**APPENDIX 1**

**TITLE OF PROJECT**

ARCHERY KING

**END TERM REPORT**

***by***

**Names of Candidates separated with coma**

# ISA SAIKIRAN,

# A SIVA POLI REDDY,

# CHRISTIANA KRAMPAH MENSAH.

Section: K19PT

Roll Numbers: 07, 23, 61



**Department of Intelligent Systems,**

**School of Computer Science Engineering,**

**Lovely Professional University, Jalandhar**

November, 2020

**APPENDIX 2**

# Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Signature: isa saikiran

Name: ISA SAIKIRAN

Roll Number: 08

Place: LOVELY PROFESSIONAL UNIVERSITY,

PUNJAB

Date: 18-11-2020

**APPENDIX 3**

**TABLE OF CONTENTS**

**TITLE**

1. **Background and objectives of project assigned** 
   1. Abstract
   2. introduction

1. **Description of Project** 
   1. process
   2. image processing
   3. rectification
   4. circle detection
   5. arrow detection

1. **Conclusion**

### 1.1 ABSTRACT

We present a method for automatically determining the score of a round of arrows lodged in an archery target face. That is, given an image consisting of a complete target face, and given a set of arrows that have struck within the target face, we generate a score for each arrow with regard to the two circles it is between.

To do this, we present a multi-step process to determine the location and shape of the set of concentric and evenly distributed circles (that may be distorted by perspective) representing the target face, locate and orient the each arrow, and find the pinpoint location of where the arrowheads pierce the target face.

We are able to handle images of a target face at arbitrary angles by automatically rectifying the image so that the target becomes circular. We test the performance of our system on a set of real images taken from mobile phones and find that it performs reasonably well.

Categories and Subject Descriptors

[Image Processing]: Scene Analysis – Object Recognition

General Terms

Planar Rectification, Object Recognition, Detection

Keywords

Archery, Scoring, Arrows, Targets, Arrowhead

#### 1.2 INTRODUCTION

Archery is a growing sport around the world, with competitors from the junior level to the collegiate and Olympic levels. During a competition, archers line up at a measured distance from a target and attempt to fire arrows into the target center.

The target consists of 10 nested concentric circles, with the innermost one worth 10 points and the outermost one worth 1 point. One of the most time consuming tasks of archery training and competition is manually determining a score for each arrow on a target. In a typical competition, archers shoot for only 4 minutes before having to walk to the target and determine the score - a process that takes up to 5 minutes, meaning the scoring time can consume more than half the competition! Since competitions usually last 2 to 4 days, a computer-assisted scoring mechanism can save quite a significant amount of time.

Finally, many archers don’t keep score during training because it is too much of a hassle, although keeping score is one of the best ways to track progress.



#### 2.1 PROCESS

Our current algorithm has two distinct stages, the second of which relies heavily upon the first. The first task is rectifying the target face and finding the circles. We then analyze the arrows afterwards.

#### 2.11 IMAGE PROCESSING

The images we analyze will most likely be taken from mobile devices, and thus, the quality is always a concern. Images will be of varying exposure, contain noise, have distortion and perspective shifts, and contain differing sized target faces that are not necessarily centered. Also, one of the first things we noticed was that for our purposes, pixel color values and textures were more likely to be detrimental than useful to the detection problem, and that all we really needed were the edges. Thus, we tried several variations of the Canny edge detector method and found that Canny with a simple Gaussian blur worked best



#### 2.12 RECTIFICATION

Rather than detecting ellipses for planar rectification, we take advantage of the fact that target faces are printed on a square sheet of paper. We detect the four corners of the sheet of paper to calculate a homography matrix which is used to rectify the image.

We use a Harris corner detector, which is able to detect the corners of the paper given a low threshold, but that also means it detects many other unrelated corners in the image. In order to single-out the corners of the paper, we rely on medium-to-long lines in the image, found using a Hough transform. Since there are many lines present in any image, we pick out a set of 4 lines that roughly intersect at 4 or more Harris corners.

