International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 23 (2017) pp. 13879-13883

Empirical Analysis of Object Detection in the Virtual World

JueSeok Kang[1], DongKeun Kim[2] and EunJoo Rhee[3]

[1,2] M&S Center 1, SIMNET, Daejeon 34127, Korea.
[3]Professor, Department of Computer Engineering, Hanbat National University, Daejeon 34158, Korea.

[1]Orcid Id: 0000-0001-8454-4248; [2]Orcid Id: 0000-0001-6093-126X; [3]Orcid Id: 0000-0002-3783-2359
(Corresponding Author)

Abstract

Artificial Intelligence has been widely used as a methodology to replace Computer Generated Force (CGF) in combat simulator. Due to the advent of Deep Learning, especially in image recognition by using Convolutional Neural Network (CNN), Artificial Intelligence used in combat simulator has taken a step forward. As a part of the combat simulator, we are willing to develop object detection and tracking module in the virtual environment that we constructed. We use YOLO (You Only Look Once) v2 model to detect object and analyze the performance of object detection depending on target size in the virtual world.

INTRODUCTION

For simulating the behavior of combat objects in a virtual environment, a shooting system is one of the most important parts with a driving system.

Conventionally, several algorithms are used to detect object in the image in the focus on image processing such as frame differencing, optical flow, and background subtraction. For the purpose of tracking object, Kalman Filter, Mean Shift Method, Support Vector Machine (SVM), and Shape Matching methods are used [1].

From the perspective of developing shooting simulator, X. Xiang et al., are based on Embedded Star platform and used OpenCV to implement Mean Shift Method to track object [2].

With the advances in the Deep Learning techniques, CNN (Convolutional Neural Network) based models have become dominantly used for object detection. R. Girshick et al., suggested R-CNN (Region-based Convolutional Neural Networks) for object detection and semantic segmentation that showed better performance than gradient based algorithm such as HOG and SIFT [3]. SPPNet, Fast R-CNN, Faster R-CNN, and ResNet have been developed the speed and accuracy of object detection more and more ([4-7]). J. Redmon et al., suggested YOLO (You Only Look Once) model and showed that their model can be used as a real-time object detection model ([8-9]).

For the use of virtual environment, in [10] they constructed the virtual world for data acquisition and training. They trained for object detection in the virtual environment and also confirmed that it can be applicable to real world object detection.

In this research, followed by development of driving module, we implemented object detection and target tracking module. We used YOLO v2 model to detect object in the virtual world and assessed the characteristics of detection performance empirically.

SYSTEM MODEL AND METHODS

In this paper, a 3D-based virtual world is used to generate training data and used as a combat simulator.

First, we selected 7 objects to detect such as barrier, car, cone, motorcycle, person, soldier, and tank among 48 classes which are predefined to perform semantic segmentation in the virtual environment.

Second, we captured each object image from the virtual world and trained the Darknet-19 classification model for feature extraction.

Third, we captured image which contains objects from the virtual world and labeled bounding box.

Next, we loaded pre-trained darknet-19 model and re-trained by using captured images and labeled data.

Finally, we analyzed the performance of object detection such as mAP (mean Average Precision) according to the number of pixels that its bounding box occupied.

A. Virtual world and Data generation

In this research, we used a 3D-based virtual world based on the Unity3D engine which is a popular game engine. We used prebuilt virtual world that previously constructed. Fig 1 shows some examples of the captured images from the virtual world we used in this research.

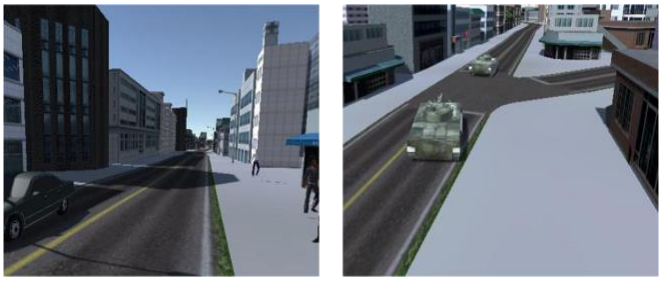

Figure 1: Examples of captured images in virtual world.

Our data generation process consists of two steps.

First, we captured each of 7 objects (barrier, car, cone, motorcycle, soldier, person, and tank) from the virtual world and 3D-object resources. We captured 100 images per class and augmented to 1000 images per class. A total of 7000 captured images are used to pre-train darknet-19 classification model for feature extraction. **Fig 2** shows some examples of data used to pre-train classification model.

