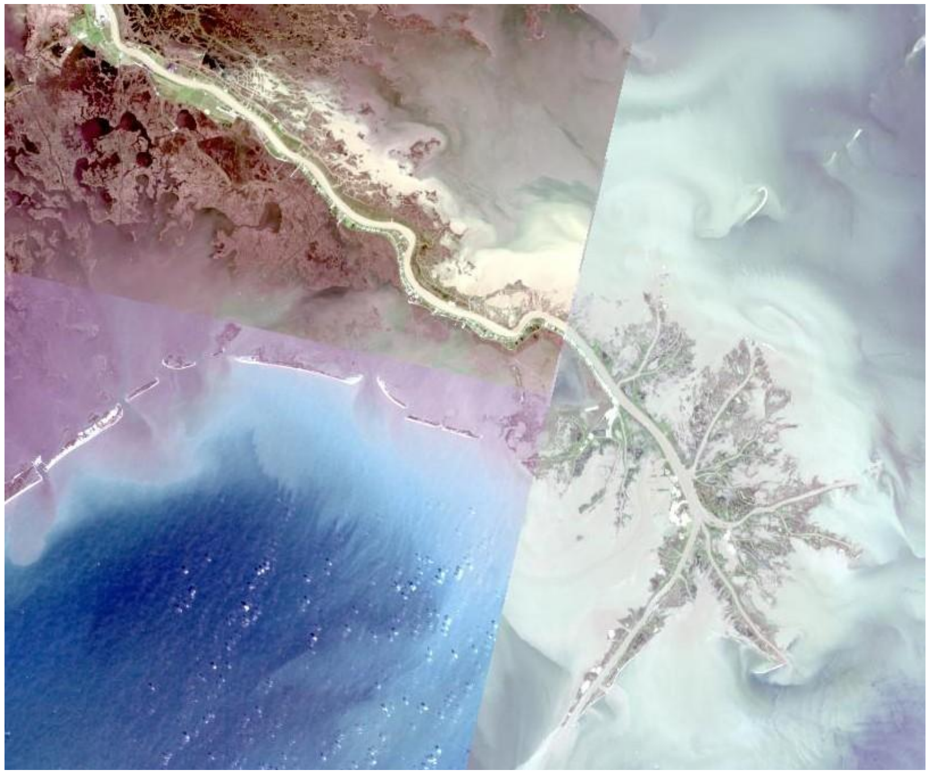
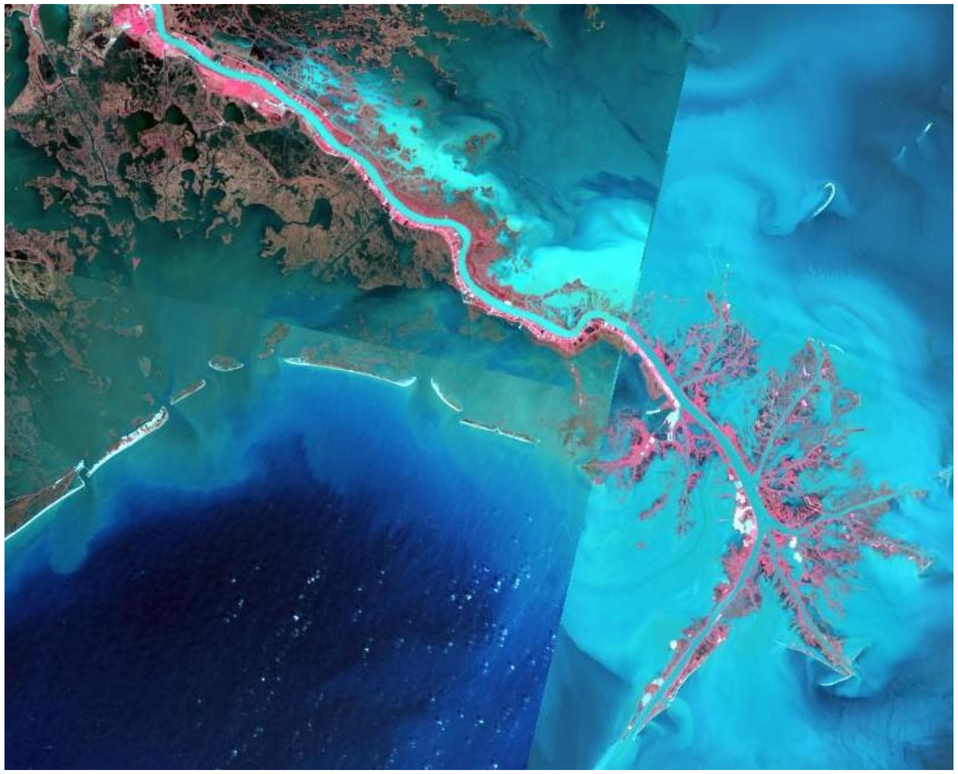
Image Processing





Figures: Esri, HERE, Garmin, USGS, NGA, EPA, USDA NPS

Image Processing

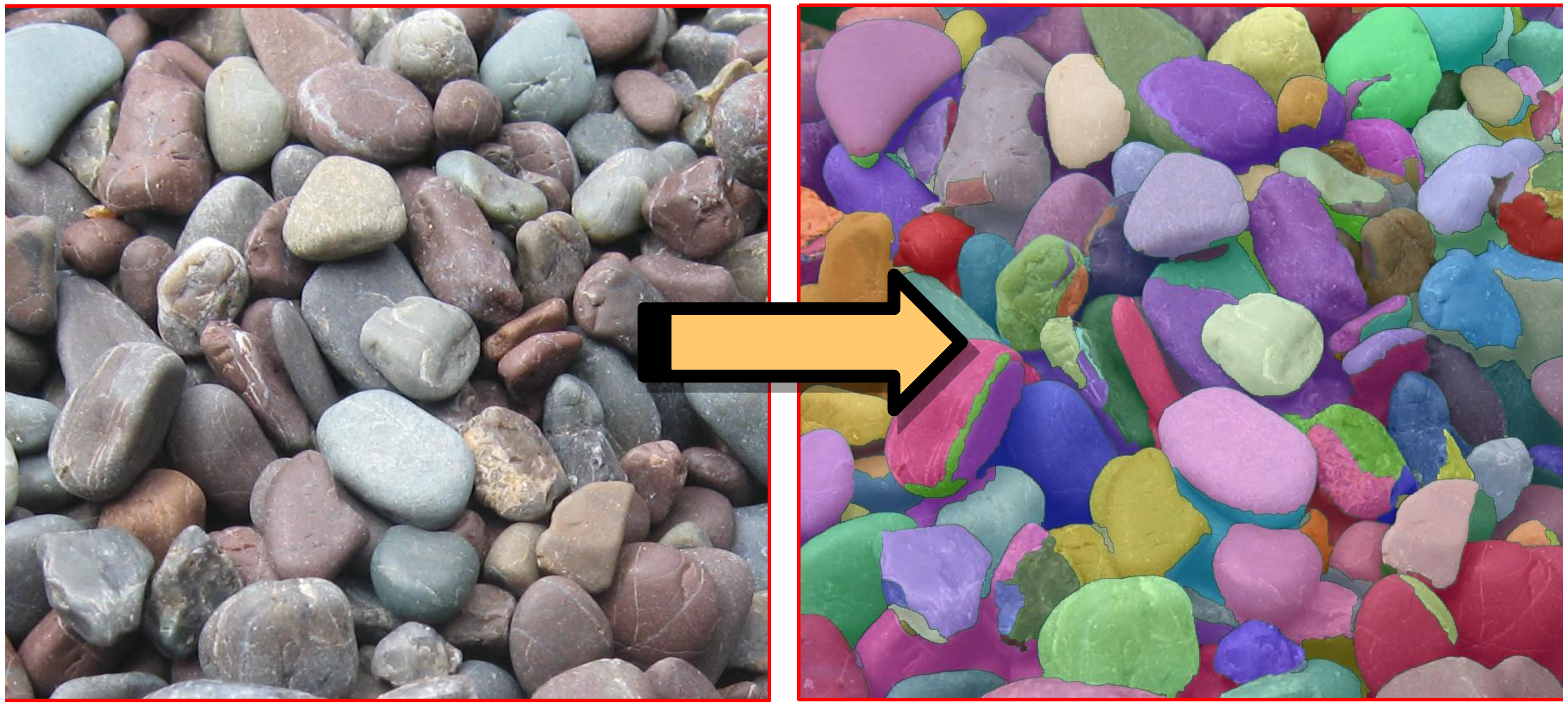


Image processing packages

- There are a number of python packages specifically designed for image processing and manipulation. In this unit, we will make use of the following packages:
  - **Scikit-image**
  - **Pillow (PIL)**
  - **OpenCV**

Loading and Viewing images

Scikit-image package

```
from skimage import io

image = io.imread('hawk.jpg')
```

