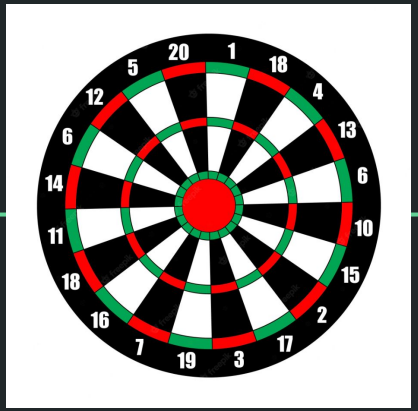
DIGITAL IMAGE PROCESSING PROJECT
Dartboard Scoring (Project ID - 26)

Team - It’s Something
Rahul Garg - 2020115006 (CHD)
Chinmay Deshpande - 2020102069 (ECE)
Sushil Kumar Yalla - 2020102071 (ECE)
Geet Dassani - 2020102001 (ECE)

Git repo link - Project
Mentor - Eshan Gupta

Introduction

This project demonstrates an innovative technique for automatically segmenting the dart board into each scoring section and identifying the placement of darts fired into the board using photos captured from a stationary camera and generating score of the player.



In the sport of darts, players throw projectiles towards a circular target that is fixed to a vertical platform. There are several locations on the target that correlate to various point and multiplier values. It would be really advantageous if this project allowed the players to keep score more readily.

Problem Statement

Dart board scoring is often done by eyeballing the board and calculating the points. Nowadays, even plastic dart boards with inbuilt sensors are available, but the sensors in the boards are highly prone to damage from dart impacts over time. This leaves image processing methods. The problem dealt with in this paper is thus to score a dart board using image processing techniques. We should be able to accurately determine the location of a dart on the board and score it accordingly.

Project Solution - At a glance

The point of this project as a whole is to create a point-to-number pixel mapping, which maps each pixel to a value relating to the number of points the each region on the dart board is worth. The detection of each region is done is based on the intensity values obtained from the colored image.

Deliverables -

- Regional masks for two-point and three-point regions (binary mask)
- Single-point region segments on the board (binary mask)
- Filters for removing any unnecessary glare and finding the dart's location

Dart Board

The bullseye is formed by two concentric circles in the middle of a conventional dartboard, which is circular in shape and divided radially into twenty parts. These sections alternate in colour between light and dark regions, each of which represents a different point value. The board also has two outside rings with alternately green and red spaces that split each scoring area into regions with single, double, and triple point values.

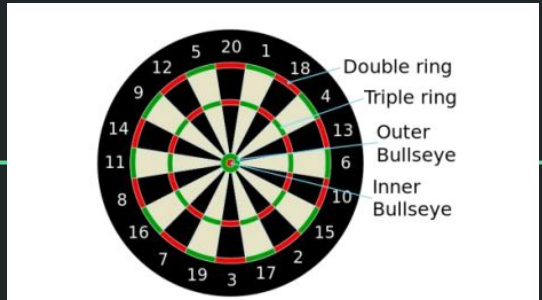


Fig. 1. Regulation Dart Board

Algorithm Segmentation

- Region Segmentation: to give each pixel in the dart picture a point value.
- Dart localization: must find the dart in the picture and figure out which pixels stand in for the dart tip's contact with the board.

Region Segmentation

- Input a image I, convert image into grayscale.
- Threshold the image using Otsu's Method.
- Taking the intensities of red and green channel of original image and create separate masks
- Thresholding again using Otsu to get just a mask of red and green region along with the bullseye point.
- By here we got the double and triple multiplier region along with bullseye.

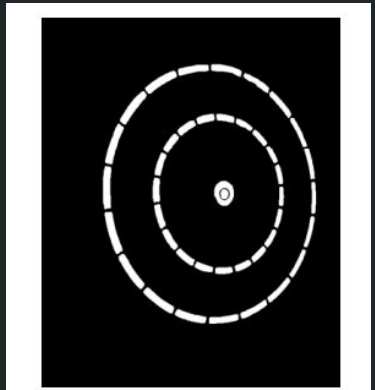
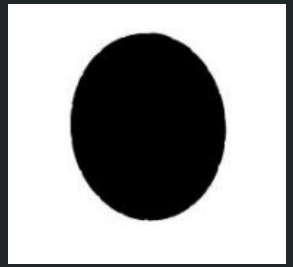


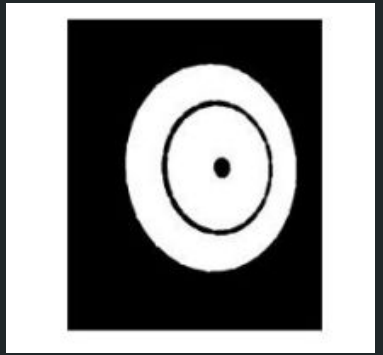
Fig. 2. Red + Green Pixel Mask

Now we extract the multiplier region from our red-green pixel mask.

