AN SVM APPROACH FOR REALTIME SOCCER BALL AND ROBOT DETECTOR

[1] *Cheng-Shau Chiang* (蔣承劭), [1] *Chiou-Shann Fuh* (傅楸善),

[1] Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan,
E-mail: r06922131@csie.ntu.edu.tw fuh@csie.ntu.edu.tw

ABSTRACT

In this paper we present an algorithm for objects recognition based on Support Vector Machine (SVM). The scenario of our approach will be applied is in the Robocup SPL game, which is a soccer game composed of five robots in each team. Thus the ability to find the ball and tell where are our enemies is an important thing to the game, thus our approach is aiming to construct a real-time object recognizer which can recognize balls and robots. The input image data are first applied with some pre-processing like color labeling, then we look for the possible candidates, and finally the candidates will be classified into three categories. The performance of the method is discussed in an experiment.

Keywords: *Object Recognition, Support Vector Machine, Classifier*

1. INTRODUCTION

Object recognition is a popular research topic recent years, and with the progress of Machine Learning, the object recognition become more robust and more reliable, and hence have already been applied to many different tasks. So as in Robotics, especially humanoid robots, usually have to deal with real-time objection recognition, since the humanoid are usually designed to imitate human's behavior, like doing daily chores, interact with human, or playing soccer. Thus a real-time object detector is very important to the humanoids, since the detector is acting like the humanoid's eyes, so it is not feasible if the eyes cannot find anything or costing too much time. And for the humanoid robots nowadays, the power is still a big issue, causing the humanoid usually don't have enough computing power to perform complex computation, especially in our scenario, thus we have to make our algorithm more efficient, consume less computation power, and most important of all, to be real-time.

2. SCENARIO DESCRIPTION AND BACKGROUD KNOWLEDGE

In this paper, we are mainly dealing with a simple robot soccer game scenario, RoboCup Standard Platform League, which is a RoboCup robot soccer league, in which all teams compete with identical robots. The robots operate fully autonomously, i.e. there is no external control, neither by humans nor by computers. The current using robots in SPL is the humanoid NAO by SoftBank Robotics. NAO, the humanoid robots, except for the latest version V6 have only CPU for computation. Since NAO doesn't have GPU, we can't take the advantage of Convolution Neural Network (CNN), which is current the state of art in object recognition. Thus we turn into using support vector machine to achieve our goal, recognizing balls and robots.

Fig.1: RoboCup SPL Game and NAO humanoid.

2.1 CAMERA CALIBRATION

Since we have to know where the ball exact is in the real world, that is we have to know the distance between our robot and the ball, thus we have to calibrate our robot's cameras.
But since the robot's cameras are installed on its head, we have to take its body's transformation and rotation into consideration, and thus the ordinary method of using black white chessboard for camera calibration doesn't work well. We turn into using the method proposed by Team B-Human [1]. The extrinsic matrix is mainly composed by the neck joints' rotation matrix and the robot's feet to torso's rotation matrix, thus we want to find the correct parameters that will fix the camera. We take 7 parameters into account, neck joints' yaw and pitch, torso's yaw and pitch, and the correction of robot's position which will include robot's transformation and rotation.
For the first step of calibration, we have to manually collect points on the field lines (Fig 2.), and since we know the dimension of the field and the coordinate of field lines, we could find the parameters that will make the coordinates of the points we collected fit the field lines' coordinate.
By using Levenberg–Marquardt algorithm, we can get the correct parameters.

Fig.2 : Camera Calibration [1]

2.2 IMAGE PRE-PROCESSING

As mentioned before, our goal is to achieve real-time object recognition, and because of the limitation of our platform, we can't afford pixel-wise computation, we thus perform color labeling on the input image.
We first set up the ROI with respect to the Robot head's current pitch and yaw, and set index onto the image (Fig. 3). Then either vertically or horizontally scan in a straight line along the indexes, and judge the color by the pixels HSV value. The category of color's HSV value is tuned manually.
After performing color labeling (Fig. 4), we then transform our raw image into a sparse color map, thus reduce the complexity of our task by transforming from using pixel value into using discrete color label map.