View using matplotlib

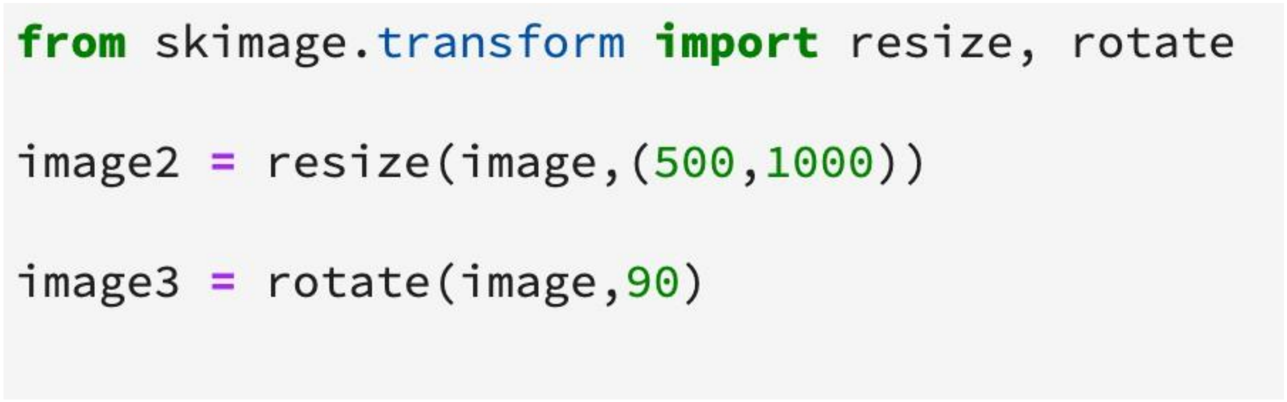
```
plt.axis('off') # turn off axis labels

plt.imshow(image)
```

Or the Scikit-image viewer

```
from skimage.viewer import ImageViewer

viewer = ImageViewer(image)
viewer.show()
```

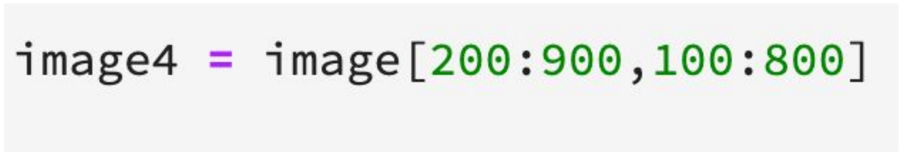
Transformations Scikit-image

- Scikit-image allows you do many of the transformations you would expect in a modern image viewing software



```python
from skimage.transform import resize, rotate

image2 = resize(image,(500,1000))

image3 = rotate(image,90)
```

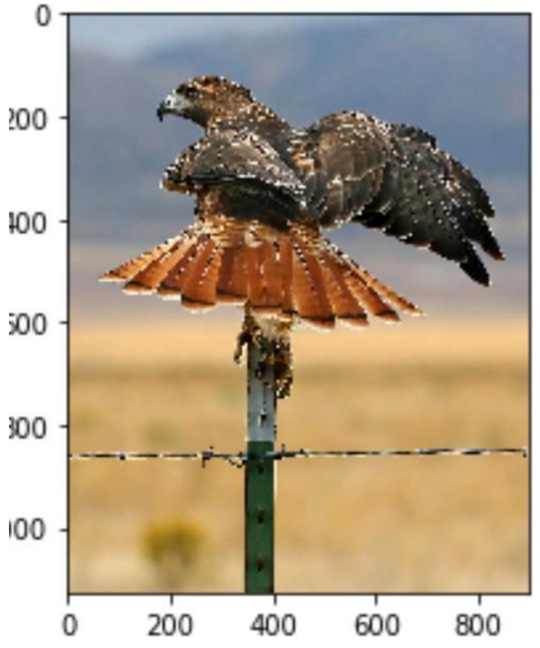
- Because images are stored as arrays, you can crop images using standard numpy array indexing



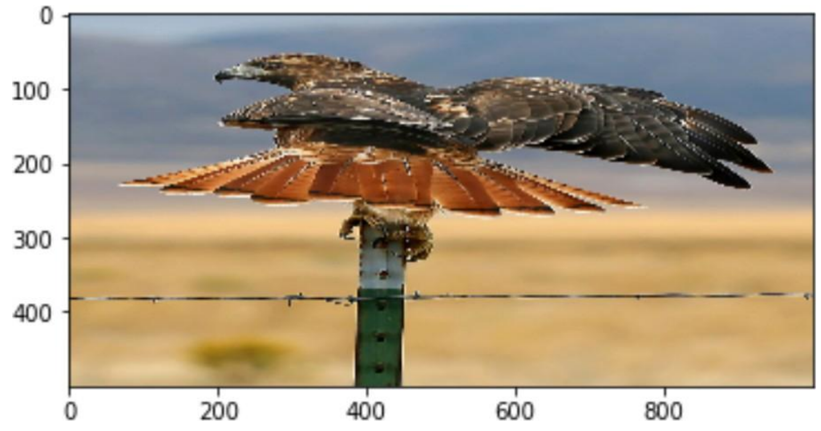
```python
image4 = image[200:900,100:800]
```

Transformations Scikit-image

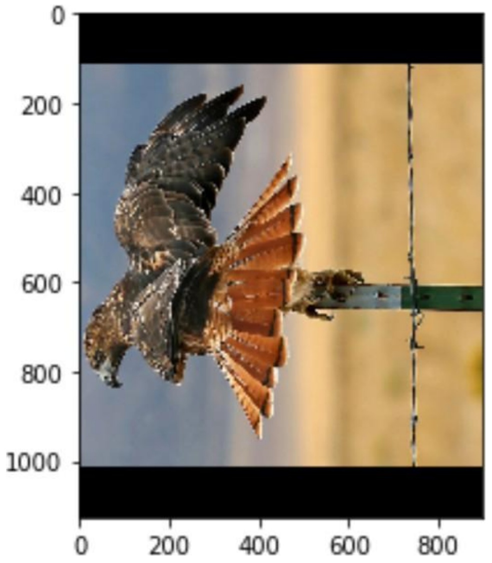
Original



Re-scaled



rotated



cropped

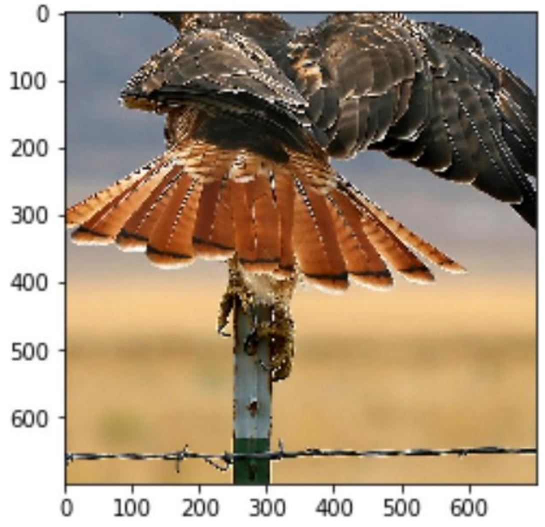
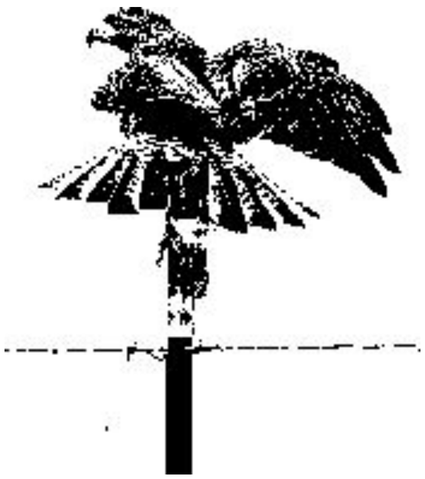
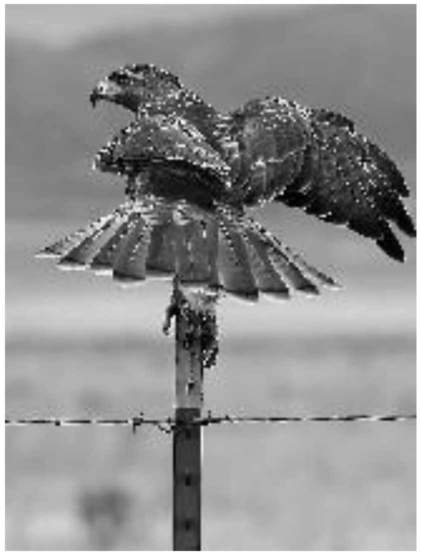


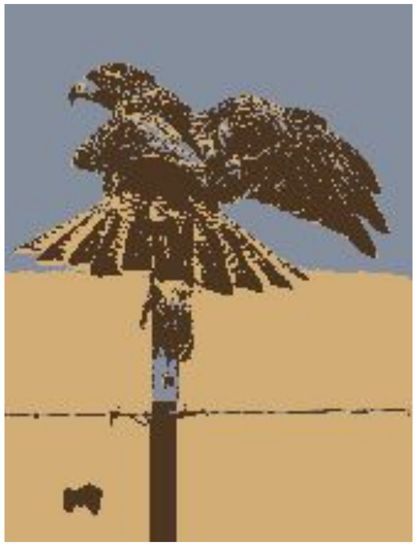
Image Types

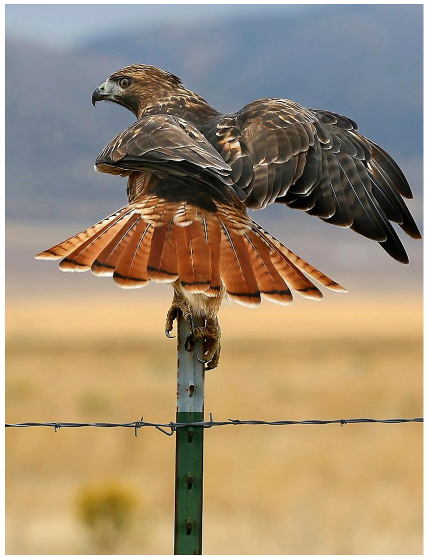
Image types





- Binary
  (black & white, bi-level)
- Grayscale
  (or gray level, intensity)





- Indexed (or pseudo-color)
- True color (or RGB color)

All of these image types can be expressed using Numpy arrays!