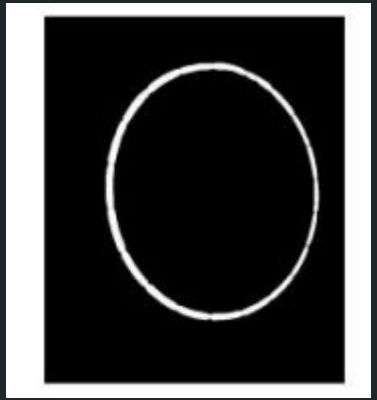
- Dilating along with the eroding is done on binary image for closing and removing the space between red and green masks to get continuous rings.
- To produce a binary map of the full scoring region, the holes are filled inside the resultant continuous rings. The zero multiplier region is the opposite of this region (outside the scoring area).



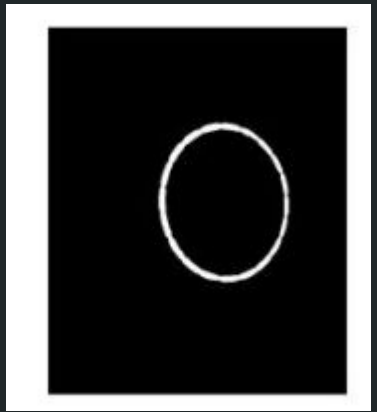
- The single multiplier region is extracted by subtracting the red and green masks from the scoring region.



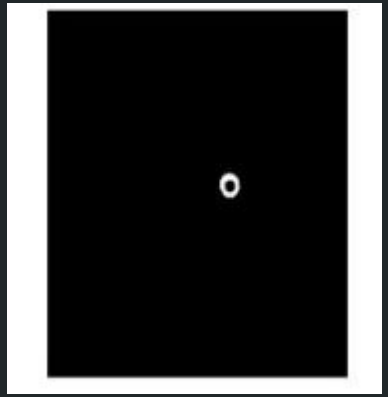
- The double multiplier region is extracted by filling the holes in the single region and subtracting from the entire scoring region.



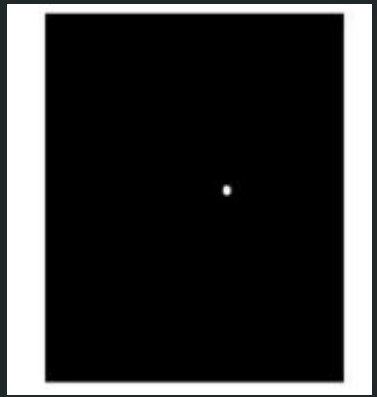
- Triple Region: Add all the masks i.e outer bullseye, inner bullseye, double ring, single ring, outer area of board. Now inverting this image will get you Triple Region.



- Outer Bullseye: Average of the green mask to get the centre and nearest white pixel from it is inner bullseye.



- Inner bullseye: Average of the red mask to get the centre and nearest white pixel from it is inner bullseye.



- Canny Edge detection operation on grayscale image of I, to create strongest edges
- Standard Hough Transform is calculated of this binary edge map. The peaks of this corresponds to the angle of the metal dividing bars that bisect the board
- Using the dart board's standard numbering pattern, a point value is determined for pixels that fall within each straight line angle.

The Bit Masks

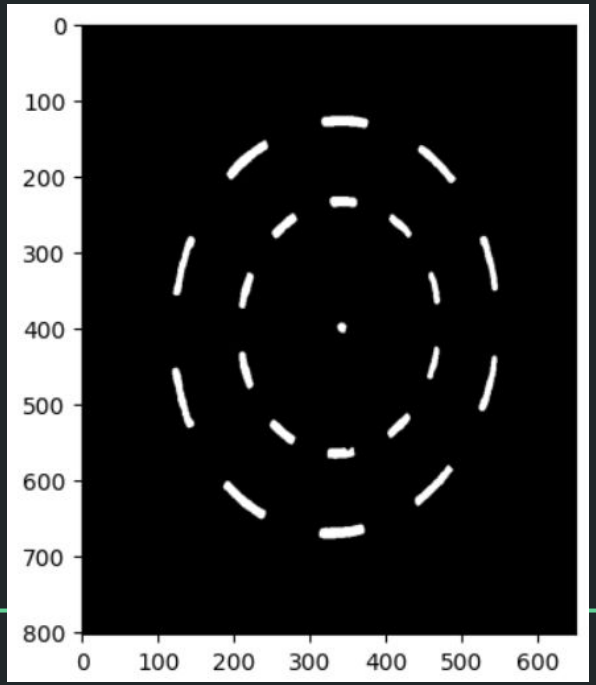
- In order to be able to properly assign the scores on the dartboard, we require the bit masks of the individual regions.