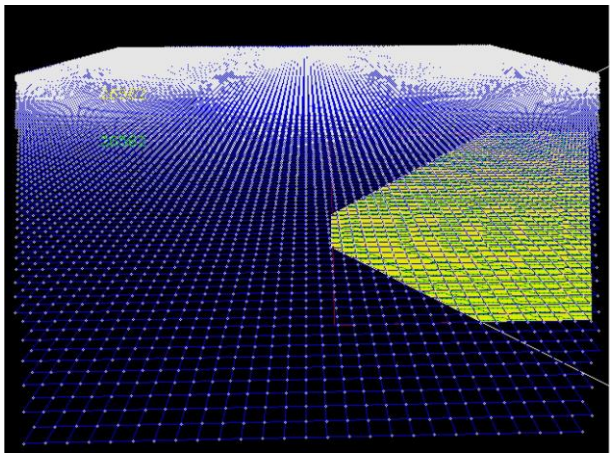

Fig.3: Image's index

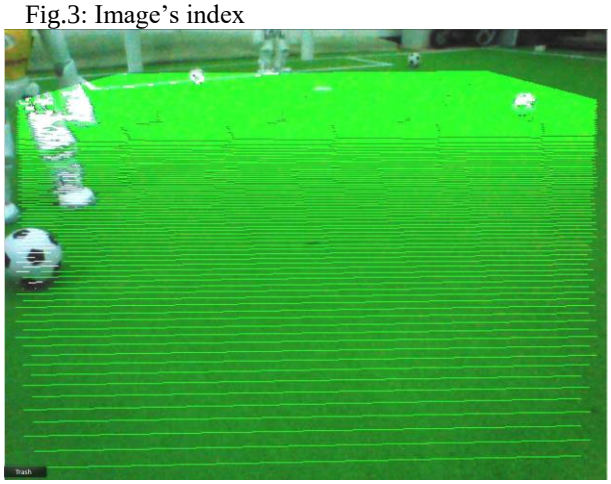

Fig.4: Horizontal Color Label

2.3 YOLO: UNIFIED, REAL-TIME OBJECT RECOGNITION

YOLO is refreshingly simple: see Figure 5. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. [2]

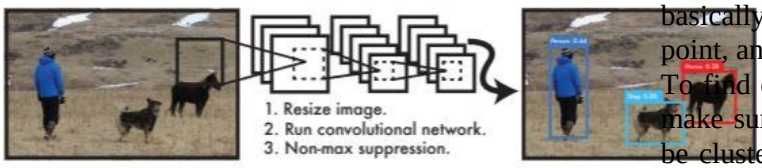

Fig 5: The YOLO DETECTION SYSTEM [2]

3.DESIGN OF BALL AND ROBOT DETECOR

The workflow of our design will be discussed in this section, first will show how did we collect the ground truth, then how do we find out the possible candidates and finally how do we train the classifier

.

3.1 GATHERING GROUND TRUTH

In order to train the classifier, we have to be able to find the ground truth of the input image. Since this could be done offline, we can use many powerful tools. Taking advantage of YOLO, a real-time object detection system using CNN, all we need is to have an annotated dataset then we are able to train YOLO. And thanks to team SPQR, one of the RoboCup SPL team, they have published an annotated dataset, thus we are able to train our own YOLO to help us label the ground truth. (Fig. 6).

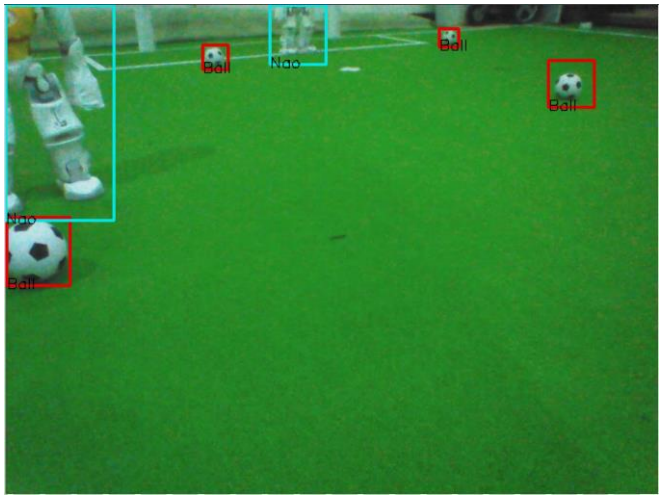

Fig.6: Ground truth labeled using YOLO.

