Vol: 1 Issue: 1

International Journal of Data Science and Advanced Analytics

Object Identification in Maritime Environments for ASV Path Planner

Oren Gal[1], Sean Levy[1]
[1]Technion. Israel Institute of Technology, Haifa
email: orengal@technion.ac.il

Received: 25-Aug-19; Accepted: 08-Sep-19; Published: 08-Sep-19
Corresponding author email: orengal@technion.ac.il

Abstract— **In this paper, we present real-time object identification algorithm, that can be implemented in the motion planning in Autonomous Surface Vehicles (ASV) at sea based on Faster Regional Convolutional Neural Networks (RCNN) model. These vehicles' control algorithms and path planning systems depend heavily on the surrounding area and the objects in sight, thus perception capability in near field is crucial for ASV's safety. Such systems can also be used for security applications, maritime traffic control and port management. The dataset we used consists of thirty six videos, captured in HD resolution, moreover, we shot another 3 videos for visual testing. We trained a Faster RCNN model in addition to a YOLOv3 model and compared the results. The first framework gave the best results while being the slowest, while the second one was fast and slightly less accurate. We showed that such a system can be implemented on a real-time camera feed and can be used as a part of the developments of an ASV as integral part of path planner for ASV .**

Keywords—**Object detection; Neural Networks; Marine Transportation; Unmanned Autonomous Vehicles**

1 Introduction

With the recent advances in Deep Learning in general and object detection in particular, in addition to the recent growth in learning algorithms, sparked the research focused on combining the two systems – computer vision and path planning.

These systems have been studied for decades. Most of the algorithms we use today were developed in the past but had not succeeded then because of the lack of data and computing power. Nowadays, we see improvements every year, as computers gain more power and more complex methods are developed. One type of algorithm that really thrived in recent years is Neural Networks. In the last few years, Neural Networks were responsible for all the advancements in the field of computer vision. When scientist and researchers sat down to tackle the problems of object detection using Neural Networks, they created RCNN [1] - Regional Convolutional Neural Networks. After 3 years, Faster RCNN [2] was created, which not only detected objects faster, but also had very high accuracy. Concurrently, two more frameworks were being developed – YOLO [3] and SSD [4], each having several versions. Although these frameworks are faster than RCNN and its ancestors, the accuracy is lower. These 3 networks are the main competitors today, YOLO being the only one that can sustain a real-time workload.

A vision-based robot navigation system is one that allows an autonomous robot to move throughout its environment under constrains, such as avoiding obstacles. The availability of low cost, low power cameras and high speed SOCs (system on a chip) are the main reasons for the rapid development and growth of image sensor applications in motion and path planning. A mobile robot navigation system based on vision can be divided into indoor and outdoor type of navigation. The former can be a map-based or a mapless form of navigation, while the latter depends whether it is a structured or an unstructured environment. Both use object detection and tracking.

In this project we performed detection on onboard/onshore video streams for the use of an autonomous ship motion planner around ports and docks. One of main the requirements here is the ability to operate at a real-time rate, running on a live video feed without much delay. We trained two largely known frameworks to be used for object detection in the future development of the ASV.

2 Materials and Methods

Before the rapid growth in learning algorithms in recent years, computer vision problems were solved by using classic mathematical models and algorithms. Object detection, in particular, was an important area of research because of the tremendous amount of real-world applications that require using it.

2.1 Classic approach

The classic approach for extracting ships from camera images is to segment the image using a segmentation algorithm and then to do a form of texture and shape analysis [5]. Another approach that was popular is focused on calculating image correlation between two images, in video-streams [6]. More ways and approaches were developed through the years.

Krüger and Orlov [7] developed a whole system to automatically detect and track distant vessels in the images generated by the thermal or visual imager. It consists of a camera, pan-tilt unit, inertial measurement unit and image exploitation computer. The first layer in the model calculates an estimation of camera orientation, using a Kalman filter on the raw data to estimate time-varying pitch and roll angles. The second layer uses the results prior to localize the horizon line in the images. The third layer is the boat detection layer. It uses the information about the horizon line to set up search areas for the algorithms and stabilize the image. The framework utilizes track-before detect algorithm using blobs, detection exploiting stable image regions and detection based on tracking salient image points. The fourth layer generates the detections.

Object detection in 4K video was the subject of [8]. The advantage of this higher resolution is providing more information in the harsh sea conditions. An alternative way of exploiting rigidity analysis (segmentation of a multi-body fundamental matrix) was proposed by using keypoint tracking. The first stages use a SIFT descriptor to find keypoint objects, and then perform some image processing in order to identity full object zones. Later, PCA is applied to extract the main directions of textural variation. The detection is achieved by merging image blocks ranked as object by the texture discrimination algorithm and containing one stable SIFT keypoint. Furthermore, the algorithm is implemented over multiple resolutions of the same image.