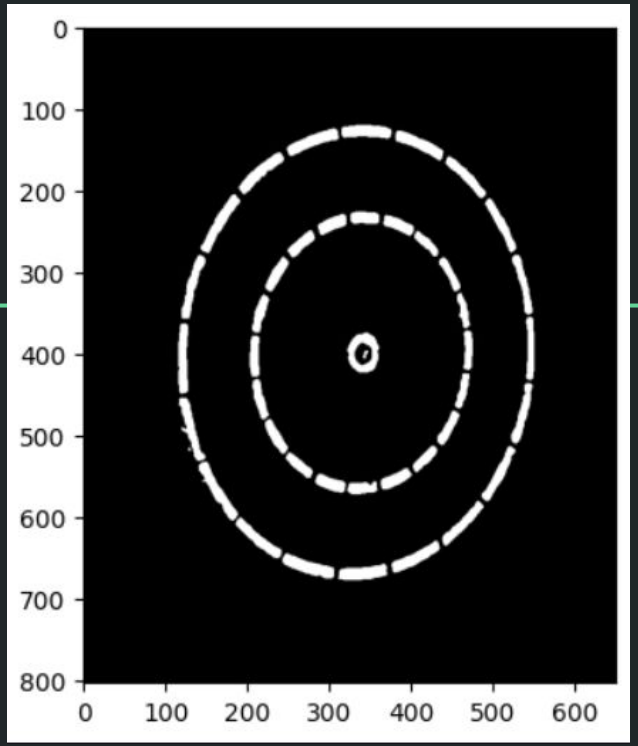
- This is because each region has its own multiplier or points value associated with it, such as the triple scoring ring multiplies by 3 or the inner bullseye gives 50 points

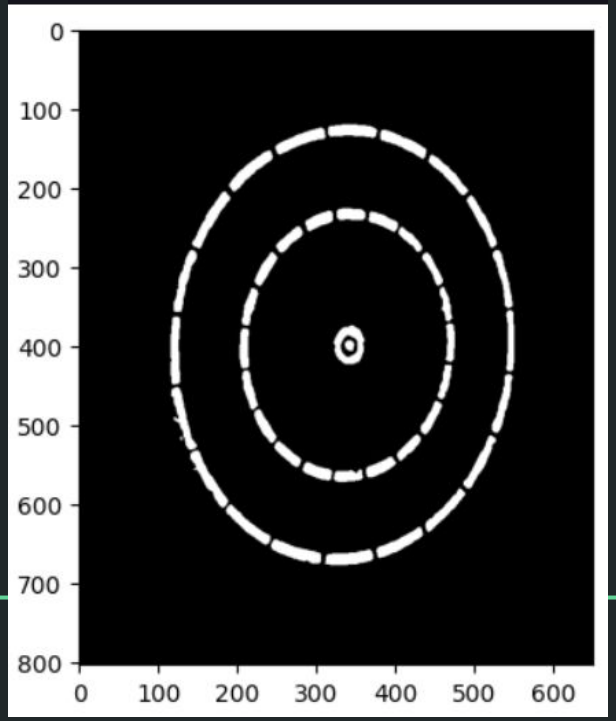
---

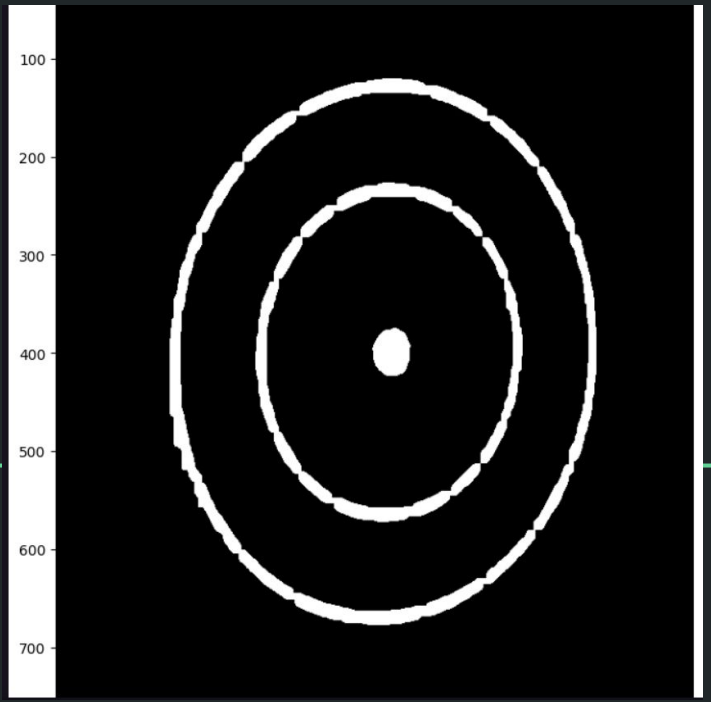
- This is done by thresholding to obtain a binary image. The process followed is the one detailed in the previous few slides on region segmentation