3.2 GENERATE CANDIDATES OF BALL AND ROBOT

After the image pre-processing, we have got the color label of the input image, the color label is basically a line segment, with the start and end point, and the color of the segment.

To find out the possible candidates, our goal is to make sure that the color label of an object should be clustered into same candidate. In our scenario, the robots and ball will be on the soccer field, thus the green segments could be ignored. In our systems, we have built lines and curves detector which could tell whether the white color segments are white field lines or not, although if the input images are blurred, due to the robots' movement, our lines and curves detector may be malfunctioning, eliminating white field lines still reduce the amount of color segments need to be processed.

We perform a greedy approach to look for candidates by scanning the color label map from bottom to top and from left to right, and will calculate each segment's distance to the candidates, the segment will be merged into the nearest candidate, and if there are no candidates or the smallest distance is not smaller than the threshold, a new candidate will be generated. The candidates we got from this still need some further processing. (Fig 7.)

Fig 7: The candidates got from the horizontal scanned color label.

The result in Figure 7 could be enhanced by further processing. Since the result in Figure 7 is using only horizontal scanned color label, we perform vertical scanned on the image to get the vertical scanned color label map. By performing the process of generating candidates, we could get another set of candidates. Then we merge two sets of candidates into one set. And discard the candidates which cannot be our desired objects by its real width, since in our scenario, the width of

robots and ball are fixed, we can use this property to reduce the number of candidates. (Fig 8.)

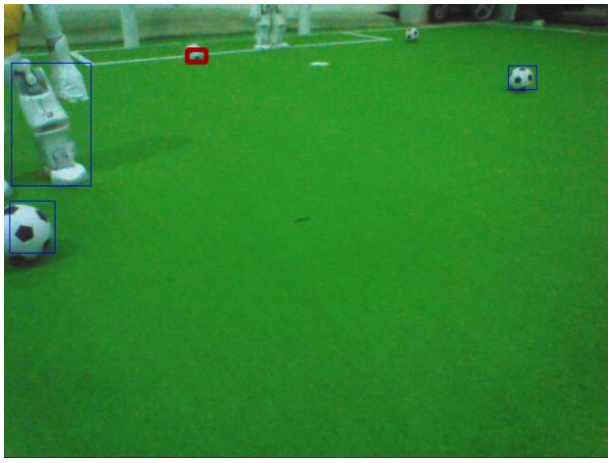

Fig 8: Refined candidates, the red rectangle is discarded due to its size.

Figure 8 showed that if the objects are placed in the field, we could get desirable results, but if the robots captured images out of the field or the line and curve detector is not functioning, we may capture the candidates that are neither robots nor balls. (Fig 9 and Fig 10)

The whole procedure of generating candidate boxes will take at most 10 milliseconds on a 640 by 480 image.