However, since the arrows themselves create long lines, they could be misconstrued as a paper border. Thus, we restrict the 4 lines such that the longest line is not much larger than the shortest line, forcing the shape to be somewhat of a skewed square. We also require that none of the line segments are part of the same line.

The main challenge presented by the hard samples was the extreme angle of the paper and the fact that one or more corners of the paper were sometimes outside of the image. This is one of the limitations of our algorithm. The algorithm was also confused when the edges of the paper were not completely flat, causing the edges of the paper to not intersect at its corners.

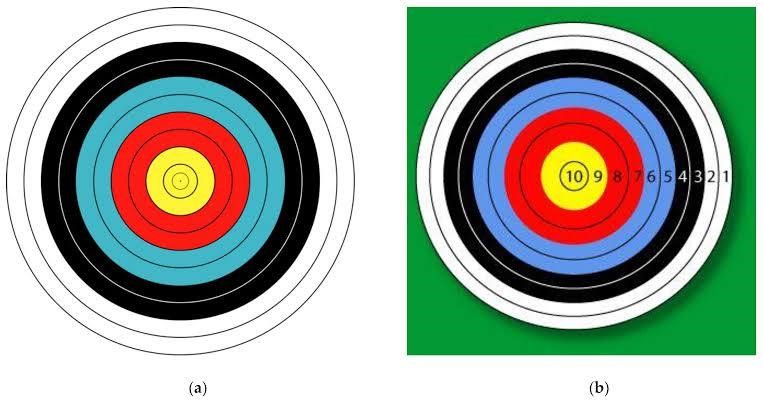
Although using 4 points is enough to calculate a usable perspective transformation, it can sometimes result in the center of the scoring rings to be transformed to an incorrect location. To fix this, we would need to detect a 5th point for the Homography equations which would ideally be the center of the rings.



#### 2.13 CIRCLE DETECTION

We started by detecting circles using a basic Hough transform. However, Open CV’s Hough Circles library discards concentric circles, perhaps to avoid false positives on the same object, and gives us the strongest circle it can detect (Figure 6). Rather than finding ways to detect all the circles, we exploit the fact that the circles on a target are evenly spaced and instead try to detect the outermost circle, from which all the inner circles can be more precisely calculated. Since Open CV allows a radius range to be specified, we performed binary search on the range of radii in order to arrive at the largest circle. As a final step, we applied a Gaussian blur to the image before applying the Hough transform for additional precision.

When we calculated the inner rings, we immediately saw how the error of the outermost circle had a much more noticeable effect on the smaller circles. In order to calculate a more precise location for our outermost circle, we used the Hough circle as an initial approximation for a more precise template matching process, in which we tested at scales of .7 to 1.3 in a 50 by 50 range (our image is 800x800) using a pyramid sliding window approach. Since the Hough circle helps us narrow down the search domain of the template matching, we can be very precise without being too expensive. This will be critical since the algorithm is intended to be used on mobile phones.



#### 2.14 ARROW DETECTION

Arrowhead intersection was a particularly difficult task, and one we had to make a lot of assumptions for initially. We tried many solutions in our attempt to solve the problem, but in the end, the simplest turned out to be the most elegant.

Our first attempt was to use template matching to find sections of the photograph that were similar to the what an intersection of an arrow with a target face looked like. However, this was not successful as efficient template matching is not rotationally invariant, in addition to the complexity of varying backgrounds that take up most of the matching template (the arrowhead is really small and narrow).

We also tried Harris corner detection, with the hopes that the intersection would appear as a corner that we could distinguish. However, as you can see in Figures 8a and 8b, Harris corner detection could not separate the intersection point from any of the other corners in the image, and because of the aliased edges of the circle that no amount of blurring could solve (Figure 8b uses a Difference of Gaussian with Canny and with a particularly low threshold), it ended up returning more circle edges than anything else.