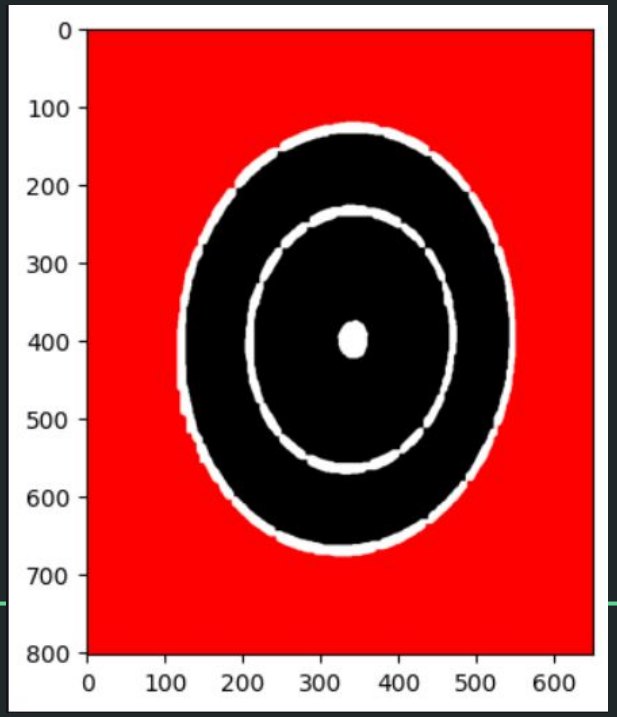
These are the bit masks we obtained

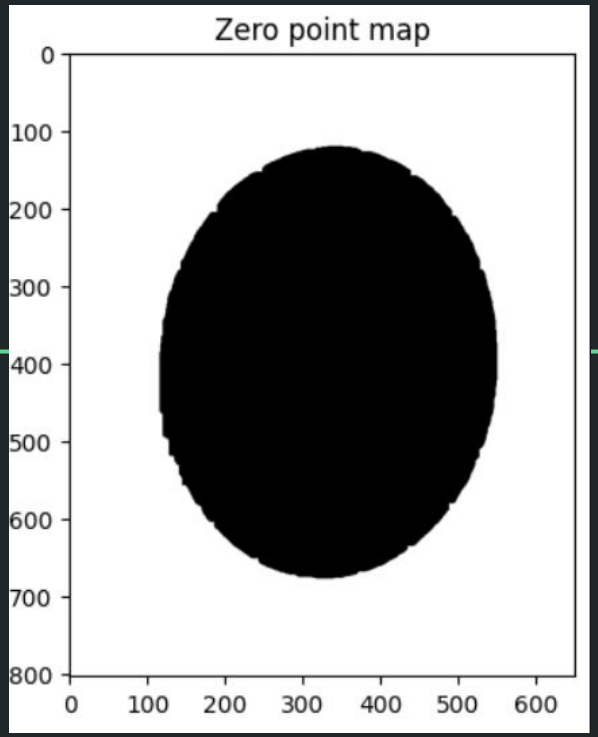

0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

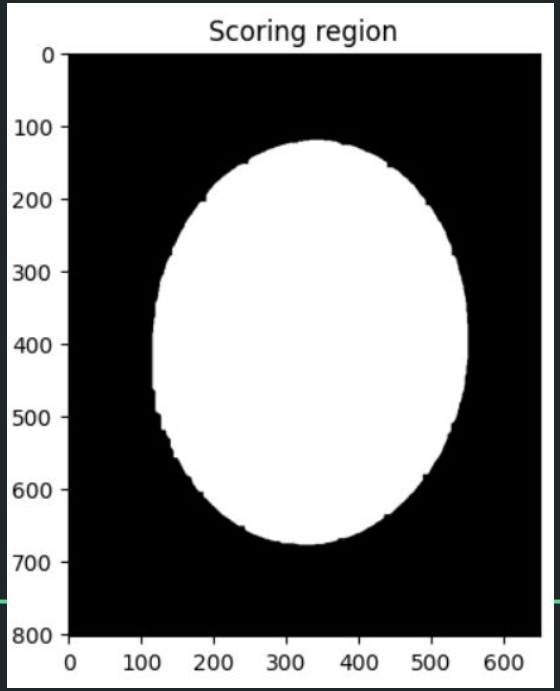

0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

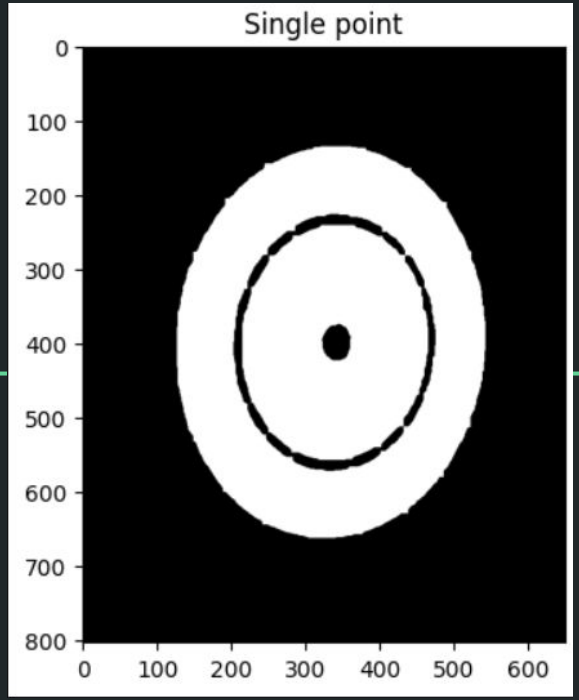

0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

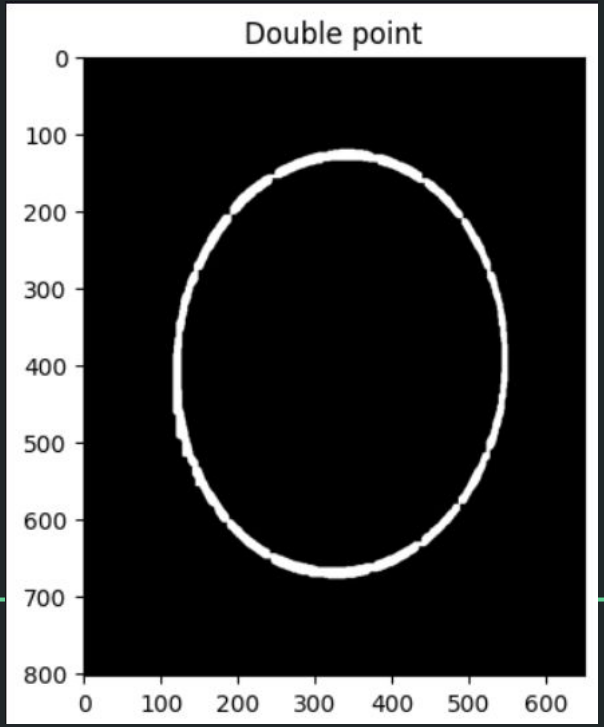

100
200
300
400
500
600
700

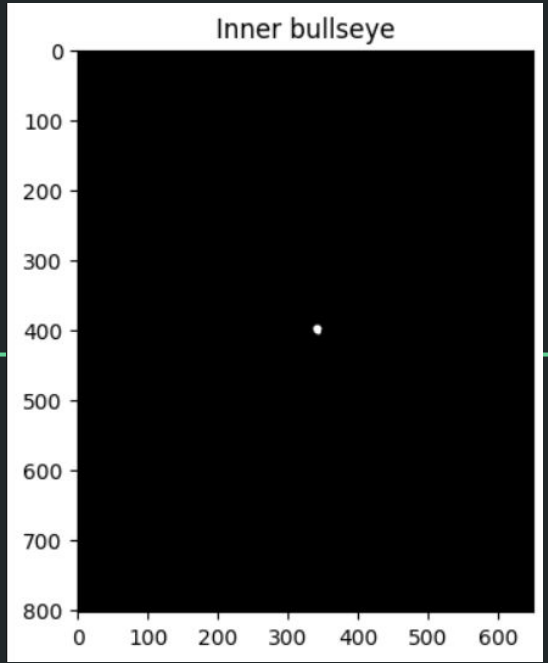

0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

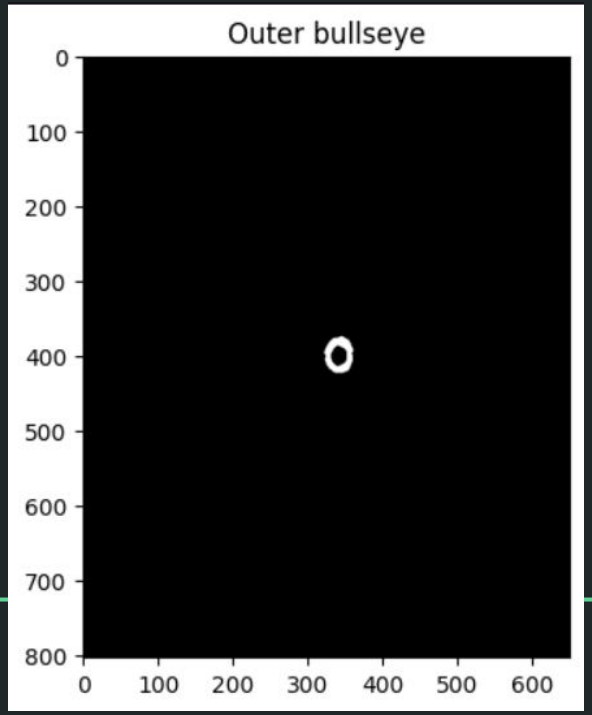

Zero point map
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

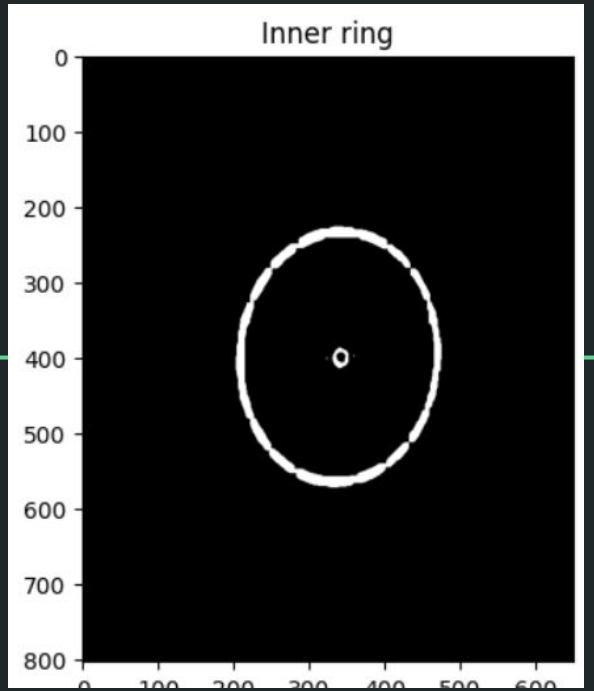