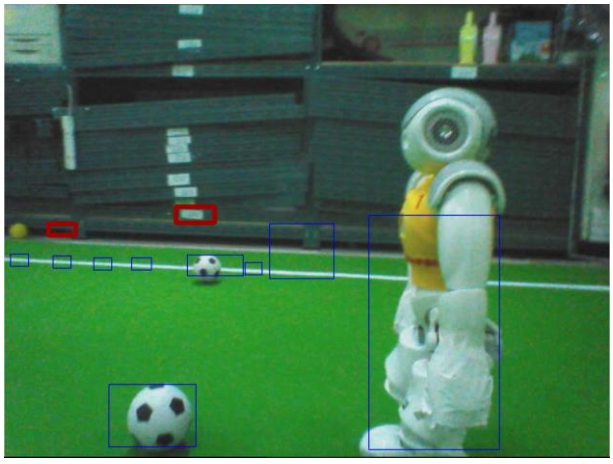

Fig 9: Results due to our line and curve detector malfunction

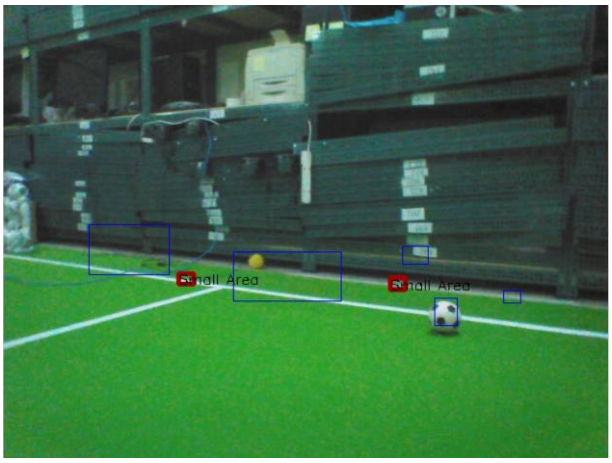

Fig 10: Results due to out of field color segments

3.3 COLLECTING TRAINING DATA

Now we are able to find the candidates, next step is to collect the training data and train our classifier. What we have now is the ground truth labeled by using YOLO, and candidates of ball and robots, thus the candidates could be classified into ball, robots or others, depends on the Intersect Over Union (IOU) with each bounding box created by YOLO.

3.3.1. R-Tree

To fulfill this task, we chose to use R-Tree which fit our requirements perfectly, each of the candidate are inserted into the R-Tree as a node, each bounding box of the ground truth is set to be a query box, if the candidate is totally inside the query box, then it belongs to the class of the query box, that is robots or balls, if the candidates only have intersected with the query box, then we take the candidate if its IOU is acceptable.
After queried all of the ground truth boxes, the candidates that have not been assigned a class will be labeled as Others.

3.3.2. Feature Selection

After classifying the class of the candidate, next step is to extract the features from the bounding boxes.
Our goal is to build a real-time object detector, and due to our platform's computation power limit, we can't use costly Haar feature or HOG, the former include integration of pixel value, the latter include differentiation. A RoboCup SPL team, Camellia Dragon, come from Aichi Prefectural University, have done implementing ball detector

using these 2 techniques, and the computation time needed are at least 44 milliseconds which are unacceptable.

In our work, we choose to use statistical features, e.g., the average length of each color, the portion of each color, and the length/width ratio of each bounding box.
And we also divide each bounding box into 16 grids, inspired by Local Binary Pattern, each grid will show the most representative color of that grid, depends on the portion of each color. We wish to find the pattern of each class. (Fig 11)

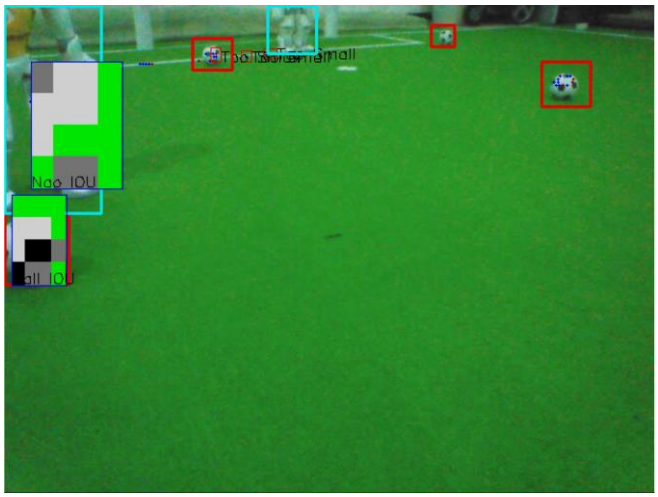

Fig 11: Visualization of 16 grids. Compare to Figure 8, we could find that the pattern of the soccer ball can be represent by the grids.