Binary Image (Black & White)

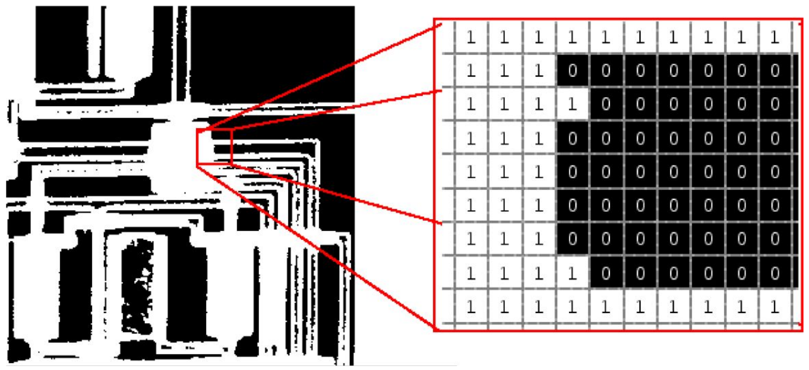
Two values:
0 or 1, background or foreground

Typically stored in a *logical* array Initialized with :

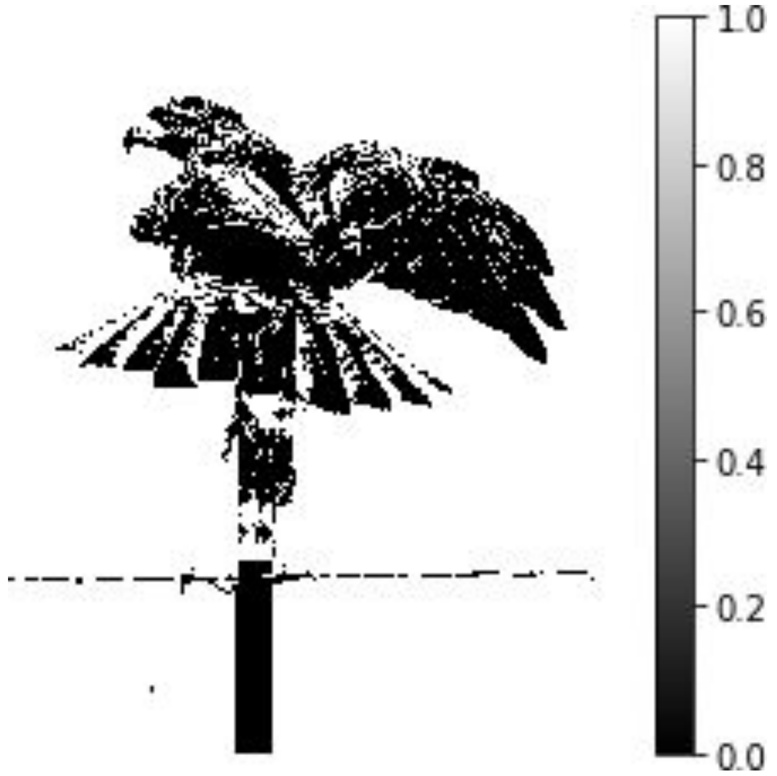
```
np.zeros([rows,cols],dtype=bool)
```

Converted (from a grayscale image) with :

```
thresh = 0.4 # Threshold can be picked manually
binary = image > thresh
```



Pixel Values in a Binary Image



Grayscale Image

Array values represent intensity

Conventionally stored in a *uint8* array with values ranging from 0 to 255 (1 byte per pixel)

Can also be stored in a *Float64* array with values ranging from 0 to 1

- **(the default for scikit-image)**
- **To convert back to the conventional:**

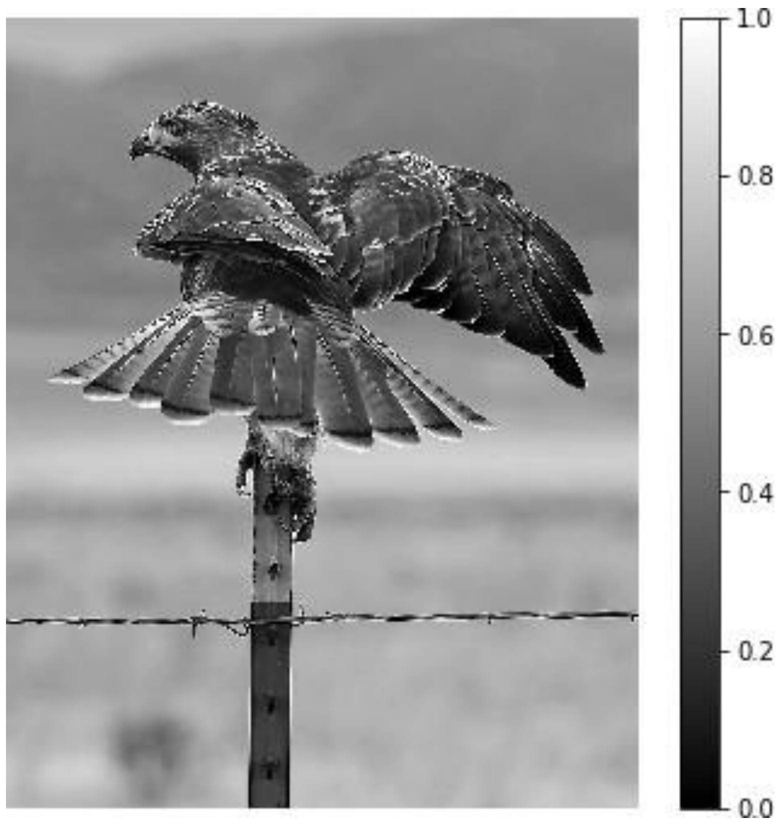
```
from skimage import img_as_ubyte
img_as_ubyte(image_grey)
```

Initialized with :

```
np.zeros([rows,cols], dtype=np.uint8)
```

Converted with :

```
from skimage.color import rgb2gray
image_grey = rgb2gray(image)
```



Intensities can be displayed with different Matplotlib colormaps:

```
plt.imshow(image_grey,cmap='gray')
plt.colorbar()
```

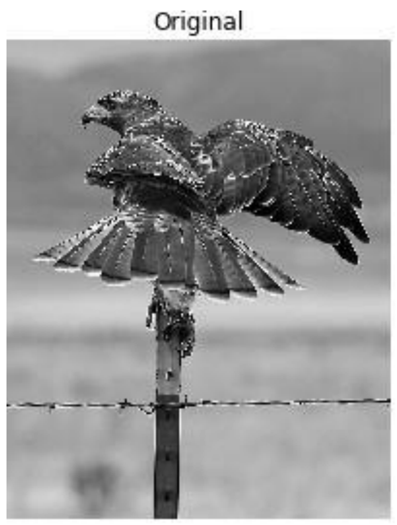
Thresholding (Grayscale to binary)

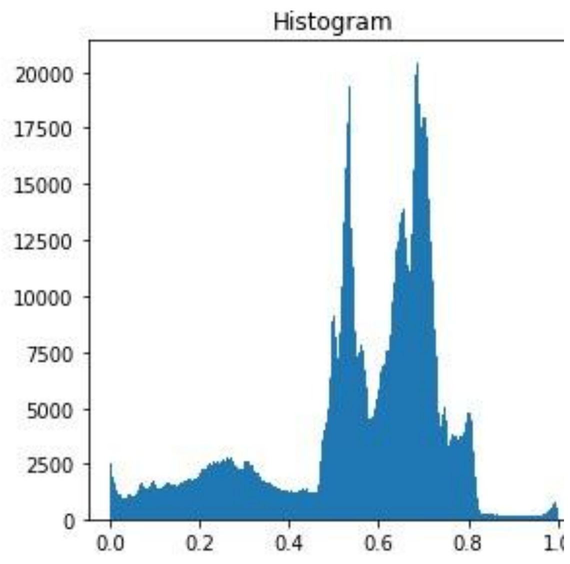
It is helpful to look at a histogram of intensities when selecting a threshold.

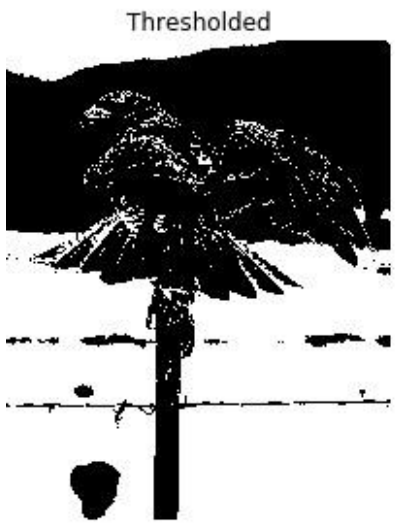
```
plt.hist(image.ravel(), bins=256)
plt.axvline(thresh, color='r')
```

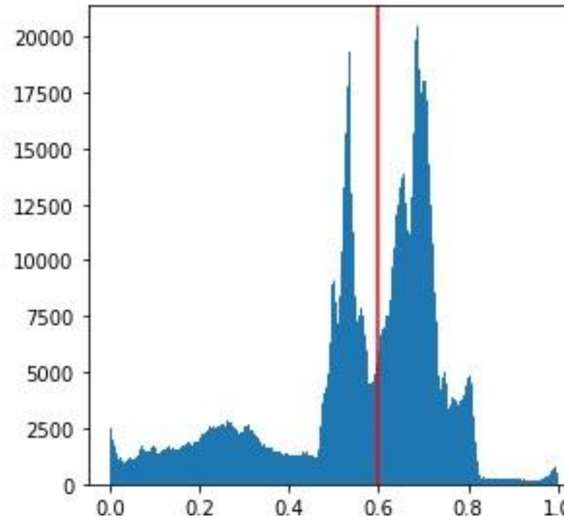
Scikit-image provides several different functions for selecting a threshold, e.g.

```
from skimage.filters import threshold_mean

thresh = threshold_mean(image)
binary = image > thresh
```









Thresholding (Grayscale to binary)

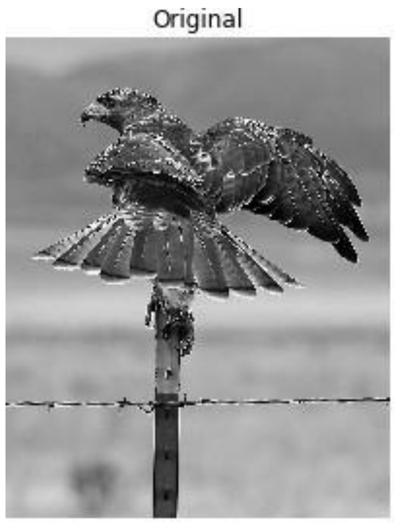
It is helpful to look at a histogram of intensities when selecting a threshold.

