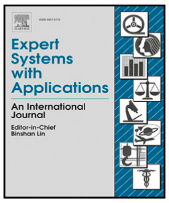
Expert Systems With Applications 164 (2021) 113833

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Detecting the shuttlecock for a badminton robot: A YOLO based approach

Zhiguang Cao[a], Tingbo Liao[b], Wen Song[c,*], Zhenghua Chen[d], Chongshou Li[a]

[a] *Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore*
[b] *School of Automation, Guangdong University of Technology, China*
[c] *Institute of Marine Science and Technology, Shandong University, China*
[d] *Institute for Infocomm Research (I2R), Singapore*

A R T I C L E   I N F O

Keywords:
Deep learning
Object detection
YOLO
Badminton robot

A B S T R A C T

The ability to identify objects of interest from digital visual signals is critical for many applications of intelligent systems. For such object detection task, accuracy and computational efficiency are two important aspects, especially for applications with real-time requirement. In this paper, we study shuttlecock detection problem of a badminton robot, which is very challenging since the shuttlecock often moves fast in complex contexts, and must be detected precisely in real time so that the robot can plan and execute its following movements. To this end, we propose two novel variants of Tiny YOLOv2, a well-known deep learning based detector. We first modify the loss function to adaptively improve the detection speed for small objects such as shuttlecock. We then modify the architecture of Tiny YOLOv2 to retain more semantic information of small objects, so as to further improve the performance. Experimental results show that the proposed networks can achieve high detection accuracy with the fastest speed, compared with state-of-the-art deep detectors such as Faster R-CNN, SSD, Tiny YOLOv2, and YOLOv3. Our methods could be potentially applied to other tasks of detecting high-speed small objects.

1. Introduction

Computer vision plays an important role in many applications of expert and intelligent systems, such as security surveillance (Gómez et al., 2015), autonomous driving (Hassannejad et al., 2015; Wu et al., 2017), unmanned aerial vehicles (Al-Kaff et al., 2018), and industrial robots (Cong et al., 2018; Kumra & Kanan, 2017). As a longstanding and fundamental problem in computer vision, object detection aims at automatically determining the existence and spatial location of certain instances of objects in the images (Liu, et al., 2020). It can effectively help autonomous and intelligent systems such as machines or robots to perceive and understand the world better by automatically analyzing digital signals from cameras, and provide critical information for latter decisions such as navigation, motion planning, safety control, etc.

Recently, deep learning based approaches have achieved state-of-the-art performance for the object detection task. As will be mentioned in Section 2, these approaches can be categorized as two-stage and one-stage detectors. The former ones, such as Faster R-CNN (Ren et al., 2015), propose a set of candidate regions first and then use a classifier to output the final prediction. In contrast, the later ones such as YOLO (Redmon et al., 2016), directly perform detection on the images without the proposal stage. While accuracy is a key metric for object detection, for many real-world intelligent systems, computational efficiency of the detector is also critical to ensure satisfactory performances (Chakraborty et al., 2016; Wooden et al., 2010). For example, autonomous robots often need to detect obstacles or tracking targets promptly via their vision systems, so that they can plan their actions and control motors based on the detection results in a real-time manner (Alabachi et al., 2019; Guo, et al., 2019). Another example is autonomous driving, where detection of various objects (e.g. cars, pedestrians, road signs) must be done in real time to ensure the right and safe vehicle control (Wu et al., 2017). Moreover, detection tasks in these applications are often done by mobile or embedded devices, which have only limited computing power (Wang et al., 2018). Consequently, though the accuracy is a bit lower, one-stage detection approaches are more ideal in these applications with real-time requirements, since they are often significantly faster in terms of computational efficiency compared with the two-stage ones (Liu, et al., 2020).

In this paper, we focus on the challenging task of detecting shuttlecock for a badminton robot. More specifically, the robot first detects the shuttlecock and fits its trajectory via a visual subsystem, then moves to the specific position and ultimately hit the shuttlecock to the

* Corresponding author.
E-mail addresses: zhiguangcao@outlook.com (Z. Cao), tingbo_liao@hotmail.com (T. Liao), wensong@email.sdu.edu.cn (W. Song), chen0832@e.ntu.edu.sg (Z. Chen), iselc@nus.edu.sg (C. Li).

https://doi.org/10.1016/j.eswa.2020.113833
Received 18 September 2019; Received in revised form 28 June 2020; Accepted 31 July 2020
Available online 7 August 2020
0957-4174/© 2020 Published by Elsevier Ltd.

opponents using its hitting device. Due to the high-speed movement of shuttlecocks, the object detection task must be performed in a real-time manner. Generally, the shuttlecock detection task is considered as a single object detection task where the goal is to detect a target object accurately within a series of sequential frames. Although significant progress has been made for general object detection in terms of efficiency and accuracy, it remains an open problem on what is the best deep architecture for different applications. In our case, simply applying existing deep detectors may engender poor performance due to several reasons: (1) scale of the shuttlecock changes quickly during flight, (2) the shuttlecock often moves in complex contexts involving moving objects, athletes, etc., (3) the shuttlecock always moves at high speed, (4) the shuttlecock is usually in white color, which can be very similar to that of the background, leading to background clutter (Cai, et al., 2019; Chen et al., 2019; Henriques et al., 2014; Suzuki, 2017; Zhang, et al., 2020).

