Deep Learning-Based Real-Time Object Detection in Inland Navigation

Metzli Ramirez-Martinez, Wided Hammedi, Philippe Brunet, Sidi-Mohamed Senouci, Mohamed-Ayoub Messous

Deep Learning-Based Real-time Object Detection in Inland Navigation

Wided HAMMEDI, Metzli RAMIREZ-MARTINEZ, Philippe BRUNET,
Sidi Mohammed SENOUCI, and Mohamed Ayoub MESSOUS
DRIVE Laboratory EA1859, Univ. Bourgogne Franche Comté, F58000 Nevers, France
{wided.hammedi,metzli.ramirez-martinez,philippe.brunet,sidi-mohammed.senouci,mohamed-ayoub.messous}@u-bourgogne.fr

Abstract—Semi-autonomous and fully-autonomous systems must have knowledge about the objects in their environment to ensure a safe navigation. Modern approaches implement deep learning techniques to train a neural network for object detection. This project will study the effectiveness of using several promising algorithms such as Faster R-CNN, SSD, and different versions of YOLO, to detect, classify, and track objects in near real-time fluvial domain. Since no dataset is available for this purpose in literature, we first started by annotating a dataset of 2488 images with almost 35 400 annotations for training the convolutional neural network architectures. We made this data set openly accessible for the community working on this area. The other contribution of this research is the adaptation and the configuration of deep learning techniques used in other domains such as maritime and road domain to fluvial domain for autonomous vessels in which high accuracy and fast processing are vital. Experiments demonstrated that detecting objects in such environment is plausible in near real time with the selected algorithms.

Index Terms—Real time, Object detection, Intelligent vehicles, Inland waterway vessels, Deep learning.

I. Introduction

Inland water transport is a viable alternative to road and rail transport on European corridors. It is highly competitive with respect to other modes of transportation, environmentally-friendly, reliable, safe and could provide lower transportation costs when moving large volumes of bulk cargo. However, the specificities of the waterways bring as many opportunities as new challenges for the automation of the transport vessels. Autonomous vehicles have made significant inroads into the area of transportation, but specifically in the terrestrial and marine environments. Thus, the idea in this work to study the performances of the methods used in these latters in inland environment in order to ensure a safe navigation of an autonomous vessels.

Specially, the object detection module of autonomous vessels plays an essential role in safe maritime navigation. The vessel needs to detect and avoid other nearby vessels and infrastructure. The current research trend is employing deep learning algorithms to learn useful features instead of hand-designing them. In fact, since the rise of Convolutional Neural Networks (CNNs) within the ImageNet [9] challenge, they have gotten to be the foremost prevalent arrangement for com-mon question acknowledgment issues such as classification, localization, and location. They accomplished lower blunder rates than the past state-of-the-art results. For this reason, CNNs have been widely adopted in object recognition systems in almost all domains in [2], [5] and [14].

The objective of the current work is to compare and identify the most appropriate algorithm for object detection in a fluvial environment in terms of accuracy, run-time and resources consumption. The novelties of this paper, compared to the state of art, consist of:

- We produce a unique dataset for training deep visual detection models for object detection in inland environment, since there is no public dataset is available,
- We evaluate the accuracy and performance of six state-of-the-art object detection algorithms for the problem of object detection in inland environment.

The rest of this paper is structured as follows. Section II gives an overview on the object detection problem and existing public datasets. In Section III, we detail the characteristics of our proposed dataset. Then, we describe the training process implementation in Section IV. Finally, discussion on results and performance evaluation are drawn in Section V and Section VI.

II. Related Works

In this section, object detection models, datasets and areas of application are reviewed. We mainly focus on the suitability of integrating the existing approaches to our problematic.