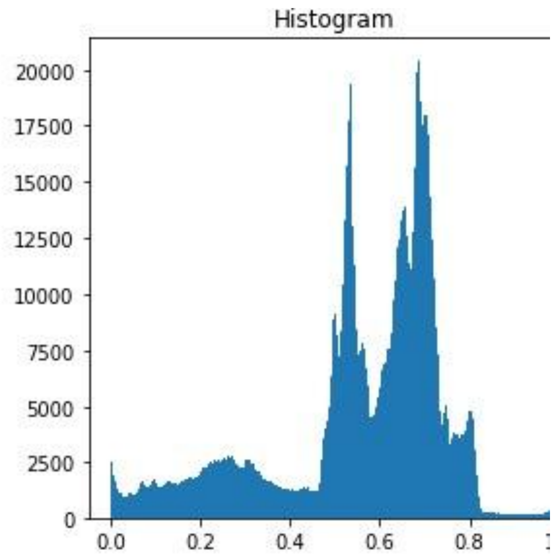
```
plt.hist(image.ravel(), bins=256)
plt.axvline(thresh, color='r')
```

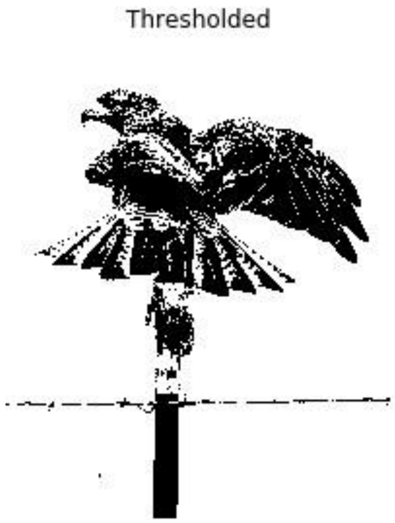
Scikit-image provides several different functions for selecting a threshold, e.g.

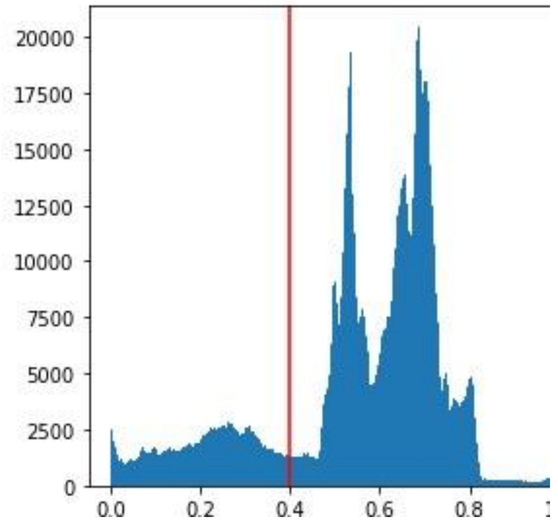
```
from skimage.filters import threshold_mean

thresh = threshold_mean(image)
binary = image > thresh
```

Different features may be emphasized by choosing a different threshold









True color Image

Values represent intensities for Red, Green, and Blue colors

Typically stored in a *uint8* array with values ranging from 0 to 255 (or in a *double* array with values between 0 and 1)

Note: scikit-image manipulations return Float64 between 0 and 1 (to avoid limits to precision)

Initialized with :

```
np.zeros([rows,cols,3], dtype=np.uint8)
```

Most graphic formats store images as 24-bit images (8 for red, 8 for blue and 8 for green)

This yields about 16 million colors!

Truecolor Image

Can also be stored in a *double* array with values ranging from 0 to 1

For a pixel (i,j):

- The red component is stored in

  `RGB[i,j,1]`

- The green component is stored in

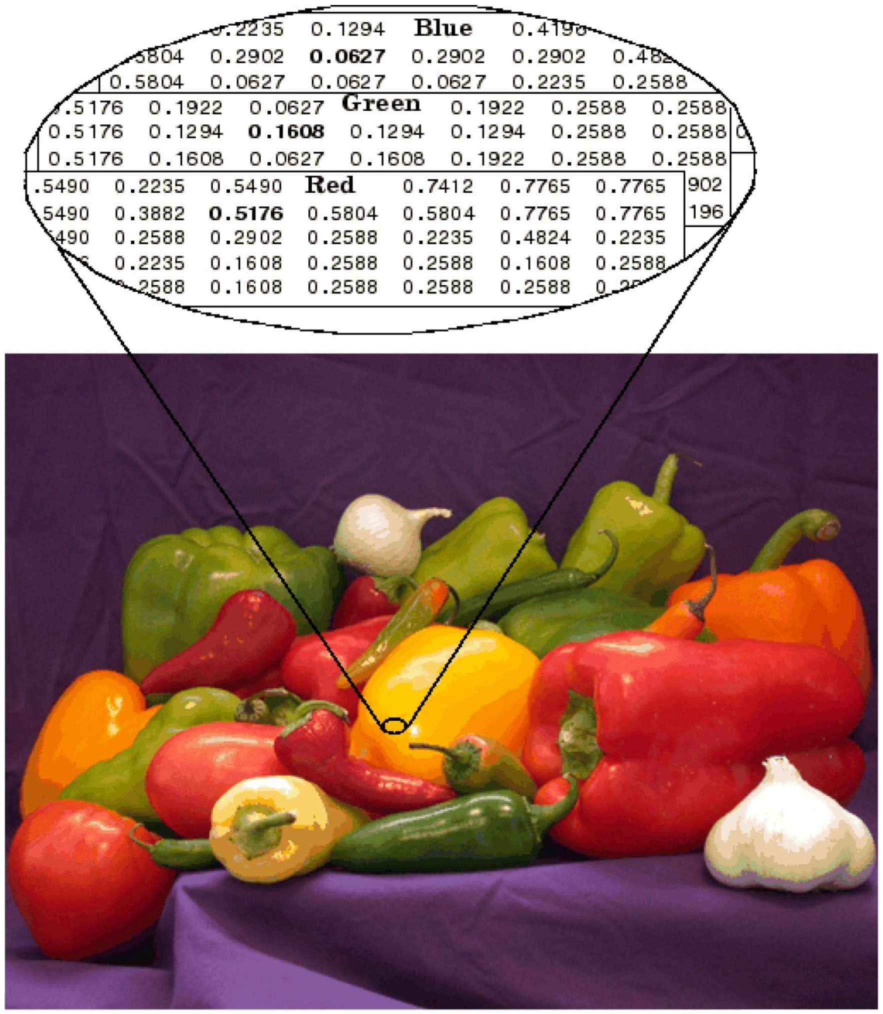
  `RGB[i,j,2]`

- The blue component is stored in

  `RGB[i,j,3]`

- The RGB triplet is stored in

  `RGB[i,j,:]`

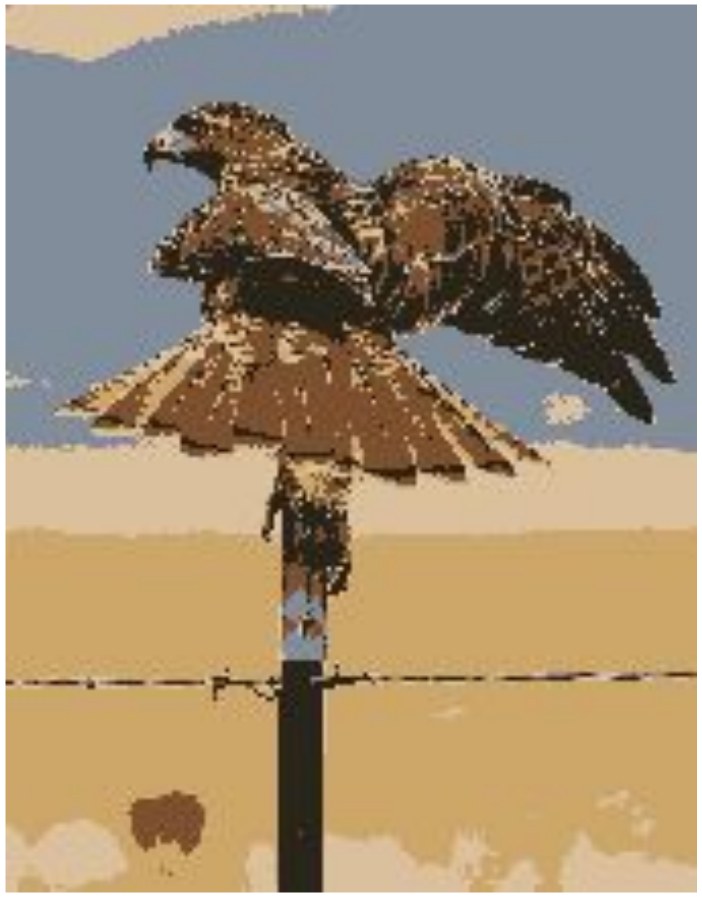


The Color Planes of a Truecolor Image

Indexed Images (Color Quantization)

