Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility

Zhihao Chen, Redouane Khemmar, Benoit Decoux, Amphani Atahouet, Jean-Yves Ertaud

Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility

1st Zhihao Chen
UNIRouen, Normandy University. ESIGELEC/IRSEEM
Saint Etienne du Rouvray, France
zhihao.chen@groupe-esigelec.org

2nd Redouane Khemmar
UNIRouen, Normandy University ESIGELEC/IRSEEM
Saint Etienne du Rouvray, France
nicolas.ragot@esigelec.fr

3rd Benoit Decoux
UNIRouen, Normandy University ESIGELEC/IRSEEM
Saint Etienne du Rouvray, France
benoit.decoux@esigelec.fr

4th Amphani Atahouet
UNIRouen, Normandy University ESIGELEC/IRSEEM
Saint Etienne du Rouvray, France
surduf.atahouet@groupe-esigelec.org

5th Jean-Yves Ertaud
UNIRouen, Normandy University ESIGELEC/IRSEEM
Saint Etienne du Rouvray, France
jean-yves.ertaud@esigelec.fr

***Abstract*—In this paper, we will introduce our object detection, localization and tracking system for smart mobility applications like traffic road and railway environment. Firstly, an object detection and tracking approach was firstly carried out within two deep learning approaches: You Only Look Once (YOLO) V3 and Single Shot Detector (SSD). A comparison between the two methods allows us to identify their applicability in the traffic environment. Both the performances in road and in railway environments were evaluated. Secondly, object distance estimation based on Monodepth algorithm was developed. This model is trained on stereo images dataset but its inference uses monocular images. As the output data, we have a disparity map that we combine with the output of object detection. To validate our approach, we have tested two models with different backbones including VGG and ResNet used with two datasets : Cityscape and KITTI. As the last step of our approach, we have developed a new method-based SSD to analyse the behavior of pedestrian and vehicle by tracking their movements even in case of no detection on some images of a sequence. We have developed an algorithm based on the coordinates of the output bounding boxes of the SSD algorithm. The objective is to determine if the trajectory of a pedestrian or vehicle can lead to a dangerous situations. The whole of development is tested in real vehicle traffic conditions in Rouen city center, and with videos taken by embedded cameras along the Rouen tramway.**

***Index Terms*—Pedestrian Detection, Pattern Recognition, Object Detection, Tracking, YOLO V3, SSD, Deep Learning.**

I. Introduction

The work presented in this paper is a part of ADAPT[1] project (Assistive Devices for empowering disAbled People through robotic Technologies) which focuses on smart and connected wheelchair to compensate for user disabilities through driving assistance technologies. The work presented in this paper is based on pedestrian detection is part of ADAPT project. In general, ADAS is used to improve safety and comfort in vehicles. ADAS is based on the combination of sensors (RADAR, LIDAR, cameras, etc.) and algorithms that ensure vehicle, driver, passenger and pedestrian safety based on different parameters such as traffic, weather etc. Here in this project, ADAS aims to detect object like pedestrian, vehicles, etc. Our contribution aims on the development of perception system based object detection-based deep learning with different approaches such as YOLO V3, and SSD.

The principal objective is to apply robust approaches to detect object and avoid traffic accidents on real time system. The detection of object and estimation of its distance are the most important tasks to determinate the object's position. To carry out these tasks, we had, in old research work, evaluated traditional image processing approaches like HOG and DPM approaches. To get a more robust result, we have applied deep learning based object detection and distance estimation approaches, and then combined them together to get the final result. Results show that the object are detected with high accuracy and is considered as satisfied result. Furthermore, the object distance estimation is carried out with some errors which need to be improved. Our training platform is a computing cluster with Nvidia P100 and K80 GPUs (Graphical Processing Units), and inference platform is portable computer with Nvidia GTX960m, which has about 1.5 TFLOPS of calculation capacity.

This paper is organized as follows: Section 1 introduces the motivation of the paper. Section 2 presents the state of the art about object/pedestrian detection based deep learning. Section

[1]This work is carried out as part of the INTERREG VA FMA ADAPT project "Assistive Devices for empowering disAbled People through robotic Technologies" http://adapt-project.com/index.php. The Interreg FCE Programme is a European Territorial Cooperation programme that aims to fund high quality cooperation projects in the Channel border region between France and England. The Programme is funded by the European Regional Development Fund (ERDF)

3 presents the object detection and tracking based on YOLO V3 and SSD deep learning algorithms. Section 4 illustrates the object distance estimation approach based on deep learning. The pedestrian behavior analysis will be presented in Section 5. Results are presented through the different section 3, 4, and 5. Finally, in Section 6, we will conclude this paper.

