Iteration 1
Status

GUI Automation using OpenCV

https://github.com/utastudents/ShapeReceipe.git

CSE-6324-001 TEAM 1

Harsh Chaludia

Deep Patel

STATUS **SUMMARY**

Who will be the target audience ?

What will be the Tech Stack?

What will be the features ?

How are we different ?

- Cross Platform Support
- A learning platform
- Web, Android, IOS

PROGRESS

From last presentation, we had the following observations.

IMPLICATIONS

- We saw many loopholes, when we started this project, and one of them was not knowing our target audience.
- We observed, we only took shapes as the method to draw elements like footer, header, etc. This limited us to include limited elements.
- For mobile, Xcode, and android studio is already into drag & drop.

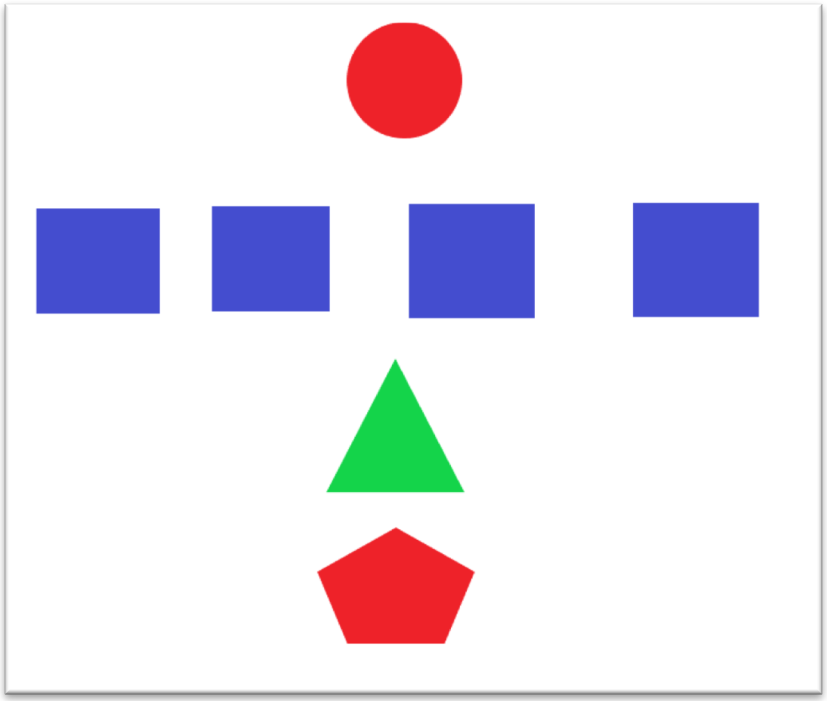
MODIFICATIONS

- We actually did an informal survey with people whom I and Harsh knew in our circle.
- We decided to use more colors to also have styling of elements associated to geometric shapes. Also, we are planning to have numeric digits to have more choices for users.
- To avoid already existing features, and since we are using React Native, we will incorporate themes made by open source theme providers.