- Values are indices into a colormap
- Colormap: a nColors x 3 double array containing values in the range [0,1]
- This can be helpful for characterizing the dominant colors and quantifying coverage
- More than one way to accomplish this, but a common method is "K-means Clustering":

5 Colors



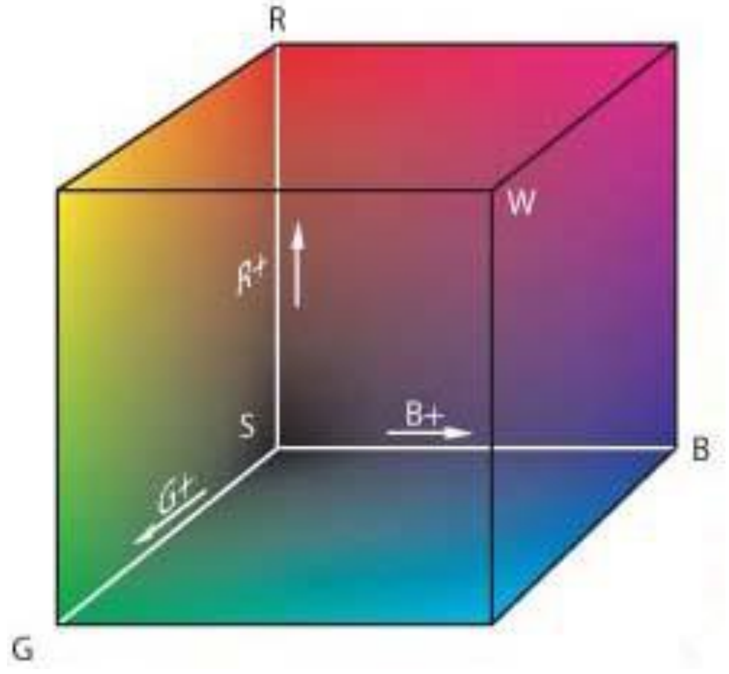
Pixels labeled like "Paint by numbers"

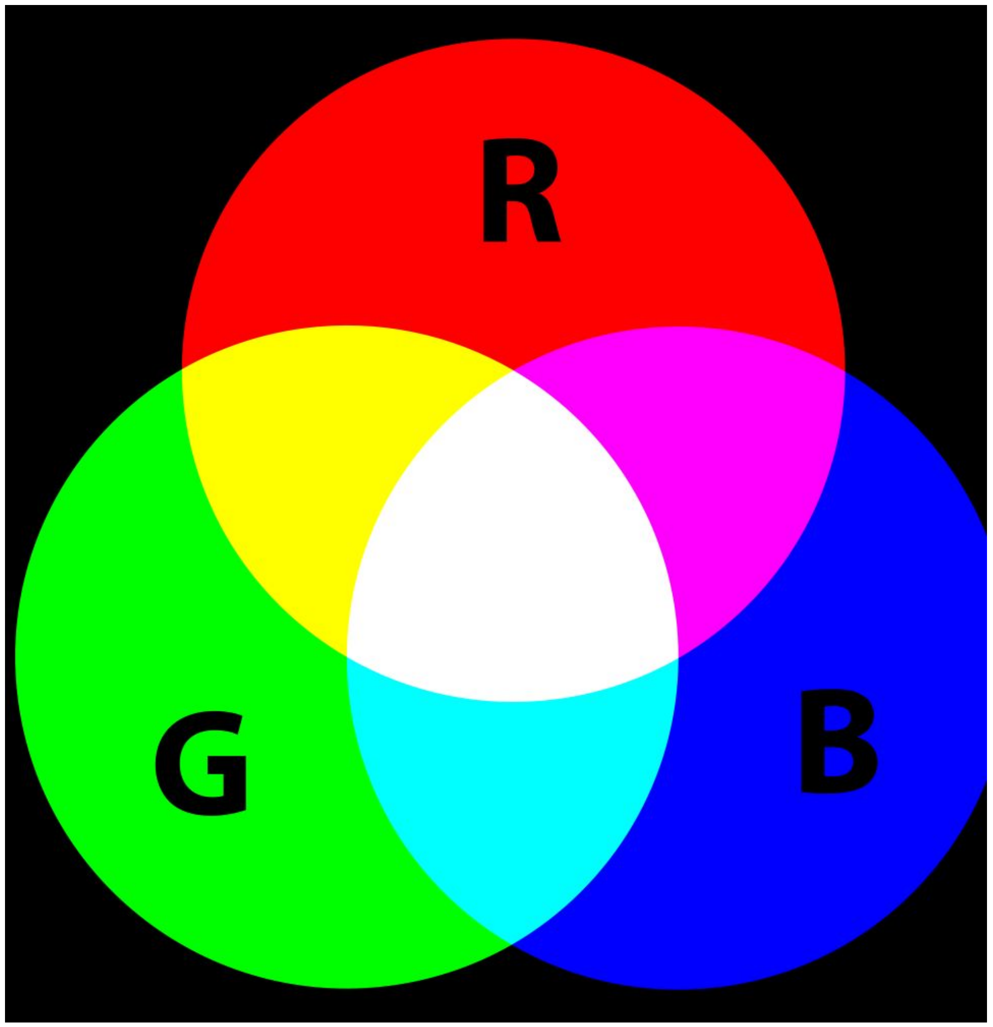
RGB Color Space

Red, Green, Blue

Additive: mixing of light

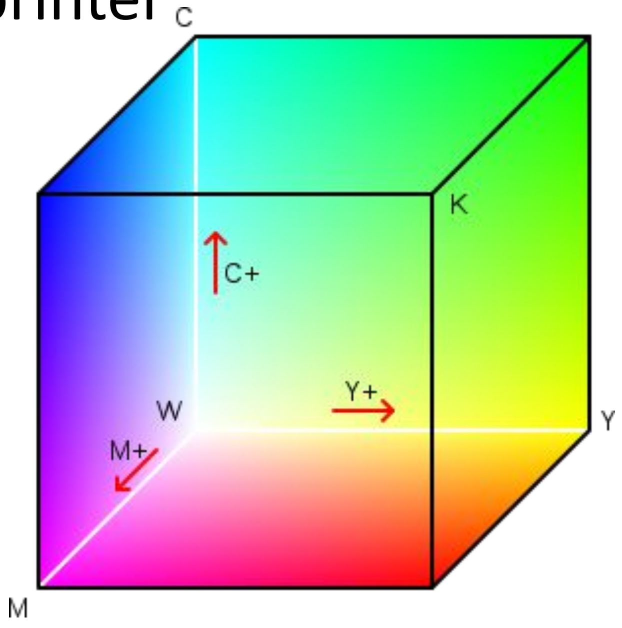
R=0, G=0, B=0: No light: black
R=1, G=1, B=1: Full light: white Example: your monitor

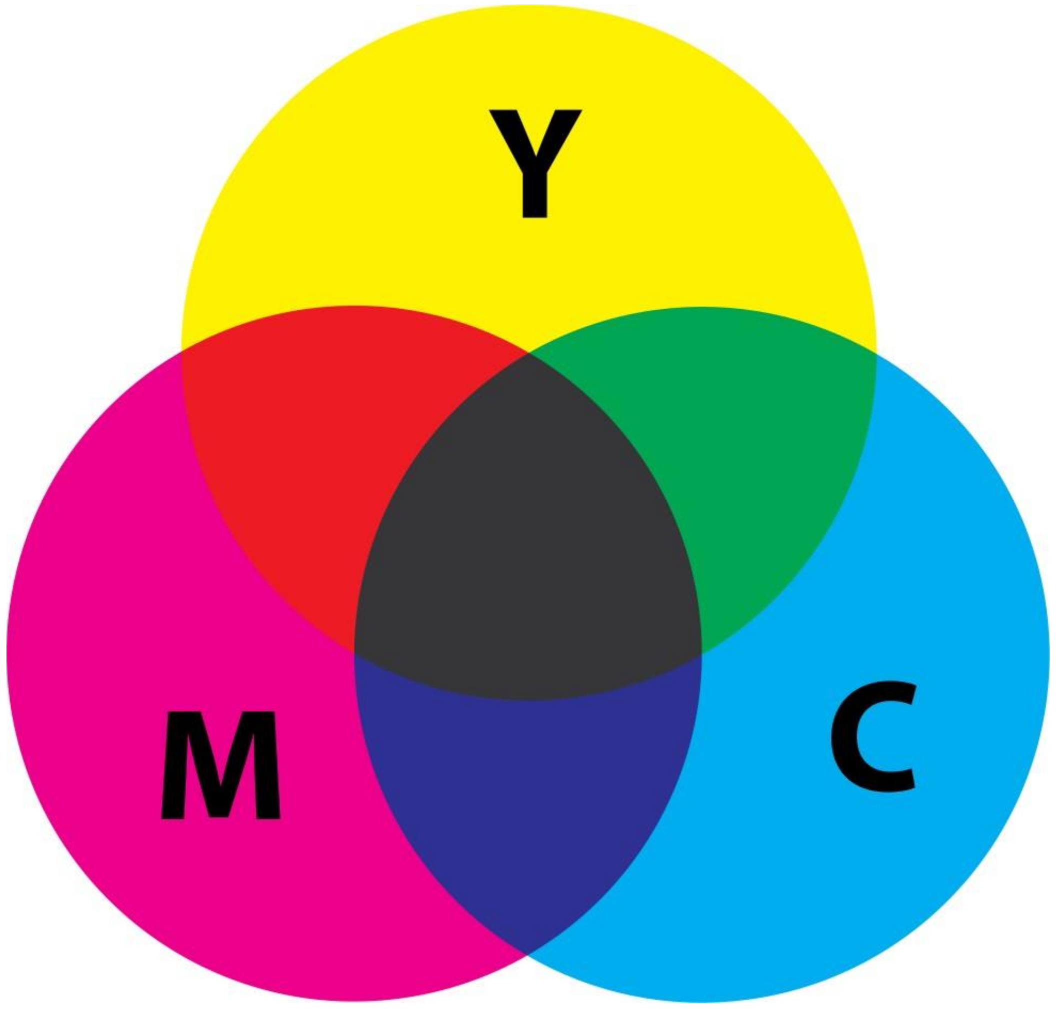




CMYK Color Space

Cyan, Magenta, Yellow
Subtractive: mixing of ink
C=0, M=0, Y=0: No ink: white C=1, M=1, Y=1: Full ink: black Example: your printer





