Are You A Potato Drawer?

CSS 487 FINAL PROJECT WRITE-UP (2016/12/07)

KUNLAKAN (JEEN) CHERDCHUSILP | KULSOOM MANSOOR | JESSICA ORIONDO

What We Are Trying to Accomplish

We want to recognize shapes drawn by the user by using computer vision techniques. For our application, we present the user with a canvas and ask them to draw a random shape (from our template image library). The user then draws on the provided canvas and pressed the “Enter” button when finished. After the user is done drawing, our program detects if the user drew the correct shape.

Our application was inspired by the Google application, “Quick, Draw!”. Here is a link to the application: https://quickdraw.withgoogle.com/

What We Have Accomplished

UI && Canvas

- Instructions are given at the start of the application and the user can press enter to continue.
- The user can draw on the canvas and save the image as a jpg when pressing the enter button.

- User can clear the canvas by pressing the backspace key.
- User can move on to the next random shape game by pressing the “R” key.
- User can press the “H” to see what shape they are supposed to draw in case they forget.
- User must press the “ESC” key to quit the application.

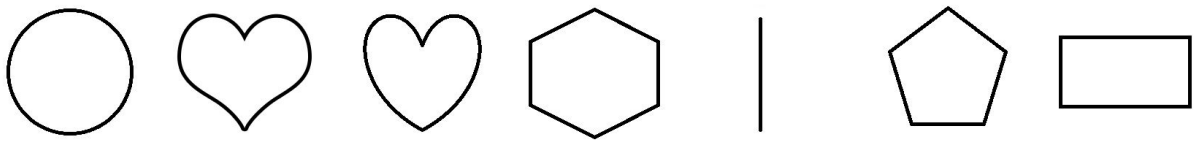
Template Image Storage

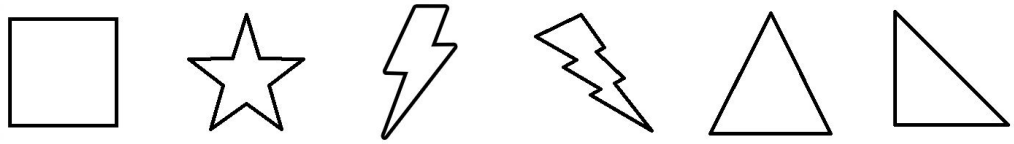
- We store all the template images in a hashmap, the key being the number of contours of the shape. This helps in getting specific set of template images to match instead of using all the template images.

Hashmap

key	values
2	Template 1
3	Template 2 | Template 3
4	Template 4 | Template 5
...	...