We address the above challenges and achieve accurate real-time shuttlecock detection, by analyzing the visual data obtained from a ZED binocular camera mounted on the robot. To meet the requirement on computational efficiency, we design our detection algorithm based on the widely used one-stage detector Tiny YOLOv2 (Redmon & Farhadi, 2017). However, the original Tiny YOLOv2 does not give satisfactory performance in detecting shuttlecock. As the main contribution of this paper, we propose two novel networks named M-YOLOv2 and YOLOBR based on Tiny YOLOv2. More specifically, the former one employs a new loss function to better reflect the characteristics of shuttlecock detection. The latter one has a novel neural architecture that ameliorates the issues of losing useful semantic information and accumulating prediction error, which affects the performance of Tiny YOLOv2. Extensive experimental results in various environments show that, compared with state-of-the-art detectors, the two proposed networks exhibit superior performance in terms of both accuracy (measured by precision and recall) and computational efficiency (measured by FPS), and is robust in both simple contexts and complex ones with background clutter.

The remainder of the paper is organized as follows. Section 2 briefly summarizes existing works related to our task. Section 3 describes our method in detail. Section 4 displays and analyzes the experimental results of our method and other state-of-the-art deep detectors. Section 5 concludes the paper and states our future works.

2. Related work

Over the past decade, two popular research streams have been formed to investigate and devise efficient and robust object detection algorithms. The first research stream focuses on shallow detection algorithms such as AdaBoost and Random Forests, which shows good robustness against illumination, scales, translations, etc., by exploring handcrafted features like SIFT, HOG and Harr (Breiman, 2001; Dalal & Triggs, 2005; Lowe, 2004; Viola & Jones, 2001; Zhang et al., 2017). For instance, given a specific scenario, the AdaBoost algorithm based on HOG and Harr can detect various faces rapidly with high detection rates. Additionally, the introduction of the integral channel features (Dollár et al., 2009) and DPM (Felzenszwalb et al., 2010) advances the development of detection for complicated objects. Though they perform satisfactorily in well-controlled scenarios, shallow detectors tend to be inferior in cases where significant background variations exist (Girshick et al., 2014).

The second research stream is based on deep learning, which reigns undisputedly as the new de-facto method for object detection. Several methods among them, such as Faster R-CNN (Ren et al., 2015), R-FCN (Dai et al., 2016), SSD (Liu, et al., 2016), Multibox (Erhan et al., 2014), Overfeat (Sermanet, et al., 2013) and YOLO (Redmon et al., 2016), yield state-of-the-art performance on benchmark datasets. More specifically, Faster R-CNN is a classic two-stage detector which introduces a region proposal network (RPN) on the basis of R-CNN and Fast R-CNN (Girshick, 2015). In the first stage, it generates a sparse set of class-agnostic box proposals via RPN. Then in the second stage, it predicts a class and box regression refinement for each proposal using convolution neural network. Thus, it can simultaneously predict the bounding box and class scores of the objects for each region proposal. In particular, Faster R-CNN obtains a mean average precision (mAP) of 78.8% while running at 5 frames per second *(FPS)* on PASCAL VOC2007 (using VGG-16 as the backbone network) (Ren et al., 2015).

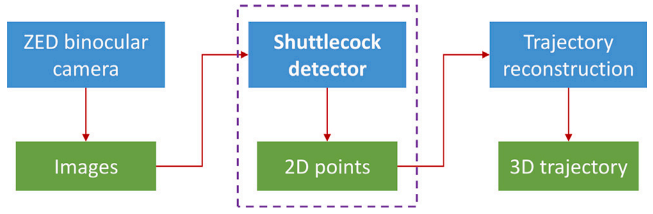


Fig. 1. The operational mechanism of the vision subsystem.

Two-stage detectors have achieved big success that propelled their applications in some real products such as Google Photos and Snapchat, which mainly pursue high detection accuracy. However, two-stage detectors are generally slow in computation, rendering them impractical for many computational demanding scenarios where efficiency also matters much, such as the shuttlecock detection task of the badminton robot. As a consequence, many one-stage detectors are devised to improve the computation efficiency. Among them, YOLO (Redmon et al., 2016) and its variants YOLOv2 (Redmon & Farhadi, 2017), YOLOv3 (Redmon & Farhadi, 2018) show promising performance in terms of detection speed. With a Pascal Titan X, YOLOv3 processes images at 20*FPS*, which engenders a mAP of 57.9% on COCO test-dev (Redmon & Farhadi, 2018). Obviously, the one-stage detectors have significantly enhanced the computation performance, but there is still much room to further improve the accuracy (Huang, et al., 2017). In the next section, we will analyze the limitations of YOLO and address them in our approach.

3. The proposed approach

In this section, we present our approach in detail. We first briefly introduce the badminton robot and its vision system, as well as the shuttlecock detection task. Then, we show how we modify the loss function, and how we improve the overall architecture of Tiny YOLOv2.