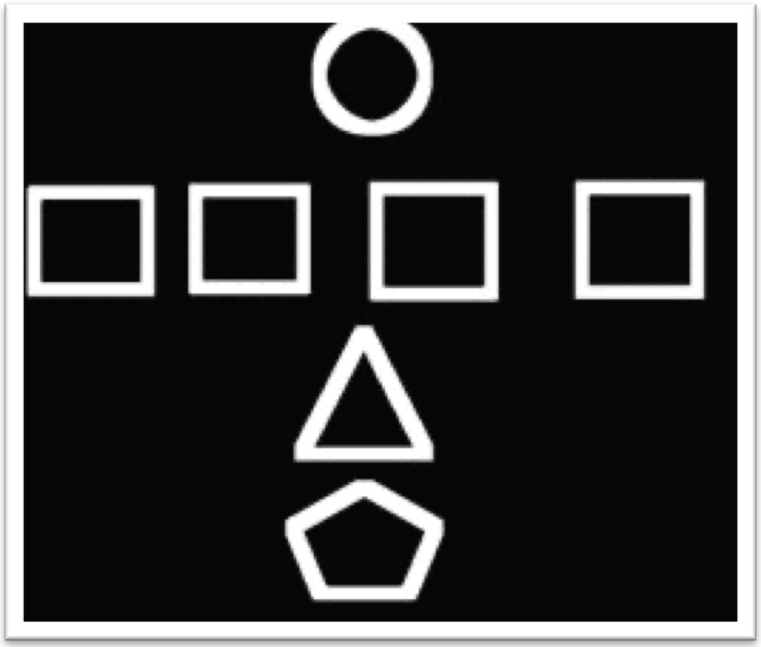
ALGORITHM

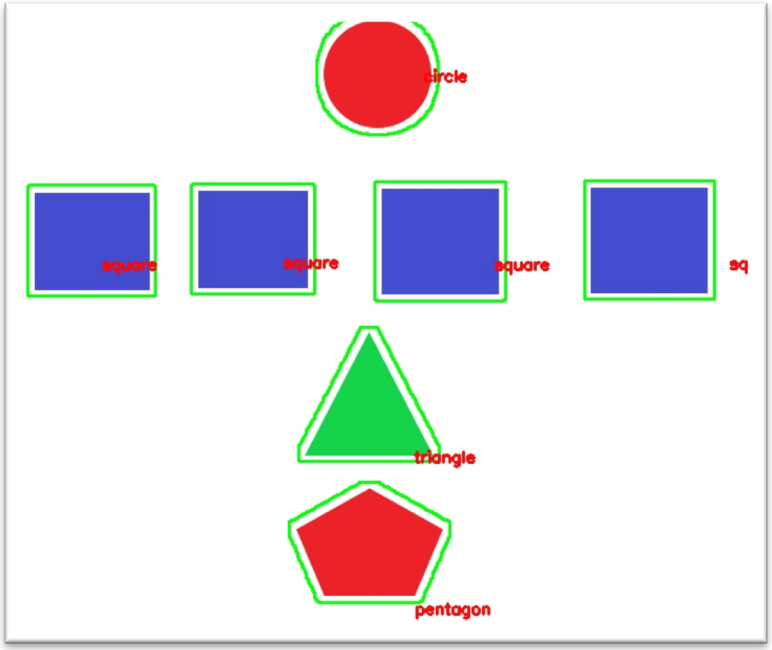
For Shape Detection,

- We are using the Ramer–Douglas–Peucker algorithm.
- “An algorithm that decimates a curve composed of line segments to a more simpler curve with fewer points” [1]



ALGORITHM

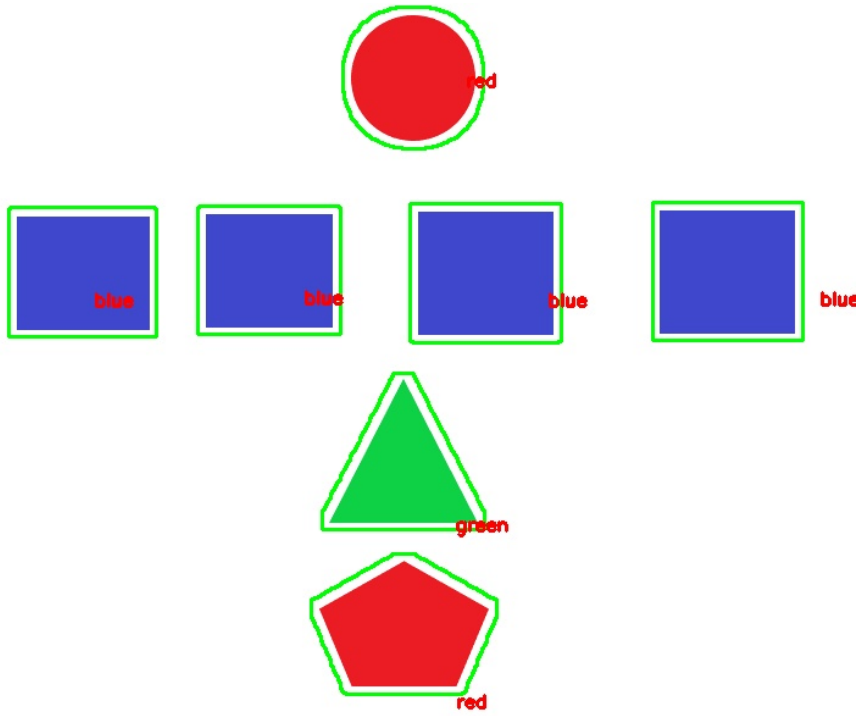




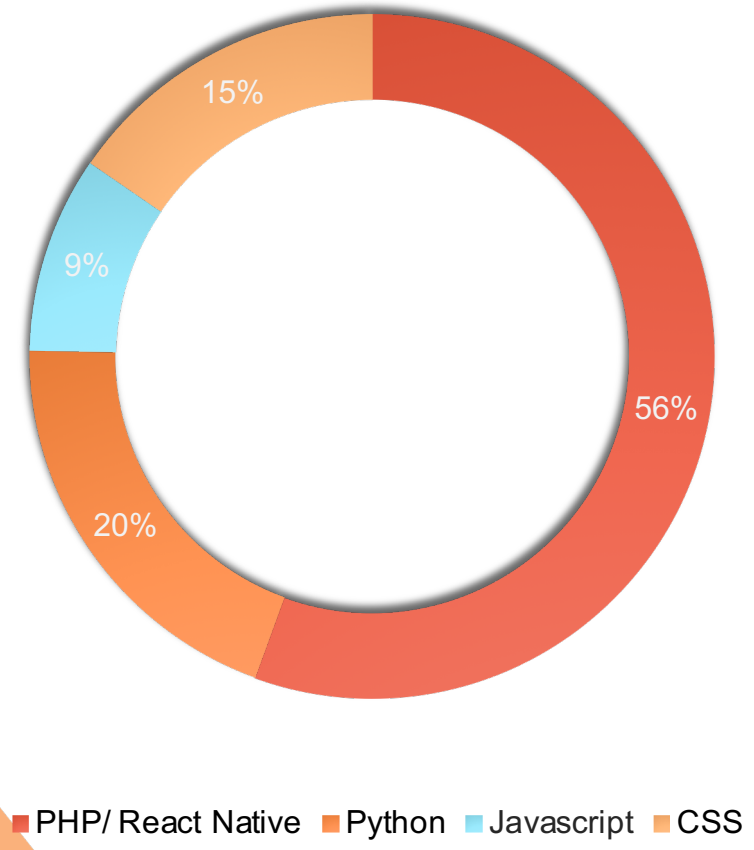
ALGORITHM

For Color Detection,

- We are using LAB color space method to detect colors in the image.
- Converting the RGB values to LAB Color space values (L*,a*,b*)
- It takes consideration of Lightness, and saturation.



TECHNOLOGIES



Language Breakdown:

- For Website – **PHP**, and Mobile – **React Native.**
- Designing Styles – **CSS**.
- Dynamic Element like Slider, Mobile Menu – **Javascript.**
- OpenCV processing of image – **Python.**

DELIVERIES

CRITICAL DELIVERABLES

- Converting sketches to skeletal design for website with PHP platform.
- Converting sketches to skeletal design for mobile with React Native platform.
- Incorporate shapes, colors, numeric digits to have more choices for users.

More..

- Once we achieve the above goals, this could be a big tool which includes multiple styling elements for both website and mobile.

GOALS FOR **NEXT ITERATION**

DATE OF NEXT STATUS UPDATE

- 20th of October 2020.

LIST GOALS FOR NEXT REVIEW

- Show a demo for Sketch of React Native product.
- Incorporate digits with shapes, and colors.
- Correcting existing issues.

ACTION PLAN REVIEW

- Use a cloud service to deploy the product online using any of the cloud platforms that let's us use command line interface.

REFERENCES

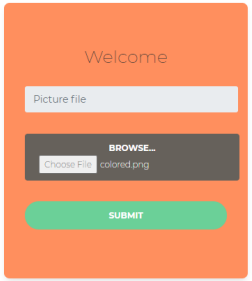
ONLINE RESOURCES

- https://en.wikipedia.org/wiki/CIELAB_color_space [2]
- https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm [1]

Specification and Design

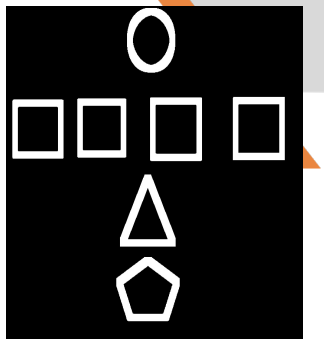
User

Uploads Image



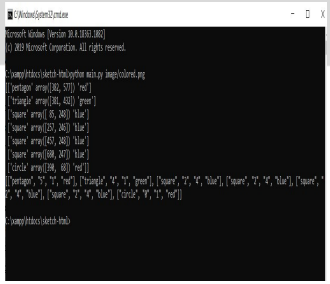
Web Form

Dilate Image



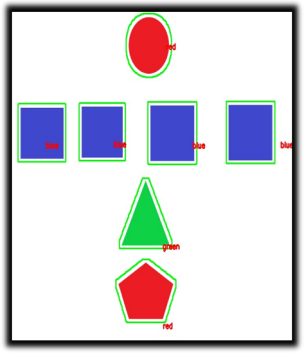
Black & White image

Recognize Shapes, Colors

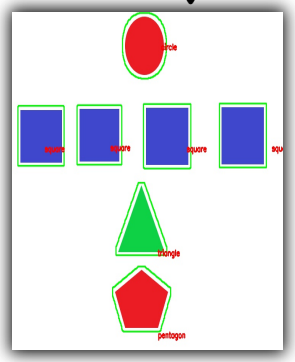


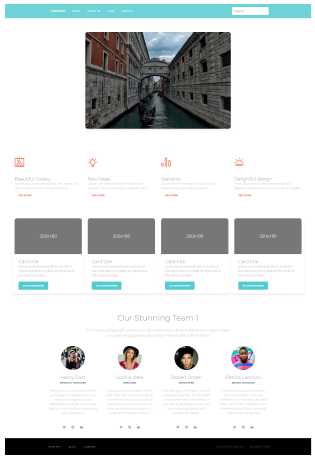
Open CV recognize total shapes

Label Colors



Label Shapes





Final output Website

Target.php Maps shapes with Web/React Native Element based on Receipe/Code.

Shape Recipe

RISKS

- Considering the edge detection method, it could be a tidious task if we draw with a doodle element where the sketch could slightly change the polygons.
- To render the application on React Native platform, we need to save the results/ outcome in an api call to save progress.
- Without knowing instructions to sketching, would confuse the users, and they might end up making an incorrect version of their design
- We saw many loopholes, when we started this project, and one of them was not knowing our target audience.
- We observed, we only took shapes as the method to draw elements like footer, header, etc. This limited us to include limited elements.
- For mobile, Xcode, and android studio is already into drag & drop.

RISKS EVALUATION

- To over come this issue, we can ask the user to drag and drop shapes from the toolbar, to approximate the shapes.
- We need to have a temporary database, and the ideal solution would be using noSQL database.
- We can make an instruction manual in a separate page on the website.
- We actually did an informal survey with people whom I and Harsh knew in our circle.
- We decided to use more colors to also have styling of elements associated to geometric shapes. Also, we are planning to have numeric digits to have more choices for users.
- To avoid already existing features, and since we are using React Native, we will incorporate themes made by open source theme providers.