2.2 Deep Learning approach

Many researchers in this field started using deep learning and neural networks, some implementing existing ones and some creating new architectures based on known ones. When performing object detection with deep learning there are several facts to be aware of:

1. The need for a large dataset
2. Preferably a large variety of objects should be used to increase the strength of the algorithm.

In [10] transfer learning is applied to improve the results over a small dataset. Training a CNN using random parameter initialization with a small dataset will results in low performance. In such situations, transfer learning is implemented – the last fully connected layer of the network is replaced and retrained with the new data. A CNN based on Inception and ResNet models was trained on the ImageNet database. Then the pre-trained network was fine-tuned with the MARVEL dataset. After training several times and choosing the right hyper-parameters, the top accuracy was around 78%. In [11] a Faster RCNN network detects the objects. The authors built a sequence for the bounding boxes using the boxes returned from the network, the sequence is gathered over time and combined by a Bayesian fusion. It is assumed that the target ships do not move rapidly, therefore in each time step the bounding box with the largest IoU is selected. Moreover, an offset to the sequence was added, which was calculated between consecutive frames. The dataset used in this paper was collected manually from Google images - 7000 images of 7 classes - and annotated. The single image detection result was 94.65% (mAP) and when tested on seven video clips from YouTube the accuracy was 93.92%.M. H. Zwemer et al. [12] used an SSD framework to improve a visual tracking system. An SSD512 model detects the objects at 5 frames per second. This model takes as an input 512x512 images, thus it′s unique name. After object detection is performed on the input image, the obtained vessel coordinates are converted to GPS coordinates using the camera calibration.. To create trajectories over time, visual tracking is performed by feature points. A total of 48966 image were collected from video footage. The authors used transfer learning with a VGG16 network to train the single shot decoder and reached 88% (mAP) accuracy.

Vision-based robot navigation requires mapping of the sensory signals to motion control commands, which involves complex feature processing in environmental perception. The classic methods for this task do not take advantage of learning algorithms and use computer vision algorithms. In [13] stereo disparity was used to find the 1d displacement of corresponding pixels in the two images. Then, using disparity-based segmentation, seed points are found which are then used to determine the location of obstacles. A path planner was implemented based on the path cost.

2.3 DRL and path planning

Recent Deep Reinforcement Learning models provide an end-to-end framework for transforming pixel information into actions. In [14], the method developed tackles the problem of navigating a space to find a given target goal using only visual input. To train and evaluate the model, a new simulation framework with high-quality 3D scenes was developed, called The House Of inteRactions (AI2-THOR), which is designed by integrating the Unity 3D physics engine with Tensorflow. The model is a deep reinforcement learning model that takes as input an RGB image of the current observation and another RGB image of the target. The output of the model is an action in 3D such as move forward or turn right. It’s worth noting that model is based on a Deep Siamese Network which is a type of two-stream neural network. Information from both embeddings is fused to form a joint single representation. This joint representation is passed through scene-specific layers (room layouts and object arrangements), andthen the model generates policy and value outputs. In training phase,the framework ran several copies of training threads in parallel, but instead of running copies of a single game, every agent got a different navigation target. In [15], a motorized wheelchair was equipped with a front facing camera and a few LIDARs providing a 270 degrees coverage of objects present in the area surrounding the system. The visual path prediction system uses a YOLOv2 net [16], which recognizes 80 different classes and provides a bounding box around the object in the image. Each bounding box is accompanied with a label, top, bottom, right, left coordinates and the prediction confidence. Then the system adjusts the Lidar and vision data frames to overlay the field of views of one another, converts the image resolution to that of the scan resolution and matches the two frames.

These works inspired us to develop a system that will be implemented on an unmanned ship, and will be used as an integral part of the path planning algorithm.

3 Preliminaries

3.1 Basic object detection metrics

When evaluating object detection, we come across two different result measurements:
1. Classification- whether an object exists in the image
2. Localization- finding the location of the object

In large datasets there are many classes and thus we cannot use simple accuracy-based metrics. That is why object detection algorithms introduce new metrics.