Scoring region
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

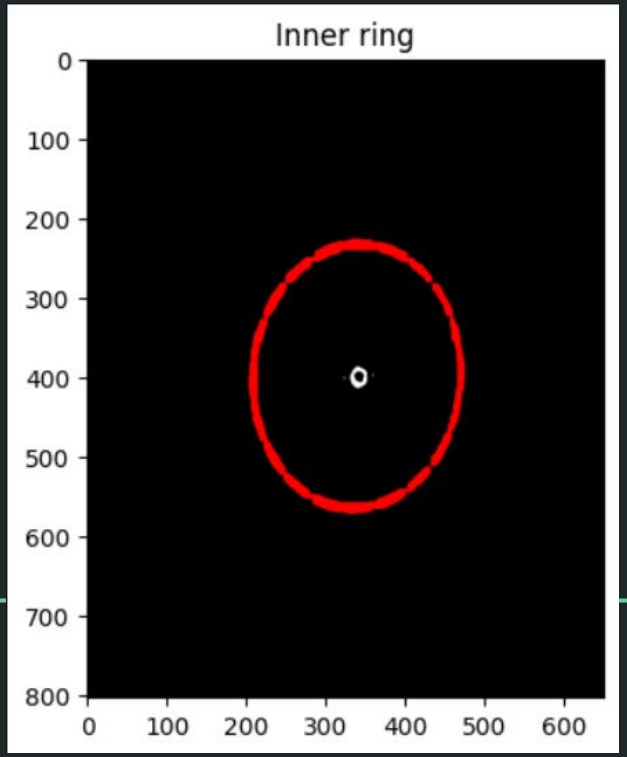

Single point
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

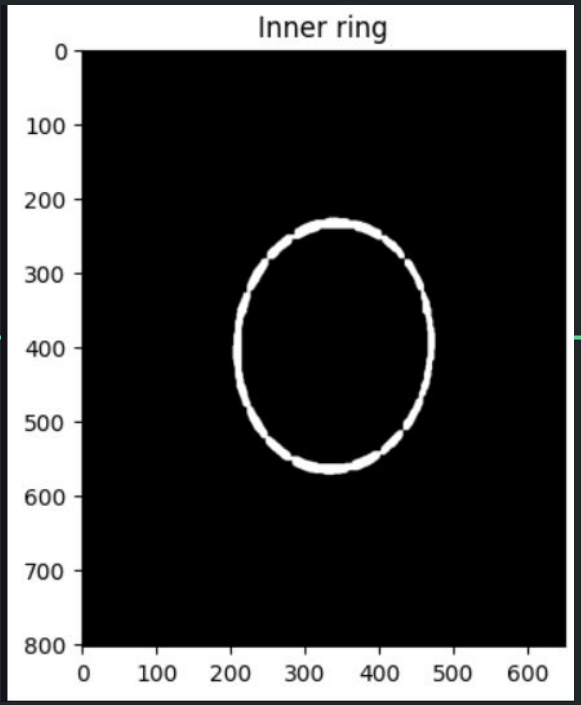

Double point
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600


Inner bullseye
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600


Outer bullseye
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600


Inner ring
0
100
200
300
400
500
600
700
800

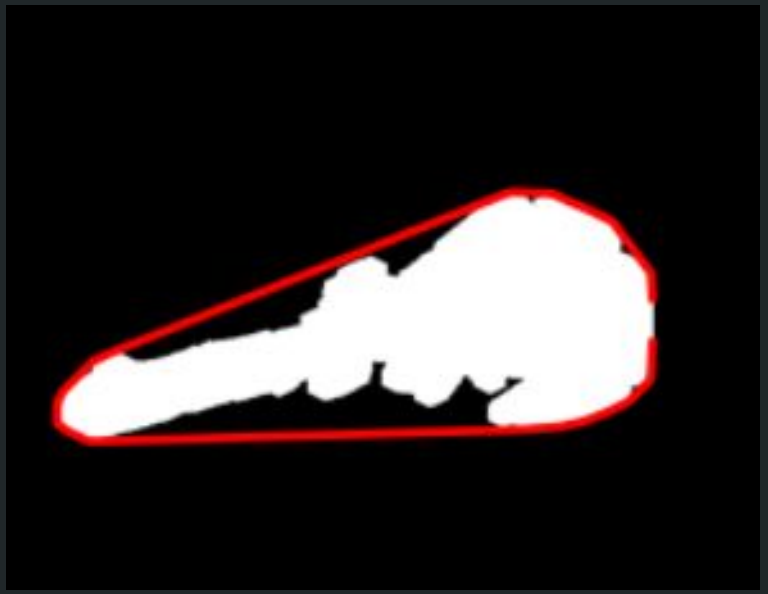

Inner ring
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600


Inner ring
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600