In the end, while the simplest method of using a Hough transform to find the lines with the right amount of parameter adjustment resulted in the best possible outcome for our contrived image,

we ended up finding out that correctly parameterizing a Difference of Gaussian for the real world images resulted in better arrow line detection. One of the issues we faced was finding a fine balance of the arrow shaft between hyperextension as a result of too much noise and hyper flexion as a result of too much blur non-definition.



**3 CONCLUSION**

Our system can successfully detect and score arrows given an image taken from a standard mobile phone. The chance of success is heavily influenced by the angle at which the image is taken. When the angle sharpens, many factors come into play that challenge our system, such as rectification accuracy and arrow occlusion.

While we initially relied on SIFT features for detection of arrows, we found better performance by using line and gap information to accurately locate arrows and eliminate false positives. Our system can further be improved by detecting more points for rectification and finding ways to separate clusters of arrows while joining together fragmented or occluded arrows. Moreover, for the rare instance where the actual intersection is further from the center than the closer of the two segment points, it would be useful to identify the ends of the arrows properly.

Finally, we’d love to port this to a mobile device and that would be able to calculate the arrows in real time.



**PROGRAM**

import graphics

from graphics import \*

import math

win = GraphWin("Archery Target", 500,500)

win.setBackground('gold')

win.setCoords(-250, -250, 250, 250)

cen = Point(0,0)

msg = Text(Point(0,230), 'Click to hit the target.')

msg.setFill('blue')

msg.draw(win)

def score\_text(): #put the score numbers on the target

v = 190

z = 18

bullseye = Text(Point(0,0), 'x')

bullseye.setSize(14)

bullseye.draw(win)

for i in range(1,11):

score\_number = Text(Point(0,v), i)

score\_number.setSize(z)

score\_number.setFill('blue')

score\_number.draw(win)

v = v - 19

z -= 1

def hit\_target(): #draw a circle where the user clicks and add to the count

count = 0

for i in range(10):

p = win.getMouse()

x = p.getX()

y = p.getY()

p = Circle(Point(x,y),4)

p.setFill('purple')

p.draw(win)

d = x\*2 + y\*2

string = 'You hit the target. Score is %s.'

if d > 200\*\*2:

msg.setText(('You missed the target. Score is %s.' %count))

elif d <= 10\*\*2:

count += 11

msg.setText(('Congrats! You hit the bulls-eye. Score is %s.' %count))

elif d <= 29\*\*2:

count += 10

message = msg.setText((string %count))

elif d <= 48\*\*2:

count += 9

message = msg.setText((string %count))

elif d <= 67\*\*2:

count += 8

message = msg.setText((string %count))

elif d <= 86\*\*2:

count += 7

message = msg.setText((string %count))

elif d <= 105\*\*2:

count += 6

message = msg.setText((string %count))

elif d <= 124\*\*2:

count += 5

message = msg.setText((string %count))

elif d <= 143\*\*2:

count += 4

message = msg.setText((string %count))

elif d <= 162\*\*2:

count += 3

message = msg.setText((string %count))

elif d <= 181\*\*2:

count += 2

message = msg.setText((string %count))

elif d <= 200\*\*2:

count += 1

message = msg.setText((string %count))

msg.setText('Game over. Score is %s. Click anywhere to quit.' %count)

win.getMouse()

win.close()

return count

def main():

#draw the target

rad = 200

color = ['white', 'black', 'cyan3', 'red', 'yellow']

for i in range(5):

for j in range(2):

c = Circle(cen, rad)

c.setFill(color[i])

c.setOutline('gray')

c.setWidth('2')

c.draw(win)

rad = (rad - 19)

#draw the bull's-eye

center = Circle(cen,10)

center.setOutline('gray')

center.setWidth('2')

center.draw(win)

score\_text()