3.1.1 Intersection Over Union

In order to decide whether the predicted bounding box is accurate and measure the prediction accuracy we use Intersection over Union (IoU). Using this algorithm, we calculate the area of the IoU of the ground truth box and the framework's prediction. We choose a threshold and then we can decide which bounding boxes will be selected and which will be discarded. The IoU's formula is given by:

$$IoU = \frac{Area\ of\ Overlap\ (yellow)}{Area\ of\ Union\ (green + red)}$$

3.1.2 Non-maximum suppression

There is another approach that can improve our results called NMS. This is extremely useful when we want to eliminate multiple boxes around one object. Given the probabilities associated with each detection one having the best results is selected.

3.1.3 Average Precision and mAP

There are two main components to the Average Precision:
1. Precision- the ratio of the true object bounding box detections to the total number of bounding boxes predicted (on a single image).
2. Recall- the ratio of the true bounding box detections to the total number of bounding boxes present in the dataset.

In order to calculate the Average Precision, we create a precision-recall curve for each of the classes. For each class we look at all the outputs when using the dataset and all the ground-truths. Then, we loop over 11 recall values, find the maximum precision there and then compute the average.

$$AP = \frac{1}{11} \sum_{r \in \{0.0, 0.1, \ldots, 1.0\}} AP_r = \frac{1}{11} \sum_{r \in \{0.0, 0.1, \ldots, 1.0\}} \max\ p(r)$$

The AP across all the classes in the dataset is an average on all the AP values, and it is called Mean Average Precision, or mAP. mAP is the main parameter for evaluating object detection algorithms. Different datasets (VOC/COCO) add their own rules for the evaluation, i.e. what is considered a true prediction based on IoU, therefore there are several formats. It is important to note that one format can give a higher score or a lower one depending on the task itself. The calculation using the 11 precision-recall values is used in the PASCAL VOC 2007 format.

3.2 YOLO – You Only Look Once

The YOLO architecture [3] is widely used today in real-time applications. It takes an entire image at once and predicts the bounding boxes coordinates and class probabilities for these boxes. Thusthe high speed of operation. At the beginning, the framework takes an input image and divides it to a grid (normally 11 by 11 grid). Then, the net classifies each grid image and localizes it, from which YOLO predicts the bounding boxes and their class scores.

3.2.1 Anchor boxes

When a new object is assigned to a new grid (in this framework in each grid only one object is identifiable), the mid-point of the object is taken and assigned to the corresponding grid based on its location. In the case there are two midpoints of two objects lie in the same grid, only one of them will be selected. To solve this problem, Anchor Boxes are used. Two different shapes are predefined, and for each grid there will be two results. In practice, we use more than two shapes. The object is assigned to the bounding boxes based on the similarity between them and the anchor boxes' shape.

The YOLO training composes of 2 phases. First, a classifier network like VGG16 is trained. Then the fully connected layers are replaced with a convolution layer and retrain it end-to-end for the object detection. During training, the framework takes an image and maps it with a target- depends on the grid size, number of anchor boxes and number of classes. While testing, the input image will be divided by same grid size. Finally, Non-Maximum Suppression will be applied to obtain only one bounding box per object.

3.2.2 Loss function

During training the loss function is optimized to improve the predictions of the network. The loss function formula is:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2]$$
$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\widehat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\widehat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2$$
$$+\sum_{j=0}^{S^2}1_{ij}^{obj}\sum_{c\in classes}(p_i(c)-\widehat{p_i}(c))^2$$

The parameter λ is used to differently weight parts of the loss function. The first line computes the loss of the predicted bounding box position. It is a sum over each bounding box j of each grid cell i. 1^{obj} is 1 if an object is present in grid cell i and bounding box j is around it. It is zero otherwise. The second line is the loss related to the predicted box width and height. We use square root to account for small deviations in large boxes. The third line is responsible for calculation of the loss associated with the confidence sore for each box. 1^{noobj} is the opposite of 1^{obj}. The fourth part is the classification loss. It is written like a regular sum-squared error and the 1^{obj} parameter is used to not penalize classification error when there is no object in the grid cell.

3.3 YOLOv2

YOLOv2 is the second version of YOLO, improving the accuracy and speed of the notorious framework. The new architecture uses a model named Darknet-19 which is similar to VGG models by using 3x3 convolutional filters and doubles the number of channels after every pooling layer. Several new adaptations are made to increase the accuracy:

3.3.1 Batch normalization

A technique widely used while working with CNN's. It works by using small batches of the data in training. Removes the need for dropout layers (where neurons are deleted).

3.3.2 High resolution classifier

YOLOv2 trains using 224×224 images, and also uses 448×448 images for fine-tuning the classification network for 10 epochs.