1) Object Detection: Computer vision has been especially a curiously intrigue field in later a long time since self-driving vehicles have taken centre stage. Generic object detection aims at locating and classifying existing objects in any one image and labeling them with rectangular bounding boxes to show the confidences of existence. Nowadays, the performance of object detection has been improved with the Deep CNNs methods which have recently come to dominate object recognition re-search due to their excellent performance on many challenging datasets harvested from the web, such as ImageNet [9]. They achieved top-1 and top-5 error rates of 37.5% and 17.0%, respectively, which is impressively superior than the past state-of-the-art [9]. Among these methods, Faster R-CNN, SSD and YOLO show a better performance of detecting objects with different sizes in other application domains such as recognizing the types of marine vessels in sail [2], detecting people, cars, and roads [5], [13] and detecting marine litter [14].

2) Public datasets: One of the biggest problems of artificial intelligence, in general and image processing, is having a good dataset that properly relates to the problem. Besides that, the dataset has to be processed in a way that the model can make sense of the information. That way the model can successfully learn from that dataset. Scientific datasets are, at least, intermediate results in many scientific research projects. For some time, datasets were not even published and even if they were published it was mostly as a not re-usable by-product of the publication.
In maritime environment, we find public datasets VAIS (*Visible and Infrared Spectrums*) and Marvel [10], used essentially for classification in [18], [21] and [6]. Details of the VAIS dataset, the most used dataset, are collected in Table I.

TABLE I: VAIS dataset.

Total images	2865
IR Images	1242
Visible Images	1623
Night IR Images	154
Number of classes	6

Among popular datasets used for Image detection, we men-tion Pascal Visual Object Classes (Pascal VOC) [17] dataset which contains 17 125 annotated images with 20 classes at the time of writing the article. This dataset was created within a challenge in visual object recognition funded by PASCAL network of excellence and then used to test and evaluate the performance of different models.
However, there is not a public dataset annotated destined for fluvial environment with its different challenges that we detail below.

3) Object Detection in Maritime Environments: Target detection in maritime environments has a place to the investigate points that pick up small consideration in the field of computer vision. Some applications exist such as collision avoidance for autonomously operating vessels [3] or object detection in general in [12] and [1].However, these results can not be adapted to the fluvial environment since the two environments are different as we will detail in the following section.

III. Dataset and Ground Truth Labelling Process

The characteristics of the fluvial environment are different from those of the maritime environment. There are differences in natural features (rocks, waterfalls, rapids, etc). Seas are wider than rivers. So, the canal size is smaller that the waterway in the maritime environment. In addition, there are other differences related to the infrastructure. In fact, the river ships are smaller, the itineraries are more port-intensive and the ships are closer to the land. In addition, locks and bridges do not exist in the maritime environment and limit of the canal is not marked by a physical infrastructure, but generally with trees. Consequently, rules of inland navigation differ from the rules of maritime navigation.

So, to evaluate the performances of Neural Networks, we need a specific dataset that meets these requirements.

Since no public dataset exists as described above, we make our own dataset[1] following a data model we propose and provide annotated images in two formats, as detailed below.

A. Data Model

The objects we aim to detect are divided in five categories:

Riverside: As there is no lane markings in rivers, specifying the exact location of a lane marking in a camera image may be very ambiguous. So, in this category, we try to get as close as safely possible to the riverside,

Vessel: In this category, we added different types of vessels that can be found in a river,

Person: We added this category to detect fishermen or other persons on the river's waters or persons in the riverside, since ships could be very close to the riverside,

Infrastructure: In this category, we find two types of infrastructure that are locks and bridges,

Road signals: As for cars as for boats, vehicles follow the traffic code by respecting the road signals. So, in this category we collect road signals we can find in real situations.

Other versions of this data model were tested: one including a label for waves (rejected due to lack of data of this class and in consequence they present a lower accuracy) and models with multiple classes for different infrastructure and ships (rejected for lack of data on some classes and lower detection performance overall). Both versions achieved lower accuracy than this model on all networks. Examples of images annotated to have class-labels and bounding boxes from this data model can be seen in Fig. 1.

