

COMP4102 Final Project Report

TIC-TAC-BOT

Nicholas Aube - 101032093

Virginia Saurer - 101037563

April 15, 2020

1 Abstract

The initial goal of this project was to create a program that can analyze an image drawn by the user. The program is able to recognize shapes like a cross, or a circle. It takes this information and passes it into a secondary program that takes this information and plays a game of tic-tac-toe against the player. This secondary program will represent the state of the board in its internal memory, and submit moves with the intelligence to know which ones are possible, and which spaces are already taken. If the player or the computer agent wins the game, or the game ends in a draw, the program will terminate and a message is displayed notifying the user of this.

2 Introduction

The main challenge of our project was implementing an algorithm that could accurately detect the hand drawn symbols in an image. Every one has different handwriting, which makes it difficult to detect symbols like Os and Xs accurately for different written images. We decided to use the OpenCV shape recognition algorithm that uses the `findContours()` function. Retrieving the contours of an image is the first step to identifying different shapes and objects within the image. Contours are curves of joint continuous pixel points that have the same colour or intensity. To retrieve more accurate contours, it is best to pass in a gray-scaled and binary image into the `findContours()` function. This is why we applied a threshold to our gray-scale image. The function then returns a found contours of the passed-in image. It is through the analyzation of the contours that the O and X symbols can be recognized. This type of algorithm is a good computer vision method for object and shape recognition. Thanks to the OpenCV library it is easily accessible and makes for efficient object and shape recognition.

3 Background

A paper that relates to our problem is the ACM article ABSTRACT SHAPE RECOGNITION BY MACHINE by Mary Elizabeth Stevens from 1961. She was part of the National Bureau of Standards in Washington, D. C. and developed a hypothetical machine model for shape recognition. She developed her model with the use of a contour-projection principle a similar principle we emulated in our program. Her model was created to identify shapes in graphic patterns that can be different in size, location and certain rotational transformations. The model also considers the recognition of hand-drawn figures. This relates to our problem of recognizing a hand drawn tic-tac-toe board. Stevens method of shape and handwritten symbol recognition is based on a contour-projection technique that is able to distinguish different variations of similar shapes. The method best works when analyzing black and white images. Similarly, our application passes in a gray-scaled, binary image into the OpenCV contour detection function. Another algorithm related to our program is described in the ACM article the Precise Pupil Boundary Detection Using Angular Integral Projection and Active Contour Model, published in 2018. It presents a new iris identification algorithm. It now successfully detects the exact pupil contour instead of detecting the pupil boundary as a perfect circle. This has immensely enhanced the performance of modern-day iris matching. In comparison to our contour-identifying algorithm, this one also is comprised of two shape detection phases. The first phase detects the circular pupil contour and the second goes on to detect the exact pupil contour. This is done by first detecting a set of pupil boundary points of a binarized iris image. The accurate pupil contour is then identified by an active contour model that is based on a greedy optimization algorithm. Our algorithms first phase is a detection of the extreme outer contours of an image. These are the identified board lines contours of the hand-drawn tic-tac-toe board. The second detection phase begins by filtering out the small contours of the image. It is through the analyzing of these contours that the exact O and X symbols can be identified. These two articles are good computer vision algorithm examples of shape detection that have a similar function as our algorithm. The main sections of our algorithm makes use of the OpenCV library that offers many shape detecting functions. OpenCV offers many open source examples of the use of these functions such as their tutorial of the use of `findContours()` and `drawContours()` functions. These different sources provided us with inspiration for coming up with our idea for creating our own shape and object detection program.

4 Approach

Our implementation of a board image recognition reads in a user selected image (one of several hand drawn images) and converts it to gray-scale. The gray-scaled image has a threshold applied to it that converts it to a binary image. A kernel along with the binary image are passed into the `Opencv` function,