HSV Color Space

Motivation:

Neither additive nor subtractive color models define color relationships the same way as the human eye, we distinguish colors in a more "perceptive" sense:

***Hue*:**

The "attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them"

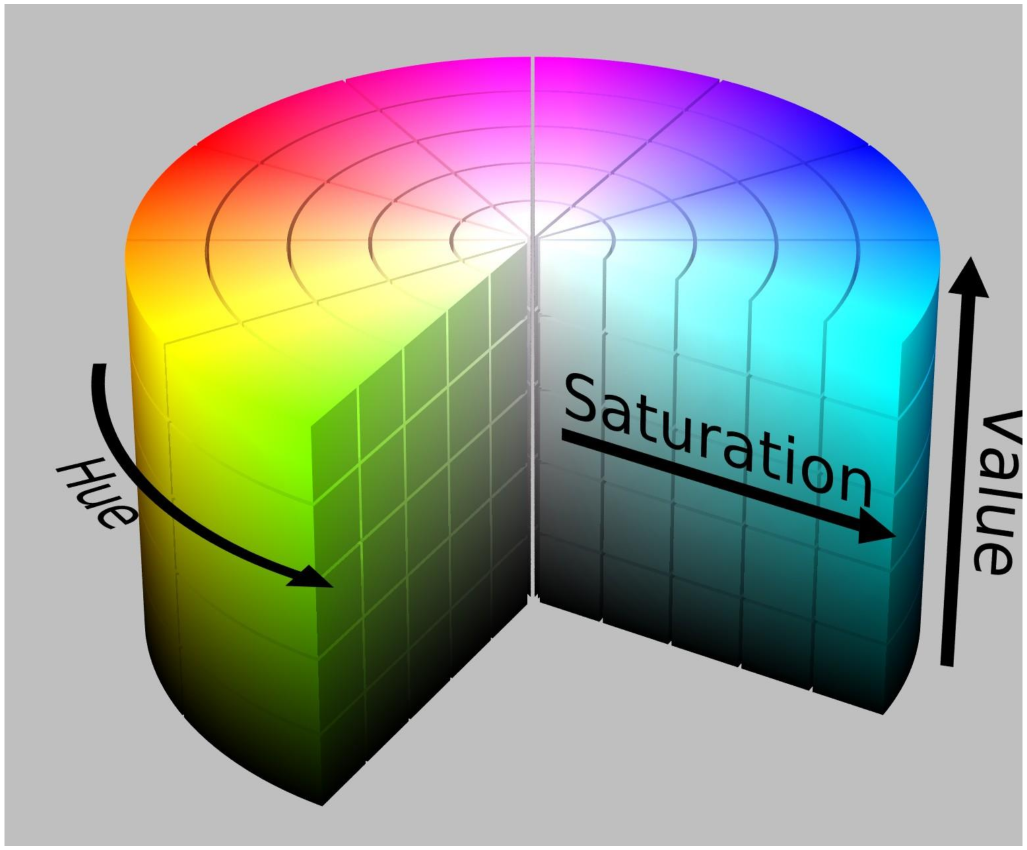
***Saturation*:**

The "colorfulness of a stimulus relative to its own brightness"

***Value*:**

The "brightness relative to the brightness of a similarly illuminated white"

```
from skimage.color import rgb2hsv
rgb2hsv(image)
```



Exporting Images with Scikit-image

Important note on exporting processed images!

- **Your operating system viewer expects images with a *uint8* array with values ranging from 0 to 255**

- **Scikit-image returns *Float64* array with values ranging from 0 to 1 by default (Matplotlib can display image in either format)**

```
img_as_float64(image)
```

- **Convert the data type when you export the array to an image file!**

```
io.imsave('my_image.png',img_as_ubyte(image))
```

Creating an **indexed image** from a truecolor image

Indexed image from RGB

- Here we use a function from another package (PIL)

```
from PIL import Image

#Load Image
imageRGB = Image.open('hawk.jpg')
imageRGB.load()
```

- We use Image.quantize() specifying K-means clustering, for N colors.

```
N=5
imageIndexed = imageRGB.quantize(colors=N kmeans=1)
```

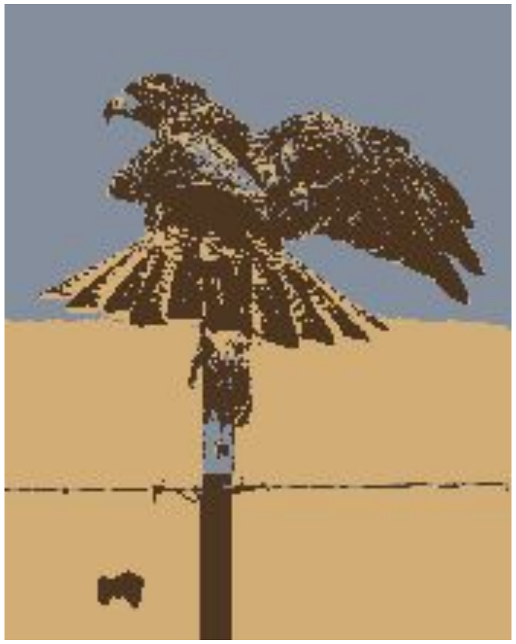
- The quantized image can be converted back into RGB to be displayed/exported.

```
# Comvert back to RGB tp display with imshow
rgbFromIndexed = imageIndexed.convert("RGB")

plt.axis('off') # turn off axis labels
plt.imshow(rgbFromIndexed)
```

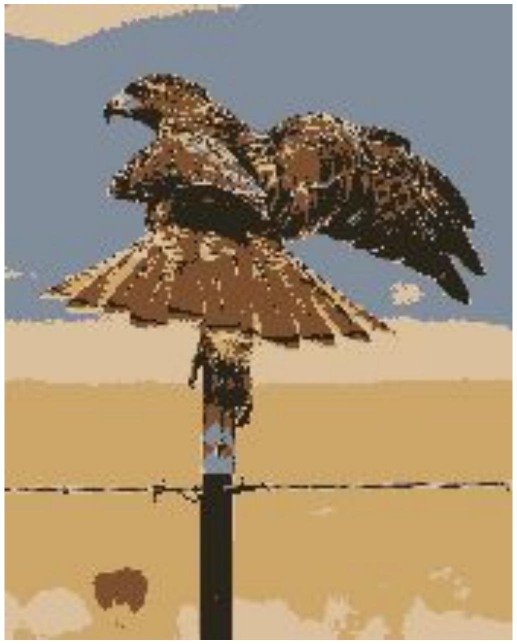
Indexed image from RGB

- The Kmeans method considers each [R,G,B] pixel as a 3D vector, and then attempts to find ”clusters” of similar vectors.
- The central [R,G,B] value of that cluster is then taken represent the entire cluster
- All pixels of a similar color (RGB vector distance) are then set to the central value.

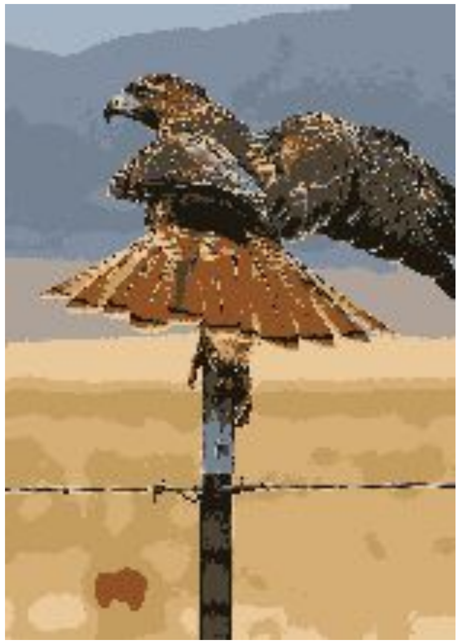
N=3



N=5



N=10



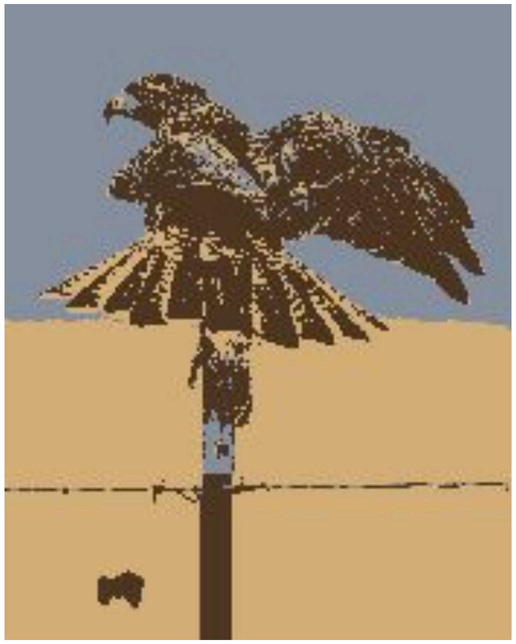
Indexed image from RGB

- The clustering also returns a label for every pixel to show what cluster it corresponds to (from 0 to 4 for N=5)
- Each index corresponds to a color in a “palette” (a list of N RGB colors)