3.3.3 Convolution and anchor boxes

YOLO makes arbitrary guesses on the size and shape of the boundary boxes. These guesses may work for some cases but not for others. Hence, if the model starts with diverse guesses that are common for real-life objects the training will be more stable. N anchor boxes are created, with certain shapes, and the model predicts offsets of each of them. The the input size was changed from 448x448 to 416x416 to create an odd number of spatial dimensions, because the center of the image often contains an object and it will be easier detecting. The authors removed one pooing layer to make the spatial output of the network 13x13, downsampling by 32

3.3.4 Dimension clusters

By using anchor boxes, two issues were found: the first one was that the bounding box dimensions were hand-picked, and while the network could learn to adjust those boxes it really depended on the anchors picked, better picked ones resulted in better performance. The suggested way is to use k-means clustering on the training set bounding boxes to find the centroids of the top-K clusters.
Using standard Euclidean distance-based k-means clustering is not good enough, thus the distance metric used is:

$$d(box, centroid) = 1 - IoU(box, centroid)$$

When the number of anchors increases, the accuracy plateaus. The authors chose 5 anchors –it was found as a good compromise.

3.3.5 Direct location prediction

The second problem with anchor boxes was model instability during early iterations. The main reason was the formulation used in the original YOLO–it was unconstrained and any box could end up at any point in the image, despite the location that predicted the box. To fix this the authorsgave predictions on the offsets to the anchors. YOLO predicts 5 bounding boxes at each grid cell in the output, or more specifically 5 sets of 5 parameters t_x, t_y, t_w, t_h, t_o and applies σ (sigma function, value between 0-1):

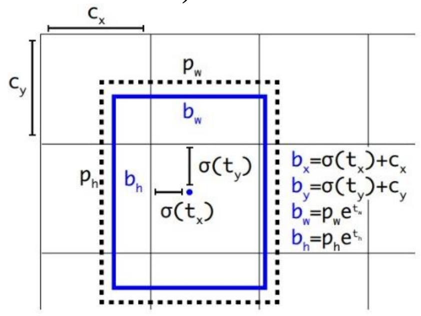


Figure 1: YOLOv2 anchors [16]

(b_x, b_y) is the location of the bbox; (p_w, p_h) is the anchor

3.3.6 Fine-grained features

YOLOv2 uses a passthrough layer which concatenates the higher resolution features with the low-resolution features by stacking them in order to detect smaller objects

3.3.7 Multi-scale training

YOLOv2 downsamples the input by 32 in the last layer, and after removing the fully connected layers it can take images of different sizes. Instead of fixing the input image size each time, every 10 batches the model selects a size randomly, in the condition that it is a multiple of 32 (i.e. 320x320, 352x352, the final one being 608x608). This forces the network to learn to predict accurately across a variety of input dimensions.

YOLO9000 [16] uses the YOLOv2 architecture to detect 9000 object classes using hierarchical classification. It combines samples from COCO and the top 9000 classes from the ImageNet by taking four ImageNet images for every COCO image. It learns to find objects using the detection data in COCO and to classify these objects with ImageNet samples.

3.4 YOLOv3

YOLOv3 [17] comes as an improvement to YOLOv2 when the latter failed in comparison to other methods of object detection, like RetinaNet and SSD. The creators give up the speed in order to improve the accuracy. YOLOv2 architecture lacks some important features that are considered as a state of the art in the way for good results, which are residual connections and upsampling. The main base network is darknet53. It is trained on ImageNet with 53 layers, and then 53 more layers are added for the process of detection alone. Darknet53 has less BFLOPS (billion floating point operations) than ResNet-152, but achieves the same classification accuracy faster.
The new architecture makes detections at three different scales – down sampling the input image dimensions by 32,16,8. Objects are detected in 3 different sizes features maps at three different locations in the network. The different layers help address the issue of detecting small objects. The low-resolution layers are responsible for detecting the large objects, whereas the large resolution ones should detect smaller objects. To set the anchors YOLOv3 applies k-means clustering, from which it selects 9 clusters- three for each scale. To obtain the multi-scale detection it assigns the three biggest anchors for the first scale, the next three for the second one and the last three for the third. That is why YOLOv3 predicts 10 times more the number of boxes that YOLOv2 predicts.