morphologyEx(), which returns the image with reduced noise. The contours of this image are retrieved by passing it into the findContours() function along with the RETREXTREXTERNAL parameter. This particular parameter only retrieves the extreme outer contours of an image which are the identified board lines in the hand drawn image. To retrieve the contours of the drawn X and O symbols there are a few more processing steps that need to be done. Each individual board tile needs to be analyzed. This is done by looping through the found outer contours of the image. A check is applied that filters out the small contours of the image and keeps those that are greater than 200000 contours. This will help the symbol identification in the board tiles. By trimming the binary image at all of its edges, the individual board tiles are created. Each tile image is then passed into the findContours() function along with the RETRTREE parameter which retrieves all the images contours and reconstructs a full hierarchy of nested contours. These found tile contours are then looped through. Each contour area in the hierarchy needs to be less 100000 to ensure that each tile does not identify itself as a contour. Checking at numbers between 80000-100000 contours give good results for most drawn images. The final step in recognizing the O and X shapes within the board tiles, is calculating the solidity of the tile contour area. The formula for calculating solidity of a contour area is dividing the contour area by the convex hull of the contour area. The convex hull of a shape or a group of points is a tight-fitting convex boundary around the points or the shape. This will help identify the round O shape in the tile area. If the solidity of the contour area is greater than 0.5 then the character O is identified. Otherwise the character X is identified. These are then stored as string characters in the 2D gamestate array at the array indices of the located tile. The gamestate array is then passed into the TicTacToeGame class as the board input. Compared to other shape recognition algorithms, ours is unique in terms of the way we cut the board image down to a tile to recognize the shapes. We are also using the solidity check instead of using the opencv HoughCircles() function to identify the drawn circles or Os in images. This is because it is more accurate in identifying any type of O written by the user on the image.

5 Results

Here is a run-through of the tic-tac-bot with the first board image selected by the user:

Figure 1. As soon as the user starts the program they are prompted with the following question.

They are asked to choose between 4 different hand-drawn images that will get read in and analyzed for symbol detection:

```
Welcome to our Tic-Tac-Toe Bot Program!
```

```
Enter a number between 1 and 4 choose a board to read in and play,  
or enter 0 to play with an empty board> 1
```

```
After viewing the image windows, make sure to close them so you can start playing the game.
```

Figure 2. The image then gets displayed with blue contour outlines highlighting the handwritten X and O symbols and black contour lines highlighting the board lines:

```
Welcome to our Tic-Tac-Toe Bot Program!
```

```
Enter a number between 1 and 4 choose a board to read in and play,  
or enter 0 to play with an empty board> 1
```

```
After viewing the image windows, make sure to close them so you can start playing the game.
```