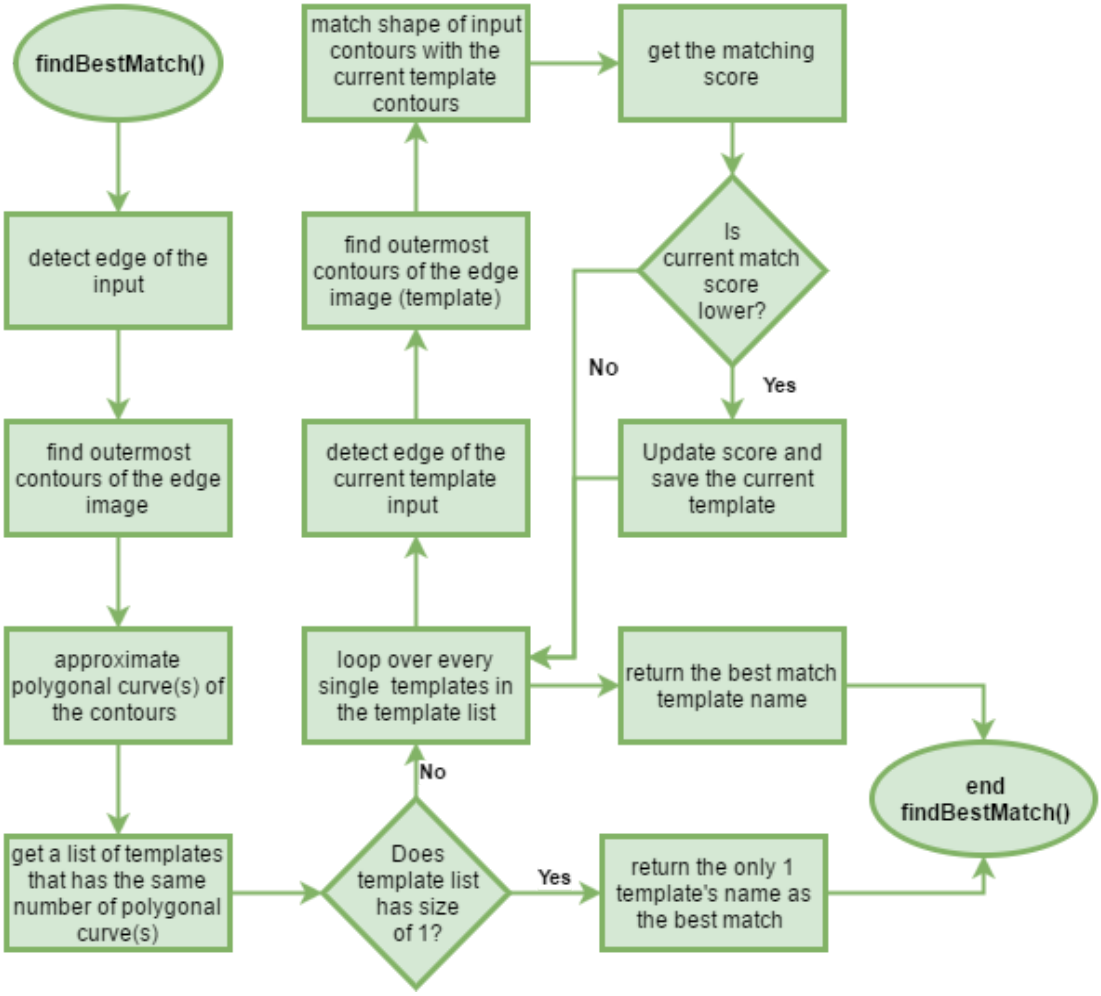
- Currently, our program only use perfect shapes of line, circle, triangle, square, rectangle, hexagon, pentagon, star, heart, and thunderbolt.





Detecting Drawn Shapes

Algorithm:



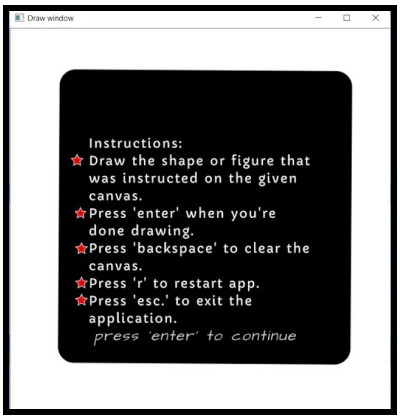
Notes:

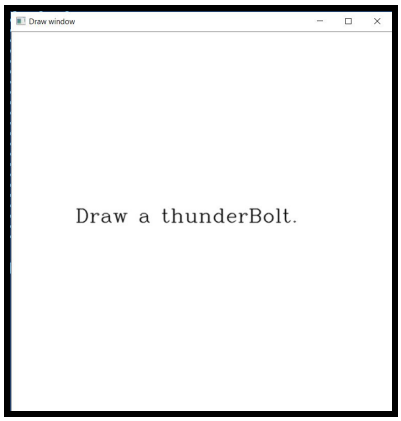
1. It's very difficult to detect the shape accurately at this moment, due to the various doodling style of each individual. However, we believe that if we have helps of machine learning, the program would detect much more accuratel.
2. For better results in general, the user should try to avoid drawing uneven lines or too small, as it might affect the detection process. Something to note is that the

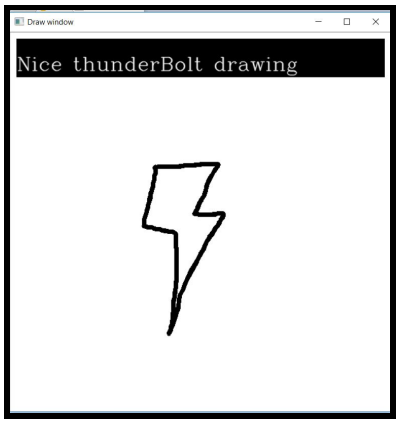
number of contours for the heart shape varies depending on how the user draws it, so it is very difficult to detect.