II. Related Work

Object detection is a key problem in computer vision. It has two big challenges: detection of objects in images and estimation of their position, and estimation of their class. Over the past few years, several methods based on Convolutional Neural Network (CNN) have been proposed to tackle this problem, with great success. Those methods can be divided into two main categories : one-stage methods, which provide estimation of position and estimation of classes in one step, and two-stage methods, which first detect regions of the images where object could be present, and then apply these regions to a classifier.

Two of the most popular methods in the one-stage category are Single-Shot Detector (SSD) [1] and You Only Look Once (YOLO) [2]. They both provide as outputs the probability of each possible classes, not for the whole image but for a set of regularly spaced positions and for different scales and aspect ratio of rectangles called boxes. One of the two-stage methods giving the best results is RCNN (Region-proposal Convolutional Neural Network) [3] and its improved versions [4] [5], base on two independent neural networks: a region-proposal network and a classification network. Those three methods have similar performance from the point of view of the mean Average Precision (mAP), a criterion which quantifies quality of detections (proportion of correct detections) as a function of the recall, which is the proportion of objects that are detected.

However, two independent networks make the prediction slow on the most embedded computation platforms. In one-stage detection architectures, the classification is made on fixed size and fixed number of bounding boxes at predefined layers, then tune the localization of detected objects by regression. Those architectures are generally faster than two-stage ones, with similar performance [1].

Considering that our application is based on sequential media (video for example), loss of too small objects and imprecision on position are less critical than the real-time performance. Some trade-off like increasing number of bounding box and resolution of inference also could be adopted to reduce it. Based on the above analysis, we choose two widely applied one-stage object detection approaches: SSD [1] and YOLO V3 [2] to evaluate.

For estimation of distance from objects to the vehicle, many sensors are available : laser, ultrasonic, infrared ray, etc. Those kinds of sensors are widely used in civil field. But the use of multiple types of sensors make the system more complex, and more expensive. Since we plan to use a low cost monocular camera for object detection, it seems valuable to estimate the distance with the same sensor as for object detection, that is a monocular camera. If the datasets used for object detection had the distance from object as ground-truth information, information of distance could be directly learned by adding a regression output to the CNN, but it is not the case. One solution is to use an unsupervised approach to estimate depth for monocular images, like the one called Monodepth [7]. This model is trained on stereo images but infers disparity maps from monocular images, so it is intersting for our needs.

III. Object Detection and Tracking based on Deep Learning

A. Choice of the Approach to be Developed

In order to analyze the performance of object-detection models, speed and accuracy are important parameters to consider. Comparison of the performance of different approaches must be done carefully because the experiments are not always done under the same conditions. Indeed, there are several parameters that can vary from one experience to another. This is for example the learning database which is used, the resolution of the input image or the threshold of the Intersection-over-Union criterion (IoU, which is used to evaluate the quality of the detections) etc. We have analyzed the properties of three well known approaches for object detection : SSD [1] YOLO V3 [2], and Faster-RCNN [6]. Faster RCNN seems to be more accurate than SSD and YOLO V3. On the other hand, it is slower. Which means that if the task requires high quality precision, RCNN Faster is the right solution. On the other hand, if it is the speed that is essential, the RCNN Faster is not the best candidate. YOLO V3 is faster than SSD and Faster RCNN. So, if speed is the main criterion of choice, YOLO V3 would to be the best candidate. If we want at the same time a good accuracy and a good speed, SSD could be a better solution, as fastest approach after YOLO V3, and its accuracy is almost as good as Faster-RCNN. So it represents a good compromise between accuracy and speed.

B. SSD vs YOLO V3

During the course, YOLO V3, the newer version of YOLO is released. According to the literature, it would have better performance than the SSD in terms of speed and accuracy. So we decided to make the inference of SSD and YOLO V3 on the same environment to compare them. Table III shows the evaluation environment used.

TABLE I
Inference Environment.