3.4.1 Predictions

The framework makes 3 predictions per location. Each prediction consists of boundary box, an objectness and class scores. The pipeline makes predictions at three different scales:

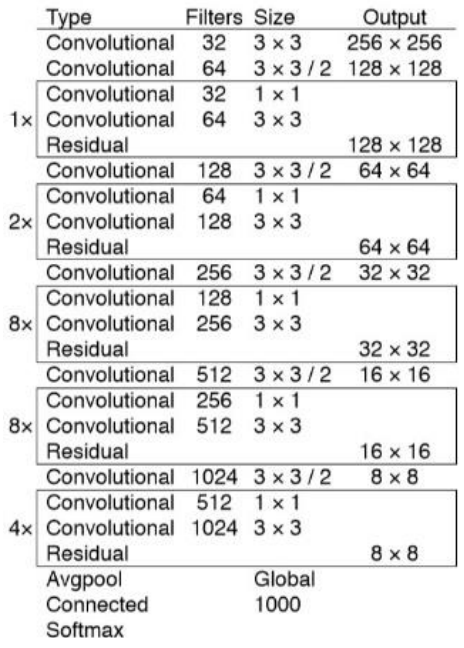
1. In the last feature map layer
2. It goes back and upsamples features by 2. Later, it takes a higher resolution feature map and combines them together. Then convolutional filters are applied on the new data to make the second set of predictions.
3. It repeats stage 2 so the resulted feature map layer has better high-level information.

3.4.2 Classes

Originally, a YOLO framework applied a softmax function layer to convert scores into probabilities. In contrary, YOLOv3 uses an independent logistic classifier (called also logistics regression) to calculate the score for the class-label pair. Instead of using mean square error in evaluating the loss, it uses binary cross-entropy loss. A threshold is used to predict multiple labels for an object.

3.4.3 Bounding-box prediction

The authors of YOLOv3 changed the way of calculating the cost function. Each ground truth object is associated with one boundary box prior only, if a bounding box is not assigned, only confidence loss is calculated, skipping classification loss and localization loss.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2: Darknet53 base model [17]

3.5 RCNN- Regional CNN

RCNN [1] is the first step towards Faster RCNN. The proposed method uses a naive algorithm called Selective Search to extract 2000 regions from the input image by combining similar pixes based on values and textures. Later, these 2,000 areas pass to a pre-trained CNN model. The outputs (feature maps) are then passed to a SVM for classification. The regression between predicted bounding boxes and ground-truth boxes are computed. The problem with RCNN is that the not only the training (and testing) is very slow because it has to classify 2000 region proposals per image, but also the selective search algorithm is a primitive one, and can be improved.

3.6 Fast RCNN

A year later,one of the creators who created RCNN, wrote a better version of the algorithm called Fast RCNN [18]. Instead of feeding the region proposals to the CNN, it feeds the input image to the CNN to generate feature maps. Then, using selective search on those feature maps the identified regions of proposals are wrapped into squares by using a ROI Pooling layer. These outputs are passed to a fully connected layer as inputs, and finally two output vectors are used to predict the object dimensions with a softmax classifier and adapt the bounding box locations with a linear regressor. This framework is faster than it's predecessor, but when inspecting the results selective search slows down the algorithm, is less accurate and is a major bottleneck.

3.7 Faster RCNN

Faster RCNN fixes the problem of selective search by replacing it with a Region Proposal Network (RPN) [3]. Itis the third official iteration of RCNN. Like Fast RCNN, the image is provided as an input to a CNN which outputs a feature map.

3.7.1 Anchors

Using the features that the CNN computed, the RPN is used to find a predefined number of regions (bounding boxes), which may contain objects. Those bounding boxes have predefined shapes and sizes and are called Anchors (just like in YOLO). In the default configuration of Faster R-CNN, there are 9 anchors at a single position in an image – 3 different sizes (e.g. 128,256,512) at three different ratios (e.g.1:2,1:1,2:1). Each point in the computed feature map is considered as an anchor, and a set of anchors for it is created. Even though anchors are defined based on the convolutional feature map, the final anchors reference the original image.

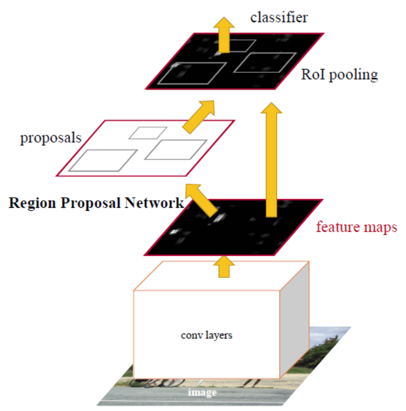


Figure 3: Faster RCNN model, consist of 2 networks running in parallel to each other and then fed to a Fast RCNN [19]

3.7.2 Region Proposal Network – RPN

RPN proposes objects by having two different outputs for each of the anchors. The first output is the probability that an anchor is an object, not depending on class - this score is used to filter out the bad predictions. The second output is the bounding box regression, in order of adjust the anchors. The RPN is connected to a convolutional layer with 3x3 filters, 1 padding, 512 output channels. The output is connected to two 1x1 convolutional layers for classification and box-regression. At the same time, non-maximum suppression is applied to make sure there is no overlapping between the proposed regions.

