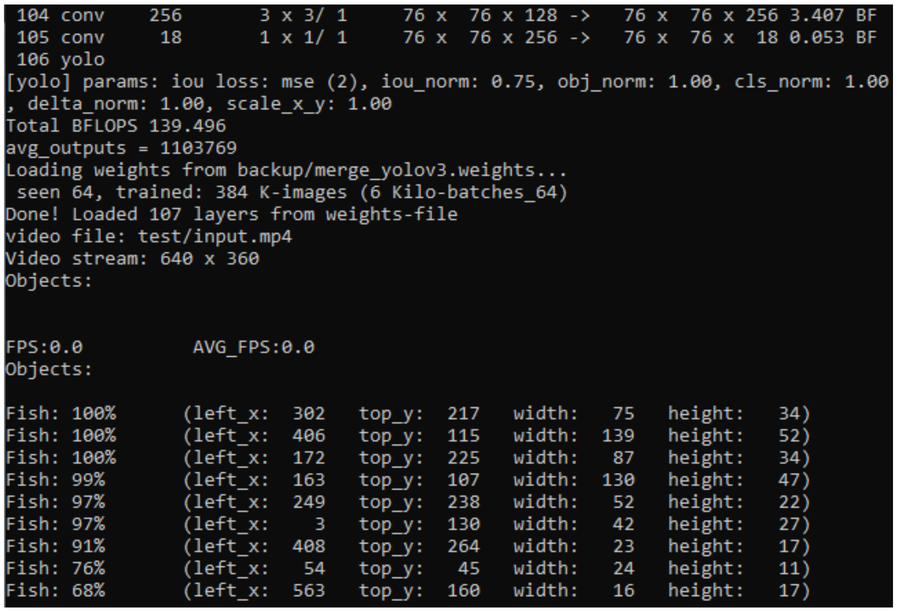
```
C:\Python 3.7.6\RetinaNet_Project\tamim662\darknet\build\darknet\x64>darknet_no_
gpu.exe detector demo data/obj.data cfg/yolov3.cfg backup/merge_yolov3.weights t
est/input.mp4 -dont_show -ext_output -out_filename test/output.avi
 GPU isn't used
 Used AVX
 Used FMA & AVX2
 OpenCV version: 3.3.0
Demo
mini_batch = 1, batch = 32, time_steps = 1, train = 0
```

Figure 3.1 Terminal containing RetinaNet.exe, Train data, Test Video file, Model and Weight files



```
mini_batch = 1, batch = 32, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639 BF
   1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407 BF
   2 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379 BF
   3 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407 BF
   4 Shortcut Layer: 1,  wt = 0, wn = 0, outputs: 304 x 304 x  64 0.006 BF
   5 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407 BF
   6 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
   7 conv    128       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x 128 3.407 BF
   8 Shortcut Layer: 5,  wt = 0, wn = 0, outputs: 152 x 152 x 128 0.003 BF
   9 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  10 conv    128       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x 128 3.407 BF
  11 Shortcut Layer: 8,  wt = 0, wn = 0, outputs: 152 x 152 x 128 0.003 BF
  12 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407 BF
  13 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  14 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
  15 Shortcut Layer: 12,  wt = 0, wn = 0, outputs:  76 x  76 x 256 0.001 BF
  16 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  17 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
  18 Shortcut Layer: 15,  wt = 0, wn = 0, outputs:  76 x  76 x 256 0.001 BF
  19 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  20 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
  21 Shortcut Layer: 18,  wt = 0, wn = 0, outputs:  76 x  76 x 256 0.001 BF
  22 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  23 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
  24 Shortcut Layer: 21,  wt = 0, wn = 0, outputs:  76 x  76 x 256 0.001 BF
  25 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379 BF
  26 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
```