GPU	RAM GPU	RAM	Operating System	Computational Performance
GTX 960m	2G	8G	Ubuntu 16.04 64bit	1.5 TFLOPS

The model of the SSD that we evaluated was trained with the PascalVOC learning base. While the YOLO V3 model we evaluated was trained with the Coco learning base. We conducted the evaluation of all the classes of each learning base. This is the reason why instead of presenting the information of the AP(Average Precision) which is relative to the average accuracy of a class, we presented the information

of the mAP(mean Average Precision) which represents the average of the APs of all classes in the training base. Table **??** presents the results of the evaluation and the inference of YOLO V3 and SSD.

TABLE II
SSD AND YOLO V3 INFERENCE.

	mAP	fps
SSD	79.5	8.6
YOLO V3	85.0	11.2

Figure 8 present the results of the inference performed on the SSD and YOLO3.

Fig. 1. SSD inference (left) vs YOLO V3 inference (right).

According to the results of our tests, the YOLO V3 gets better performance. The YOLO V3 at the resolution 416*416 has the similar inference FPS as the SSD at the input resolution 300*300 but the precision is markedly higher. On the other word, with the parameter of the same precision, the YOLO V3 is quicker than SSD.

C. SSD Algorithm

We made inferences using the Keras and Tensorflow libraries. Below are the characteristics of the inferences made with the SSD approach.

- VGG (SSD, 300x300, 120000 iterations): model trained with an SSD architecture having a VGG base. Model taking in pictures of size $300x300$. This is the model on which the inference was made.
- SSD: name of the approach used to perform the inference
- PascalVOC: database dedicated to pedestrian traffic containing 20 classes
- nVIDIA GTX 660: graphics card used
- 2G: size of the ram used for the graphical calculation

Fig. 2. SSD Inference Result.

We obtain a processing time of 6.2fps and a confidence rate approaching 100 % in the majority of case.

The purpose of this method is to determine the performance of the model trained. This is the VGG (VOC0712, SSD 300x300, 120000 iterations), model presented above. There are two main parameters that come into play in this context. This is the accuracy and the rate of callback. The figure below [9] presents the operations to be applied in order to obtain these two parameters.

This evaluation allows us to know how much the model we use is accurate with respect to each class in the learning base on which it was trained. The model we used was trained on the PascalVOC learning base. But we did not evaluate the 20 classes of this learning base. We evaluated just six classes that we considered likely to end up in the tramway environment. These include the following classes: person, car, motorcycle, bike, bus and dog.

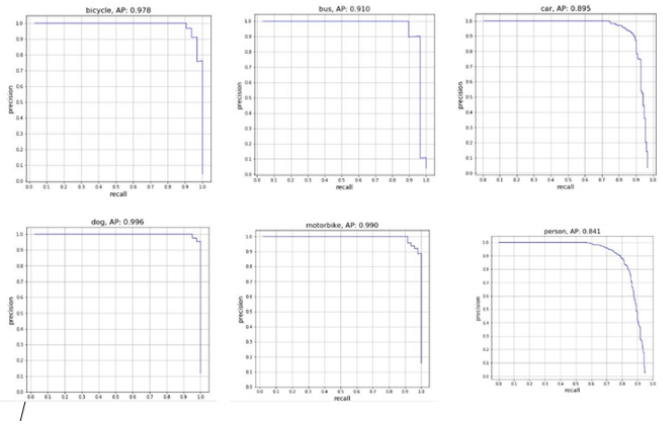

Fig. 3. SSD Inference Evaluation.

The graphs in the figure above show the accuracy according to the recall rate. This precision is calculated relative to a parameter named confidence rate which is set at 0.5 % in our case. If we were at the top right corner of the graph, that would mean that we have an average accuracy of 100%. Unfortunately, this is not the case but we are very close to it. In fact, of the six classes evaluated, the lowest rate is that of the person class and it corresponds to an AP of 84%. Apart from that, all other results are around 90% AP. The 84% AP rate for the person class is a good result, although it is slightly lower than the rest of the classes. This can be explained by the fact that the person class is more complicated to detect than a class like the car class for example. Indeed, because of its movement during the detection of a person, one can not detect in some cases the totality of the person. That is to say, sometimes there may be missing in the box delimiting the perimeter of the detected object an arm or a leg. This will result in the person being detected but their confidence will be weakened. A car during its detection does not lose a tire or a door because it is moving. When it is detected, it is detected completely. The table summarizes the AP of each of the evaluated classes. In view of this table and the previous analysis, we can conclude that the model VGG (SSD $300x300$, iter: 120000) allows to have very good results in terms of

specify. As for speed, with a GTX 660 we are at 6.2 fps. We could improve this performance by using much better GPUs.

