Graphics Programming

Assignment 06 - Hough Transformation

The Hough Transform

The Hough Transform is a technique for detecting features of an image that is useful in image processing and computer vision. It converts shapes, such as lines, ellipses, and circles, from traditional x,y coordinates to parameter space. When converted to parameter space, the peaks present in this accumulator space represent the likely parameters for the x,y coordinates of the desired shape. These peaks can be filtered through and used to detect the likely locations of the shapes.

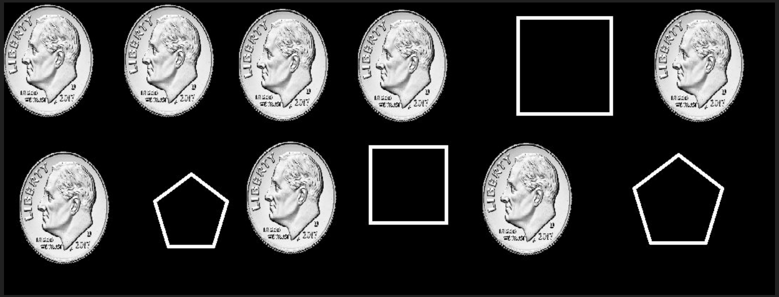
Detecting Circles Code

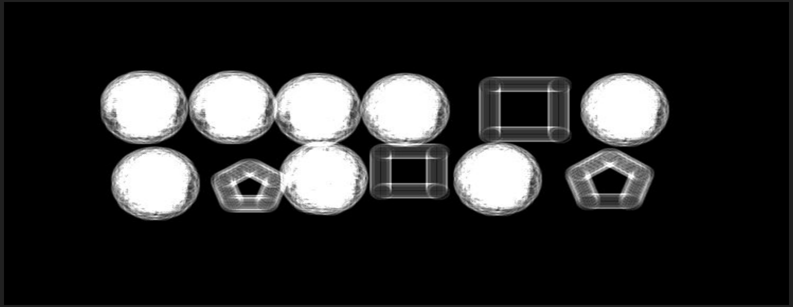
```
edgeMap = cv2.Canny(img, cannyThreshold1, cannyThreshold2)
Y, X, D = np.mgrid[:d, :d, :d]
distances = np.hypot(Y-d//2, X-d//2)
circles = ((D>3)*(distances>D/2-1)*(distances<D/2+1))/(D+.1)
circles = circles[:, :, ::-1]

votes = scipy.signal.correlate(edgeMap[:,:,None]*1.0, circles*1.0)
_, _, v_d = votes.shape
dilated_votes = morphology.grey_dilation(votes, (5, 5, 5))
peaks = ((dilated_votes==votes)*(votes>votes.max()*N))

circled_img = img*1
peaks_y, peaks_x, peaks_d = np.where(peaks)
for i, j, r in zip(peaks_x, peaks_y, peaks_d//2):
      cv2.circle(circled_img, ((i-v_d//2),(j-v_d//2)), r,
(0,255,0), 2)

show(circled_img)
```

An edge map is created from the input image. Then, numpy is used to create a three-dimensional image where all possible circles in that image and their radii are contained. However, this can use lots of memory and so it is often necessary to downsize the input image before hand. Then, scipy is used to ‘rub’ the circle image over the original. The peaks of this vote map are where the circles are.

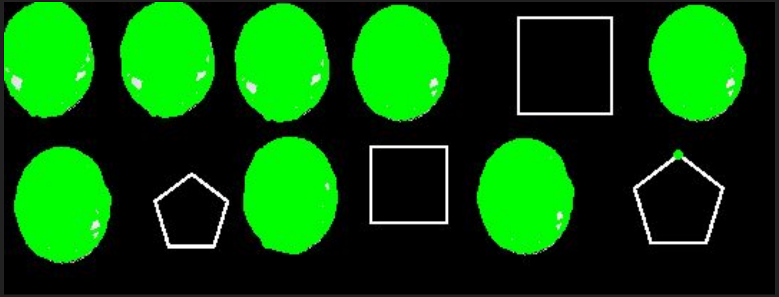
Detecting Circles Result

Instead of detecting the overall circles in the coin image, it detected a bunch of tiny circles within the coins. However, in the votes map you can see it is differentiating between the clearly not circles and the coins.





votes.png



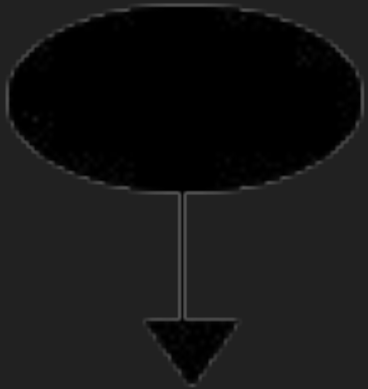
Detecting Irregular Shapes

To detect irregular shapes, the shape is uploaded as a png. Since it is a two dimensional image, it is not necessary to use scipy.signal.correlate but it will still work. Almost all the code is essentially the same, with the circle being replaced with the custom patch.

