Automatic Dart Detection

Jonathan Schlink
Walter Scott Jr. College of Engineering
Colorado State University
Fort Collins, CO, United States
jschlink@rams.colostate.edu

***Abstract*—This paper explains the Automatic Dart Detection project, including the object detection model, hardware, and extra software used.**

I. Introduction

The Automatic Dart Detection project is a full prototype project. The idea of the project is to create an object detection algorithm that works on a raspberry pi device with a pi camera to detect metal darts on a cork dartboard. Using a raspberry pi model 3B+ with pi camera module v2, the complete cost of the product comes to about $150 including darts and dartboard. Comparing this price to current automatic scoring steel tip dartboards, which run on the low end of $800, it is a great reduction. I decided to use the yolov5s model, which is a small version of the yolov5 object detection model. I used the small version due to it having to run on a raspberry pi with limited resources, so it was the best option. From this model, the detection detects with a confidence of about 95% for a dart, and a confidence of about 95% for the tip. With this information it is very possible to map the location to an image to find the score of the dart.

II. Related Work

A. Prodigy Automatic Scoring Steel Tip Dartboard System

Uses a patent pending bi-ocular dart recognition technology with infrared lighting and location algorithms to determine score accurately. My approach differs from this work because I am using a raspberry pi, and the total cost of only $150 compared to the current price of this product of $999. My project is similar to this product because I plan to implement an app and wi-fi integrated connectivity, which the prodigy dartboard includes.

B. Darts Score Detection by Kawasaki-Kento

Dart score detection using an SSD-Mobilenet with transfer learning. This project determines the score of the dart using vectors from the bullseye.

III. Methods

A. Embedded System

For this project, I started by first looking into different embedded systems. My first choice was the Raspberry Pi Model 4 model B, which had a quad core CPU with speeds of 1.8Ghz, with the option of getting up to 8Gbs of RAM. This product was out of stock for many months, so I decided to go with the Raspberry Pi Model 3B+, which has a 1.4GHz quad core CPU, but with only 1GB of RAM. This became to be a problem later.

B. Object Detection Model

When choosing an object detection algorithm, I was stuck between two models, an SSD, or the YOLO model. I was stuck between these two because both had lightweight options, which was a must for my project being run on a Raspberry Pi. I ended up choosing the YoLoV5s model by Ultralytics, due to its ability to have much faster speed and efficiency, with less accuracy than the SSD.

C. Dataset

When looking online, I did not see many dart datasets, so I decided to create my own. My dataset included about 500 images, with about 5-10 images per score segment, along with some extras of darts outside the scoring zone. The images I took used only one dart at a time.

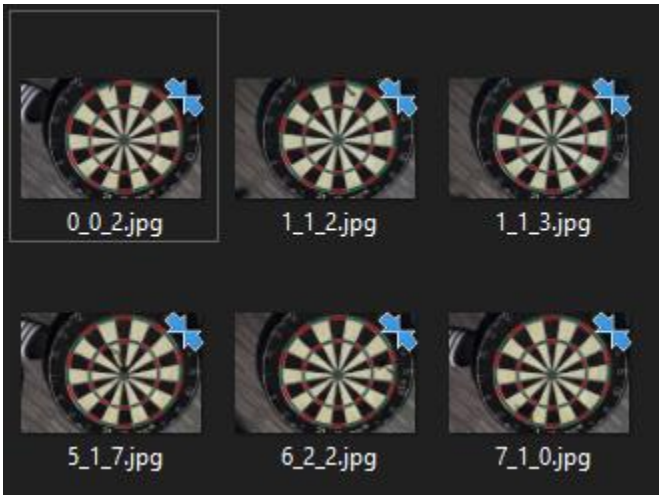

Figure 1: Example of Dart Dataset

I then used the program ImgLabel to create bounding boxes for each image. When creating the bounding boxes, I created one for the entire dart, and another for where the tip of the dart is located, naming the score it should be. This program then exported an XML file with the location of the bounding boxes. The next issue I ran into was needing to change the bounding box files to txt files, with different format taken by the yolov5 model. When noticing this, I also noticed that a YoLov5s model with over 60 classes would not work on only 500 pictures, so I switched to only two classes, the full dart, and the tip of the dart.

```
<?xml version="1.0"?>
- <annotation>
    <folder>images</folder>
    <filename>0_0_2.jpg</filename>
    <path>/home/schli/darts/images/0_0_2.jpg</path>
  - <source>
      <database>Unknown</database>
    </source>
  - <size>
      <width>720</width>
      <height>480</height>
      <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
      <name>0_0</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>348</xmin>
        <ymin>5</ymin>
        <xmax>371</xmax>
        <ymax>19</ymax>
      </bndbox>
    </object>
  - <object>
      <name>Dart</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>349</xmin>
        <ymin>4</ymin>
        <xmax>384</xmax>
        <ymax>55</ymax>
      </bndbox>
    </object>
</annotation>
```