Figure 3.2 First batch run with 107 convolutional layers.



```
 104 conv    256       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 256 3.407 BF
 105 conv     18       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x  18 0.053 BF
 106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, obj_norm: 1.00, cls_norm: 1.00
, delta_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 139.496
avg_outputs = 1103769
Loading weights from backup/merge_yolov3.weights...
 seen 64, trained: 384 K-images (6 Kilo-batches_64)
Done! Loaded 107 layers from weights-file
video file: test/input.mp4
Video stream: 640 x 360
Objects:


FPS:0.0          AVG_FPS:0.0
Objects:

Fish: 100%      (left_x:  302   top_y:  217   width:   75   height:   34)
Fish: 100%      (left_x:  406   top_y:  115   width:  139   height:   52)
Fish: 100%      (left_x:  172   top_y:  225   width:   87   height:   34)
Fish: 99%       (left_x:  163   top_y:  107   width:  130   height:   47)
Fish: 97%       (left_x:  249   top_y:  238   width:   52   height:   22)
Fish: 97%       (left_x:    3   top_y:  130   width:   42   height:   27)
Fish: 91%       (left_x:  408   top_y:  264   width:   23   height:   17)
Fish: 76%       (left_x:   54   top_y:   45   width:   24   height:   11)
Fish: 68%       (left_x:  563   top_y:  160   width:   16   height:   17)
```

Figure 3.3 The input.mp4 is given with weights set.

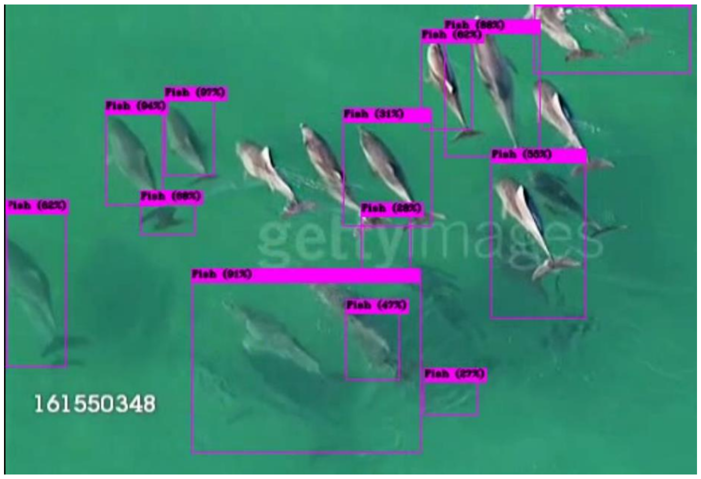
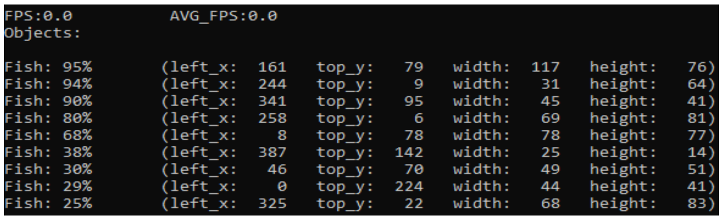


Figure 3.4 The output.avi in aerial view detection



```
FPS:0.0          AVG_FPS:0.0
Objects:

Fish: 95%       (left_x:  161   top_y:   79   width:  117   height:   76)
Fish: 94%       (left_x:  244   top_y:    9   width:   31   height:   64)
Fish: 90%       (left_x:  341   top_y:   95   width:   45   height:   41)
Fish: 80%       (left_x:  258   top_y:    6   width:   69   height:   81)
Fish: 68%       (left_x:    8   top_y:   78   width:   78   height:   77)
Fish: 38%       (left_x:  387   top_y:  142   width:   25   height:   14)
Fish: 30%       (left_x:   46   top_y:   70   width:   49   height:   51)
Fish: 29%       (left_x:    0   top_y:  224   width:   44   height:   41)
Fish: 25%       (left_x:  325   top_y:   22   width:   68   height:   83)
```

Figure 3.5 Each frame of video is interpreted as an Image and a fish detection algorithm is applied. The position of the detected fish is found for Proximity calculation.

