COMP PHOTOGRAPHY FINAL PROJECT

Bridget Cave

Spring 2017
bcave3@gatech.edu

COMIC BOOK SELFIES

Create a cool comic book style photo from any of your selfies!

Project Goal

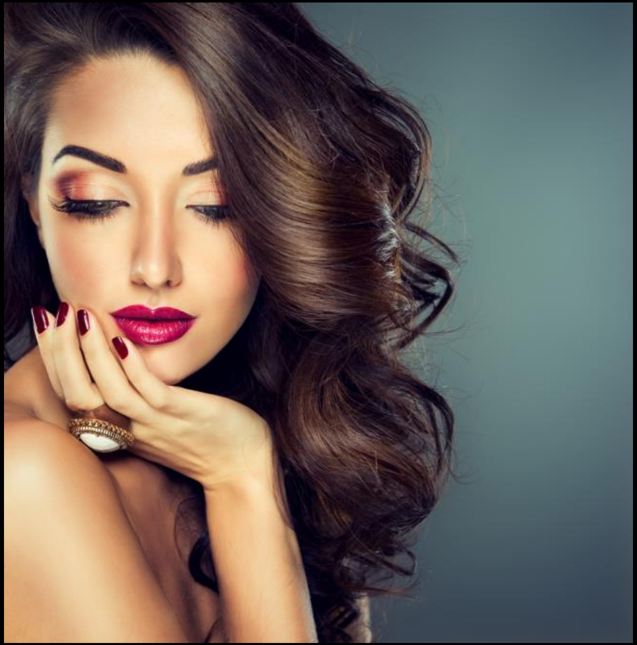
I. Original project scope:
   - To take in an image and use filters, blending, halftoning and feature detection to return a comic book style version of the photo.

II. What motivated you to do this project?
   - There was a post on Piazza entitled “Automating photoshop workflows using code for final project” which included a YouTube video about creating text portraits from an image. In the suggestions for this video, I saw a Photoshop tutorial for transforming a photograph into a digital painting. A few suggestions later, I found the video that inspired my project, “Photoshop Tutorial: How to Make a Comic Book, Pop Art, Cartoon from a Photo.” I found this video to be very interesting and thought this project would be a good match for me because of my interest in comic books. [5]

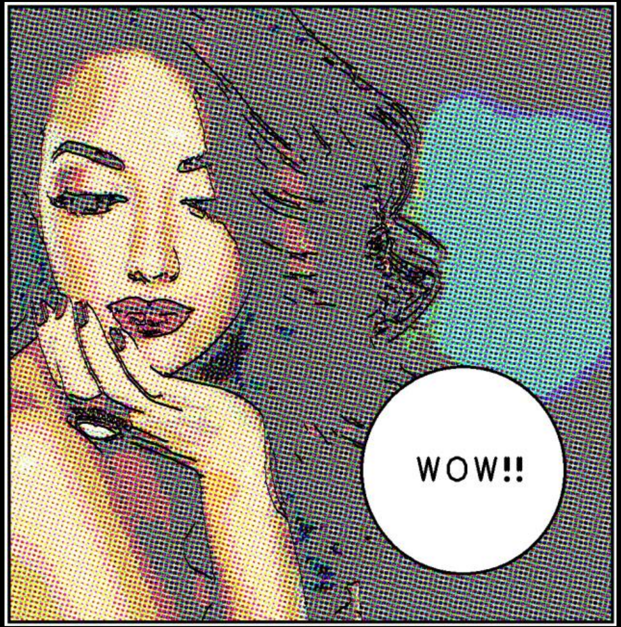
Scope Changes

Originally, I intended to use feature detection to find the light reflection in the eyes of the subject as shown in the video. However, most photos I found did not have this. I instead chose to use feature detection to determine the placement of a speech bubble on the photo. Other than this, the scope of the project remained the same including the fact that all of the steps are automated. I still used filters, blending, halftoning and feature detection to create a comic book inspired output image.

Showcase

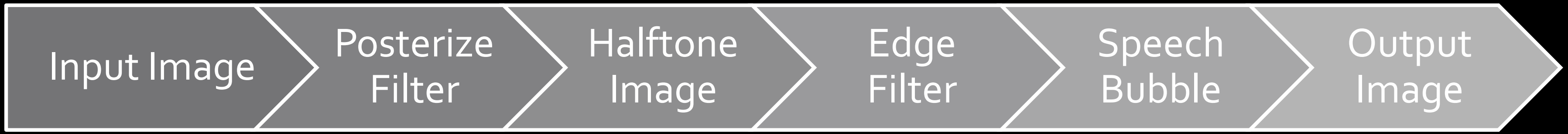


Input



Output

Pipeline

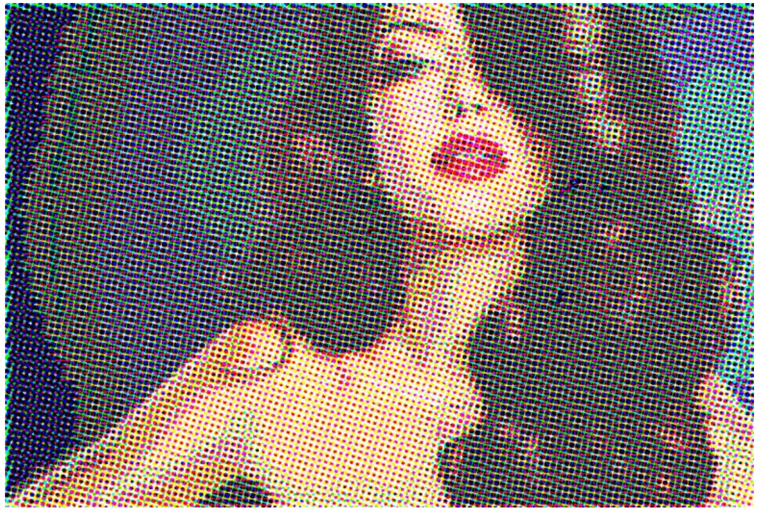

Input Image
Posterize Filter
Halftone Image
Edge Filter
Speech Bubble
Output Image

Posterize Filter

- Posterization is the process of converting gradation of tone to several regions of fewer tones, creating abrupt changes from one tone to the next. [2]
- Photoshop does this by quantizing each channel individually (reducing the number of distinct colors), given a number of specified levels (n). [1] The fewer levels, the more color quantization.

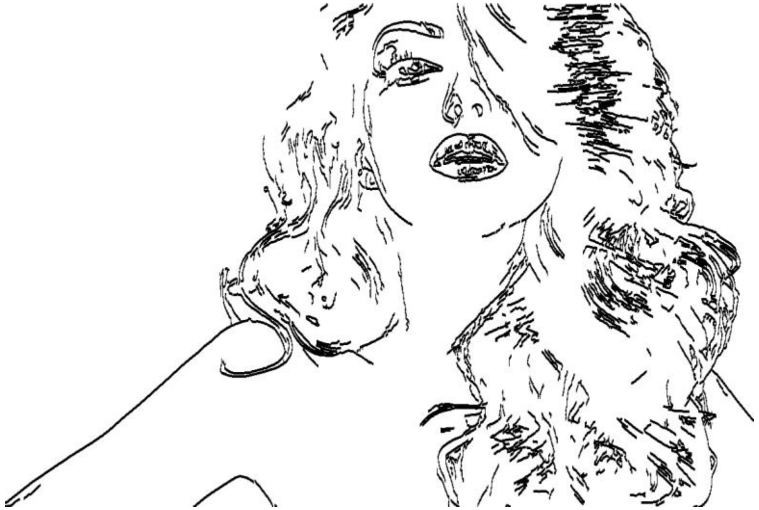
Halftone Image

- Rather than continuous tones, halftone images are created by a series of dots. In this case, the dots represent the different channels of a CMYK image and their size is representative of intensity.



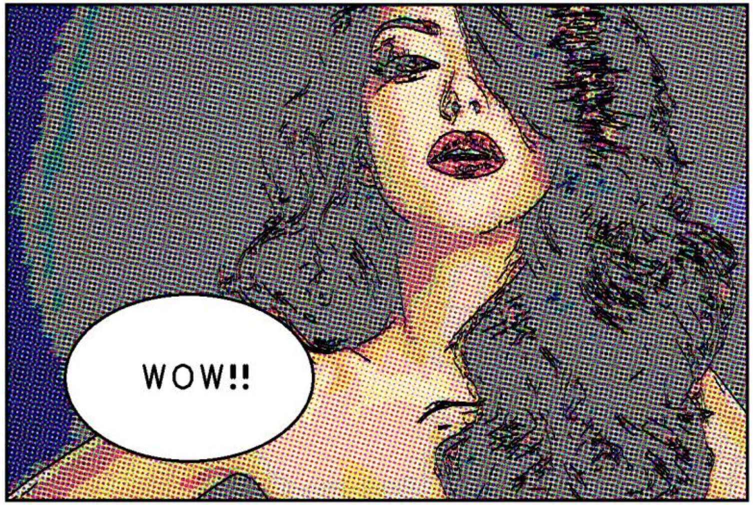
Edge Filter

- Edge detection is a method that finds where there are abrupt changes in brightness.
- I used Canny Edge Detection to create an Edge Map and took the inverse of this to create my Edge Filter than shows the edges of an image in black.

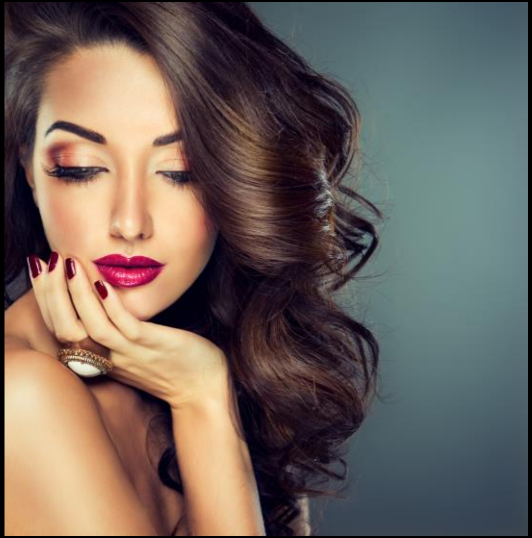


Add Speech Bubble Using Feature Detection

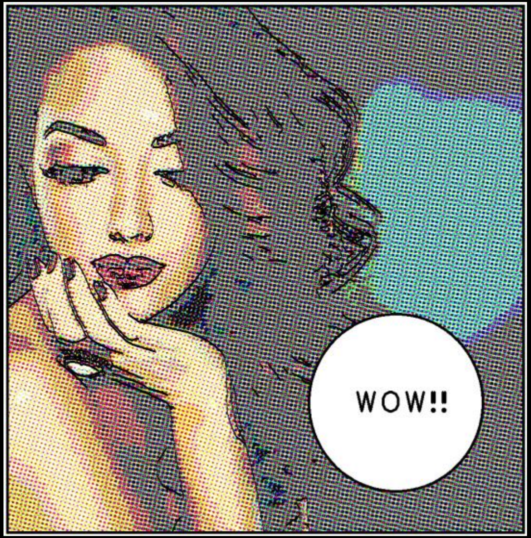
- I used feature detection to find the front of a face and then add a speech bubble in the lower quadrant opposite of the face.
- This will work as long as a face is detected, otherwise no bubble will be added.



Result Set 1

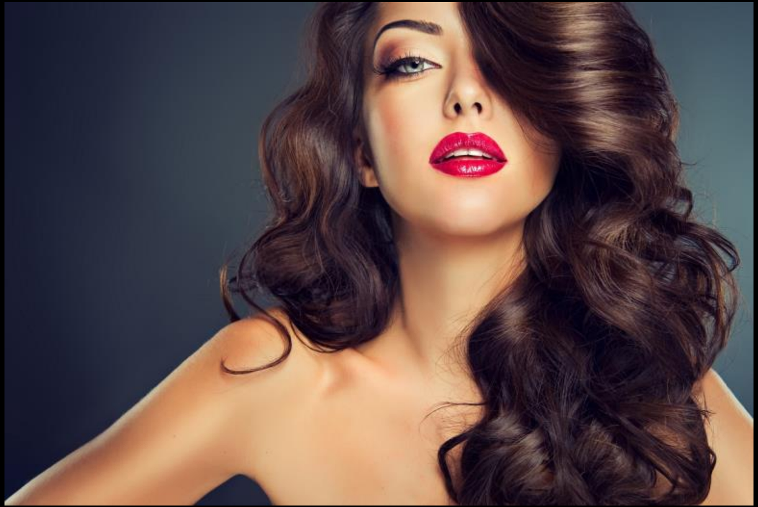


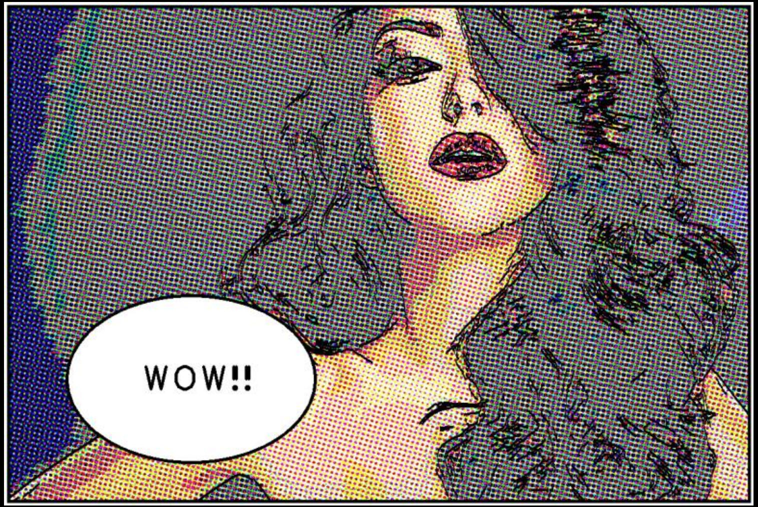
Input



Output

Result Set 2

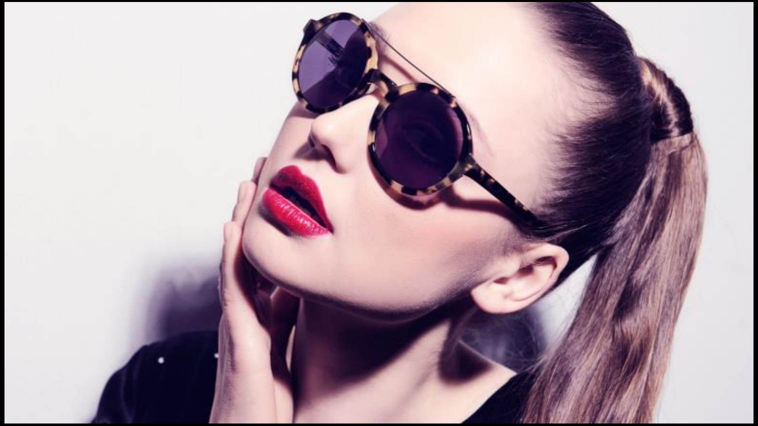


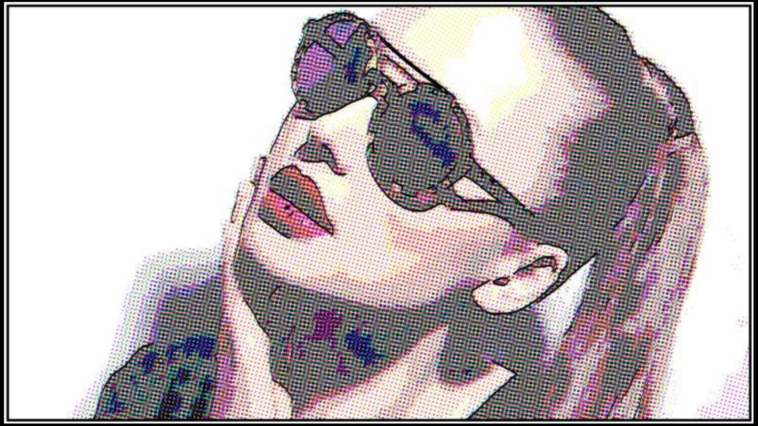


Input

Output

Result Set 3



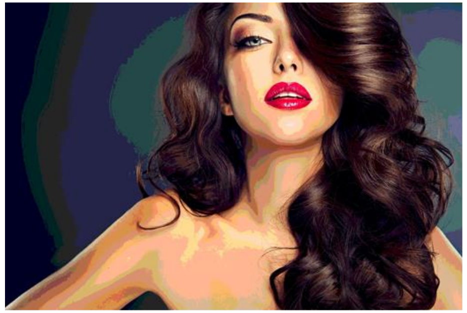


Input

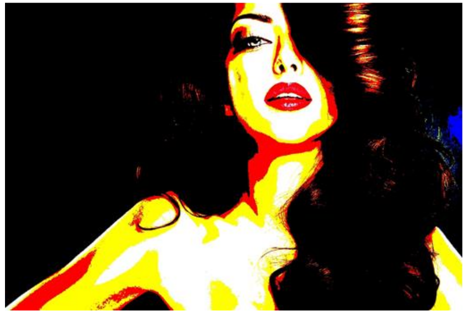
Output

Project Development: Posterization Filter

- The video I based my project on began by applying a "Poster Edges" filter in Photoshop.
- The "Poster Edges filter can be created using a Posterize filter and an Edge Filter.
- The only problems I faced at this step was finding the best value of n, the number of levels of quantization. I went with eight because it create the effect while maintaining the image.



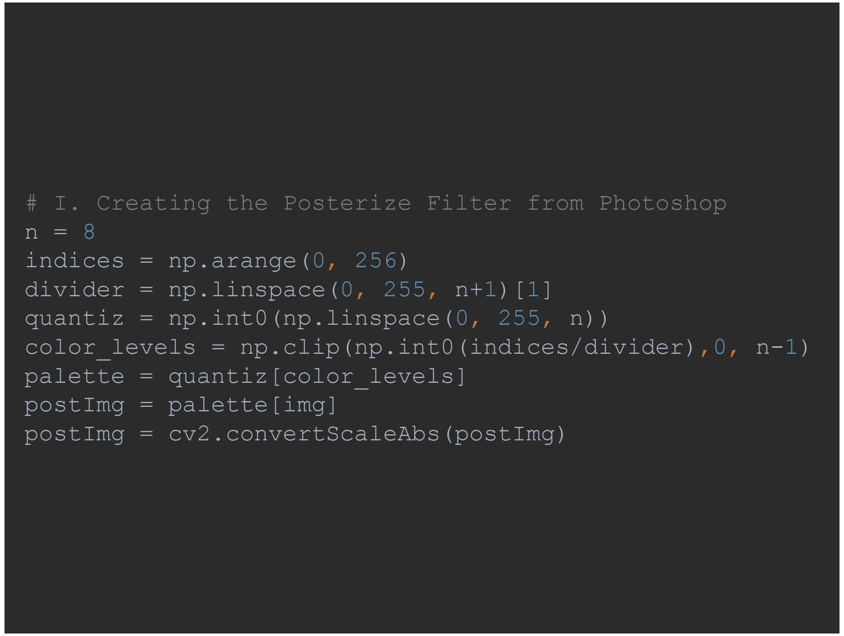
n = 8



n = 2

Code: Posterize Filter

- I began by deciding on the level of quantization.
- Then, I found a list of all the colors, called indices.
- I calculated the divider and found the color levels using n.
- I created a new palette using my number of color levels and then applied it to the image.
- Lastly, I applied the palette to the input image.

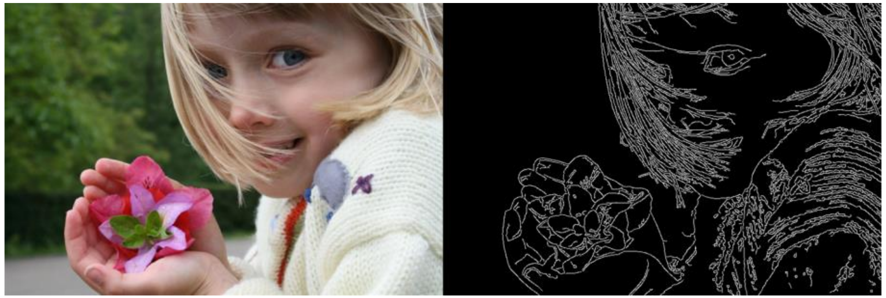


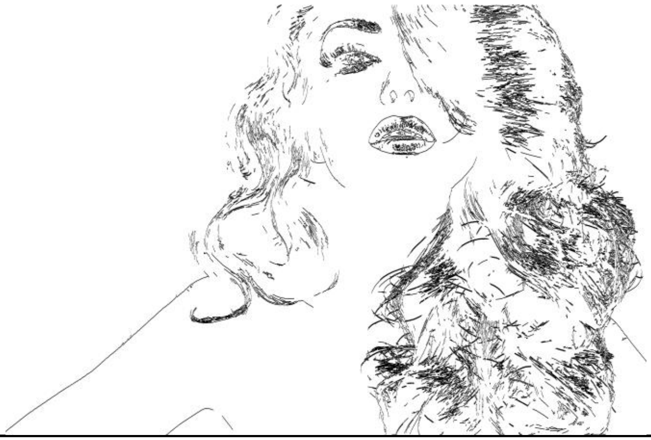
```python
# I. Creating the Posterize Filter from Photoshop
n = 8
indices = np.arange(0, 256)
divider = np.linspace(0, 255, n+1)[1]
quantiz = np.int0(np.linspace(0, 255, n))
color_levels = np.clip(np.int0(indices/divider),0, n-1)
palette = quantiz[color_levels]
postImg = palette[img]
postImg = cv2.convertScaleAbs(postImg)
```

[1]

Project Development: Edge Filter

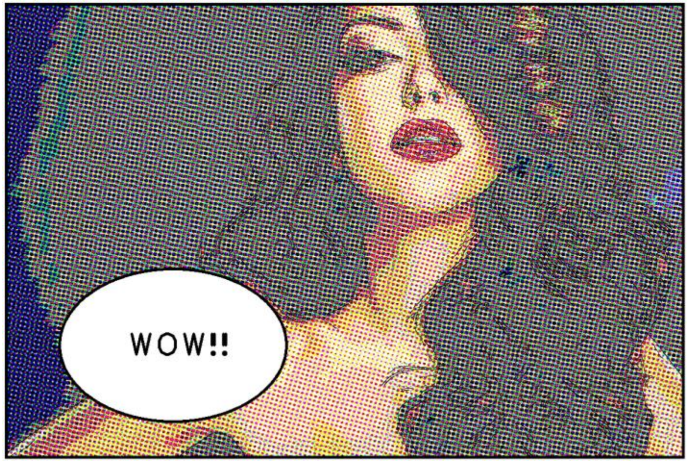
- Although the Edge Filter comes after the Halftone image in the pipeline, I did this second.
- I struggled with this for a while before realizing that the images I was using were too large. I only realized this after running the image from the Wikipedia page on the "Canny_Edge_Detector" and getting the correct results. (top) [6]
- For reference, the top input image is about 700 x 500 while the bottom image (inverse of the edge filter shown) was 3983 x 2675. This created a lot of extra noise and looked bad. Not to mention, the processing time was very long.



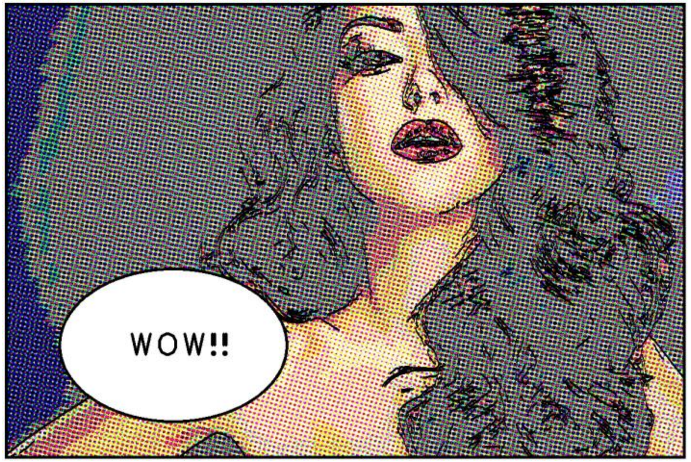


Project Development: Edge Filter

I used a method in cv2 called dilate to make the black outlines more visible. I had to experiment with different levels of dilation to find a good balance. [8]



Original



Dilated

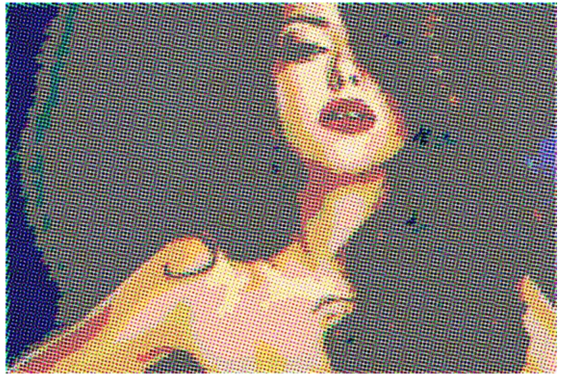
Code: Edge Filter

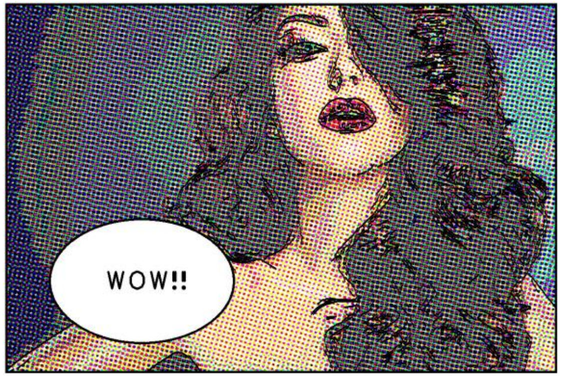
- I used Canny Edge Detection for my Edge Filter. I originally used fixed upper and lower bounds instead calculate the upper and lower bounds for each image.
- Then, I dilated the resulting filter and found the inverse of it.

```python
sigma = 0.33
v = np.median(edgeFilterImg)
lower = int(max(0, (1.0 - sigma) * v))
upper = int(min(255, (1.0 + sigma) * v))
edgeFilterImg = cv2.Canny(edgeFilterImg, lower, upper)
for i in range(0, 3):
    edgeFilterImg = cv2.dilate(edgeFilterImg, (3, 3))
inverseImg = (255 - edgeFilterImg)
inverseImgColor = cv2.cvtColor(inverseImg,
cv2.COLOR_GRAY2RGB)
```

Project Development: Halftone Image

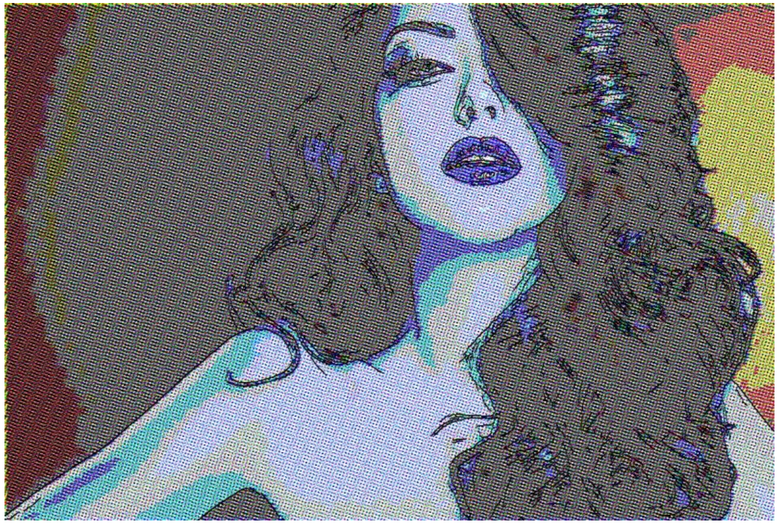
- Originally, I applied the edge filter before I crated the halftone image. This made the dark outline impossible to see. (top)
- I also had to play around with what to make the "sample" variable, which determines the diameter of the circles. The bottom image has a sample of 10, which creates circles with a larger diameter. I thought a slightly smaller diameter looked better, going with an 8.





Project Development: Halftone Image

- The halftone image needed to be combined with the posterized image and so it needed to be converted to an np array. However, the conversion caused a bizarre change in colors.
- I had to Image.save and then cv2.imread the photo in order to fix this problem.



Code: Halftone Image

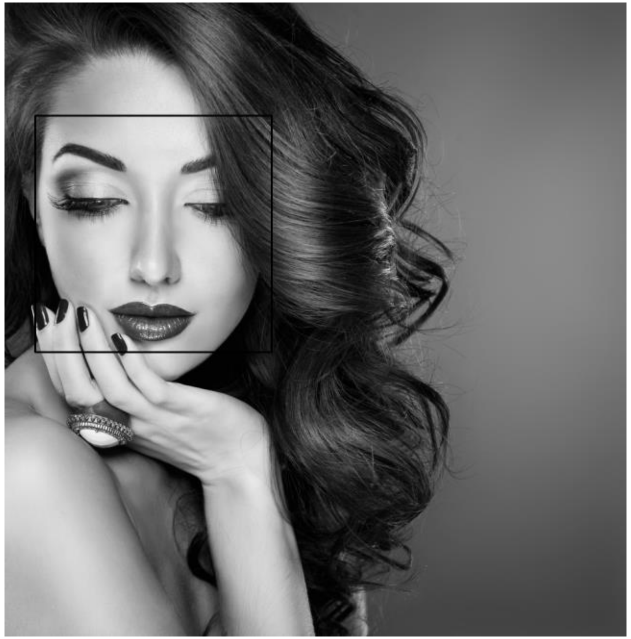
- This was done by first converting the image to CMYK format and splitting the channels apart.
- Then, each of the four separated channels were rotated 0, 15, 30 and 45 degrees respectively. (shown)
- Next, the half-tone of each channel was taken and they were rotated to their original position. (shown)
- Lastly, the channels were all merged back together.

```
for channel in cmyk:
    # 2. Rotate each separated image by 0, 15, 30 and
45 degrees
    channel = channel.rotate(angle, 0, 1)
    size = channel.size[0], channel.size[1]
    half_tone = Image.new('L', size)
    draw = ImageDraw.Draw(half_tone)
    # 3. Take the half-tone of each image
    for x in xrange(0, channel.size[0], sample):
        for y in xrange(0, channel.size[1], sample):
            box = channel.crop((x, y, x + sample, y +
sample))
            stat = ImageStat.Stat(box)
            diameter = (stat.mean[0] / 255) ** 0.5
            edge = 0.5 * (1 - diameter)
            x_pos, y_pos = (x + edge), (y + edge)
            box_edge = sample*diameter
            draw.ellipse((x_pos, y_pos, x_pos +
box_edge, y_pos + box_edge), 255)
    # 4. Rotate back each half-toned image
    half_tone = half_tone.rotate(-angle, 0, 1)
    width_half, height_half = half_tone.size
    xx = (width_half - cmyk_image.size[0]) / 2
    yy = (height_half - cmyk_image.size[1]) / 2
    half_tone = half_tone.crop((xx, yy, xx +
cmyk_image.size[0], yy + cmyk_image.size[1]))
    dots.append(half_tone)
    angle += 15
return dots
```

[3]

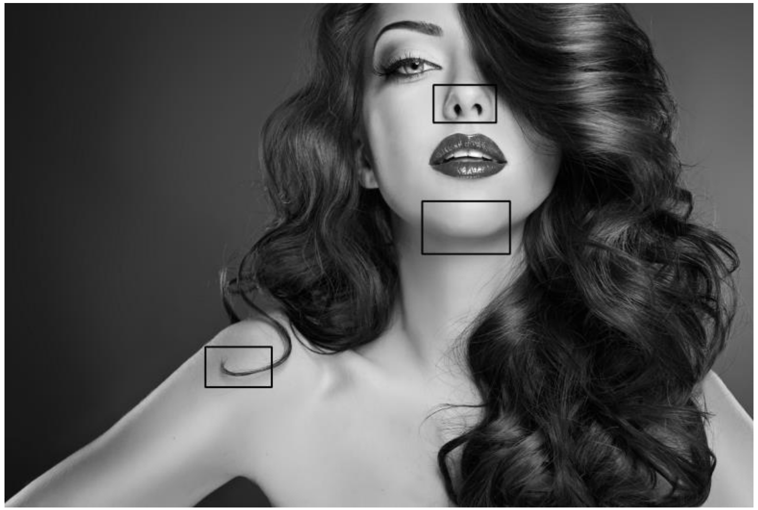
Project Development: Add Speech Bubble Using Feature Detection