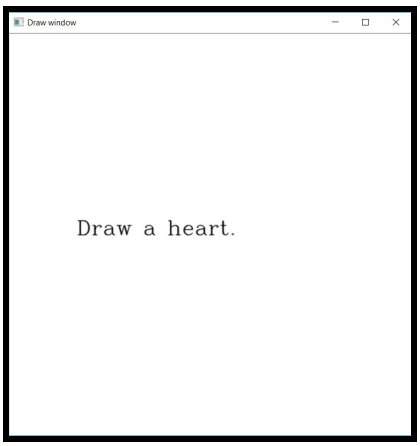
3. Heart shape is the most difficult shape to detect since heart can be drawn in various way and involves a lot of curves.

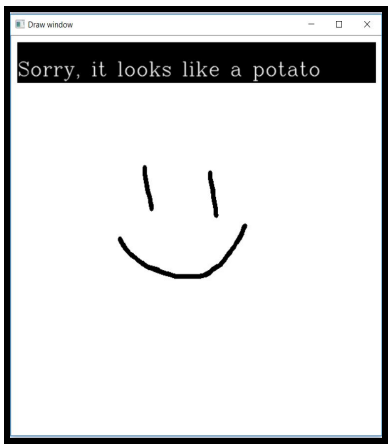
Results

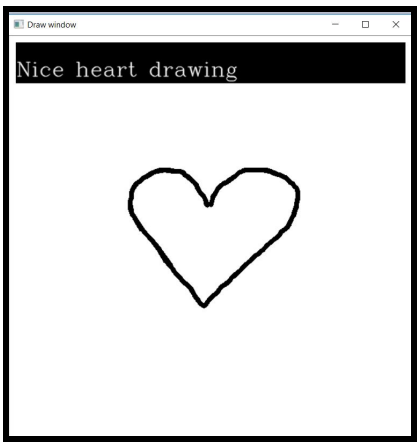


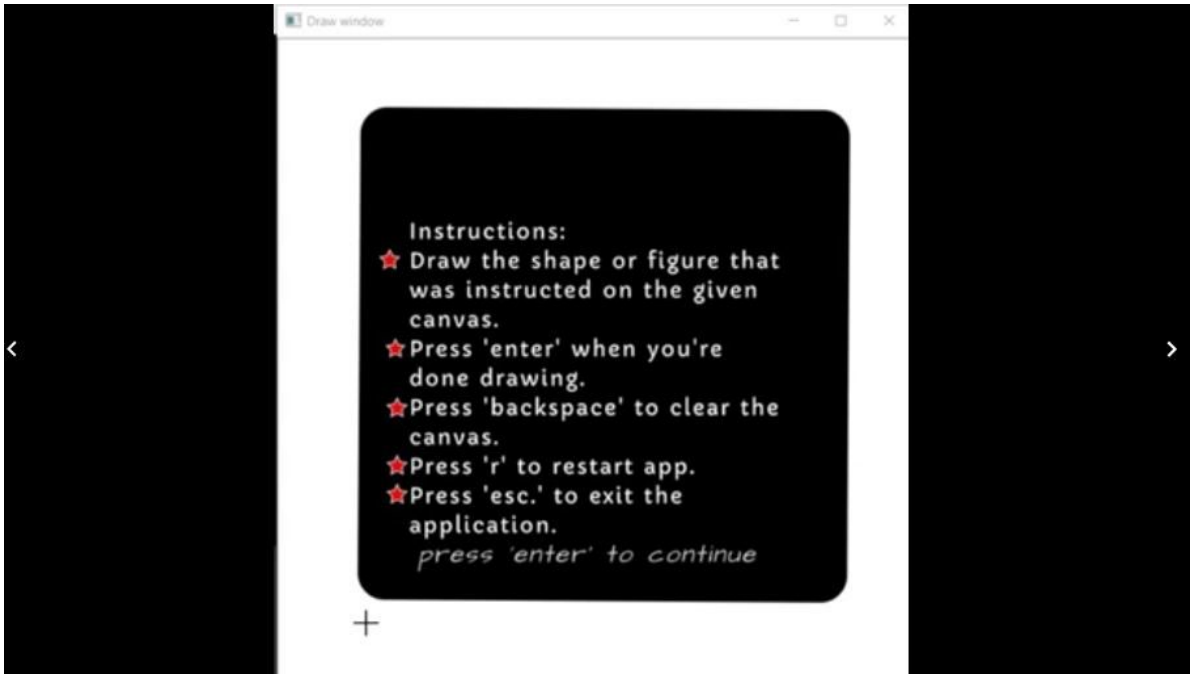












click the image above to see a video for more results

Output

The application is able to detect the shapes that was drawn by the user. However it is not 100% right due to some circumstances which are listed above under notes.