CODE

Main Function – Detect Shapes, and Colors from the input image.

```
    # Read the image data from argument
    image = cv2.imread(argv[1])
    # Resize if it is necessary(faster but not accurate)
    resized = imutils.resize(image, width=800)
    ratio = image.shape[0] / float(resized.shape[0])
    blurred = cv2.GaussianBlur(resized, (5, 5), 0)
    gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)
    lab = cv2.cvtColor(blurred, cv2.COLOR_BGR2LAB)
    thresh = cv2.threshold(gray, 60, 255, cv2.THRESH_BINARY)[1]
    # Get the edge by canny effect
    canny = cv2.Canny(gray.copy 80, 120)
    # Set the kernel for dilation(make the white line thicker)
    kernel = np.ones((3, 3), np.uint8)
    dilation = cv2.dilate(canny.copy(), kernel, iterations=7)
    cv2.imwrite('dilate.jpg', dilation)
    # Get the contour from dilated picture, EXTERNAL contour used
    cnts = cv2.findContours(dilation.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    cnts = imutils.grab_contours(cnts)
    # ColorLabel
    # initialize the ShapeDetector class
    sd = ShapeDetector()
    cl = ColorLabeler()
    oracle = []
    for c in cnts:
        try:
            # Do if the contourarea is larger than specific area
            if cv2.contourArea(c) > np.shape(image)[0] / 40 * np.shape(image)[1] / 40:
                # Get the moments of the shapes
                M = cv2.moments(c)
                cX = int(M['m10'] / M['m00'])
                cY = int(M['m01'] / M['m00'])
                # Detect the shape
                shape = sd.detect(c)
                color = cl.label(lab, c)
                # Put additional information in the image
                c = c.astype('float')
                c *= ratio
                c = c.astype('int')
                cv2.drawContours(image, [c], -1, (0, 255, 0), 2)
                cv2.putText(image,shape,(cX, cY),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0, 0, 255),
2,)
                # Store the shape information to oracle
                oracle.append([shape, np.array([cX, cY]),color])
        except:
            continue
    # Save the image file
    cv2.imwrite('Shape_recognized.jpg', image)
    alsoOracle = oracle
    oracle = np.array(oracle)
```

```
    #print(oracle)
    # Create the json file to return the following value to PHP
    # ["Type of shape", "RowID", "number of shape in the same row"]
    jsonfile = get_block(image, oracle, (6, 6),alsoOracle)
    print(jsonfile)
```

Get Block Function -- "Type of shape", "RowID", "Total shapes in row", ”Color”

```
def get_block(image, oracle, split, alsoOracle):
    # Get the distance of unit
    unit_X = np.shape(image)[0] / split[0]
    unit_Y = np.shape(image)[1] / split[1]
    # Get the row ID and return in data
    data = []
    for instance in oracle:
        row = math.floor(instance[1][1] / unit_X)
        data.append([instance[0], row])
    data = np.array(data)
    # Get the set of rowID
    index_full = set(data[:, 1])
    # Calculate the number of shapes in the same row
    post = []
    for index in index_full:
        sum = 0
        for i in range(0, len(data)):
            if data[i, 1] == index:
                sum += 1
        post.append([index, sum])
    post = np.array(post)
    # add the row which specifies the number of shape in the same rows
    data = np.hstack((data, np.zeros((np.shape(data)[0], 1))))
    for i in range(0, np.shape(post)[0]):
        for j in range(0, np.shape(data)[0]):
            if post[i, 0] == data[j, 1]:
                data[j, 2] = post[i, 1]
    # return json file to process in json
    data = data.tolist()
    for x in range(len(data)):
        data[x].append(alsoOracle[x][2])
    return json.dumps(data)
```