3.7.3 ROI Pooling

After the previous step, the model has several object proposals with no class assigned to them. Instead of taking each proposal, cropping it, and running it through the network, Faster RCNN reuses existing feature maps. This is done by extracting fixed-size feature maps for each proposal using region of interest (ROI) pooling.

The final step in the Faster RCNN's pipeline is to use the features extracted from the ROI pooling layer for classification. The last two goals:

1. Classify object proposals into classes
2. Adjust the bounding box dimensions

The algorithm flattens the feature map for each proposal and uses 2 fully-connected layers with ReLU activation. Then softmax and bounding box regression are applied.
After getting the final objects and ignoring those predicted as background, class-based NMS is applied.
For the final list of objects, a probability threshold is set and a the limit on the number of objects per class is chosen.

3.7.4 Loss function

After combining all the models, the loss function consists of 4 different losses (2 for RPN layer and 2 for RCNN). Those four losses are combined using a weighted sum, for being able to give one type of loss more weight than other ones. The loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{reg}$$

$$\mathcal{L}(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i \mathcal{L}_{cls}(p_i, p_i{}^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^*\mathcal{L}_{reg}(t_i, t_i^*)$$

Where $\mathcal{L}_{cls} = -p_i^* log p_i - (1-p_i^*)log(1-p_i)$

is the log loss function over two classes. A multi-class classification can be easily translated into a binary classification by predicting a sample being a target or not. The regression (box) loss is defined as:

$$\mathcal{L}_{reg} = \sum_{i\epsilon(x,y,w,h)} L_1^{smooth}(t_i - t_i^*)\ ,\ L_1^{smooth} = \begin{cases}0.5x^2 & if\ |x| < 1\\ |x| - 0.5 & otherwise\end{cases}$$

4 Methodology

4.1 Dataset

The dataset used in this project is the Singapore Maritime Dataset (SMD) [19]. The creators used several Canon 70D cameras to film around Singapore waters. All the videos acquired are in high definition, 1920x1080 pixels. The dataset was divided into parts: 40 on-shore videos taken from the shore and 4 on-board videos taken from a moving vessel. For each frame the creators annotated the horizon and object bounding boxes, and that is why this dataset is suitable for object detection. Moreover, the dataset contains 30 videos shot on a near-IR camera, which we did not use.

4 videos had no object annotation files, therefore 36 videos were used. Theon-board videos were not used, because they were very unstable, the camera was trembling. All the individual frames were extracted from the videos and saved as JPEG images. Next, the annotations were converted from the ".mat" format and saved in XML/TEXT files, depending on the architecture in use. Afterwards, each tenth frame was left behind and as a result it lowered the image number from 17967 to 1796. The videos were captured in 30 frames-per-second, thus some data could be discarded as there was a similarity between consecutive frames.

In order to demonstrate our solution's results, we videotaped 3 videos in different parts of the port of Haifa during different weather conditions and extracted a number of frames as a minitest set. These will be used mainly for the visual test, as they are not annotated.

4.2 Base network

For the base architecture in training Faster RCNN VGG-16 was chosen. In was developed in 2014 at the Oxford University in England [20]. As the name suggests, it consists of 16 layers- by today’s standards it would not be considered very deep, but at that time it more than doubled the number of layers commonly used and kick started the “deeper = better” wave.
The input has to be of size 224x224x3 and is passed through 16 layers.

The VGG-16 at the base was pre-trained on ImageNet dataset. This technique is called Transfer Learning and it is commonly used for training a classifier on a small dataset using the weights of a network trained on a bigger dataset. It is a common practice to always load pre-trained weights into the model and avoid using random weights, as it helps the network to converge. These weights, even if the dataset is entirely different, help the lower level feature maps "learn" better and faster.

4.3 Preprocessing and Training

The architectures were trained using fine-tuned parameters and for each one we calculated the mean average precision (mAP) and the FPS (Frame per Second). From the FPS measurements we can infer if an algorithm can be used in real-time applications.

After extracting the frames, we randomly selected 80% of the data for training and 20% for validation. As a result, 1436 images were considered as training and 360 were considered as validation.