3.4 CASCADE CLASSIFIER

We found that in our scenario, directly using multiclass classifier won't get desirable accuracy. Thus we tend to use a cascade classifier, it is like a decision tree, first we separate one class from the other, then we deal with the rest, by doing this we may found the real class of an object. The difference between cascade classifier, cascade multiple binary classifier, and directly use a multiclass classifier is that due to the SVM's characteristic, the hyper plane of multiclass classifier found may be different, because the parameter could be tuned each binary classifier, like the gamma value for the SVM using RBF kernel, and the C, which is a penalty of making wrong classification, for soft-margin SVM. By using multiple binary classifier, we may simplify the multiclass classification problem into multiple binary classification problem, and thus get a better result. (Fig 12)

4. EXPERIMENT RESULTS

.
Out experiment environment is SoftBank's humanoid NAO V5, its CPU is Intel ATOM Z530 1.6GHz, and it has 1GB RAM.

The procedure of candidate takes around 10 milliseconds at most, take around 5 milliseconds to gather the features of all candidates at most, and also take 10 milliseconds for inference for all candidates. It is depending on the scene Robots see. So it takes 26 to 27 milliseconds for the whole procedure.

We used lib-SVM [3] to train our classifier, the accuracy of using 1 classifier is about 75%, we have tried out using method like decision tree, we first train a model that will classified the ball out of robots and others, and then train another classifier to classify robots and others, the accuracy of this method could be around 83% in our test case.

The reason of using the decision tree is we found that the robots is difficult to be distinguished from others, but balls can be easily distinguished from robots and others.

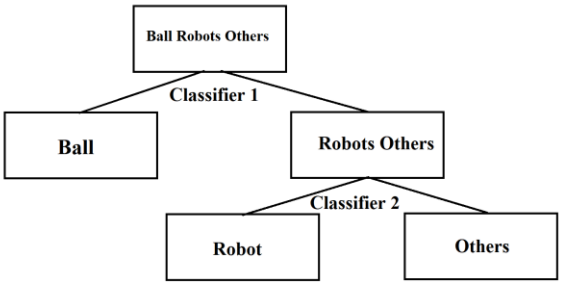

Fig 12: Ideal Classifier Tree

5. DISSCUSSION

The results we have got are not so acceptable.

First, the time of generating candidates is too long, but it is due to the environment of our field, the white wall you may see at figure 7 will have lots of white segments and also there are lots of things that usually will not be placed beside the field standing beside our fields, which are very time consuming on the process of generating candidates since they are not green, and thus cannot be

skipped. That means we will have a lower latency when we are on the real competition field, since we may generate less candidates, due to the tidier environment which means less non-ball and robots objects will be seen by the Robots.

Second, the accuracy needs improvement, now we had only around 80% of accuracy, the major problem is the robots cannot be separated from other object, this may have several reasons, first is that "Others" should contains the objects that are not robots and soccer ball, and the robots are not like soccer ball, compared with soccer ball, the robot has no repeat patterns, and looks different if the viewing angle varies. So, if we did not have enough data we may not be able to classify all the robots. Another problem is that if the candidate's bounding box is too small, often there will only have lots portion of white, it is difficult to classify which class the candidate belongs to in such case. But if we discard candidates that are too small, we may not be able to detect the balls as before, since the balls are often have small size and it is difficult to find the threshold of what size's candidates should be discarded.

Last, the proposed detector's performance is totally based on the quality of the color labels, in our test case, sometimes the leg of the robot or the bottom of the balls will be labeled as green due to shadow's influence. Thus our system need to be re-train if the lighting condition is different, or we have to improve the quality of our color labels.

6.CONCLUSION AND FUTURE WORK

In this paper we have proposed a method combining using accurate object detector and r-tree to gather training data, and using cascade classifier to improve the accuracy of classification, and a color-based object detector. Though the result might not be good enough, but since we have very limited computation resources, and we are performing a near real-time process, we have to achieve the balance between the accuracy and the computation time.

For the future work, we may focus on improving the quality of the color label, since we think that maybe with the better color label, the statistical features we used for SVM may perform in a better way. And we may try to use the localization data we already have to further discard the candidates that are out of our field.

REFERENCES

[1] Team B-Human, "CodeRealease 2017", P.24

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015

[3] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm