Module 1c Review & Masking

Today's class was a review of the concepts we learned up to this point plus the additional tool of masking. The pixel array, color spaces, and histograms were all reviewed in this notebook.

Review of Markdown

Sizes of headers

Module

Module

Module

Module

You can use HTML too!

1. item one
2. item two

Python List & NumPy Array

In [ ]:

```
# They both used Brackets []

# Python List
list = [1,2,3]
print("Python list:", list)

# Numpy array
import numpy as np
arr = np.array([1,2,3])
print("NumPy array:", arr)
```

```
Python list: [1, 2, 3]
NumPy array: [1 2 3]
```

In [ ]:

```
# Both can access the elements with an index.

# Python List
print("Python list:",list[0])

# Numpy Array
print("Numpy Array:",arr[0])

# And slice/filter out elements
```

```
# Python List
print("Python list:",list[:2])

# Numpy Array
print("Numpy Array:",arr[:2])
```

```
Python list: 1
Numpy Array: 1
Python list: [1, 2]
Numpy Array: [1 2]
```

In [ ]:

```
# Python lists don't support mathematical operations

list = list*2
print("Python list:", list)

# Multiplication repeats the array elements
```

```
Python list: [1, 2, 3, 1, 2, 3]
```

In [ ]:

```
# NumPy array's support mathematical operations.  Very important for us!

arr = arr*2
print("NumPy array:",arr)
```

```
NumPy array: [2 4 6]
```

In [ ]:

```
# NumPy array's can set the data types.   Very import for us!
arr = np.array([1,2,3], np.uint8)
```

In [ ]:

```
print("NumPy array:", arr.dtype)
```

```
NumPy array: uint8
```

In [ ]:

```
# Import Libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:

```
# Import image using openCv
img = cv2.imread("Graphics/Faces/tng.jpg")
```

In [ ]:

```
# Total Pixels in the image
print(img.size)
```

```
9024000
```

In [ ]:

```
# What is the shape of the image
print(img.shape)
```

```
(1504, 2000, 3)
```

In [ ]:

```
# What type type is the image.
print(img.dtype)
```

```
uint8
```

In [ ]:

```
# Print the first pixel.
print(img[0][0])
```

```
[29 18  0]
```

In [ ]:

```
#  Convert BGR to RGB
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [ ]:
```python
# Print the first pixel.
print(img[0][0])
```

```
[ 0 18 29]
```

In [ ]:
```python
# Convert image to HSV
img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
```

In [ ]:
```python
# Print the first pixel.
print(img[0][0])
```

```
[101 255  29]
```

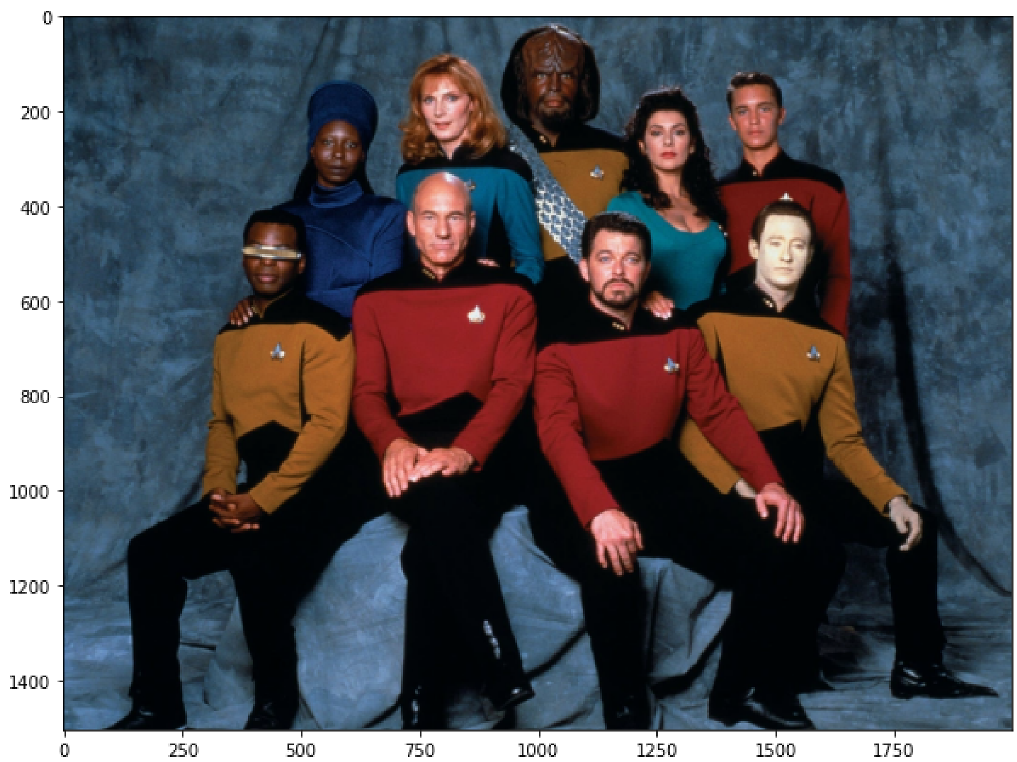
In [ ]:
```python
# Convert back to RGB
img = cv2.cvtColor(img, cv2.COLOR_HSV2RGB)
```

In [ ]:
```python
# Print the first pixel.
print(img[0][0])
```

```
[ 0 18 29]
```

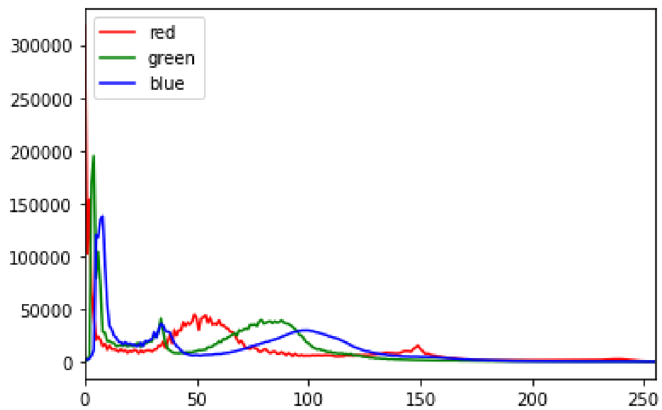
In [ ]:
```python
# Use plt.figure to adjust the size of your plotted image.
fig = plt.figure(figsize=(10,10))

# Plot image
plt.imshow(img)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c57b3280>



In [ ]:
```python
# Create a RGB histogram of our image to examine the color data.

# Red Channel
hist = cv2.calcHist([img],[0],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='red')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])
```

```
# Green Channel
hist = cv2.calcHist([img],[1],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='green')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])

# Blue Channel
hist = cv2.calcHist([img],[2],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='blue')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])

# Add a legend
plt.legend(('red','green','blue'), loc = 'upper left')
# Plot the histogram
plt.show()
```

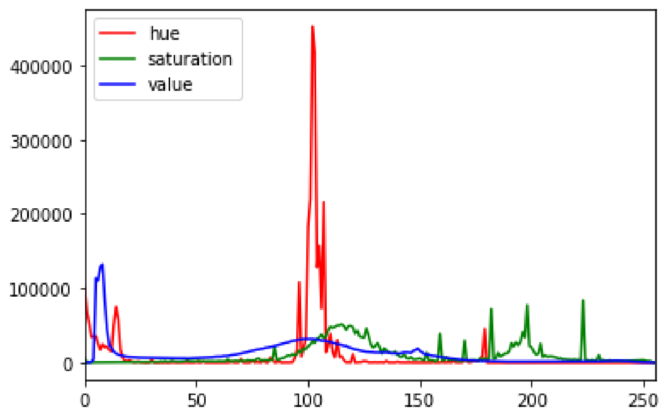


In [ ]:

```
# Convert to HSV
img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
```

In [ ]:

```
# Create a histogram of HSV values.