3.1. The vision system and detection task

The badminton robot used in our case consists of three major components, i.e. the vision, control, and mechanical systems. Particularly, the vision system is responsible for detecting the shuttlecock and analyzing its trajectory. This information is then fed into the control system which drives the motor to move the robot to a specific position, where it hits the shuttlecock back to the opponent using its hitting device in the mechanical system. Apparently, vision system is the most crucial and challenging part of the badminton robot since it provides control singles for the following actions. A more detailed illustration of how the vision system works is given in Fig. 1. As shown, the ZED binocular camera is used for image acquisition. After that, a pre-trained shuttlecock detector that identifies shuttlecocks in images is employed. Finally, the detected shuttlecock points in the 2D images are restored in the 3D space, and the corresponding trajectory can be obtained via 3D reconstruction and the least square method (Yang, 1993). In the remaining part of this paper, we focus on the shuttlecock detection in the 2D images, which is the core task of the vision system.

As mentioned in Section 2, compared with most of the two-stage detectors that are based on selective search (Uijlings et al., 2013), sliding window (Chang & Lee, 2004), or RPN (Ren et al., 2015), the

"You Only Look Once" mechanism of YOLO enables faster detection speed while maintaining relatively high average precision. This is ideal for our case, because for the badminton robot, real-time detection with low false positive and low false negative rates are necessary and crucial. Essentially, YOLO frames the task of object detection as a regression problem. As a result, the detector is a single network that outputs the bounding box coordinates and class probabilities simultaneously, and can be trained in an end-to-end fashion.

Although the YOLO model can satisfy the requirement of real-time detection for robots, there are some drawbacks that limit the application of YOLO. More specifically, there are two limitations of YOLO based networks. Firstly, the spatial constraints of YOLO limit the number of nearby objects that the model can predict. In Redmon et al. (2016), Redmon and Farhadi (2017), YOLO divides an image into 7×7 grids, predicts two bounding boxes for each grid, and outputs confidences and 20 probabilities for these bounding boxes. Since YOLO predicts two bounding boxes, the prediction capacity of the model is limited to the suggested number of nearby objects. Moreover, there are some difficulties for detecting small objects. Although some issues have been addressed in YOLOv3 (Redmon & Farhadi, 2018) by adopting residual connections and skips, it still struggles with small objects such as birds and kites. This also applies to our situation, since the average resolution of shuttlecock in the dataset is only about 15×15 (in pixel). Secondly, YOLO is a one-stage method, where the detector predicts and outputs the class and box regression over a whole image at once. Thus, the relatively coarse features of images will result in larger location error compared with two-stage detectors.

To resolve the above limitations of YOLO, we modify its network by designing several novel strategies. Among all the variants of YOLO, we choose Tiny YOLOv2 (Redmon & Farhadi, 2017) as the baseline model to ameliorate, mainly for the sake of efficiency.

3.2. Loss function modification: M-YOLOv2

In the Tiny YOLOv2 model, the neural network is optimized by minimizing a loss function defined as a sum-squared error of two parts, i.e. the location error and the classification error. For our shuttlecock detection task, considering the fact that the size of shuttlecock is relatively stable in real scenarios and there is no requirement for multi-class detection, we can modify the loss function of Tiny YOLOv2 defined in Redmon et al. (2016) accordingly for better performance and efficiency. These modifications are detailed below.

Firstly, the original Tiny YOLOv2 model predicts for each bounding box the square roots of its width and height and uses them in the loss function, instead of the width and height themselves. This is to partially reflect that the small deviations in large boxes matter less than those in small boxes. However, in our situation, since the shuttlecock is always regarded as a small object during flight in the sport courts, we can take the width and height of bounding box directly. Secondly, we notice that there are only two classes in our scenario, i.e. the shuttlecock and the background. Therefore, we remove the classification part of the loss function for efficiency.

Combining the above modifications, we formulate the new loss function as below:

$$\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2) \\
&+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} ((w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2) \\
&+ \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_{ij})^2 \\
&+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{i}^{obj} (C_i - \hat{C}_{ij})^2,
\end{aligned} \tag{1}$$

Table 1
Architecture of the Original Tiny YOLOv2.

Type	Filters	Size/Stride	Output
Input			416 × 416
Convolution	16	3 × 3/1	416 × 416
Maxpool		2 × 2/2	208 × 208
Convolution	32	3 × 3/1	208 × 208
Maxpool		2 × 2/2	104 × 104
Convolution	64	3 × 3/1	104 × 104
Maxpool		2 × 2/2	52 × 52
Convolution	128	3 × 3/1	52 × 52
Maxpool		2 × 2/2	26 × 26
Convolution	256	3 × 3/1	26 × 26
Maxpool		2 × 2/2	13 × 13
Convolution	512	3 × 3/1	13 × 13
Maxpool		2 × 2/2	13 × 13
Convolution	1024	3 × 3/1	13 × 13
Convolution	1024	3 × 3/1	13 × 13
Convolution	125	1 × 1/1	13 × 13
Detection (Fully connected)			

where 1_i^{obj} denotes whether the object appears in cell i and 1_{ij}^{obj} denotes that the jth bounding box predictor in cell i is "responsible" for that prediction, 1_{ij}^{noobj} denotes that the jth bounding predictor in cell i is "irresponsible" for that prediction. The coordinates (x, y) represent the center of the bounding box, while the coordinates (w, h) are the weight and height of the box. C_i represents the groundtruth, and its value is 1 only when the label is the shuttlecock, otherwise $C_i = 0$. In Eq. (1), the symbols with ^ are the predictions, and the symbol without ^ is the tag value for training. λ_{coord} is a parameter used to improve the stability of the model, similar to those in Redmon et al. (2016). More specifically, λ_{coord} is applied to increase the loss from bounding box coordinate predictions, and λ_{noobj} is introduced to balance the positive and negative samples. In the remaining part of this paper, the modified Tiny YOLOv2 network with the loss function defined in Eq. (1) is referred to as "M-YOLOv2" for simplification.