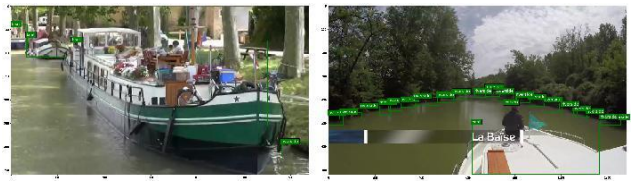

Fig. 1: A sample from the training dataset are shown. Multiple rectangular bounding boxes are drawn to delimit objects with corresponding corresponding class labels.

Although following a single object is the most common object-following scenario, detecting multiple objects is neces-sary for our application. As shown in Fig. 1, we add multi-object detection capabilities by drawing multiple bounding boxes for multiple objects in the same image.

B. Dataset Construction

Our dataset is composed of 2 488 images. The number of annotations for each class is detailed in the Table II. In order to get close to real conditions, we selected images with different backgrounds, camera positions, weather conditions and day time. This allows us to create a dataset for training which

[1]The dataset and trained models will be made available for academic research purposes

TABLE II: Proposed dataset.

Total images	2488
Riverside	28084
Vessels	3496
Persons	1545
Infrastructure	2051
Road signals	193
Number of classes	5

closely conforms to real-world conditions, unlike previous works, which mostly rely on internally generated datasets with simulation software.

C. Annotation

We annotated our dataset using the LabelImg. This open source graphical image annotation tool can be downloaded on GitHub [2]. It was selected because it saved the annotation files to the PASCAL VOC format XML and annotation files are saved separately for each image in the source folder. So, it will be easier for future users of this dataset to reuse the annotations.
Unfortunately, right now there seems to be no standard on how to represent this information. In other words, each of the object detection systems expect a different format to represent the bounding box and class of each object:

Darknet uses a text file, with the following format:

[category number] [object center in X] [object center in Y] [object width in X] [object width in Y]

Darkflow expects the annotations in the same format of the PASCAL VOC dataset in xml files with the following format:

```
<width>width</width>
<height>height</height>
<depth>depth</depth>
<bndbox>
<xmin>xmin</xmin>
<xmax>xmax</xmax>
<ymax>ymax</ymax>
</bndbox>
```

Then we change XML files to text files using these equations:

* Object center in $X = \frac{(xmin+xmax)}{2 \cdot width}$
* Object center in $Y = \frac{(ymin+ymax)}{2 \cdot height}$
* Object width in $X = \frac{(xmax - xmin)}{2 \cdot width}$
* Object width in $Y = \frac{(ymax - ymin)}{2 \cdot height}$

So, as we explained above, we build an annotated dataset where each image has two separate files (text file and XML file) in the two formats.

IV. Networks Configuration and Training

This section explains the neural network architectures se-lected, the pre-processing alterations done and the training process suggested.

[2] https://github.com/tzutalin/labelImg

A. Network Architectures

We choose for this project some network architectures from the most popular and successful object detection networks used today. Each one has its own benefits and drawbacks, with varying levels of accuracy and run-time speeds.
In this section, we will briefly discuss their methodologies and the related design choices.

1) Faster R-CNN: It is a project created by Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun [8]. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, Faster R-CNN is a very accurate region-based deep detection model which improves Fast R-CNN by introducing the Region Proposal Networks. The predicted region proposals are then reshaped using a Region of interest (ROI) pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

2) Single Shot Multibox Detector (SSD): It is a popular algorithm in object detection [15]. It's generally faster than Faster RCNN. It uses some middle layers, called feature maps,
in the network to detect objects of different sizes instead of predicting using the last layer, as done in traditional networks. Single deep learning network can significantly reduce computation time and improve inference accuracy.

3) Four versions of YOLO and Tiny-YOLO: The You only look once (YOLO), first proposed in [22], is an object detection system targeted for real-time processing, which performs the region proposal and classification in one pass over the input image. It leads to a significant speed-up when compared to two-stage methods like Faster R-CNN. Two updates to the original YOLO method are described in [20], [21] with some modifications to increase the detection precision and speed.
In this project, we use the improved versions of the models which are Yolo v2, Tiny-Yolo v2 [20], Yolo v3 and Tiny-Yolo v3 [21].