Shape Detection Function – Using Ramer-Douglas-Puecker algorithm using approxPolyDp function from cv2, and edge detection method.

```python
# import OPENCV
import cv2

class ShapeDetector:
    # Nothing to initialize
    def __init__(self):
        pass

    # Method to detect the shape
    def detect(self, c):
        # Return "unidentified" if the shape doesn't match any shape
        shape = "unidentified"
        # Approximate the contours by polinomial shape
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.04 * peri, True)
        # Triangle
        if len(approx) == 3:
            shape = "triangle"
        # Rectangle
        elif len(approx) == 4:
            (x, y, w, h) = cv2.boundingRect(approx)
            por = w / float(h)
            # if the threshold satisfies==> square, if not rectangle
            shape = "square" if por >= 0.6 and por <= 1.4 else "rectangle"
        # Pentagon
        elif len(approx) == 5:
            shape = "pentagon"
        # Hexagon
        elif len(approx) == 6:
            shape = "hexagon"

        # if many edges than octagon --> Circle
        else:
            shape = "circle"

        # return the type of the shape
        return shape
```

Color Detection Function – Using Lab Color Space, by converting RGB values to (L*,a*,b*)

```python
class ColorLabeler:
    def __init__(self):
        # initialize the colors dictionary, containing the color
        # name as the key and the RGB tuple as the value
        colors = OrderedDict({'red': (255, 0, 0), 'green': (0, 255, 0),'blue': (0, 0, 255)})
        # allocate memory for the L*a*b* image, then initialize
        # the color names list
        self.lab = np.zeros((len(colors), 1, 3), dtype='uint8')
        self.colorNames = []
        # loop over the colors dictionary
        for (i, (name, rgb)) in enumerate(colors.items()):
            # update the L*a*b* array and the color names list
            self.lab[i] = rgb
            self.colorNames.append(name)
        # convert the L*a*b* array from the RGB color space
        # to L*a*b*
        self.lab = cv2.cvtColor(self.lab, cv2.COLOR_RGB2LAB)
    def label(self, image, c):
        # construct a mask for the contour, then compute the
        # average L*a*b* value for the masked region
        mask = np.zeros(image.shape[:2], dtype='uint8')
        cv2.drawContours(mask, [c], -1, 255, -1)
        mask = cv2.erode(mask, None, iterations=2)
        mean = cv2.mean(image, mask=mask)[:3]
        # initialize the minimum distance found thus far
        minDist = (np.inf, None)
        # loop over the known L*a*b* color values
        for (i, row) in enumerate(self.lab):
            # compute the distance between the current L*a*b*
            # color value and the mean of the image
            d = dist.euclidean(row[0], mean)
            # if the distance is smaller than the current distance,
            # then update the bookkeeping variable
            if d < minDist[0]:
                minDist = (d, i)
        # return the name of the color with the smallest distance
        return self.colorNames[minDist[1]]
```

TESTS

Example Usage:

Getting the files

```
git clone https://github.com/utastudents/ShapeReceipe.git
```

Choose an image or Sketch one.

1. Upload an image, or sketch with shapes.
2. Submit the image for processing.
3. Run it over a localhost.

Prerequisites

- Python 3 (not compatible with python 2)
- pip
- Php

Test Case Table:

ID	Test Case Objectives	Pre-requisites	Steps	Input Data	Expected Output	Status
1	Detect Shapes	Image should be given	1. Resize 2. Ratio 3. Blur 4.grayscale 5. Thresholding 6. Dilation 7. Find Contours 8. Detect Shape with edges	1. image.png	Classified Shapes	PASS
2	Detect Colors	Image should be given	1. Resize 2. Ratio 3. Blur 4. Grayscale 5. Convert the RGB to LAB Color Space (L*,a*,b*). 6. Detect Colors with LAB values.	1. image.png	Classified Colors	PASS
3	Generate Website	Image, and filter for website needs to be given	1. Click an image. 2. Upload the image. 3. Select Website filter. 4. Submit the image for processing. 5. Download the design with source code.	1. image.png 2. type = website	HTML, CSS, JAVASCRIPT	PASS
4	Generate Mobile App	Image, and filter for website needs to be given	1. Click an image. 2. Upload the image. 3. Select Website filter. 4. Submit the image for processing. 5. Download the design with source code.	1. image.png 2. type = app	React Native Emascript	PASS