Dart Score Detection

Yurii Voievidka

June 13, 2024

1 Task Statement

Who doesn't love a great game? Maybe during Beer Friday or after a workday. But nobody likes to calculate. Let's delegate it to a PC and let it decide who is the winner of the dart game.

Analyze the images of the darts game, find the score, and determine the winner (red or green) using methods of computer vision. I can use algorithmic, deep learning approaches or a mixture of both to demonstrate my skills.

Deliverables: - Working code (it should be possible to install all the necessary packages to run it from requirements.txt using pip). - A report [in English] on the principle of operation of your solution (better with illustrations and explanations of the chosen solution), as well as a discussion of the results obtained. - The initial game images with found scores.

2 Non-Deep Learning Approach

The first idea that I came up with was to use OpenCV library capabilities to segment the image. This includes the following steps:

1. Detect the boundaries of each dart (green or red) as a rectangle.
2. Calculate the midpoint of that rectangle and consider it as the point that was hit.
3. Calculate the radius and center of the biggest outer circle.
4. Divide the radius by the number of sectors (8 in our case).
5. Assign points depending on how far the hit point is from the center.

2.1 Preprocessing

Figure 1: Initial image of boards

To work with arbitrary images, we will first resize them proportionally to 1024 px width.

2.2 Cropping Spare Parts

In further work, I will use the OpenCV library to detect circles in the image. To make it more accurate, I will grayscale the image and find the boundaries of the black rectangle to crop the image to its size.

Figure 2: Cropped image

2.3 Detection of Red and Green Darts

After cropping and resizing, we want to detect boundary boxes for darts on the board. Fortunately for us, they are quite distinguishable in the image.

In Python, we can create a range for green color and a range for red color to detect. First, we will convert the image to HSV format as it is commonly more useful for color-based image processing. Then I will define a red mask as a range from [0, 120, 70] to [10, 255, 255], and the same for the green boundary: [36, 45, 45] - [86, 255, 255]. Using the function 'opencv.findContours', I will find bounding boxes of red and green tails and draw them on the board.

Figure 3: Detected Green and Red boundaries

2.4 Detection of Dashboard Center and Outer Circle

Now we need to find the radius and center of the bullseye. For this purpose, I used the OpenCV method 'HoughCircles'. The Hough Circle Transform is a feature extraction technique used to detect circles in an image. It works by converting points in the image space to a parameter space, where the circles are identified based on their radius and center coordinates.

First, I will grayscale the image and add blur to the image. According to the documentation, it helps to detect circles more clearly.

Figure 4: Grayscaled and blurred image

After applying the ‘opencv.HoughCircles‘ method to this image, we will extract all circles in this image.

Figure 5: Detected circles

I want to extract only the outer circle with the biggest radius.

Figure 6: Boundaries and dartboard radius detected

2.5 Simple Math

To calculate the score, we can divide the dartboard into 8 sections.

Figure 7: Score calculation

As we know the radius, we can divide it by 8 and know the range of 1 section. Depending on how far the center of each box is from the center, we will be able to calculate the score.

2.6 Results

And so, here are the scores for each picture.

Figure 8: Board 1

Figure 9: Board 2

Figure 10: Board 3

The algorithm detected the winner correctly, however, the scores are different. The true score of the first board is 130 for green and 190 for red. The true score of the second board is 60 for green and 160 for red. The true score of the third board is 150 for green and 200 for red.

Problem: We calculate the score based on green and red boxes which are the tails of darts. We need to calculate scores based on the heads of darts. Detection of heads is a tedious task. Each head has a very dim color and we can't detect them using simple OpenCV tools. Here comes the deep learning approach for this task.

3 Deep Learning Approach

As I mentioned in the previous section, we should detect dart heads somehow to improve accuracy. I propose to take the YOLOv7 model and train it to detect dart heads. We will have 2 classes: green-dart and red-dart which represent bounding boxes around each dart head. The problem is that we have only 3 pictures of dartboards... But to train YOLO you need to have much more training data...

So I bought a dartboard and created a training dataset myself =)

3.1 Data Annotation

To fine-tune YOLOv7, I used the Roboflow framework which helped to create a training dataset of 400 images faster by manually drawing bounding boxes.

Figure 11: Data Annotation process

After annotation, I used the hosted YOLOv7 API to detect darts in the image.

3.2 Dart Detection

As a result, after applying YOLO we will have the following bounding boxes.

Figure 12: Image

3.3 Results

The results that YOLO produced were surprisingly worse than I expected. Board 2 was detected and calculated perfectly but others had issues.

Figure 13: Board 1

Figure 14: Board 2

Figure 15: Board 3

4 Conclusion

In the end, I can say that the first method of boundary detection is more robust and showed greater results. On the other hand, the YOLO method detects dart images more precisely, but due to the image color scheme and overlapping dart heads, it can be somehow hard to detect the correct score.