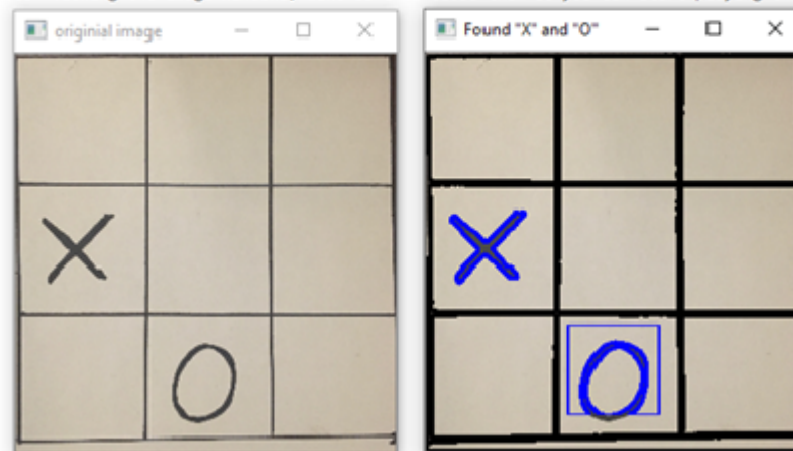


Figure 3. After closing the image window, the detected X and O symbols are added into the board array. The board is then displayed in the console and the user is prompted to make their move of placing their O:

```

Welcome to our Tic-Tac-Toe Bot Program!

Enter a number between 1 and 3 choose a board to read in and play,
or enter 0 to play with an empty board> 1
After viewing the image windows, make sure to close them so you can start playing the game.
[[' ', ' ', ' '], ['X', ' ', ' '], [' ', 'O', ' ']]

CURRENT BOARDSTATE:
  | |
  -----
X | |
  -----
  | O |

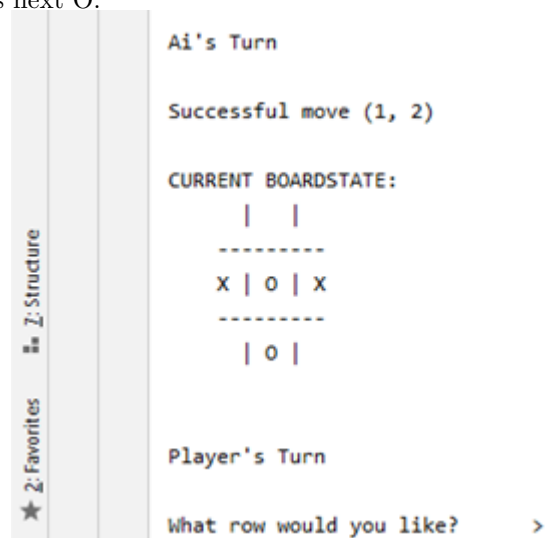
O (Player) goes first

Player's Turn

What row would you like?      >

```

Figure 4. The AI then plays his turn and the user is then asked to place his next O:



```

★ 2: Favorites  ■ 2: Structure

Ai's Turn

Successful move (1, 2)

CURRENT BOARDSTATE:
  | |
  -----
X | O | X
  -----
  | O |

Player's Turn

What row would you like?      >

```

Figure 5. As soon as there is a recognized win, the console outputs the winner of the game and the program ends:

```

Player's Turn

What row would you like?    >0
What column would you like? >1
Successful move (0, 1)

CURRENT BOARDSTATE:
  | 0 |
-----
X | 0 | X
-----
  | 0 |

The player is the winner!

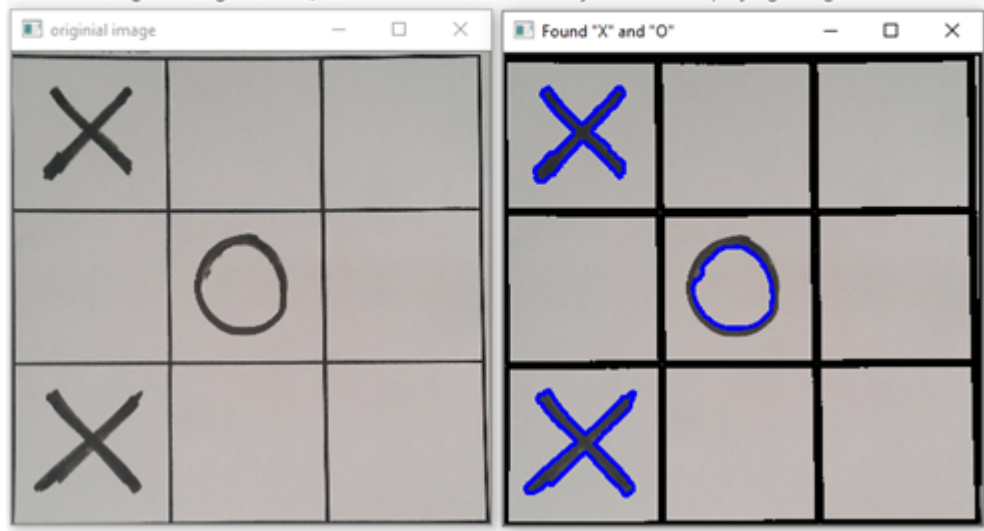
```

Here are output images of the three other hand-drawn boards that have highlighted symbol recognition:

Welcome to our Tic-Tac-Toe Bot Program!

Enter a number between 1 and 4 choose a board to read in and play,
or enter 0 to play with an empty board> 2