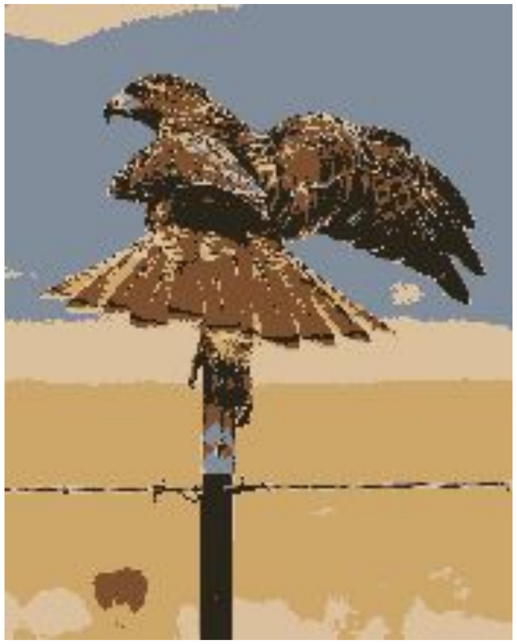
```
rgbFromIndexed.getcolors()
```

- In order to build the final image, one must build the ixjx3 array, by substituting in the [R,G,B] coordinate to each pixel in the array.

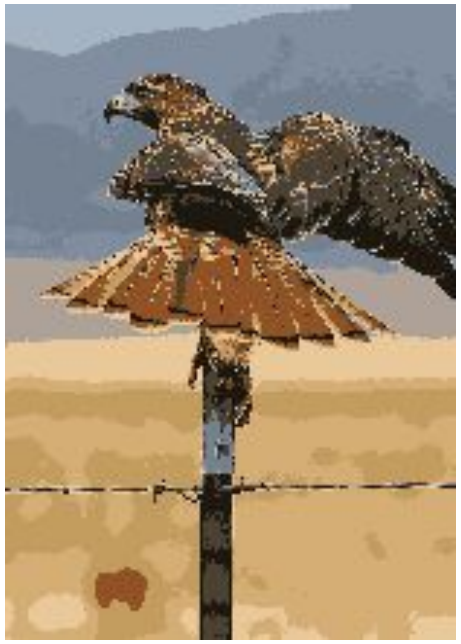
N=3



N=5



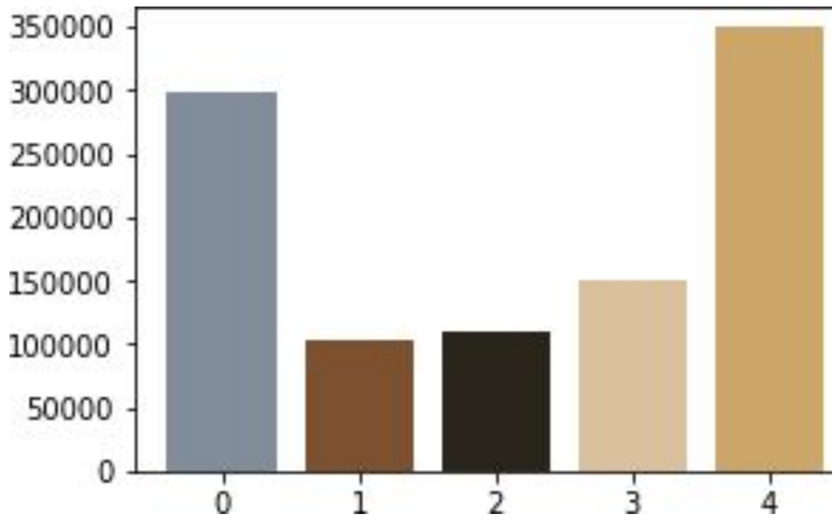
N=10



Indexed image from RGB

- By adding up the number of pixels of each color you can generate a histogram. Shown here with the corresponding fit color
- This can be used to compare the color distribution between images, or to identify
  - Agricultural land in a satellite photo
  - Mineral grains in a microscopic cross-section
- For fun, we can substitute colors by replacing the colormap with one corresponding to randomly generated RGB values.

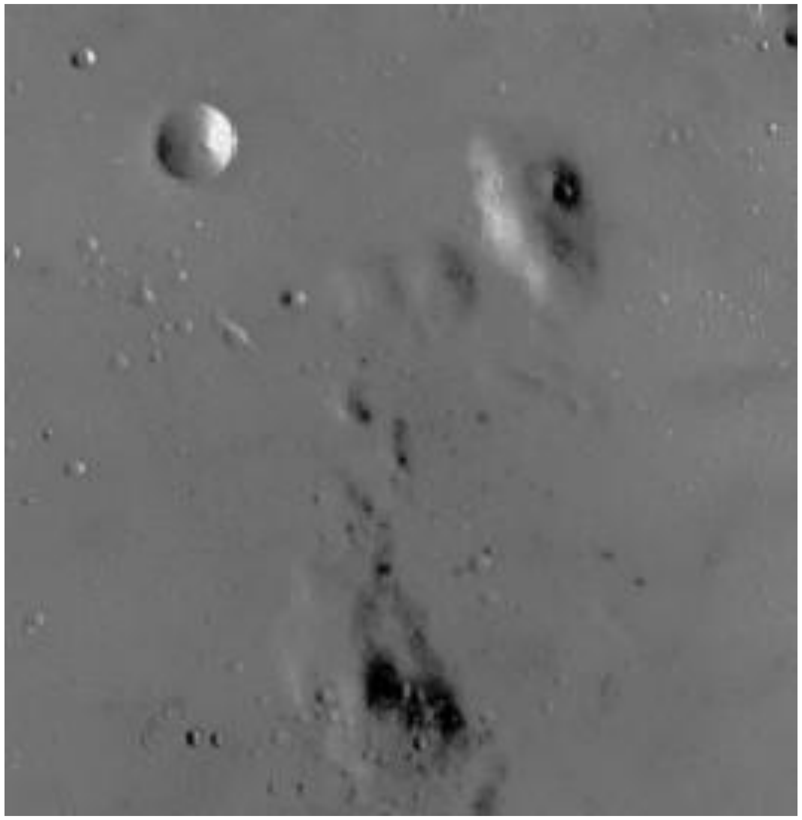
```
from random import randint
[randint(0,255), randint(0,255),randint(0,255)]
```



Increasing the **contrast** of an image of the Moon’s surface

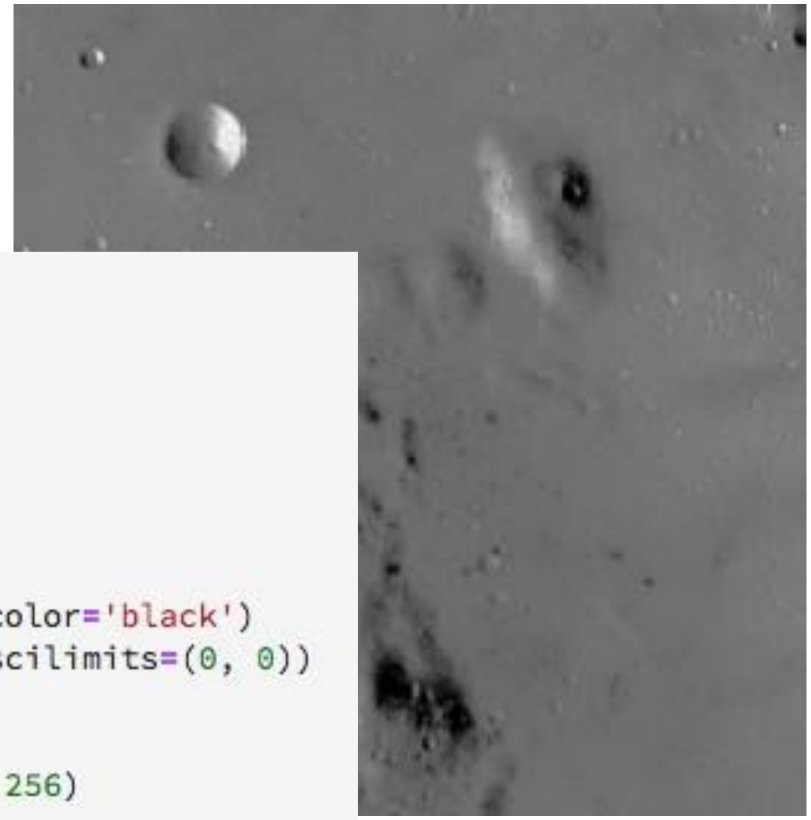
Intensity distribution (Contrast)