# Hue
hist = cv2.calcHist([img],[0],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='red')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])

# Saturation
hist = cv2.calcHist([img],[1],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='green')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])

# Value
hist = cv2.calcHist([img],[2],None,[256],[0,255])
# Add histogram to the plot.
plt.plot(hist, color='blue')
# Limit the plot range on the x-axis to 256 values.
plt.xlim([0,255])

# Add a legend
plt.legend(('hue','saturation','value'), loc = 'upper left')

# Plot a histogram.
plt.show()
```



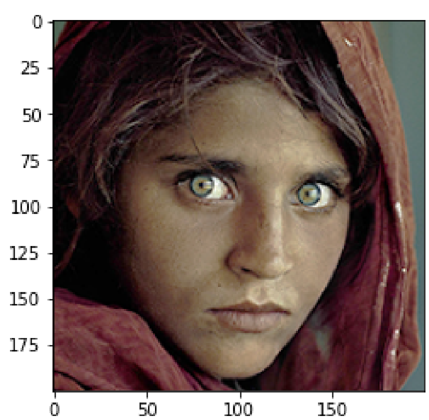
In [ ]:

```
# Import the face image.
img = cv2.imread('Graphics/face.png')
```

In [ ]:

```
# Convert from BGR to RGB
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Show the image.
plt.imshow(img)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c5c80880>



In [ ]:

```
# Convert the image back to BGR so you can use it with OpenCV functions.

img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
```

Create Masks and Bounding Boxes on the Image.

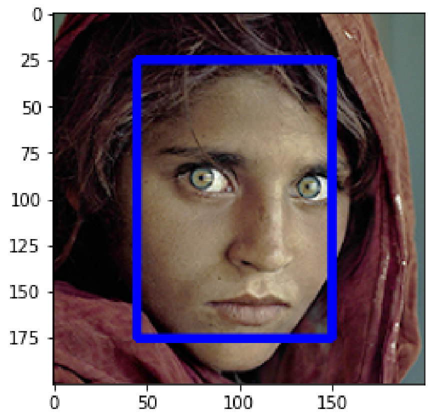
This allows us to analyze specific portions of the image. We will draw a rectangle and circle on the image. Then we will use the rectangle coordinates to create a mask on our image. Using this mask we can analyze the histogram of this region.

In [ ]:

```
# Rectangle OpenCv function
# cv2.rectangle(img, top_left, bottom_right, rect_color, thickness)

# Top left corner of rectangle
top_left = (45,25)

# Bottom right corner of rectangle
bottom_right = (150,175)

# Color
rect_color = (255,0,0)

# Thickness
thickness = 3
```

```
# Rectangle function
img_rect = cv2.rectangle(img, top_left, bottom_right, rect_color, thickness)

# Convert to RGB
img_rect = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Plot image
plt.imshow(img_rect)
```

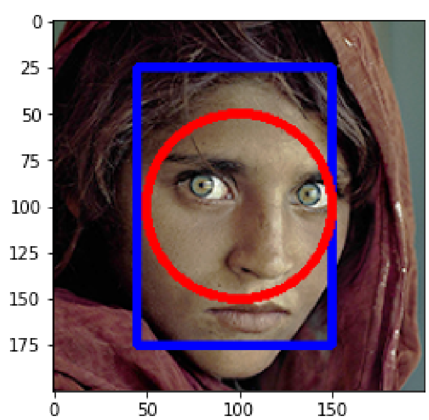
Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c621a310>



In [ ]:

```
# Circle function
# cv2.circle(img, center_circle, radius, circle_color, thickness)

# Center of circle
center_circle = (100,100)

# Radius of the circle
radius = 50

# Color
circle_color = (0, 0, 255)

# Add circle to image
img_circle = cv2.circle(img, center_circle, radius, circle_color, thickness)

# Convert color
img_circle = cv2.cvtColor(img_circle, cv2.COLOR_BGR2RGB)

# Plot image
plt.imshow(img_circle)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c6e756d0>



CREATE A MASK!

In [ ]:

```
# Paste the rectangle coordinates so you can reference them.
```

```python
# Top left corner of rectangle
# top_left = (45,25)