The faster RCNN framework was written using Keras (a high-level API running on top of tensorflow). Before starting, we opened the .mat files and saved each bounding box annotation in a corresponding text file for training/validation. Every row is a different bounding box in an image (an image can have many bounding boxes) and has the image name, class name and 4 coordinates. Later, we loaded weights of a VGG16 trained on ImageNet.

In order to work with YOLOv3, we used the Darknet API, written by YOLO's creator Joseph Redmon [21]. A large folder containing the images and annotation files was created, in which each annotation file is a text file that corresponds to one image in the dataset. In each file we wrote a list of the object bounding boxes: class names and coordinates. Then, created two text files: train.txt and test.txt each containing a list of images (from the 80-20 split).

A common deep-learning metric is epochs, which is the number of the full dataset passes through the network. We chose to train for 20 epochs, because our training dataset consist of only 1436 images. Iterations are basically epochs multiplied by the steps-per-epoch (1436 steps).

The hardware we used was a 4-core Intel CPU, 16GB of RAM and a Tesla K80 GPU, running on an Ubuntu 16.04 VM in Google's GCP.

5 Results

We evaluate the models using mAP with the VOC 2007 format, as described in the Theory chapter. In addition, we measure the time the algorithm takes to predict bounding boxes on a single image, and from that we get the FPS. Each network is trained for 28,720 epochs. The IoU threshold is 0.5, meaning each detection over 0.5 confidence is considered.

Architecture	% mAP	FPS
Faster RCNN	89.5	2
YOLOv3	79.2	25

Table 1: Results comparison

Presented here are the predictions of YOLOv3 model on 3 examples of the mini test set images taken in the port of Haifa- the videos that were not included in the dataset.

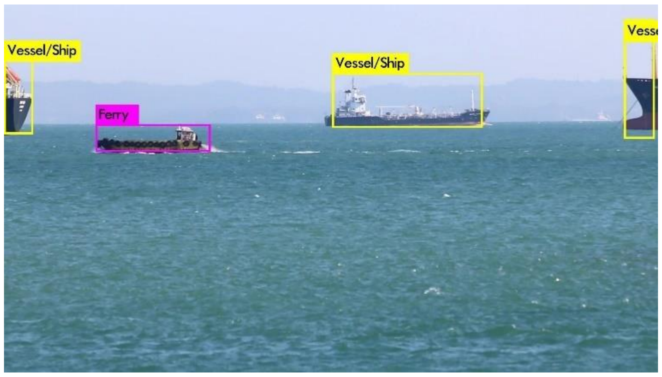


Figure 4: detection result from video 1, the framework detected 2 types of ships.

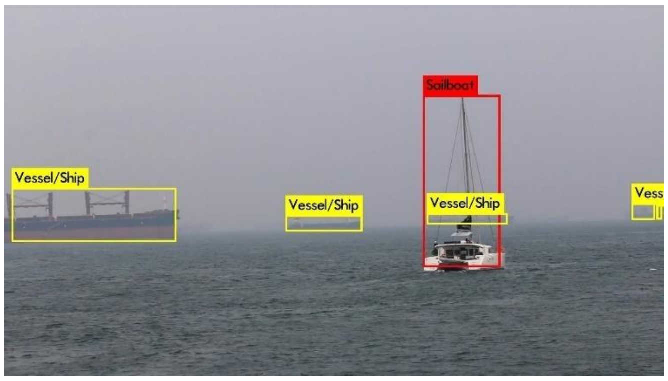


Figure 5: detection results from video 2, the framework again discriminated correctly between the 2 types of ships.

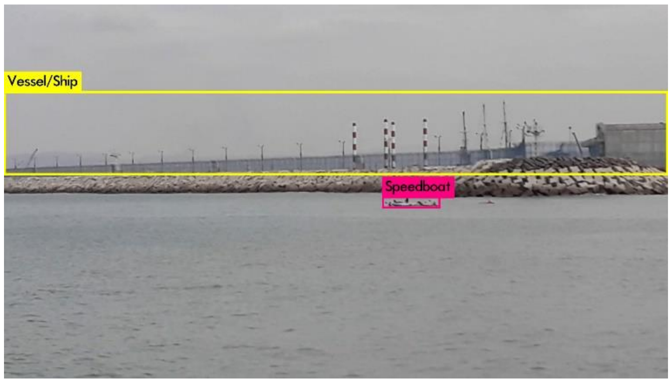


Figure 6: detection results from video 3, where a small speedboat is detected, but the environment around it is classified incorrectly.

We see that the framework can detect large ships and classify them by type. Almost all the ships we can see are detected, except the far away ones. We tested the results with small ships.