Getting the dart orientation

We need to know the orientation of the dart. By tracing along the line drawn along the angle at which the dart hit the board, we can obtain the exact location of the point of contact between dart and board.

We use saliency detection to extract the dart and apply thresholding to convert it to a binary image. We use this to determine the orientation of the dart



We use convex hull to get a polygon surrounding the darts binarized images, and draw a line between the the farthest points on the polygon. The reason we do so is because the farthest points on the polygon will always be the end of the dart shaft, which hit the board and the farthest pixel on the tail.

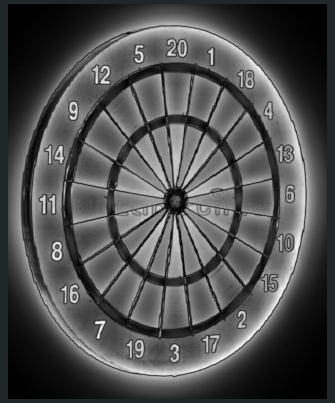
Saliency Detection

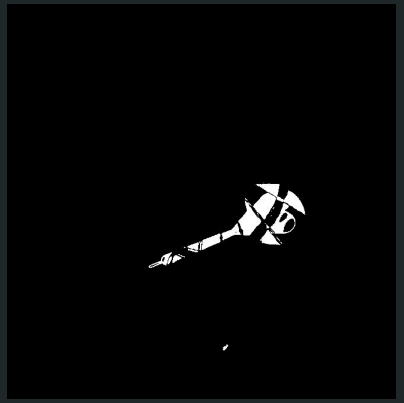
Saliency mapping is basically the highlighting of the important or prominent parts of the object. This is generally useful for foreground detection. In this project we are using this to eliminate the background of the image from the dart and dart board.

INPUT IMAGE

IMAGE AFTER SALIENCY

Here we applied saliency of an image with and without the dart and found the absolute difference of the image and apply threshold to find the image of just the dart in our final image which we can further use to pinpoint the location of where it is hitting the dart.