3.3. The YOLOBR architecture

Although the modified loss function in M-YOLOv2 is able to ameliorate the detection performance, there is still much room for further improvement in other aspects. The architecture of original Tiny YOLOv2 has 9 convolutional layers, 6 Maxpool layers across the first 6 convolutional layer, and a fully connected layer to output the detection result, as shown in Table 1. However, this architecture has two limitations. Firstly, frequent pooling tends to diverge the extracted features, making the detection of small objects such as shuttlecock extremely challenging. Secondly, since the original Tiny YOLOv2 does not adopt any proposal refinement measures, the detection results can be poor due to the accumulated error between positions of the predicted and ground truth boxes.

Motivated by the shortcomings of Tiny YOLOv2 mentioned above, we propose the YOLOBR architecture, which is built as follows. In order to retain more meaningful semantic information of small objects, we reduce the number of Maxpool layers. More specifically, there are only 3 Maxpool layers in our YOLOBR network compared with the 6 Maxpool layers in the original Tiny YOLOv2. Moreover, as the shuttlecock detection is relatively easier than the original Tiny YOLOv2's multi-class detection, we reduce the number of convolutional filters to enhance detection speed while maintaining accuracy. In YOLOBR, we reduce the number of convolution filters by 4 times compared with the original Tiny YOLOv2 (shown in Tables 1 and 2). In addition, Szegedy

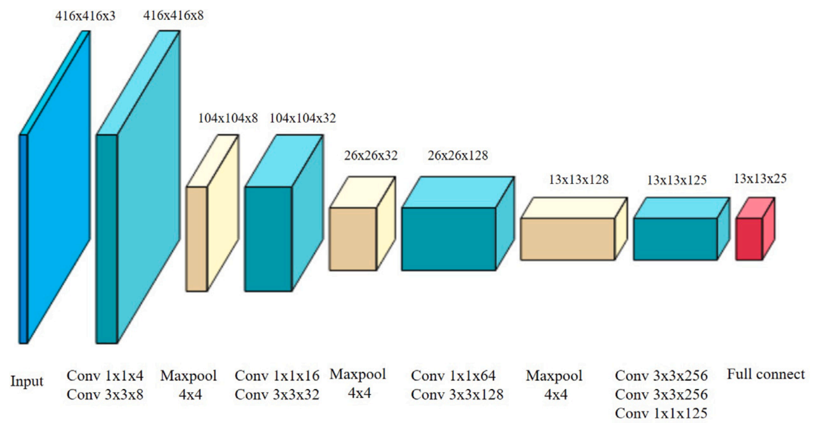


Fig. 2. Illustration of the YOLOBR architecture.

Table 2
Architecture of the proposed YOLOBR.

Type	Filters	Size/Stride	Output
Input			416 × 416
Convolution	4	1 × 1/1	416 × 416
Convolution	8	3 × 3/1	416 × 416
Maxpool		4 × 4/4	104 × 104
Convolution	16	1 × 1/1	104 × 104
Convolution	32	3 × 3/1	104 × 104
Maxpool		4 × 4/4	26 × 26
Convolution	64	1 × 1/1	26 × 26
Convolution	128	3 × 3/1	26 × 26
Maxpool		2 × 2/2	13 × 13
Convolution	256	3 × 3/1	13 × 13
Convolution	256	3 × 3/1	13 × 13
Convolution	125	1 × 1/1	13 × 13
Detection (Fully connected)			

et al. (2016) shows that a factorizing idea of replacing a 3 × 3 convolutional layer with a 1 × 3 and a 3 × 1 convolutional layers can be more efficient. Utilizing the similar idea, we adopt a 1 × 1 and a 3 × 3 convolutional layers rather than two 3 × 3 convolutional layers, which further benefits small objects detection. All the layers utilize Rectified Linear Unit (ReLU) (Nair & Hinton, 2010) as the activation function except for the final layer which uses a softmax activation function. The complete architecture of YOLOBR is shown in Fig. 2.

4. Experimental results and discussion

In this section, we compare the proposed method with state-of-art detection networks on the videos of badminton playing. Detailed setup of the networks and the collected shuttlecock dataset are presented below.