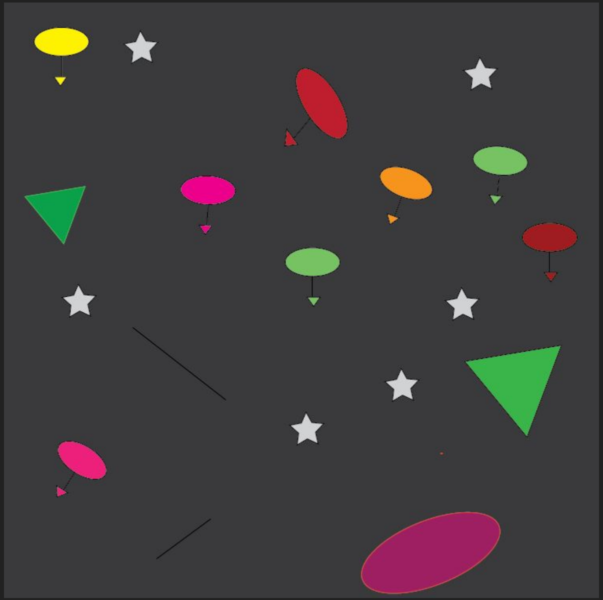
The biggest difference is drawing the patch where the votes are. The patch is drawn transparent using alpha blending.

```
    alpha = (1-patch)
    for i, j in zip(peaks_x, peaks_y):
        if j-ph >= 0 and j < h and i-pw >= 0 and i < w:
            background = alpha*(drawn_img[j-ph:j, i-pw:i])
            # show(background)
            drawn_img[j-ph:j, i-pw:i] = patch + background
```

Detecting Irregular Shapes Results



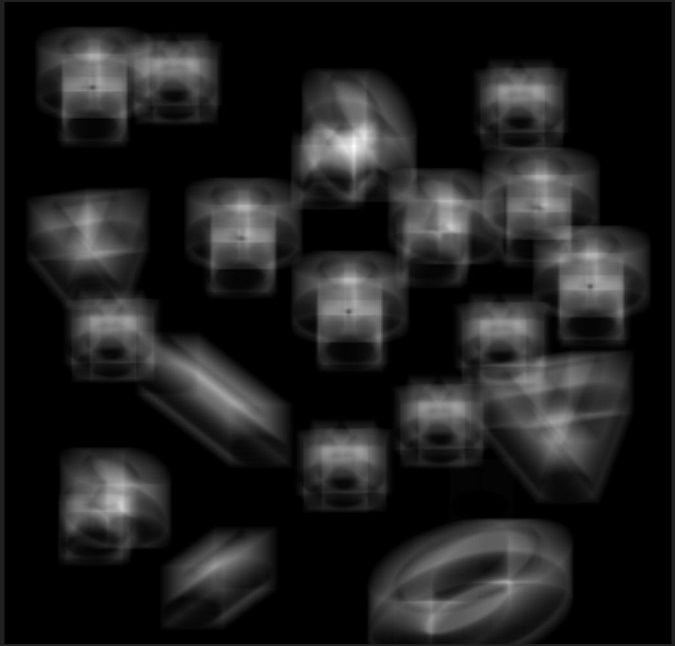
Patch / Irregular Shape

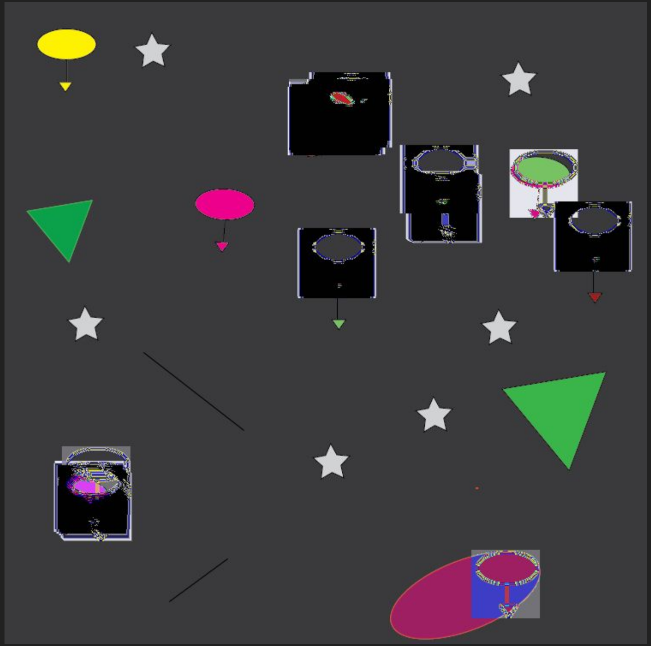


Background

Detecting Irregular Shapes Results pt. 2

For these matches, anything greater than 90% of the max was taken. The outlines look weird because there is transparent patches stacked on top of each other.


Irregular Votes


Detected Shapes