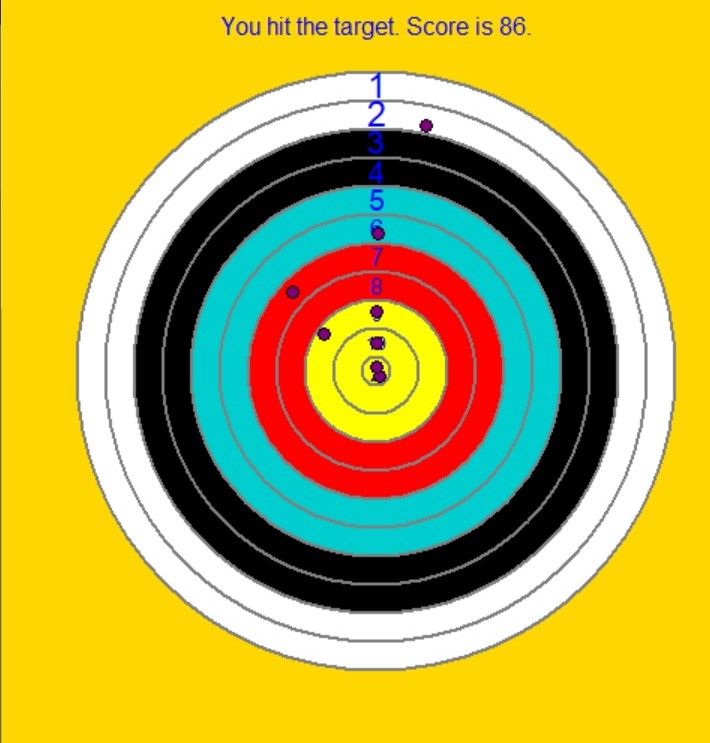
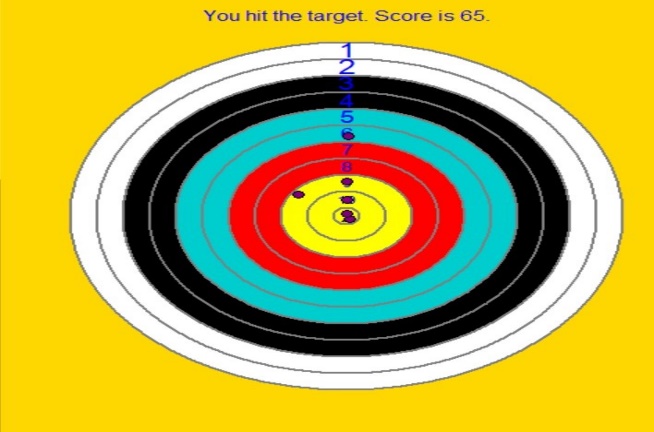
hit\_target()

if \_name\_ == '\_main\_':

main()

**OUTPUT:**

** **

** **

****

**APPENDIX 4**

BONAFIDE CERTIFICATE

Certified that this project report “ARCHERY KING” is the bonafide work of “ISA

SAIKIRAN, A SIVA POLI REDDY, CHRISTIANA KRAMPAH MENSAH” who carried out the project work under my supervision.

<<Signature of the

Supervisor>> (Due to Covid19, signature is exempted)

<<Name of supervisor>>

<<Academic Designation>>

<<ID of Supervisor>>

<<Department of

Supervisor>>

**Technology and frame work:**

* + - 1. **Python**
      2. **Spyder(anaconda 3)**
      3. **Win 10 for pc, android mobiles**
      4. **2gb ram for pc**

**SWOT analysis**

**Strength:**

**Easy to install.**

**User friendly interface.**

**Cost effective.**

**Weak:**

**Does not support Multi screen.**

**Opportunity:**

**Can deploy in cloud space.**

**Threat:**

**N/A.**

**Description of Work Division in terms of Roles among Students.**

I am A SIVA POLI REDDY from K19PT, group – 2, 61. Our project NAME is “ARCHERY

KING“. So we are discussed about our project and divided different roles among us to do project.

In this project my role is preparing of report for our project. Firstly I had searched in google about

this project and also I got some more information from my seniors. Then I got some idea about the

project and also CHRISTIANA helped to do report then I had completed the report. No part of this report

of this project is copied from other sources.