4.1. Experimental setup

Dataset collection and processing. For the shuttlecock detection task studied in this paper, currently there is no well-annotated public dataset. Therefore, we collect an annotated dataset for the classification and detection of shuttlecock based on the following procedure. Firstly, we acquire a set of video recordings of shuttlecock with the resolution of 1280 × 720 pixels, using the ZED binocular camera mentioned in Section 3.1. More specifically, we select 50 different environments, including simple and complex contexts evaluated based on the environmental lighting and background clutter, and take a video in each of them. Several examples of these contexts are shown in Fig. 4 of Appendix. Then, from these 50 video recordings, we extract approximately 18000 RGB images of shuttlecocks (about 15 s per video at 24 *FPS*) to obtain the shuttlecock detection dataset. We select 80% of the data, i.e. images from 40 contexts (20 simple ones and 20 complex ones) as the training set (about 14400 images), and the remaining 20% data from 10 contexts (5 simple ones and 5 complex ones) as the test set. Further, we split the training set into training and validation sets at a ratio of 5:3. There is no overlap in training, validation and test sets.

Baselines and evaluation metrics. We compare the performance of our approach against several state-of-the-art detectors, including Faster R-CNN (Ren et al., 2015), SSD (Liu, et al., 2016), the original Tiny YOLOv2 (Redmon & Farhadi, 2017), and the latest YOLOv3 (Redmon & Farhadi, 2018). We use the default settings for these baselines, except Tiny YOLOv2 which is the basis of our methods. Implementation details of Tiny YOLOv2 and the two proposed networks will be presented later. The comparison is conducted on the dataset we collected above, based on the following metrics:

- Precision: $N_{tp}/(N_{tp}+N_{fp})$;
- Recall: $N_{tp}/(N_{tp}+N_{fn})$;
- FPS: the number of images processed every second.

For precision and recall, N_{tp}, N_{fp} and N_{fn} are the numbers of true positives, false positives, and false negatives, respectively. These metrics measure the performance from two aspects: precision and recall are the most widely-used metrics for the detection accuracy (Hosang et al., 2016), and FPS evaluates the computational efficiency of shuttlecock detection. Following previous research, we also use the intersection over union (IOU) criterion to check whether a prediction is correct or not, computed as follows:

$$IOU = \frac{Prediction \cap GroundTruth}{Prediction \cup GroundTruth} \tag{2}$$

The IOU value is between 0 and 1, and the closer to 1, the better. We define a predicted bounding box as positive when its *IOU* is greater than 0.6, otherwise it is negative. In our experiments, we set the threshold to be 0.6 to balance the number of true positives and false negatives while maintaining real-time performance.

Implementation details. We use the Darkflow framework for all training and inference. The hardware we use is a DELL mobile workstation with i7 CPU, NVIDIA M1200 (GPU supported) and 16G memory, and Linux OS. Below we explain in detail how we implement Tiny YOLOv2 and the two proposed networks, M-YOLOv2 and YOLOBR.

The optimizer we use for these three networks is the adaptive moment estimation (Adam) method, which leads to quick convergence during training in about 8000 epochs (40 h). The learning rates are set to be 0.001 initially and reduced by a factor of 10 for each 2000 iterations. Due to the hardware constraints, we use a batch size of 24

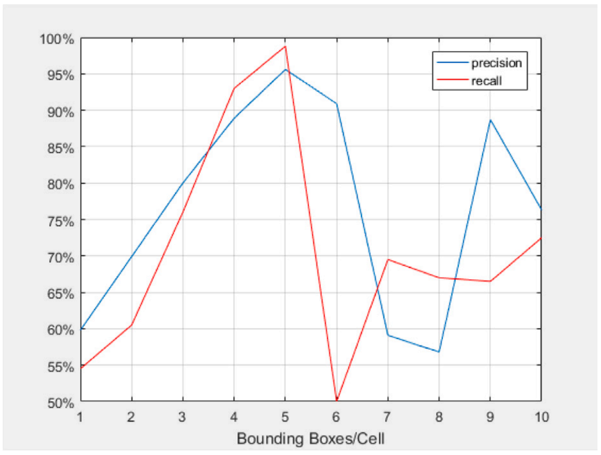


Fig. 3. Average *precision* (blue solid line) and *recall* (red solid line) curves with the number of bounding boxes per cell (from 1 to 10) under the IOU threshold 0.6 on the test subset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
FPS with the number of bounding boxes per cell (from 1 to 10) under the IOU threshold 0.6 on the test set.

Bounding Boxes/Cell	FPS (frames/second)
1	29.8
2	29.4
3	28.7
4	28.5
5	**27.6**
6	27.1
7	26.3
8	25.3
9	24.5
10	23.6

during training. For shuttlecock detection, we take the complete image as input of the original Tiny YOLOv2 networks and the two proposed variants. We choose $S = 13$, i.e. divides the input image into a 13×13 grid, which empirically leads to good performance in our experiments.