Figure 3.6 input.mp4 file

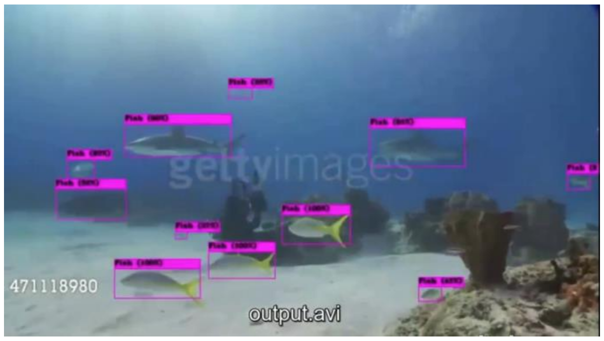
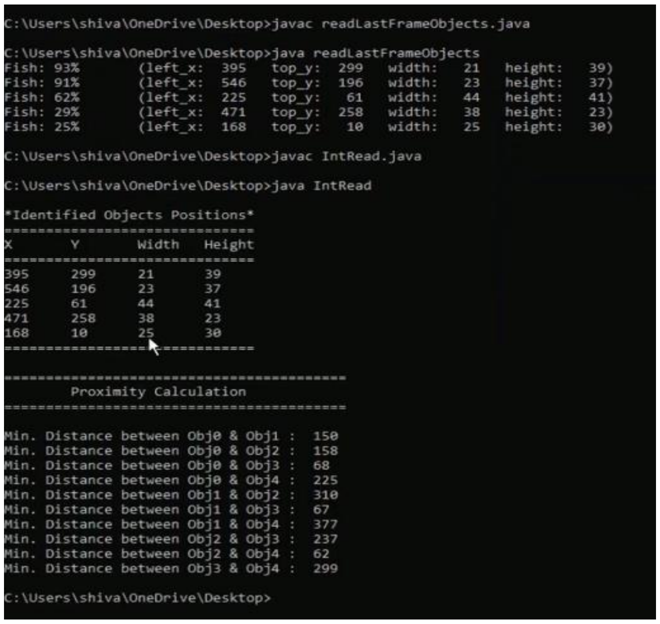


Figure 3.7 output.avi file



```
C:\Users\shiva\OneDrive\Desktop>javac readLastFrameObjects.java

C:\Users\shiva\OneDrive\Desktop>java readLastFrameObjects
Fish: 93%       (left_x:  395   top_y:  299   width:   21   height:   39)
Fish: 91%       (left_x:  546   top_y:  196   width:   23   height:   37)
Fish: 62%       (left_x:  225   top_y:   61   width:   44   height:   41)
Fish: 29%       (left_x:  471   top_y:  258   width:   38   height:   23)
Fish: 25%       (left_x:  168   top_y:   10   width:   25   height:   30)

C:\Users\shiva\OneDrive\Desktop>javac IntRead.java

C:\Users\shiva\OneDrive\Desktop>java IntRead

*Identified Objects Positions*
==============================
X       Y       Width   Height
==============================
395     299     21      39
546     196     23      37
225     61      44      41
471     258     38      23
168     10      25      30
==============================

=========================================
        Proximity Calculation
=========================================

Min. Distance between Obj0 & Obj1 :  150
Min. Distance between Obj0 & Obj2 :  158
Min. Distance between Obj0 & Obj3 :  68
Min. Distance between Obj0 & Obj4 :  225
Min. Distance between Obj1 & Obj2 :  310
Min. Distance between Obj1 & Obj3 :  67
Min. Distance between Obj1 & Obj4 :  377
Min. Distance between Obj2 & Obj3 :  237
Min. Distance between Obj2 & Obj4 :  62
Min. Distance between Obj3 & Obj4 :  299

C:\Users\shiva\OneDrive\Desktop>
```

Figure 3.8 Proximity calculation for the coordinates of detected classes.

CONCLUSION

Object detection, closely related to image understanding and video analysis, is an influential and extensively studied approach in computer vision. Analyzing marine data of vast size manually is not feasible, our project aims to offer new solutions to marine animal detection and hence facilitate the monitoring of their existence and lives. The models we built for this project can also benefit the related research and call for increasing attention to marine animal protection and human fatal deaths. This paper's Object detector model precisely estimates the locations of objects and further demonstrates their presence in given aerial view images or videos by drawing bounding boxes with confidence scores around the objects of interest correspondingly. We focused on marine animal detection with a more complicated background i.e., Aerial view, which is different from traditional object detection. The calculated distance between dangerous animals and swimmers helps reduce coastal deaths by alerting the swimmer in advance. This research might be useful in classifying any area as dangerous marine animals prevailing zones or not, by analyzing patterns of detected marine species. Extending this work to multispecies detection would be helpful in collectively identifying all prevailing dangerous species and alerting the swimmers if they are in a dangerous zone.