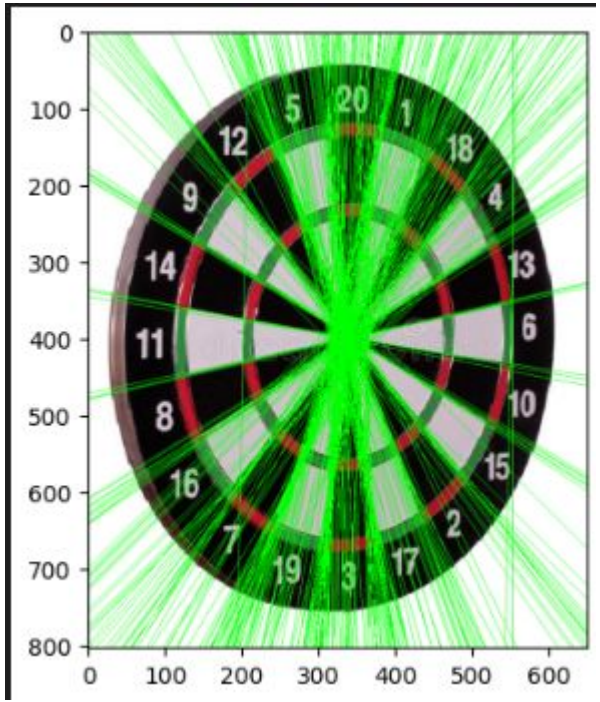


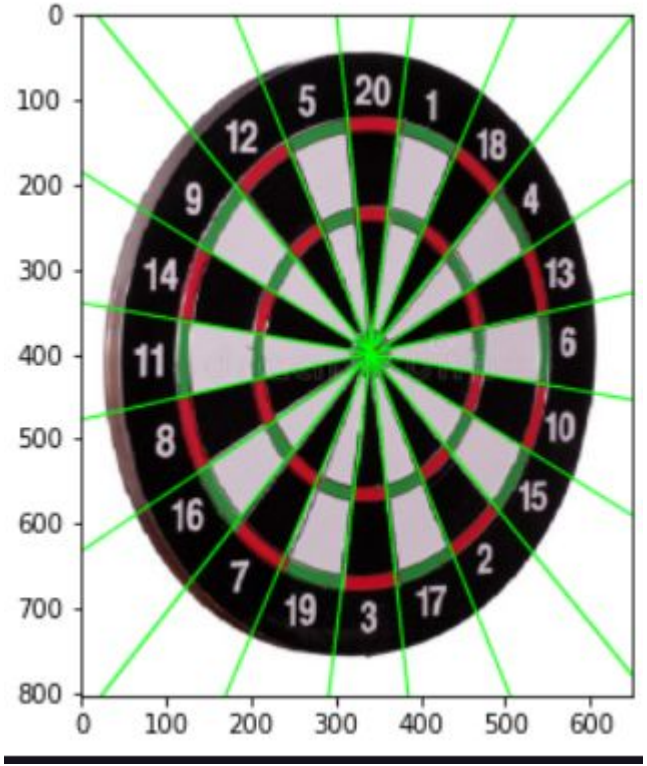
Line - Based Segmentation

- We use the linear Hough transform to split the dartboard into sectors

- There is a chance of line fraying, that is, when multiple lines are drawn very close together. This can cause problems in score assignment, since we won't have the exact segment in which the point hit, or that the point we have is on a line that is not supposed to be there

- To avoid this, we can apply k-means clustering on the lines to bring all the lines having a very small angle difference into a single line. This enables clear segmentation.





The image on the left is with improper segmentation. We can see multiple lines very close to where the actual segmenting boundary should be drawn. We use k-means clustering to bring all of these unnecessary lines together to give clear segmentation boundaries.

Score Assignment

- Now once the segments are obtained, we need to assign a score to each segment
- This is done using a simple array-based assignment
- We assign the top segment 20 and circularly iterate and assign them their respective scores
- Then once we know the point where the dart landed, we get the score from the segment first. If it is a bullseye then it is a predefined score however in case of multiplier regions we just multiply it by the required factor.

Additional features

- We implemented a sift-based image comparison for the images
- As per the research paper, this is an optional feature, but this is useful in case we have camera images of the dartboard, where there is a clear displacement between consecutive images
- The main reason this has not been considered in our project is because none of us have a camera feed of a dartboard on hand. We used online images for this project
- However, upon using sift, we can feed the key comparison points into a Homographic transform matrix and use that to correct the image perspective.

Possible Fail Cases

No project is perfect, there are always edge cases where the implementation may falter. These are ours:

- Saliency detection fails when the dart is outside the image
- There may be an error in case the dart is thrown at the board from the left side and lands pointing to the right.
- Since we are checking score in the bit masks using a 10x10 neighbourhood around the point of impact of the dart, some errors may arise near the borders of the multiplier regions

Possible Fail Cases

No project is perfect, there are always edge cases where the implementation may falter. These are ours:

- Saliency detection fails when the dart is outside the image
- There may be an error in case the dart is thrown at the board from the left side and lands pointing to the right.
- Since we are checking score in the bit masks using a 10x10 neighbourhood around the point of impact of the dart, some errors may arise near the borders of the multiplier regions

Test Cases (Successful and Fail Cases)

Output - 11

Output - 36

Output - 50

Output - 25

Output - 16

Output - 24

Output - 5

Output - 7

Output - 19

Output - 14

Ideal Output - 0

Actual Output - 0

Ideal Output - 20

Actual Output - 0

(Fail Case)

Ideal Output - 6

Actual Output - 0

(Fail Case)

Ideal Output - 15

Actual Output - 0

(Fail Case)

Ideal Output - 11

Actual Output - 0

(Fail Case)

Ideal Output - 3

Actual Output - 21

(Fail Case)

Difficulties faced in the project

- Understanding the paper itself. There was a lot of things that were assumed to be known by the reader.
- Figuring out the threshold values for various functions was a trial-and-error process, which took a good deal of our time
- Many of the methods used required a high level of mathematical proficiency, so it required multiple tries to get the idea
- Last-minute debugging took time

Individual Contributions

- Rahul: Multiplier bitmasking, assigning score values to angular segments and final scoring
- Chinmay: Line-based segmentation and segment selection, Presentation
- Sushil: Saliency dart detection, segment selection
- Geet: Presentation, dart orientation

There were no hard boundaries, everyone contributed towards bug-fixing and general assistance in essentially all topics.

THANK YOU!