B. Training Process

To train all the supervised deep models, we use an NVIDIA GeForce GTX 2080 Ti GPU with 11GB of GPU memory on a Linux machine (ubuntu 18.04). In addition we used CUDA Deep Neural Network library (cuDNN) which is a GPU-accelerated library created by NVIDIA to improve deep neural network performance. CuDNN provides optimized implementations for common routines such as forward and backward convolution, pooling, normalization, and activation layers. TensorFlow [16] and Darknet [23] libraries are used for implementation.
Collecting labeled data is expensive. However, training a large neural network on a relatively small dataset, such as our dataset, would lead to over-fitting. So, to overcome this issue, we applied two methods:

Data augmentation: This technique has been shown to produce promising ways to increase the accuracy of detection tasks. We made some alterations to our existing dataset with rotation (angle=30 and angle=45) in order to increase the amount of training data. Neural network models consider additional as distinct images without concerning about the orientation.

Fine tuning: This technique means taking weights of a trained neural network and use it as initialization for a new model being trained on data from the same domain. It is used to both speed up the training and overcome small dataset size. In our implementations, we initialize our models with weights of the same trained models on Pascal VOC dataset.

Since the objects we aim to detect are different in term of size and shape, we calculated correctly tuned anchor boxes [11] to improve the detection of irregular objects. We stop the training when the loss no longer decreases.

V. Evaluation Criteria

We evaluate the different networks mentioned above using the following standard metrics:

mAP (mean Average Precision): It is the first popular metric used for evaluating detection algorithms. Average precision computes the average precision value for recall value over 0 to 100. The mAP metric is the product of precision and recall of the detected bounding boxes. A higher value in mAP indicates a better performance. The mAP can be computed by calculating average pre-cision (AP) separately for each class, then calculating the average over the class. A detection is considered a true positive only if the Intersection over Union (IoU) is above 0.5. AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, using the increase in recall from the previous threshold as the weight:

$$AP=\sum_{n}(R_n-R_{n-1})\,P_n,$$

where P_n and R_n are the precision and recall at the n^{th} threshold.

IoU (Intersection over Union): It is the second popular metric used for evaluating detection algorithms. It is given by the ratio of the area of intersection and the area of union of the predicted bounding box and the ground truth bounding box.

$$IoU=\frac{AreaofOverlap}{AreaofUnion}$$

These two metrics concisely describe the accuracy and the quality of object detections. Adding to that, we measure the runtime speed of the algorithms using a third metric.

FPS (Frame per Second): It is the third important metric used for evaluating detection algorithms. It measures the number of frames processed by second.

VI. Performance Evaluation and Results Analysis

Detection results are obtained on a test set, with images that do not appear in the train set. Our test set contains examples of every object mentioned in the data model with different background and camera position, to provide a realistic evalu-ation.

A. Metrics

To evaluate the performance of the models in terms of accuracy, we use the mAP. We set the IoU threshold to 0.5, as suggested in [4]. If the classification of prediction matches the ground truth and IoU meets the threshold, the detection is considered as a correct prediction. Morevoer, to evaluate the performance of the models in terms of resource consumption, we use the FPS.

In Table III and Table IV, we illustrate the results obtained on the test set of the different models in terms of accuracy (mAP) and runtime speed (FPS). As we can see, all the six network architectures have similar behavior in detecting overall classes; the best predicted object is the vessel and the worst one is the riverside class. In Table III, we notice that YOLO and Tiny-YOLO have lower mAP when compared to Faster R-CNN and SSD. Conversely, Faster R-CNN and SSD have higher processing times, as shown in Table IV.