TABLE III
AP OF THE EVALUATED CLASSES.

Classes	AP
Bike	97
Bus	91
Car	89
Dog	99
Motorbike	99
Person	84

D. YOLO V3 Algorithm

YOLO V3 is an improvement version of the former YOLO and YOLO 9000, and adopts some state-of-the-art architectures like Residual Network. The implementation which we use is its original, on C language. This implementation doesn't depend on other high level deep learning libraries like Tensorflow or Caffe and it could slightly improve its efficiency. Firstly we evaluate its pretrained model which is trained by the author. This model is trained on Pascal VOC + COCO. To get quantifiable evaluation results, we evaluate this model on the dataset Pascal VOC 07 Evaluation, for the class Person. Then, we train our model to figure out if we can improve its performance by training on different dataset. We trained by ourselves 2 models, one is trained on Pascal VOC 07 with one class Person and another is trained on COCO. The training is executed on the cluster of computation MYRIA in the Normandy calculate center CRIANN. The model trained on Pascal VOC 07 takes about 9000 batches and the model trained on COCO takes more than 16000 batches with the batch size of 32. 2 Nvidia P100 can train about 1500 batches per hour and 2 Nvidia K80 can train around 800 batches per hour. To quantify our models performance, we use the same evaluation dataset as the pretrained model, Pascal VOC 07 Evaluation. Beside this quantifiable result, we also make the inference on a video of driver's perspective which is taken by us in the centre and urban of the city Rouen.

Those results show two important consequences. 1. a rich dataset is critical for the performance of precision (mAP). The model trained on COCO dataset with 64115 images of persons shows 5.6mAP improvement over the Pascal VOC dataset with 2000 images of persons. 2. The number of classes doesnt significantly influence the speed of inference. The one class model is only about 0.4FPS faster than the 80 classes model. The inference time mainly depends on the quantity of parameters in the neural network, but the suppression of classes has only impact on the number of parameters of the fully connected layer's, which is a small part of the whole model.

IV. OBJECT DISTANCE ESTIMATION

Monodepth [8] is an unsupervised CNN based approach for distance estimation. The model is trained on stereo images and makes inference on monocular images. The output of this approach is a disparity map. The authors provide the source code with many models, trained on different datasets (including Cityscape [8] and Kitty [9]) and with different backbones (including VGG and ResNet). We have tested two models on some images, with different backbones (VGG and ResNet) and with different datasets (Cityscape and Kitty). The disparity values represent relative distances because the training images and inference images are taken by cameras of different focal distance. To get the true distancewe should use calibration images to calculate the coefficient of the baseline and focal distance. Figure 4 shows the resulting disparity maps for one example of image.

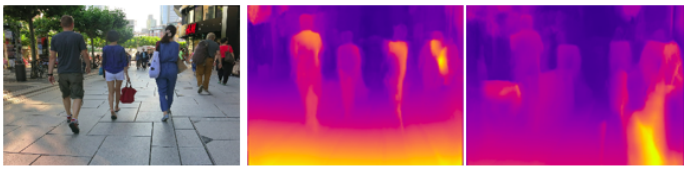

Fig. 4. Test of different datasets : disparity maps resulting from application of Monodepth on a scene of sidewalk, with backbone VGG and training on two different datasets : Cityscape (middle) and Kitty (right).

Fig. 5. Test of different backbones : disparity maps resulting from application of Monodepth on a scene of sidewalk, with training on Cityscape dataset and two backbones: VGG (middle) and Resnet (right).

These output disparity maps show the influence of the dataset and the backbone. A suitable training dataset can improve the precision of the disparity maps: the model trained by Cityscape gets better result for an input image of city scene. 5 shows that the backbone network has also influence on the result: for this example image, VGG has a better performance than ResNet, even though ResNet generally gets better precision for classification in many works [6].