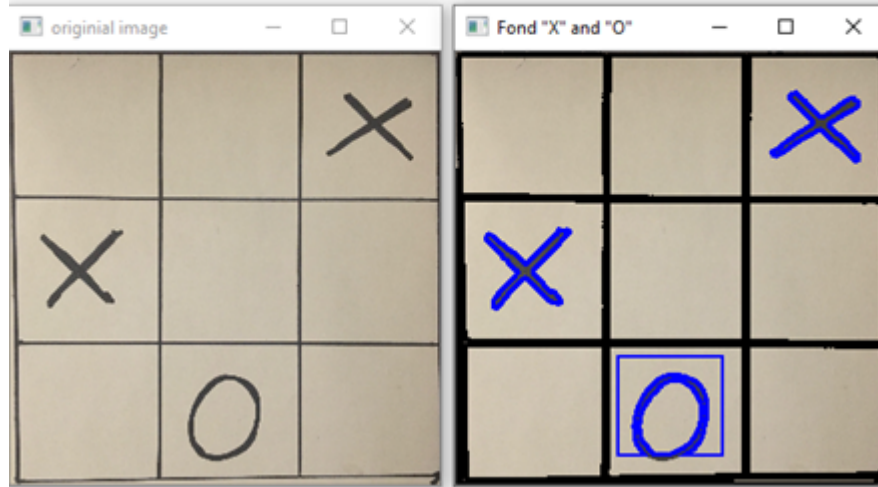
After viewing the image windows, make sure to close them so you can start playing the game.



Welcome to our Tic-Tac-Toe Bot Program!

Enter a number between 1 and 3 choose a board to read in and play,
or enter 0 to play with an empty board> 3

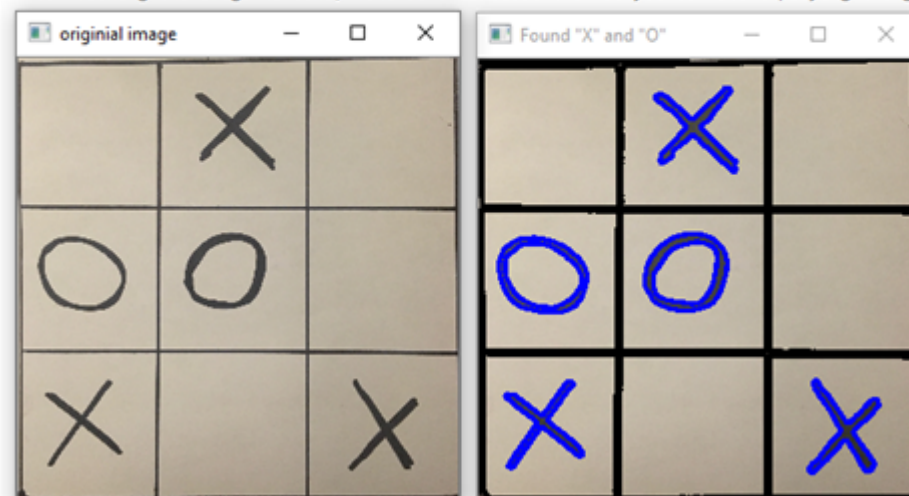
After viewing the image windows, make sure to close them so you can start playing the game



Welcome to our Tic-Tac-Toe Bot Program!

Enter a number between 1 and 4 choose a board to read in and play,
or enter 0 to play with an empty board> 4

After viewing the image windows, make sure to close them so you can start playing the game.



6 List of Work

Equal work was performed by both project members.

7 Github Page

<https://github.com/vannsaurer/Tic-Tac-Bot>

8 References

Alkazzaz A. A. J., Wang K., and Ghassan J. Mohammed Algdool G. J. M., (2018). Precise Pupil Boundary Detection Using Angular Integral Projection and Active Contour Model. In Proceedings of the Fourth International Conference on Engineering MIS 2018 (ICEMIS 18). Association for Computing Machinery, New York, NY, USA, Article 62, 1-6.
DOI:<https://doi-org.proxy.library.carleton.ca/10.1145/3234698.3234760>

Doxygen, (April 10, 2020). Creating Bounding boxes and circles for contours,
https://docs.opencv.org/3.4/da/d0c/tutorial_bounding_rects_circles.html

Doxygen, (April 10, 2020). Contours: Getting Started
https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

Rosebrock A., (February 8, 2016). OpenCV shape detection, PyImageSearch.
<https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/>

Shrimali K. R., (August 13, 2018) Convex Hull using OpenCV in Python and C++. Learn OpenCV,
<https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c>

Stevens M. E., (1961). Abstract shape recognition by machine. In Proceedings of the December 12-14, 1961, eastern joint computer conference: computers - key to total systems control (AFIPS 61 (Eastern)). Association for Computing Machinery, New York, NY, USA, 332351.
DOI:<https://doi-org.proxy.library.carleton.ca/10.1145/1460764.1460790>