In addition, we notice that the number of bounding boxes predicted for each cell in one image, i.e. B, can greatly influence the performance. We perform experiments on the test set to examine such impact, and plot the average precision and recall curves for different numbers of bounding boxes per cell (from 1 to 10) with the IOU threshold 0.6 in Fig. 3. The corresponding FPS is listed in Table 3. We can observe that the number of bounding boxes for each cell does have a great impact on the detection performance. With the increase of the number of bounding boxes per cell, the precision and recall increase firstly and then drop quickly. Ranging from 1 to 10, 5 bounding boxes per cell achieves the best result (95.6% and 98.8% for precision and recall, respectively). In addition, Table 3 shows that FPS decreases with the increase of B, due to the increasing number of parameters in the detector. When $B = 5$, which has the best precision and recall, FPS is 27.6 which meets the real-time demand for the badminton robot. Therefore, in the following experiments, we set the number of bounding boxes per cell to 5 for Tiny YOLOv2 and the two proposed variants, M-YOLOv2 and YOLOBR. Since the dataset has 2 labeled classes, i.e. shuttlecock and background, we use C to express the confident of the prediction object. Therefore, the final output of each of these networks is $S \times S \times ((x, y, w, h, C) \times B)$, namely a $13 \times 13 \times 25$ tensor of predictions. The other parameters are set to the default value in the original Tiny YOLOv2 network.

4.2. Results and analysis

In this section, we conduct experiments to provide insights on how the proposed M-YOLOv2 and YOLOBR networks improve the detection performance compared with other state-of-the-art detectors, including Faster R-CNN, SSD, the original Tiny YOLOv2, and YOLOv3.

Table 4
Offline evaluation results on the test set (* is the best across all methods, bold is the best in our methods).

Method	Precision (%)	Recall (%)	FPS
Faster R-CNN	97.3*	99.2*	2
SSD	68.1	65.8	18
Tiny YOLOv2	89.3	75.4	27
YOLOv3	74.2	88.0	20
M-YOLOv2 (Ours)	95.6	**98.8**	27.6
YOLOBR (Ours)	**96.3**	94.6	**29.2***

We first perform offline experiments to evaluate our approaches against the baselines, which guarantees that the detector can process each frame in a video pipeline. Here, we choose the test set described in Section 4.1. The results are summarized in Table 4. We can observe from this table that the two proposed methods M-YOLOv2 and YOLOBR run at 27.6 and 29.2 FPS (videos are compressed before being processed), respectively, which are faster than the frame rate of the ZED camera (24 FPS) and satisfy the real-time demand of the badminton robots. Compared with SSD, the original Tiny YOLOv2, and YOLOv3, the two proposed networks show excellent performance in terms of both detection accuracy and speed, since the values of all three metrics (precision, recall and FPS) are significantly better. Compared with Faster R-CNN which shows the best detection performance with respect to precision and recall, the proposed M-YOLOv2 and YOLOBR far outstrip Faster R-CNN in terms of computational efficiency, since the corresponding FPS is one magnitude higher than that of Faster-RCNN. This further demonstrates the limitation of Faster R-CNN in most of the applications that require real-time performance. Comparing the two proposed methods, we can observe that YOLOBR can achieve performance equivalent to that of M-YOLOv2 in terms of precision and recall while reaching a higher FPS (29.2), due to the simplification and effectiveness of its architecture. In fact, the memory consumption of YOLOBR is 4 times less compared to that of the original Tiny YOLOv2 and the size of weight file is only 4 MB.

Since the proposed methods are required to operate in the vision system of the badminton robots, we run an online test to evaluate the real-time performance of our algorithms. We summarize the experimental results in Table 5. We can observe from this table that for almost all methods, the performance is worse than those in Table 4, but our methods still perform reasonably well. More specifically, M-YOLOv2 and YOLOBR require about 48 ms and 45 ms to process each frame, resulting in 21 and 22 FPS, respectively, which are slightly slower than the frame rate of the ZED binocular camera (24 FPS). This is due to the limitation of the ZED camera's frame rate (FPS maximum) and the extra overhead caused by image processing (image reading, object detection, and shuttlecock display), which forces the detector to skip some frames and results in the decline of detection speed. The frame skipping also affects the detection accuracy. Fortunately, by comparing the results in Tables 4 and 5, we can observe that the precision and recall only decrease to a small extent, and the online detection accuracy is still acceptable. Comparing with other baseline methods, we can observe that M-YOLOv2 and YOLOBR detector still show their overwhelming superiority over SSD, the original Tiny YOLOv2, and YOLOv3 in all evaluation metrics, which is similar to the observation we made for Table 4. Although YOLOv3 (Redmon & Farhadi, 2018) improves the detection performance on general small objects via residual connections and skips, it turns out that it struggles with shuttlecock detection compared with our methods in all respects, both for the offline and online experiments. For Faster R-CNN which is the best performing method in the offline test in terms of accuracy, it still gives the highest precision in the online experiments, but the recall drops significantly to 72.6%. To summarize, the offline and online experiments fully demonstrates the

Fig. 4. Different contexts of the flying shuttlecock (including simple and complex contexts) that are used for training and validation.