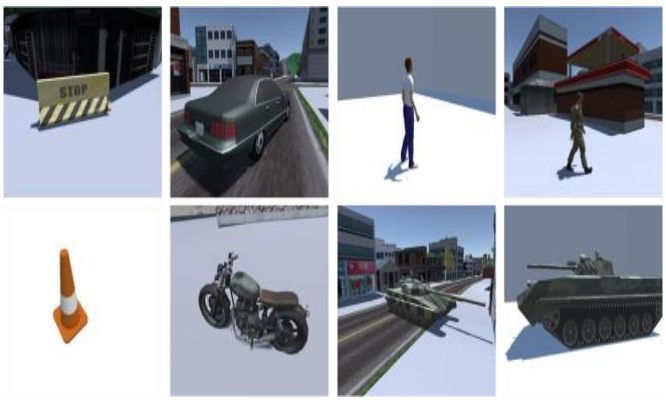

Figure 2: Examples of each object images.

Next, for the purpose of generating training data for object detection, we captured image from the virtual world and manually labeled bounding box on image. We use labeling tool which generate PASCAL VOC format XML file as an output. A total of 695 images are used to train.

B. Model Description

The model used in this research is YOLO v2. YOLO is constructed according to the Darknet-19 structure. Detailed structures are shown in Table 1. YOLO predicts position of each bounding box, confidence, and class specific probability in the end-to-end fashion, from the input image.

Table 1: The Structure of the Darknet-19.

Type	Dimension	Stride	Output
Convolutional	32 x 3 x 3	1	416 X 416
Max pool	2 x 2	2	208 X 208
Convolutional	64 x 3 x 3	1	208 X 208
Max pool	2 x 2	2	104 X 104
Convolutional	128 x 3 x 3	1	104 X 104
Convolutional	64 x 3 x 3	1	104 X 104
Convolutional	128 x 3 x 3	1	104 X 104
Max pool	2 x 2	2	52 X 52
Convolutional	256 x 3 x 3	1	52 X 52
Convolutional	128 x 3 x 3	1	52 X 52
Convolutional	256 x 3 x 3	1	52 X 52
Max pool	2 x 2	2	26 X 26
Convolutional	512 x 3 x 3	1	26 X 26
Convolutional	256 x 3 x 3	1	26 X 26
Convolutional	512 x 3 x 3	1	26 X 26
Convolutional	256 x 3 x 3	1	26 X 26
Convolutional	512 x 3 x 3	1	26 X 26
Max pool	2 x 2	2	13 X 13
Convolutional	1024 x 3 x 3	1	13 X 13
Convolutional	512 x 3 x 3	1	13 X 13
Convolutional	1024 x 3 x 3	1	13 X 13
Convolutional	512 x 3 x 3	1	13 X 13
Convolutional	1024 x 3 x 3	1	13 X 13
Convolutional	1024 x 3 x 3	1	13 X 13
Convolutional	7 x 3 x 3	1	13 X 13
Average pool	-	-	13 X 13

The output dimension of a typical YOLO model output array is S x S x B x (5 + C). S is the number of horizontal and vertical grids, B is the number of bounding boxes in each grid cell, and C is the number of classes to be detected. In this study, since S, B, and C are set to 13, 5, and 7 respectively, the size of the output array is 13x13x5x12. As a result, 845 (13x13x5 = 845) boxes are expected to exist as a result of YOLO model. The reason why the final array dimension is (5 + C) is that for each bounding box, x, y (the center point of the bounding box), w, h (the size of the bounding box), confidence (the probability of object existence), and the conditional class probability value corresponding to the number of classes. As a result, YOLO model will generate x, y, w, h, confidence, and conditional class probability values for 845 boxes, respectively.

YOLO filters out the output of 845 boxes through two processes.

In the first step, if the IOU between the generated boxes is 0.4 or more, it is assumed that the same object is detected, and the class specific confidence score is set to 0 except one box with the highest class specific confidence score.

In the second procedure, the box is selected only if among the C (seven in this study) class specific confidence scores there is a value greater than predefined threshold.

Through these two processes, the box to be output is determined among the 845 boxes generated, and the threshold is determined according to the nature of each experiment.

EXPERIMENTS AND RESULTS

Experiments are performed on an Intel Core i7-7700 CPU @ 3.60GHz, two NVIDIA GeForce GTX 1080 Ti Graphic card, 8GB RAM. We used Keras and Tensorflow as front and back-end respectively. At all times, we use 640 x 480 size image as the input image size.

For pre-training darknet-19 classification model, 5600 images are trained for 50 epochs, and 1400 images are tested. Image augmentation techniques such as rotation, width shift, height shift, shear, and zoom are applied to the image to prevent over-fitting that might occur during learning. The results of the experiment show that the model's accuracy is about 99%. The obtained model is used to extract the feature of each object to be detected before training for object detection.