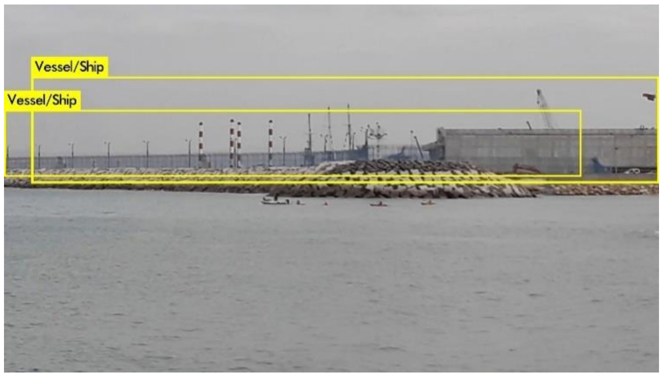


Figure 7: another detection results from video 3, the small speedboat is not detected.

In this example, the algorithm detects false alarms and can not find the speedboat. Small boats are detected sometimes but not in a reliable manner, because of the high-resolution videos and the lack of data for this category in the dataset.

6 Conclusions

Using Deep Learning in object detection yields good results when aiming for the detection of large vessels. Our system can be implemented on a real-time video feed using the YOLOv3 framework, however for it to be faster (30+ FPS) we need a GPU with more V-RAM (GPU memory) or 2/4 GPUs running in parallel. We showed that Faster RCNN has the highest accuracy, while being the slowest, as we expected.

Recommendations for future work:
1. In order to improve the results on smaller ships, we must add more data and thus represent this category better in the dataset.
2. Use a stronger GPU, or several GPUs, to improve the FPS on a real-time video feed.

References

[1] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 580-587.

[2] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017.

[3] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” Computer Vision – ECCV 2016 Lecture Notes in Computer Science, pp. 21–37, 2016.

[5] M. Selvi and S. S. Kumar, "A Novel Approach for Ship Recognition using Shape and Texture," International Journal of Advanced Information Technology (IJAIT), vol. 1, no. 5, 2011.

[6] A. Kadyrov, H. Yu and H. Liu, "Ship Detection and Segmentation Using Image Correlation," 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, 2013, pp. 3119-3126.

[7] W. Krüger and Z. Orlov, "Robust layer-based boat detection and multi-target-tracking in maritime environments," 2010 International WaterSide Security Conference, Carrara, 2010, pp. 1-7.

[8] V. Marie, I. Bechar and F. Bouchara, "Towards Maritime Videosurveillance Using 4K Video," in Lecture Notes in Computer Science, 2018, pp. 123-133.

[9] X. Yang et al., "Automatic Ship detection of remote sensing images from google images in complex scenes based on multi scale rotation dense feature pyramid networks," Remote Sensing, vol. 10, no. 1, p. 132, 2018.

[10] M. Leclerc, R. Tharmarasa, M. C. Florea, A. Boury-Brisset, T. Kirubarajan and N. Duclos-Hindié, "Ship Classification Using Deep Learning Techniques for Maritime Target Tracking," 2018 21st International Conference on Information Fusion (FUSION), Cambridge, 2018, pp. 737-744.

[11] K. Kim et al., "Probabilistic Ship Detection and Classification Using Deep Learning," Applied Sciences, vol. 8, no. 6, 2018.

[12] M. Zwemer, R. G. J. Wijnhoven, P. H. N. de With, ”Ship detection in harbour surveillance based on large-Scale data and CNNs,” VISIGRAPP 2018. Vol. 5, 2018. pp. 153-160.

[13] B. Hummel, S. Kammel, Thao Dang, C. Duchow and C. Stiller, "Vision-based path-planning in unstructured environments," 2006 IEEE Intelligent Vehicles Symposium, Tokyo, 2006, pp. 176-181.

[14] Y. Zhu et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 3357-3364.

[15] R. M. Kangutkar, "Obstacle Avoidance and Path Planning for Smart Indoor Agents," M.S. thesis. Rochester Institute of Technology, 2017. Available: RIT Scholar Works.

[16] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6517-6525.

[17] J. Redmon, A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767, 2018.

[18] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448.

[19] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally and C. Quek, "Video Processing From Electro-Optical Sensors for Object Detection and Tracking in a Maritime Environment: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 8, pp. 1993-2016, Aug. 2017.

[20] [21] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556, 2014.

[21] J. Redmon, "Darknet Neural Network Framework," [Online]. Available: https://pjreddie.com/darknet/yolo/.