In one hand, the Faster R-CNN model achieves much better detection performances compared to the other models although it is the slowest in terms of running time. In the other hand, YOLO v3 and SSD provide comparable detection perfor-mances. Although Tiny-YOLO provides fast running time, its detection performance is not as good as the other models.

TABLE III: Detection metrics in mAP and AP.

Model	mAP	*Person*	*Riverside*	*Vessel*	*Infrastructure*	*Road signals*
YOLO v3	60.36	64.87	27.53	80.16	68.08	61.14
Tiny-YOLO v3	45.42	45.24	12.69	68.65	52.36	48.19
YOLO v2	43.65	43.84	13.33	65.36	38.59	54.13
Tiny-YOLO v2	40.95	41.89	14.07	60.98	43.61	44.20
Faster R-CNN	72.14	76.2	49.67	92.96	76.35	65.52
SSD300	50.31	62.63	29.65	56.9	56.15	46.21

A model is said to be acting is real-time if its time response is less than the time response of a vessel. Then, to confirm whether the models are acting in real time conditions, we need to calculate the time a vessel make to stop, in real conditions.

The navigation authority responsible for the management of the inland waterways network [19] indicates that "*unless otherwise specified, signposts indicate limited speed on canals at 6 or 8 km/h and between 10 and 15 km/h in rivers*". The type of ship, dimensions, mass and velocity are the parameters of greatest importance for the stopping distance and time of the ship. So, to calculate the minimum value of the stopping time (t_{min}), we should consider the maximum value of stopping distance (d_{max}) [7].

d_{max}= ($load_{max}$ * $speed_{max}^2$/ (2* brake force) = 5.9 meters,

$$t^{min} = \frac{d^{max}}{vessels', sspeed} = 1.41 \text{ seconds}$$

Hence, the results demonstrate that all the models, except the Faster R-CNN, validate their applicability in real-time object detection applications for inland navigation.

TABLE IV: Detection metrics in FPS.

Model	FPS
YOLO v3	96
Tiny-YOLO v3	240
YOLO v2	120
Tiny-YOLO v2	160
Faster R-CNN	0.48
SSD300	4.54

B. Prediction Results

In this section, we draw the predictions of the six models for the same two images. Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9 and Fig. 10 show the predictions of Faster R-CNN model, YOLO v3 model, Tiny-YOLO v3 model, YOLO v2 model, Tiny-YOLO v2 and SSD model, respectively.

The result images reveal a correlation between detections in images and values in the Table III below. It is noticeable that all the models can detect almost all the classes with error percentages that differ from model to model and from image to image. By analyzing these results, we consider that Faster R-CNN is the best method from a standpoint of accuracy, but it requires much more resources than other models in terms of run-time speed. Tiny-YOLO v3 is the fastest method and Yolo v3 has the ideal balance of accuracy and run-time speed. Hence, we confirm that despite the differences, the selected state-of-art models perform as well as in other domain applications since we have achieved similar results.

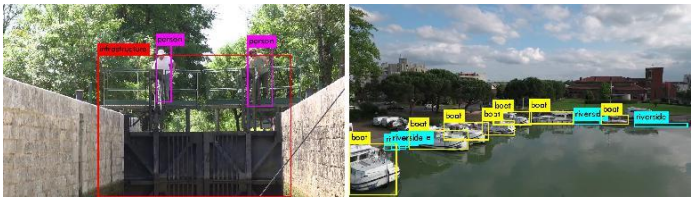

Fig. 2: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by YOLO v3 at test time.

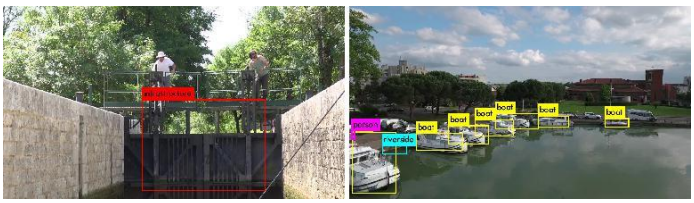

Fig. 3: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by Tiny-YOLO v3 at test time.