REFERENCES

1. Lin Meng, (IEEE), Takuma Hirayama and Shingeru Oyangi (Member, IEEE) “Underwater-drone with panoramic camera for Automatic fish recognition based on Deep Learning”, Volume-4, June 2016.
2. Alessandro Lambertini, Massimiliano Menghini, Jacopo Cimini, Angelo Odetti, Gabriele Bruzzone “Underwater drone architecture for marine digital twin”, Project- Sensors 22, 744. h, August 2022.
3. Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S. Palmer, Craig Packer, and Jeff Clune “Automatically identifying, counting, and describing wild animals in camera- trap images with deep learning”, National Academy of Sciences, June 2018.
4. Deborah Levy, Yuval Belfer, Elad Osherov, Eyal Bigal, Aviad P. Scheinin, Hagai Nativ, Dan Tchernov, and Tali Treibitz “Automated Analysis of Marine Video With Limited Data”, IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2018.
5. Longyin Wen, Dawei Du, Pengfei Zhu, Qinghua Hu, Qilong Wang, Liefeng Bo, Siwei Lyu “Detection, Tracking, and Counting Meets Drones in Crowds: A Benchmark ” IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2021.
6. Di Guo, Fuchun Sun, Tao Kong, Huaping Liu “Deep vision networks for real-time robotic grasp detection”, Volume: 14 issue: 1, December 2016.
7. Frederic Maire, Luis Mejias Alvarez and Amanda Hodgson “Automating Marine Mammal Detection in Aerial Images Captured During Wildlife Surveys: A Deep Learning Approach”, Australasian Joint Conference on Artificial Intelligence, November 2015.

8. Amarjot Singh, Devendra Patil, SN Omkar "Eye in the Sky: Real-time Drone Surveillance System (DSS) for Violent Individuals Identification using ScatterNet Hybrid Deep Learning Network", IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2018.
9. Frederik Kratzert, Helmut Mader "Fish species classification in underwater video monitoring using Convolutional Neural Networks", May 2018.
10. M.Subba Rao, B.Eswara Reddy "Comparative Analysis of Pattern Recognition Methods", Indian Journal of Computer Science and Engineering (IJCSE), January 2011.
11. Longyin Wen, Dawei Du, Pengfei Zhu, Qinghua Hu, Qilong Wang, Liefeng Bo, Siwei Lyu "Detection, Tracking, and Counting Meets Drones in Crowds: A Benchmark " IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2021.
12. Ilseo Jeon, Sangwoo Ham, Jangwo Cheon, Anna Maria Klimkowska "A Real-Time Drone Mapping Platform For Marine Surveillance", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-2/W13, 2019.
13. Mr Di Guo Mr Funchun Sun " Deep vision networks for real-time robotic grasp detection", December 27, 2016
14. Muhammad Ather Iqbal1, Zhijie Wang, Zain Anwar Ali, Shazia Riaz "Automatic Fish Species Classification Using Deep Convolutional Neural Networks", part of Springer Nature 2019
15. Alexander Ryan Pollard "Computer vision guidance for precise movement in commercial drones", Oct 2017
16. Laurin Goeller "Drone Precision Landing using Computer Vision", Jan 24, 2018
17. Sudipa Pascal "Flight Dynamics-Based Recovery of a UAV Trajectory Using Ground Cameras", IEEE-2017
18. Morten Goodwin, Kim Tallaksen Halvorsen "Unlocking the potential of deep learning for marine ecology: overview, applications, and outlook", 29 Sep 2021.
19. Artem Rozantsev, Vincent Lepetit, Pascal Fua "Flying Objects Detection from a Single Moving Camera", IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2015.
20. Amarjot Singh, Devendra Patil, SN Omkar "Eye in the Sky: Real-time Drone Surveillance System (DSS) for Violent Individuals Identification using ScatterNet Hybrid Deep Learning Network", IEEE CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2018.
21. Frederic Maussang, Liyun Guelton, Rene Garello and Alexis Chevallier "Marine Life Observation using Classification Algorithms on Ocean Surface Photographs", IEEE, May 2015.
22. Vemema Kangunde1, Rodrigo S., Jamisola Jr., Emmanuel K., Theophilus1" A review on drones controlled in real-time", International Journal of Dynamics and Control (2021) 9:1832–1846.
23. Michael Esty, Lauren Hayden "Use of Drone Technology in Ocean Research Of Saco River Estuary supporting Student Research", University of New England.
24. Fenglei Han, Haitao Zhu and Chunhui Wang "Underwater Image Processing and Object Detection Based on Deep CNN Method", 22 May 2020

25. Haitao Zhu and Chunhui Wang "Marine Organism Detection and Classification from Underwater Vision Based on the Deep CNN Method", International Journal on Mathematical Problems in Engineering, 2020.