Figure 2: XML file from ImgLabel Export

```
labels > test > 2_2_0.txt
1    1 0.773 0.844 0.026 0.029
2    0 0.792 0.911 0.069 0.177
```

Figure 3: YoLoV5 Bounding Box .txt File Format

D. Additional Software

After having a working prototype, I also created software to create a dataset for my model. I used Python and took pictures every time a key was pressed. When the key was pressed, I input the score and score multiplier. The program then saved the image named: score_score multiplier_index, where the index was the number of images with the same name. I also created a Python script to create text files of the labels from the XML files I had in YolOv5 labels format.

E. Camera

For a camera I had many options, but ended up going with the Pi Camera Module v2, due to its very low price and 8MP camera. This camera also included many abilities with the raspberry pi that others may not have. This camera also had a decent field of view, which was required to capture the entire dart board from only 3 feet away.

F. Abbreviations and Acronyms

SSD: Single Shot MultiBox Detector

YOLO: You Only Look Once

IV. Experiments

For my project, I first trained my YoLoV5s model on my desktop, due to the Raspberry Pi not being able to withstand it. This trained model had very good metrics, so I decided to go further.

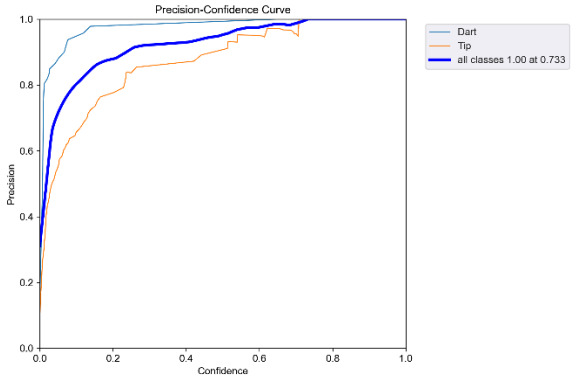

Figure 4: Precision-Confidence Curve of Model

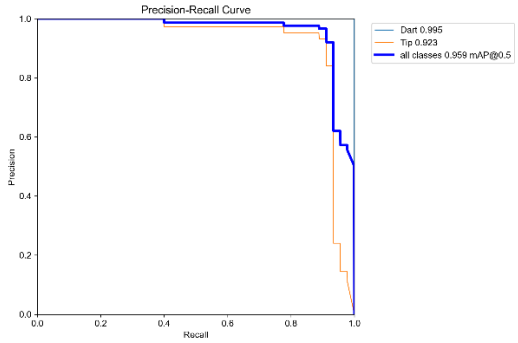

Figure 5: Precision-Recall Curve

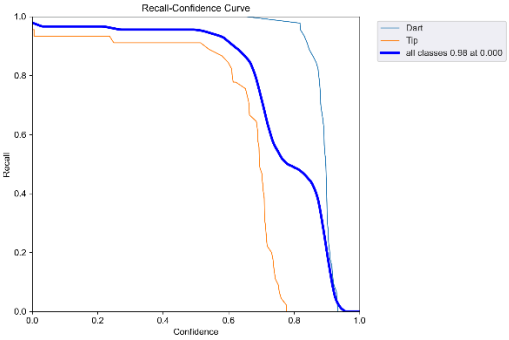

Figure 6: Recall-Confidence Curve

Figure 7: YoLoV5s Validation Training Prediction

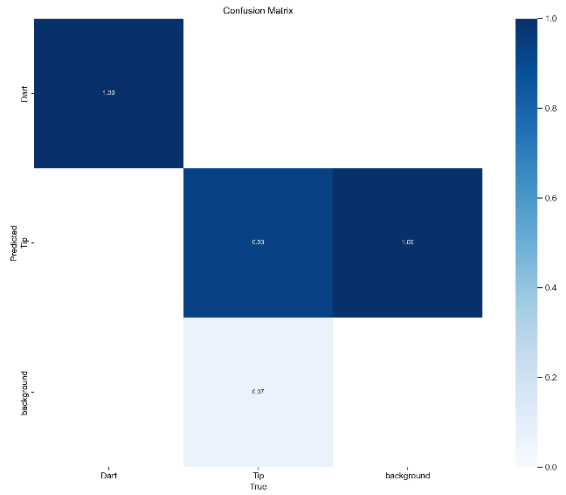

Figure 8: YoLoV5 Confusion Matrix

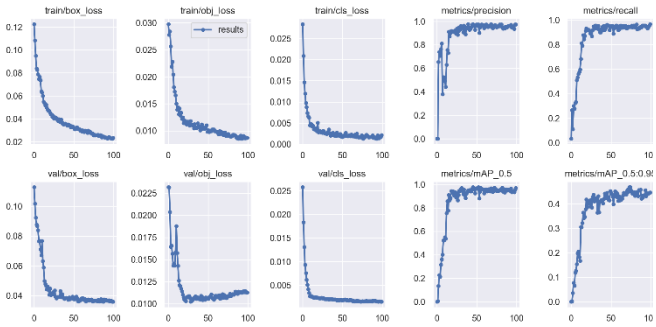

Figure 9: YoLoV5 Model Training Results

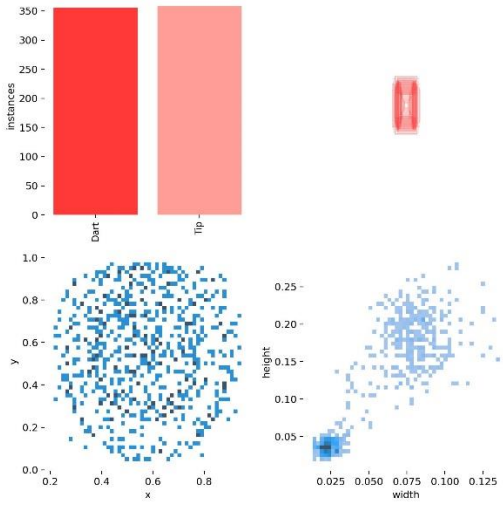

Figure 10: Label Map for Detection

With this trained model, I implemented it into the Raspberry Pi with some simple code to take an image and process it through my model using the detect.py code supplied with the model, exporting an image of the object detections. This image was then saved in a folder for future use. I was wanting to use cv2 and torch to create a live feed of images going to the model but was unable to get torch working correctly on the Raspberry Pi due to incompatibilities. I instead used the Picamera2 library to capture a frame when a key was pressed, display it in a cv2 window, and run the detect command in the Python code. When the key 'z' was pressed, the script would end and