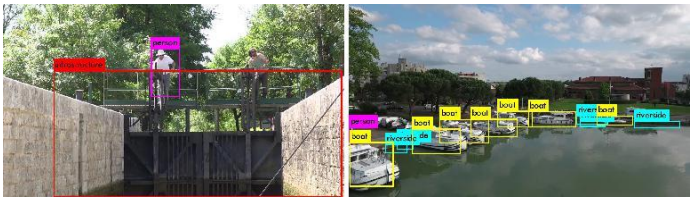

Fig. 4: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by YOLO v2 at test time.

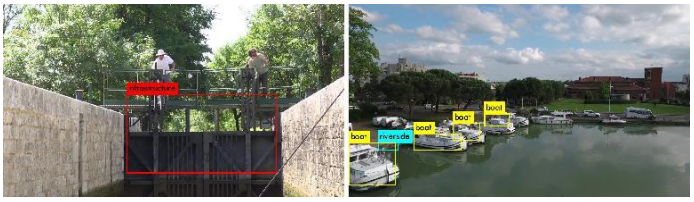

Fig. 5: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by Tiny-YOLO v2 at test time.

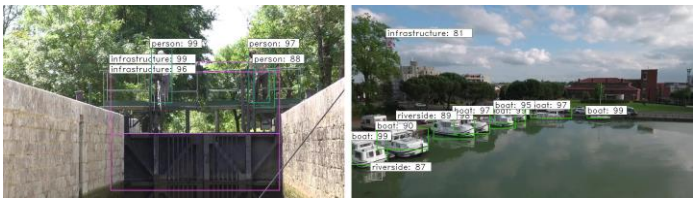

Fig. 6: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by Faster-RCNN at test time.

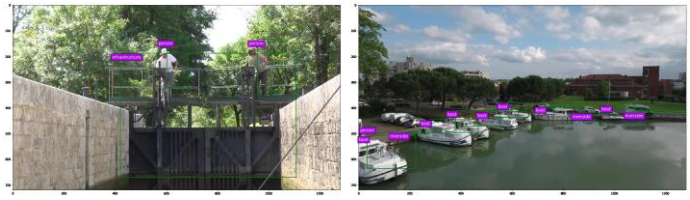

Fig. 7: A sample of predicted images where the rectangles and text overlaid on the figures are the outputs generated by SSD at test time.

C. Detection of Particular Objects: riverside

The six models show encouraging performance in general object detection. However, it still cannot identify and precisely localize particular objects, such as riverside in our case. As we notice in the Table III, the riverside class has the highest error values. In fact, this is due to the complexity of the class in terms of variations of its appearances, variations of the type of river, variations of the day time, for instance illumination appears to be totally different in the night, the presence of shadows of trees and vessels. Such situations are difficult even with human intervention.

In Fig. 8, we evaluate the detections predicted by the Tiny-

YOLO v3 model, which has the worst performance in de-tecting the riverside class, on an image with only this class. We find some bounding boxes along the river, showing the edges. So, we conclude that even with this model, we can find useful results. Besides the results shown in Fig. 7, we confirm that these bounding boxes can be sufficient for the purpose of autonomous vessels, since it can detect almost all the near objects.

Fig. 8: Detections generated by Tiny-YOLO v3 on a particular image with a complicate environment (shadow of trees).

VII. Conclusion and Discussion

Real-time object detection plays a crucial role in autonomous vehicles as the information obtained using this concept can be used to direct the vehicle along the safest possible path by avoiding obstacles.

In this work, we addressed the challenges involved in real-time inland navigation. We exploited the existing deep convolutional neural network architectures, specifically, we tested Faster RCNN, YOLO v3, Tiny-YOLO v3, YOLO v2, Tiny-YOLO v2 and SSD to solve this problem. To accrately represent the specific needs of fluvial environment and since no dataset is available in litterature, we made a dataset of 2488 images with openly accessible for the community working on this area. We annotated it with a data model that contains five classes: vessel, person, riverside, road signals and infrastructure. The first comparative results demonstrate the effectiveness of these models in the fluvial domain.