The disparity maps are not sufficient for our objective, as they are not related to the objects which are present in the images. So we need to combine an object detection approach, which provides estimations of the bounding boxes of objects, and Monodepth which can give distance information inside these bounding boxes. Firstly we apply the input image to the object detection algorithm to get objects bounding boxes, and then compute histogram in the corresponding regions of the disparity map to get the distributions of disparity values of the detected objects. The major distribution interval gives the estimation of distance of the object. Figure 6 shows an example of application of this method on the bounding box of a detected object and the region of the corresponding disparity map.

We have applied this method on several images including real and synthetic ones. The results are difficult to quantify, as

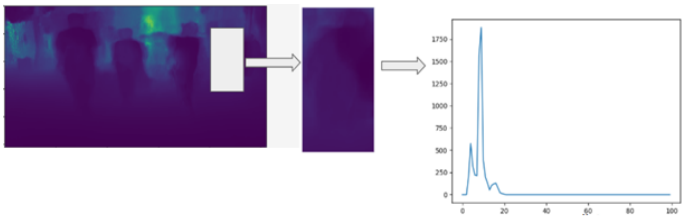

Fig. 6. Combination of the results of object detection and depth estimation : for each detected object, the histogram of estimated disparities is computed. The figure shows an example of bounding box : the corresponding histogram represents the number of disparity values (ordinate), as a function of disparities (abscissa). Those numbers are accumulated on small intervals, and the interval with maximum sum is chosen as the one corresponding to the object in the bounding box.

there is no dataset available with ground truth for the distance to the objects. Distance estimation gives good results when object are not too numerous on the images. When there are many objects with overlapping bounding boxes, the estimation is distorted, as all pixels in the bounding box are taken into account. Figure 7 shows an example of results with bad distance estimation.

Fig. 7. Example of results with bad distance estimation.

This problem could be reduced by applying a semantic segmentation to the images in parallel to the object detection process, to use only the pixels belonging to the objects inside the bounding boxes. Figure 8 shows some examples of correct distance estimation.

Fig. 8. Example of correct results of distance estimation. Estimated disparity is superimposed on each detected object.

V. Pedestrian Behavior Analysis

For our application, it is very important to be able to give an estimation of the direction of movement and speed of the object in motion. So we need to make a tracking of the detected objects in the images. However, in a video sequence, the confidence score of a detected object can vary to a large extent from image to image, and then can fall below the detection threshold on some images. So it is important to be able to estimate their positions in this case. Below are the steps that allowed us to track down detected objects:

- At time t, make the list of all boxes of the detected objects.
- At time t + 1, compare the abscissa of the boxes of each newly detected object with those of the boxes of the detected objects at time t ; associate this box with the nearest one, provided that the difference of abscissa is below a giver threshold.
- The boxes of time t + 1 which have a corresponding box at t are marked as tracked.

When tracking an object in a video sequence, we can have a loss of detection on some images. In this case, we need to estimate the new position of the object anyway. The objects which are not detected by the object-detection algorithm This means that on the current image, they could not be detected by the SSD but we could estimate their positions using the information of their positions in the previous images. For this, we apply a processing with the following sequence of steps:

- For each tracked object, calculate its speed (in pixels/s) at time t, which is the difference of abscissae of the bounding box centers between positions of this object in images at times t and t-1 (the sign of this difference gives the direction of motion)
- In order to attenuate the effect of uncertainty of the estimation of position given by the object-detection algorithm, the final speed is taken as the moving average of the values calculated over a few preceding images
- In case of no detection, consider that the object has the same speed as previous calculation, and consider that the dimensions and ordinate of its bounding box are the same as the previously calculated one
- Go back to the first step ; in case of no detection after a few iterations of this sequence of steps, exit from this sequence and mark this object as no-tracked

Figure 9 shows examples of application of this principle on two images of a sequence.