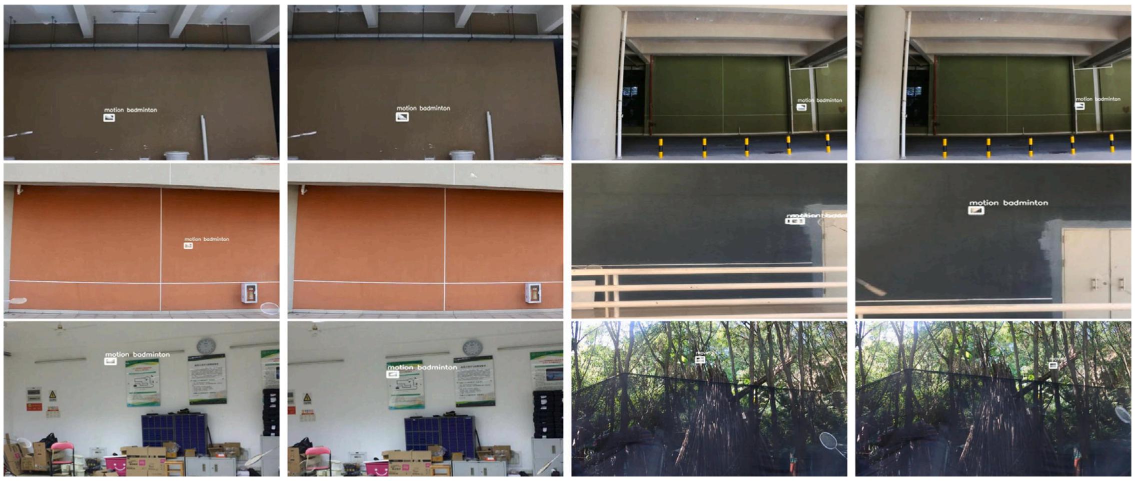


Fig. 5. Different contexts of the motion shuttlecock (including simple and complex contexts) that are used for testing. Detected shuttlecock is annotated by the white rectangle. The shuttlecocks in some contexts cannot be detected due to the background clutter, for example the second left one in the middle row.

Table 5
Online evaluation results on the test set (* is the best across all methods, bold is the best in our methods).

Method	Precision (%)	Recall (%)	FPS
Faster R-CNN	94.7*	72.6	1.5
SSD	59.3	57.8	17
Tiny YOLOv2	81.7	67.6	19
YOLOv3	77.4	85.5	16
M-YOLOv2 (Ours)	89.3	90.2	21
YOLOBR (Ours)	**91**	**92.5***	**22***

superiority of the proposed two networks for the badminton detection task in simple and complex contexts. Some detection results are shown in Fig. 5 of the Appendix.

5. Conclusions and future works

In this paper, we study the shuttlecock detection task of a badminton robot. This is a challenging task since shuttlecock moves very quickly, and detection must be done in real time. Based on the architecture of Tiny YOLOv2, we propose two novel variations, namely M-YOLOv2 and YOLOBR. We modify both the loss function and the network structure, so as to achieve a better adaption from general object detection to the shuttlecock detection task. We collect a large-scale annotated dataset of badminton playing in various contexts, including simple and complex ones with background clutter, and conduct comprehensive experimentations to examine the performance of our methods by comparing them with state-of-the-art detectors, including Faster R-CNN, SSD, Tiny YOLOv2, and YOLOv3. Results show that comparing with these methods, our networks can achieve very good accuracy with very efficient computation, which demonstrate the superiority and effectiveness of our design.

Our study suggests several ways to improve the performance of deep models for the detection of high-speed small objects such as shuttlecock. For example, the feature extraction layers could be simplified by reducing the number of pooling operations, such that more semantic information of the small objects could be retained. It would also be helpful to use larger feature maps, and directly perform regression on the height and width of bounding boxes. We believe these intuitions could be extended to the detection tasks of other small objects, and could also be applied to other deep architectures such as Faster-RCNN, SSD, and other variants of YOLO, to improve their performance. In the future, we plan to perform similar modifications on other feature extraction networks such as ResNet (He et al., 2016) and DarkNet53 (Redmon & Farhadi, 2018), and verify their performance in detecting small objects.

One limitation of our current study is that, most data is collected in outdoor contexts with natural light. However, for indoor contexts with multiple light sources, the shuttlecock in the images could be ghosting, which may affect the detection performance. We plan to extend our methods to these contexts in the future. Another limitation is that currently we use the binocular camera for 2D shuttlecock detection, and the reconstructed 3D shuttlecock trajectory may have certain error, which may further affect the actions of robot. We will investigate how to reduce such error by incorporating other sensors such as radar.

CRediT authorship contribution statement

Zhiguang Cao: Conceptualization, Methodology, Writing - original draft. **Tingbo Liao:** Software, Validation, Writing - review & editing. **Wen Song:** Conceptualization, Methodology, Writing - original draft. **Zhenghua Chen:** Methodology, Writing - review & editing. **Chongshou Li:** Software, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Zhihang Li and Haozhi Lin for their work in collecting the datasets. We thank Dr. Le Zhang for helpful comments and discussions. Zhiguang Cao is supported by the National Natural Science Foundation of China (61803104), MOE Academic Research Fund of Singapore (R266000096133, R266000096731 and MOE2017-T2-2-153) and Singapore National Research Foundation (NRF-RSS2016004). Wen Song is supported by the Young Scholar Future Plan of Shandong University, China (Grant No. 62420089964188).

Appendix

See Figs. 4 and 5.

References

Al-Kaff, A., Martin, D., Garcia, F., de la Escalera, A., & Armingol, J. M. (2018). Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Systems with Applications*, *92*, 447–463.