close all cv2 windows. This worked well, other than having an elapsed time of about 20 seconds per detection. My hypothesis as to why it took so long is that the Raspberry Pi 3B+ did not have enough RAM to run the model faster. With this new trained model, it was able to detect multiple darts and dart tips on a single image, although some shadows from darts were also labeled as dart tips with low confidence.

A. Future Work

For future work, I could implement a program to map the detected dart time location onto an image, and then find the score of that dart. If confidence of the detection was low, it could prompt the user if it was correct. Another future work possible would be to increase the size of my dart dataset for better training which will improve the detection during real time games. One thing to change on this project would be to use a Raspberry Pi 4B instead of the 3B+ to improve performance and efficiency of the model. Looking into how one could implement a real time camera capture and run the model during that would also be an improvement to my project. Once a working real-time algorithm was working, one could create a simple script for playing dart games like cricket, 301/501, etc. and use the algorithm to score said game.

V. Conclusion

In conclusion, my YoLoV5s model was successful in creating an automatic dart detection model for use with steel tip darts. The model itself was very successful, but execution of said model was not. Starting this project, I wanted real time dart detection while playing a game like Cricket, 301, etc. This became impossible due to the issues I had with torch and cv2 on the Raspberry Pi system. With this trained model, one could implement these things on any embedded system with their own Python code. Overall, the success of this project demonstrates the potential for object detection algorithms to enhance the accuracy and efficiency of dart detection, from training tools for professional dart players to automated scoring systems in dart tournaments for much cheaper than currently available.

VI. References

[1] G. Jocher, "YOLOv5 by Ultralytics." May 2020. doi: 10.5281/zenodo.3908559.

[2] "YOLO: Real-Time Object Detection." https://pjreddie.com/darknet/yolo/ (accessed May 05, 2023).

[3] "How to Train YOLO v5 on a Custom Dataset," *Paperspace Blog*, May 10, 2021. https://blog.paperspace.com/train-yolov5-custom-data/ (accessed May 05, 2023).

Demo Video: https://drive.google.com/file/d/1YEac3cYQ-BCC-h082iTpo_0fNzPLm8Wr/view?usp=share_link