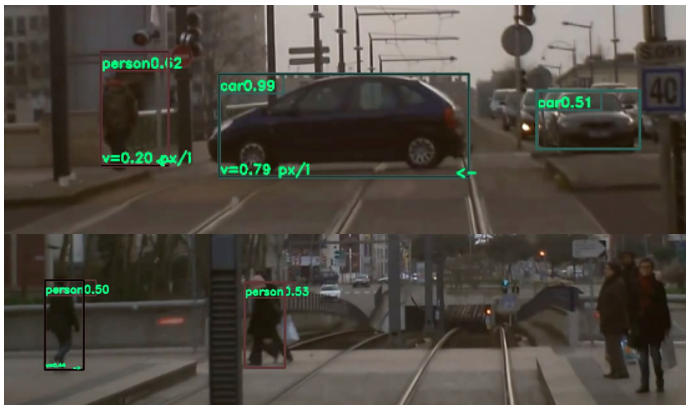

Fig. 9. Examples of tracked pedestrians and vehicle on two parts of images, with estimated direction of motion and speed (in pixels/s). When tracked, objects are marked with estimated speed (at the bottom left corner of the bounding box) and direction of motion (arrow at the bottom right corner) on the abscissa axis of the image. Objects which are tracked are bounded by two rectangles: one with the color of the estimated class and a black one to denote tracking in action

VI. Conclusion

In this paper, we have presented a contribution based three deep learning approaches for object detection, tracking, and distance estimation for smart mobility applications (traffic road and railway). The object detection approach has been well developed by taking into account not only high accuracy for object detection, but real-time applications constraints too. We have developed object detection by both SSD and YOLO V3 algorithm in order to find which algorithm is more adapted for our application. The comparison carried out illustrate that the YOLO V3 is more than SSD algorithm.

The object distance estimation based on monodepth algorithm was developed. The model was trained under stereo images dataset and makes inference on monocular images dataset. As output data, the monodepth algorithm gives a disparity map. We can merge the object detection approach and estimation distance to share the feature extraction layers, which could improve its efficiency. We have validate our approach under different datasets like Cityscape and Kitty, but also in real time within ESIGELEC vehicle in traffic road of Rouen city center. We have also validate the development under railway dataset of the Tramway of Rouen.

As a last contribution in this paper, we have presented a new method based SSD in order to analyze the behavior of object like pedestrian or vehicle. After detecting object with the SSD modified algorithm, we estimate its future position in order to have its direction of movement: pedestrian who want to cross the road, who do not cross the road, who goes through, etc.

We have presented a comparative study between SSD and YOLO V3 algorithm for object detection and tracking. To optimize their performance on low-consumption platform, a rich and suitable dataset could be very important. Change the number of detection class cant get significant improvement.

Finally, the whole development presented in this paper were validated on both real traffic circulation conditions in the city center of Rouen, and through the railway videos acquired from an embedded camera in the tramway of Rouen.

Acknowledgment

This research is supported by ADAPT Project (co-financed by the European Regional Development Fund within the framework of the INTERREG VA France (Channel) England program). Many thanks to Segula company for its contribution in the project and to the engineers of Autonomous Navigation Laboratory of IRSEEM for their support in testing phase.

References

[1] W. Liu et al., SSD: Single Shot MultiBox Detector, ECCV 2016, pp 21-37.

[2] J. Redmon et al., YOLOv3: An Incremental Improvement, arXiv:1804.02767v1

[3] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014, pp 580-587.

[4] R. Girshick et al., Fast R-CNN, ICCV 2015, pp 1440-1448.

[5] S. Ren et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS'15, Volume 1 pp 91-99.

[6] K. He et al., Deep Residual Learning for Image Recognition, CVPR 2016.

[7] C. Godard et al., Unsupervised Monocular Depth Estimation with Left-Right Consistency, CVPR, 2017.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, The Cityscapes Dataset for Semantic Urban Scene Understanding, in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2016, pp 3213-3223.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, Vision meets robotics:The KITTI dataset,Int. J. Robot. Res., vol. 32, no. 11, pp. 12311237,2013.

[10] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations 2015.

[11] Szegedy, C., Reed, S., Erhan, D., Anguelov, D., Ioffe, S. (2014). Scalable, highquality object detection. arXiv preprint arXiv:1412.1441.

[12] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016, October). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[13] Ren, S., He, K., Girshick, R., Sun, J. (2015). Faster RCNN: Towards realtime object detection with region proposal networks. In Advances in neural information processing systems(pp. 91-99).

[14] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[15] He, K., Gkioxari, G., Dollr, P., Girshick, R. (2017, October). Mask rcnn. In Computer Vision (ICCV), 2017 IEEE International Conference on (pp. 2980-2988). IEEE.