For training object detection model, 695 images are trained for 500 epochs. **Fig 3** shows the results of successful object detection.

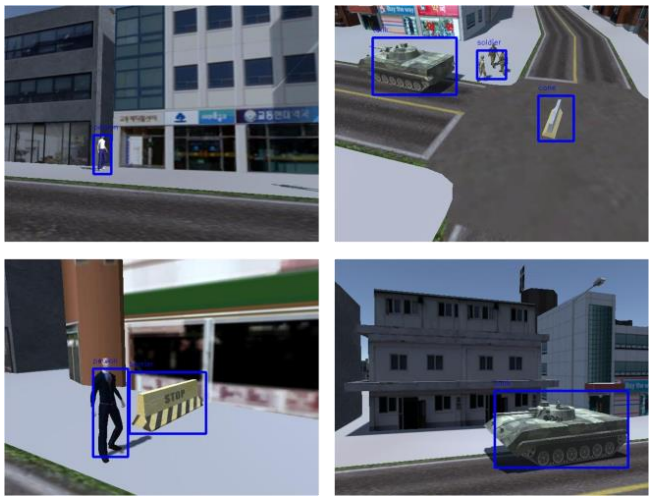

Figure 3: Examples of successfully detected images

It draws a bounding box with confidence value greater than threshold value, and prints the class with the highest probability. As shown in the second figure, if you have a collection of objects to detect, you can see that the YOLO v2 model is not well categorized because it has the disadvantage that it is difficult to distinguish objects within the grid.

Fig 4 shows the results for each threshold value. The first column shows the result when the threshold is set to 0.2, the second column shows the result when the threshold is 0.3, and the last column shows the result when the threshold is 0.4. As the threshold value decreases, the number of output boxes decreases.

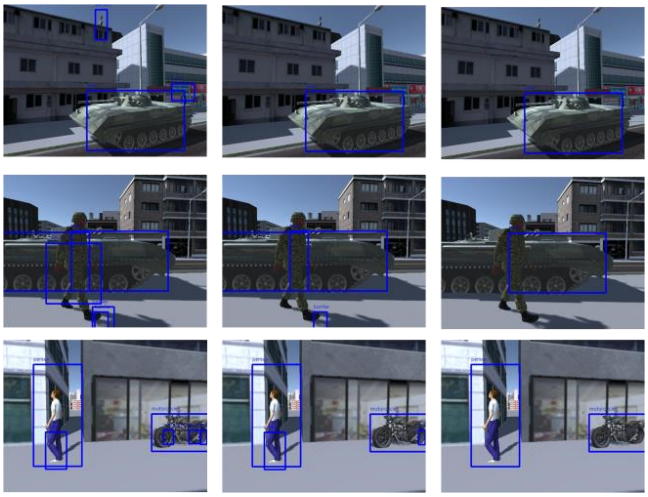

Figure 4: Examples of results for output bounding box according to the different threshold settings.

This might not be a predefined value. After training the model, we can adjust the complementary true negative and false positive number. When the threshold is set to a small value, false positive increases which mean objects are detected while there is no object. On the other hand, when the threshold is set to a large value, true negative increases which means the existing object is not detected.

A. Comparison of detection accuracy in different size

The following experiments were planned to quantify the performance of the target recognition model used in this study.

At first, 225 tank images were captured in different distances and environments, and corresponding bounding boxes were manually labeled. **Table 2** shows the categorization depending on the object size.

Table 2. Categorization depending on the object size

Classification	The number of Images
Small size object (Less than 1000 pixel)	33
Medium size object (Between 1000 and 10000 pixel)	115
Large size object (More than 10000 pixel)	77
Total Images	225

An example image for each image categorization is shown in **Fig 5**. In the case of small tank size images, the size of 640 x 480 is large enough to be difficult to identify with the naked eye.

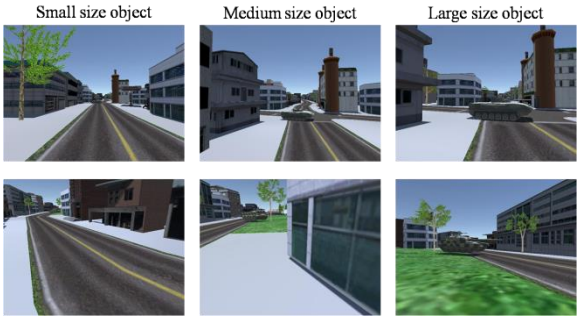

Figure 5: Examples of categorization depending on the object size.

Next, the mean IOU and the detection rate were measured for each image classification while changing the threshold value. Mean IOU is a measure of how closely the actual bounding box of each image matches the bounding box predicted by the model. Mean IOU and detection rate are defined as follows:

$$\text{IOU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

$$\text{Detection rate} = \frac{\text{Number of detected images}}{\text{Total number of Images}}$$

The experiment results for the threshold settings 0.4, 0.3, and 0.2 are shown in Table 3, 4, and 5, respectively.

Table 3: Detection result with 0.2 threshold

	Mean IOU	Detection Probability
Small size object	0.0000	0.0649
Medium size object	0.1394	0.3217
Large size object	0.4480	0.6970

Table 4: Detection result with 0.3 threshold

	Mean IOU	Detection Probability
Small size object	0.0013	0.1169
Medium size object	0.1472	0.3826
Large size object	0.4353	0.7576

Table 5: Detection result with 0.4 threshold

	Mean IOU	Detection Probability
Small size object	0.0009	0.1688
Medium size object	0.1696	0.5217
Large size object	0.4159	0.8182

In the case of the small object (less than 1000 pixels), it is confirmed by mean IOU value close to zero that it is hardly detected. For the medium size object, mean IOU is about 15% and detection rate is about 30~50%. For the large size object, mean IOU is about 40% and detection rate is about 70~80%.

In the case of the detection rate, as the threshold value is decreases, the detection rate is increased regardless of the object size. It is because the number of output bounding box is increased as the threshold value is decreased.

In the case of mean IOU, mean IOU increased with lower threshold value in small image and intermediate image, whereas mean IOU value decreased with threshold value decreased from 0.3 to 0.2 in large image.

Through these results, we can conclude that the smaller the sizes of objects to be detected, the higher detection performances are expected when we set lower threshold value. However, if the sizes of objects to be detected are large, the false alarm caused by the smaller threshold value decreases the mean IOU value.

The detailed results are shown in **Fig 6**. In both sets of images acquired under similar circumstances, it was confirmed that images smaller than 1000 pixels were incorrectly detected.

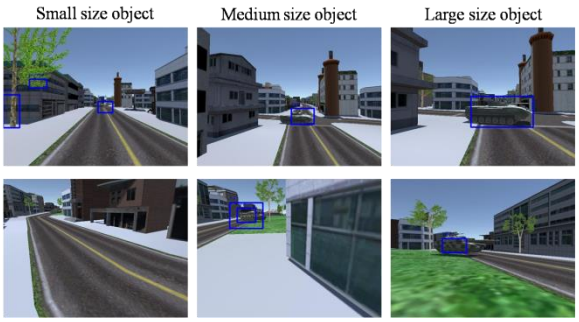

Figure 6: Detection results according to the object size.

CONCLUSIONS

In this paper, we performed a research to detect and track seven predefined classes, such as soldiers and tanks in the virtual world. We conducted two steps training processes, pre-training for extracting the features of each object and training for object detection.

As a result of training, it was confirmed that the predefined objects were detected and tracked well, so we could replace AI for simulating the behavior of combat objects in the virtual world.

In addition, we conducted empirical analysis on the relationship between object sizes, threshold settings, and detection performance, and showed that proper threshold value is needed for improving detection performance depending on the size of objects.

In further studies, extending this study, we might establish the rules of optimal threshold settings considering the size of object to be detected.

REFERENCES

[1] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods", *International Journal of Innovative Research in Computer and Communication Engineering*, volume 2, issue 2, pp 2970-2978.

[2] X. Xiang, W. Chen, and D. Zeng, "Intelligent Target Tracking and Shooting System with Mean Shift", *International Symposium on Parallel and Distributed Processing with applications*, 2008, pp 417-421.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", *Computer Vision and Pattern Recognition* (CVPR), 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 37, issue 9, pp 1904-1916, 2015.

[5] R. B. Girshick, "Fast R-CNN", *International Conference on Computer Vision*(ICCV), pp 1440-1448, 2015.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks", *Neural Information Processing Systems* (NIPS), volume 2015, pp 91-99.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *Computer Vision and Pattern Recognition* (CVPR), 2015.

[8] J. Redmon, S.K. Divvala, R. B. Girshick, and A. Farhadi,"You Only Look Once: Unified, Real-Time Object Detection", *Computer Vision and Pattern Recognition* (CVPR), 2016.

[9] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger", *Computer Vision and Pattern Recognition* (CVPR), 2016.

[10] E.Bochinski, V.Eiselein and T.Sikora, 2016, "Training a Convolutional Neural Network for Multi-Class Object Detection Using Solely Virtual World Data", *Advanced Video and Signal Based Surveillance*, pp 278-285.