- I used feature detection on a grayscale version of the original image since the image never changes size.
- I used Haar Cascades to find the front of a face. A Haar Cascade is a classifier that is trained with hundreds of photos of a particular object of similar size. [10]
- To the right is a box showing the results of the haarcascade_frontalface_alt.xml. [9]



Project Development: Add Speech Bubble Using Feature Detection

- I had trouble finding a classifier that could accurately detect facial features.
- For example, to the right is an example of the results from trying to find a nose.



Code: Add Speech Bubble Using Feature Detection

- I started by making sure that a face front was found, if not then no bubble is added.
- Then, I found the center of the face and the center of the image. Then, I placed the ellipse on the opposite side of the face.
- Lastly, I just experimented with the ellipse size and font size.

```
if len(features) > 0:
    (faceX, faceY, faceW, faceH) = features[0]
    face_center = (faceX + faceW/2, faceY + faceH/2)
    imageCenterX = posterImg.shape[1] / 2
    imageCenterY = posterImg.shape[0] / 2
    ellipseX = 0
    ellipseY = 0
    if face_center[0] > imageCenterX:
        ellipseX = int(imageCenterX * 0.5)
    else:
        ellipseX = int(imageCenterX * 1.5)
    if face_center[1] > imageCenterY:
        ellipseY = int(imageCenterY * 0.5)
    else:
        ellipseY = int(imageCenterY * 1.5)
    xAxis = int(imageCenterX * 0.333)
    yAxis = int(imageCenterY * 0.333)
    poly = cv2.ellipse2Poly((ellipseX, ellipseY), (xAxis,
yAxis), 0, 0, 360, 5)
    posterImg = cv2.fillConvexPoly(posterImg, poly, (255,
255, 255))
    posterImg = cv2.ellipse(posterImg, ((ellipseX,
ellipseY), (xAxis * 2, yAxis * 2), 0), (0, 0, 0), 4)
    posterImg = cv2.putText(posterImg, "WOW!!", (ellipseX
- 80, ellipseY + 20), cv2.FONT_HERSHEY_PLAIN, 4, (0, 0,
0), 4)
```

Product Development and Code: Add Border

- The very last thing I did was add a border to the final image.
- This just added the finishing touch to make the output image really feel like a panel in a comic book.

```
posterImg = cv2.rectangle(posterImg, (0, 0),
(posterImg.shape[1], posterImg.shape[0]), 0, 18)
posterImg = cv2.rectangle(posterImg, (0, 0),
(posterImg.shape[1], posterImg.shape[0]), (255, 255,
255), 8)
```

Project Development

- If I were to continue this project…
  - I would use feature detection to add a tail to the speech bubble that angled toward the face.
  - I would create multiple phrases to be put in the speech bubbles to be chosen at random.
  - I would use feature detection to find the pupil then offset a small amount proportional to pupil size and add a light reflection.

Resources

- Photos:
  - https://thoughtcatalog.files.wordpress.com/2014/03/shutterstock_159672665.jpg
  - http://www.bluemaize.net/im/hair/hair-model-4.jpg
  - http://old.eyewear-magazine.com/wp-content/uploads/2014/03/anderne-1.jpg
- http://stackoverflow.com/questions/11064454/adobe-photoshop-style-posterization-and-opencv [1]
- https://en.wikipedia.org/wiki/Posterization [2]
- http://stackoverflow.com/questions/10572274/halftone-images-in-python [3]
- http://stackoverflow.com/questions/3870748/implementing-photoshops-poster-edges-filter [4]
- https://www.youtube.com/watch?v=7vFdXq-So8g&t=358s [5]
- https://en.wikipedia.org/wiki/Canny_edge_detector [6]
- http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/ [7]
- http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html [8]
- http://alereimondo.no-ip.org/opencv/34 [9]
- http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html [10]

APPENDIX

https://github.com/BridgetCave/CS4475-FinalProject

Credits or Thanks

- I would like to thank Dr. Essa for a thought-provoking, memorable class that has fostered an interest in photography and computational photography in me.
- I would like to thank our helpful TAs for their assistance throughout this semester.
- I would like to thank Robert Plante for helping me with the idea for this project, pointing me in direction of face detection using Haar Cascades, and encouraging me to do more with this assignment. (Also, extra shout out to Robert for letting me turn one of the rooms in his apartment into a pinhole camera.)