Lessons Learned

Template Matching

In the beginning of the development process, our team was focusing on using template matching technique in detecting hand drawn shapes. We later found out that template matching technique will not work with our project.

As we were doing researches about template matching, we learned that template matching work best when exact same object in the template is in the search image. We have tested the template matching algorithm to find the perfect square template image in the search image. The result was that only a small part of the hand drawn square was matched with the template.

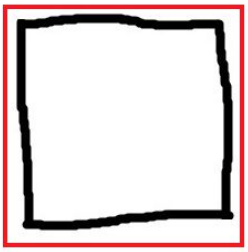


Figure 1. The result of image detection using template matching technique when the search image has the same exact object as the template image.

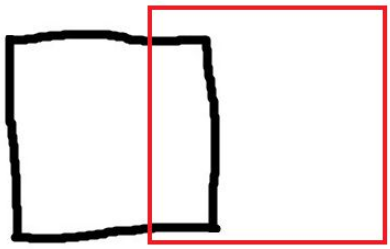


Figure 2. The result of image detection using template matching technique when the object in the search and template images are not the same.

We have also tested algorithm with search image that doesn't at all contain the object in the template. Sometimes the algorithm returns the black white spot on the search image as the best match. So, we came to the conclusion that this technique will not work with our project.

Hausdorff Distance & Shape Context Distance

Later in the process, we have decided to use Hausdorff Distance technique in detecting hand drawn shapes. However, Hausdorff Distance algorithm is very slow and expensive in calculating distance, and also giving inaccurate result of detection.

We tested the algorithm by using a single search image with multiple template images to see which template will get the best result. The distance returned from the function was very confusing. For example, we used the hand drawn square image with multiple scales to test with perfect shape templates, and the result that the Hausdorff Distance detects that hand drawn squarer is more similar to perfect rectangle than perfect square.

```
Rectangle: 209.695
Rectangle: 18
Rectangle: 127.413
Rectangle: 42
Rectangle: 70
```

```
Square: 56.8858
Square: 56.8858
Square: 56.8858
Square: 158.455
Square:  225.535
```

```
Best distance: 18
Best match template: Rectangle
```

We have also tested with other shapes such as hearts and circle, but it doesn't seems to be working the way we needed.

We also got to play with Shape Context Distance a little bit, but because it take a lot longer than Hausdorff Distance to calculate distance (longer than 5 minutes to detect one shape with a single scale), we gave up on this technique.

Contours & Approximate Polygon Curves

While we were coming up with the idea of building a data structure that will reduce time in searching for the template to match, we initially thought of using the number of strokes it takes the user to draw the shape as keys of the hashmap. However, it was troublesome to do the estimation of number of strokes that can be used to draw each template shapes since shapes can be drawn using almost any number of strokes.

We then learned about contours and its approximated polygon curves, and how it can help us detect number of lines that can be used in forming shapes drawn. We decided to not use strokes and instead use contours as the key to the hashmap.

Shape Matching & Hu-moment

Shape Matching technique work best with what we wanted to do for our project. It even allows us to detect hand drawn shapes with perfect shapes.

Shape Matching algorithm in opencv also come with a benefit, Hu-moment! According to opencv, Hu-moment "are proved to be invariants to image scale, rotation, and reflection." In addition to this benefit, the algorithm is extremely effective and fast compared to the other techniques we tried.

How To Run The Program

You have two options to run the program:

Option 1

1. Unzip the project folder
2. Go to the T-Shirt project folder
3. Go to the T-Shirt folder

4. Double click on the run_program batch file

Option 2

1. Open our program in Visual Studios 2015 and build and run the program.