Alabachi, S., Sukthankar, G., & Sukthankar, R. (2019). Customizing object detectors for indoor robots. In *2019 International conference on robotics and automation* (pp. 8318–8324). IEEE.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Cai, R., Li, Z., Wei, P., Qiao, J., Zhang, K., & Hao, Z. (2019). Learning disentangled semantic representation for domain adaptation. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 2060–2066). AAAI Press.

Chakraborty, P. R., Tjondronegoro, D., Zhang, L., & Chandran, V. (2016). Automatic identification of sports video highlights using viewer interest features.In *Proceedings of the 24th ACM international conference on multimedia retrieval*. (pp. 55–62).

Chang, J., & Lee, W. (2004). A sliding window method for finding recently frequent itemsets over online data streams. *Journal of Information Science and Engineering*, *20*(4), 753–762.

Chen, W., Liao, T., Li, Z., Lin, H., Xue, H., Zhang, L., Guo, J., & Cao, Z. (2019). Using FTOC to track shuttlecock for the badminton robot. *Neurocomputing*, *334*, 182–196.

Cong, Y., Tian, D., Feng, Y., Fan, B., & Yu, H. (2018). Speedup 3-d texture-less object recognition against self-occlusion for intelligent manufacturing. *IEEE Transactions on Cybernetics*, *49*(11), 3887–3897.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379–387).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 18th IEEE computer society conference on computer vision and pattern recognition*.(pp. 886–893).

Dollár, P., Tu, Z., Perona, P., & Belongie, S. (2009). Integral Channel Features. In *Proceedings of the 20th british machine vision conference*.

Erhan, D., Szegedy, C., Toshev, A., & Anguelov, D. Scalable object detection using deep neural networks. In *Proceedings of the 27th IEEE conference on computer vision and pattern recognition*. (pp. 2147–2154).

Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. (2010). Cascade object detection with deformable part models. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 2241–2248). IEEE.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the 15th IEEE international conference on computer vision*. (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 27th IEEE conference on computer vision and pattern recognition*. (pp. 580–587).

Gómez, M. J., García, F., Martín, D., de la Escalera, A., & Armingol, J. M. (2015). Intelligent surveillance of indoor environments based on computer vision and 3D point cloud fusion. *Expert Systems with Applications*, *42*(21), 8156–8171.

Guo, J., Liu, Y., Qiu, Q., Huang, J., Liu, C., & Cao, Z. (2019). A novel robotic guidance system with eye gaze tracking control for needle based interventions. *IEEE Transactions on Cognitive and Developmental Systems*, 1. http://dx.doi.org/10.1109/tcds.2019.2959071.

Hassannejad, H., Medici, P., Cardarelli, E., & Cerri, P. (2015). Detection of moving objects in roundabouts based on a monocular system. *Expert Systems with Applications*, *42*(9), 4167–4176.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (pp. 770–778).

Henriques, J., Rui, C., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), 583–596.

Hosang, J., Benenson, R., Dollar, P., & Schiele, B. (2016). What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(4), 814–830.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., & Fathi, A. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the 30th IEEE conference on computer vision and pattern recognition*. (pp. 7310–7311).

Kumra, S., & Kanan, C. (2017). Robotic grasp detection using deep convolutional neural networks. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 769–776). IEEE.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single shot multibox detector. In *Proceedings of the 14th European conference on computer vision*. (pp. 21–37).

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., & Liu, X. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, *128*, 261–318.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of 27th international conference on international conference on machine learning*. (pp. 807–814).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the 29th IEEE conference on computer vision and pattern recognition*. (pp. 779–788).

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In *Proceedings of the 30th IEEE conference on computer vision and pattern recognition*. (pp. 6517–6525).

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

Suzuki, S. (2017). Shuttlecock detection system for fully-autonomous badminton robot with two high-speed video cameras. In *Proceedings of the International congress on high-speed imaging and photonics*. (pp. 1–6).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the 29th IEEE conference on computer vision and pattern recognition*. (pp. 2818–2826).

Uijlings, J., Sande, K., Gevers, T., & Smeulders, A. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 14th IEEE computer society conference on computer vision and pattern recognition*. (pp. 511–518).

Wang, R. J., Li, X., & Ling, C. X. (2018). Pelee: A real-time object detection system on mobile devices. In *Advances in neural information processing systems* (pp. 1963–1972).

Wooden, D., Malchano, M., Blankespoor, K., & Howardy, A. (2010). Autonomous navigation for BigDog. In *Proceedings of IEEE international conference on robotics and automation*. (pp. 4736–4741).

Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. (pp. 129–137).

Yang, B. (1993). Subspace tracking based on the projection approach and the recursive least squares method. In *1993 IEEE international conference on acoustics, speech, and signal processing (vol. 4)* (pp. 145–148). IEEE.

Zhang, L., Chen, Z., Cui, W., Li, B., Chen, C., & Cao, Z. (2020). Wifi-based indoor robot positioning using deep fuzzy forests. *IEEE Internet of Things Journal*, 1. http://dx.doi.org/10.1109/jiot.2020.2986685.

Zhang, L., Varadarajan, J., Suganthan, P. N., Ahuja, N., & Moulin, P. (2017). Robust visual tracking using oblique random forests. In *Proceedings of the 30th IEEE conference on computer vision and pattern recognition*. (pp. 5589–5598).