In the future, in order to improve the performance of the detection, we consider increasing the size of the dataset by adding new annotated images and videos. In addition, we will apply a mathematical method to mark the edge of the river by a shape that accurately maps the navigation area.

Acknowledgment

This work is achieved as part of a project partially funded by BFC region (Bourgogne-Franche-Comté).

References

[1] S. Moosbauer, D. Koenig, J. Jaekel and M. Teutsch. "A Benchmark for Deep Learning Based Object Detection in Maritime Environments". In IEEE PBVS (June 2019)

[2] Aiswarya S Kumar and Elizabeth Sherly, “A convolutional neural network for visual object recognition in marine sector” in 2nd International Conference for Convergence in Technology (I2CT), 2017

[3] T. Statheros, G. Howells, and Klaus McDonald Maier. "Autonomous ship collision avoidance navigation concepts, technologies and techniques". in The Journal of Navigation, 61(1):129–142, 2008.

[4] Cai, Z., Vasconcelos, N. “Cascade r-cnn: Delving into high quality object detection.“ In: IEEE CVPR (June 2018)

[5] Yu-Ho Tseng and Shau-Shiun Jan, ” Combination of Computer Vision Detection and Segmentation for Autonomous Driving” , in IEEE/ION Position, Location and Navigation Symposium (PLANS), 2018.

[6] Atmane Khellal, Hongbin Ma and Qing Fei “Convolutional Neural Network Based on Extreme Learning Machine for Maritime Ships Recognition in Infrared Images” in mdpi sensors (2018).

[7] Jean Sommet and François Parthiot ”Distances d’arrêl des grands navires en canal” in La Houille Blanche, 2009

[8] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” NIPS, 2015.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database”, in IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[10] E. Gundogdu, B. Solmaz, V Yucesoy, and A. Koc. "Marvel: A large-scale image dataset for maritime vessels". In ACCV, 2016

[11] Tong Yang, Xiangyu Zhang, Zeming Li, Wenqiang Zhang and Jian Sun, "MetaAnchor: Learning to Detect Objects with Customized Anchors", in NIPS, 2018.

[12] Dilip K. Prasad1, C. Krishna Prasath, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai Quek, "Object Detection in a Maritime Environment: Performance Evaluation of Background Subtraction Methods" in IEEE Transactions on Intelligent Transportation Systems (July 2018)

[13] Petru Soviany, Radu Tudor Ionescu "Optimizing the Trade-off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction" In: Proceedings of SYNASC, pp. 1–6, (2018)

[14] Michael Fulton, Jungseok Hong, Md Jahidul Islam, Junaed Sattar, “Robotic detection of marine litter using deep visual detection models” , in IROS, 2018.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. "SSD: single shot multibox detector". CoRR, abs/1512.02325,2015.

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.

[17] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. “The Pascal visual object classes challenge: A retrospective “. IJCV, 111(1):98–136, 2015.

[18] Mabel M. Zhang, Jean Choi, Kostas Daniilidis, Michael T. Wolf and Christopher Kanan, “VAIS: A dataset for recognizing maritime imagery in the visible and infrared spectrums,” in IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015

[19] Voies navigables de France, "http://www.vnf.fr/vnf/", Accessed: 05-04-2019.

[20] J. Redmon and A. Farhadi. YOLO9000: Better, faster,stronger. In CVPR, 2017

[21] J. Redmon and A. Farhadi. "Yolov3: An incremental improvement". Technical report, CoRR, abs/1804.02767, 2018.

[22] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: "You only look once: unified, real-time object detection". In: CVPR (2016)

[23] J. Redmon and A. Farhadi, "yolo." https://pjreddie.com/darknet/yolo/, 2017. Accessed: 04-04-2019.