# Bottom right corner of rectangle
# bottom_right = (150,175)

#Create an empty 2D array filled with zeros the same shape as our image.  Notice we are defining the data ty
mask = np.zeros(img.shape[:2], np.uint8)
```

In [ ]:

```python
# Set the masked to white. Remember black is 0 and white is 255 is 8 bit color.
# The array positions are all the y coordinates and then the x coordinates of our rectangle.

mask[25:175, 45:150] = 255
```

In [ ]:

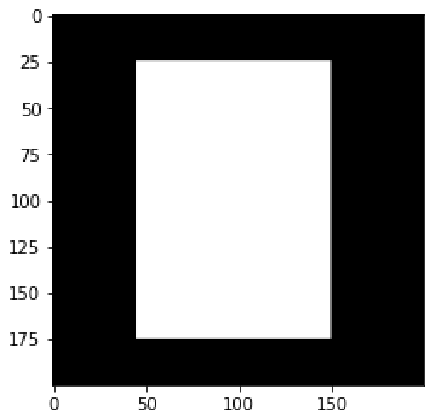
```python
# Image shape to confirm it's the same size as our image.
print(mask.shape)
```

```
(200, 200)
```

In [ ]:

```python
# Plot mask. Make sure you tell matplotlib that you it only contains gray values.
plt.imshow(mask, cmap="gray")
```

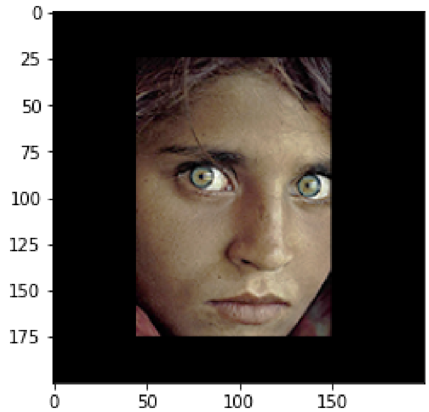
Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c58b2880>



In [ ]:

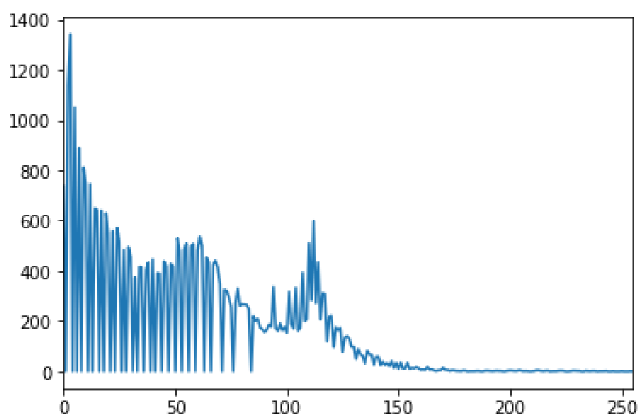
```python
# Let's add our mask to our image.

# Import the face image again.
img = cv2.imread("Graphics/face.png")

# The bitwise AND operator ( & ) compares each bit of the first operand to the corresponding bit of the seco
# If both bits are 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is s
# First image to compare.  Second image to compare.  Mask to use.
img_masked = cv2.bitwise_and(img, img, mask=mask)

# Convert our image to RGB so we can plot it.
img_masked = cv2.cvtColor(img_masked, cv2.COLOR_BGR2RGB)

# Plot the masked image.
plt.imshow(img_masked)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7fe2c7006f10>



In [ ]:

```
# Let's create a RGB histogram of our image.

# Create a histogram of the red channel.
hist_full = cv2.calcHist([img],[0],None,[256],[0,255])

# Plot
plt.plot(hist_full)
# Limit range
plt.xlim([0,255])
# Show plot
plt.show()
```



In [ ]:

```
# Let's add the mask we created with the rectangle mask parameter of the openCV histogram function.  Now we

hist_mask = cv2.calcHist([img],[0],mask,[256],[0,255])
# Plot
plt.plot(hist_mask)
# Limit range
plt.xlim([0,255])
# Show plot
plt.show()
```