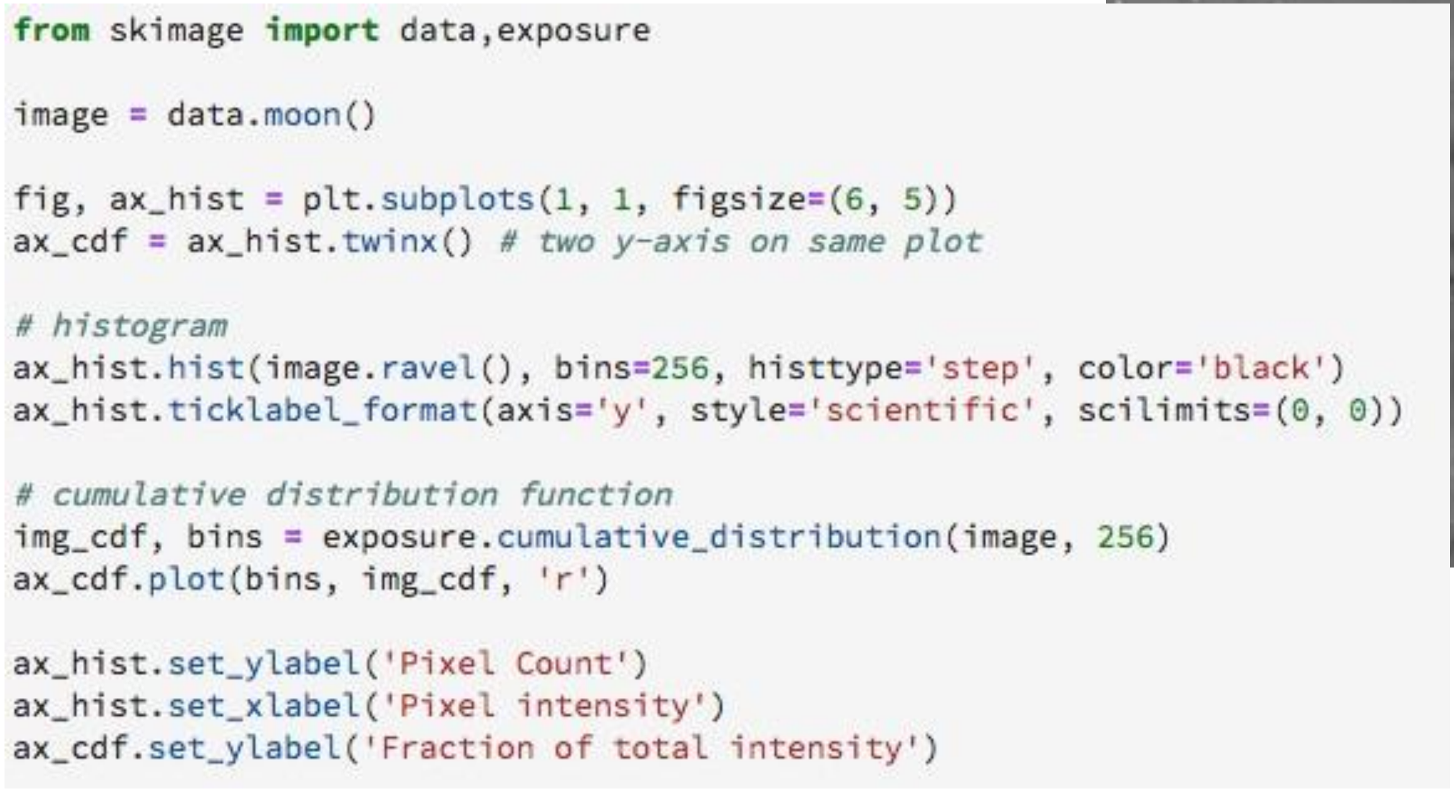
- Low contrast image of the moon (from scikit-image's example data)
- Plot histogram and cumulative distribution function of intensity



Intensity distribution (Contrast)

- Low contrast image of the moon (from scikit-image's example data)
- Plot histogram and cumulative distribution function of intensity



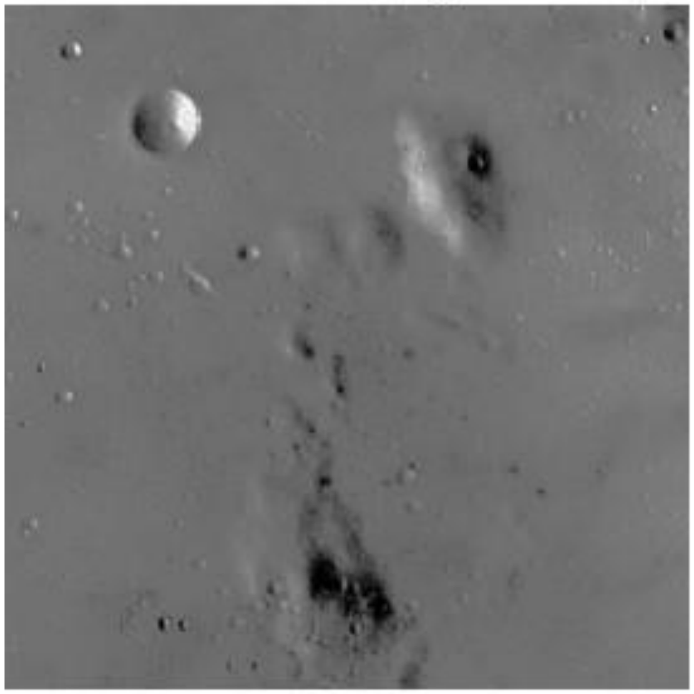


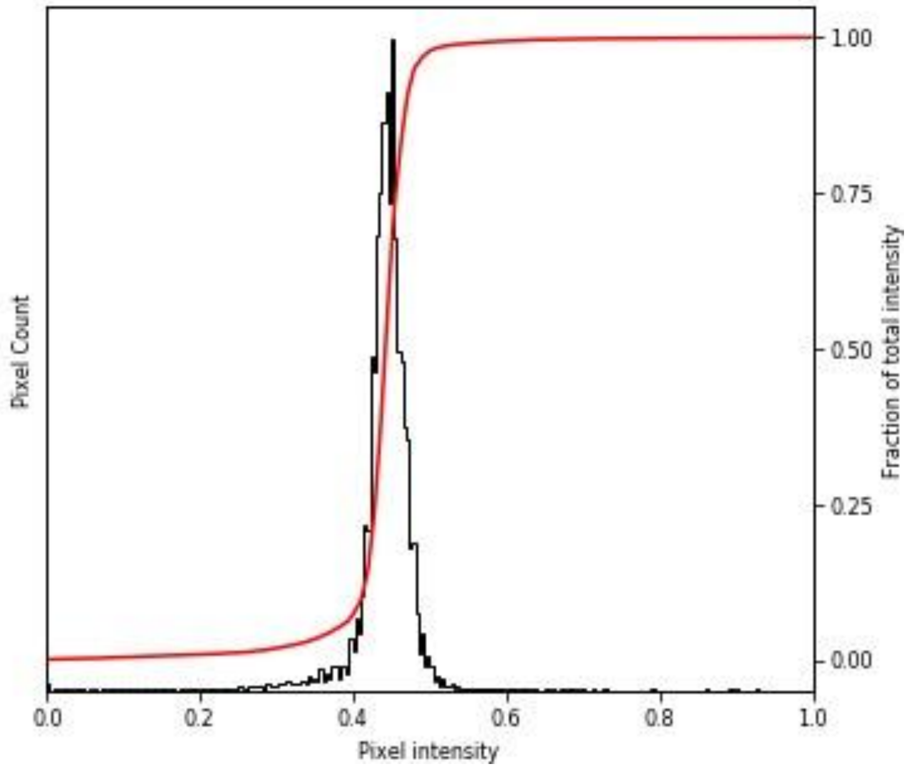
```
from skimage import data,exposure

image = data.moon()

fig, ax_hist = plt.subplots(1, 1, figsize=(6, 5))
ax_cdf = ax_hist.twinx() # two y-axis on same plot

# histogram
ax_hist.hist(image.ravel(), bins=256, histtype='step', color='black')
ax_hist.ticklabel_format(axis='y', style='scientific', scilimits=(0, 0))

# cumulative distribution function
img_cdf, bins = exposure.cumulative_distribution(image, 256)
ax_cdf.plot(bins, img_cdf, 'r')

ax_hist.set_ylabel('Pixel Count')
ax_hist.set_xlabel('Pixel intensity')
ax_cdf.set_ylabel('Fraction of total intensity')
```

Intensity distribution (Contrast)

- Sharp peak in intensities, very few bright and dark pixels!
- Topography and small craters very difficult to make out!

Low contrast image

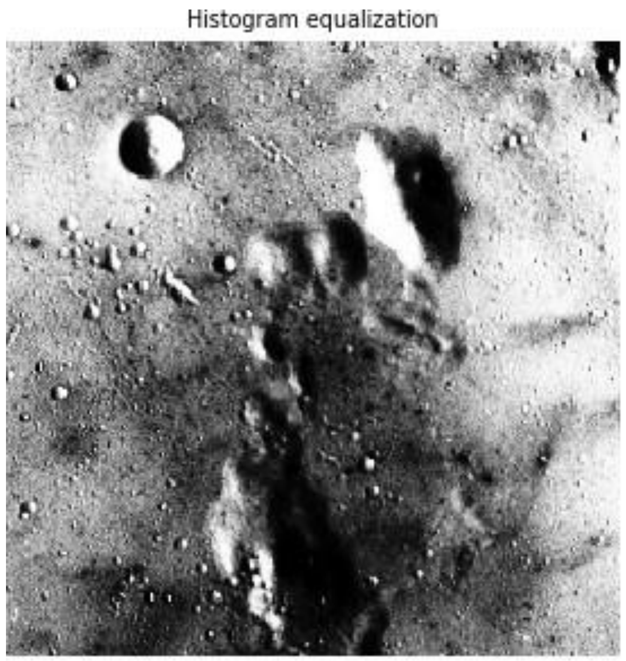


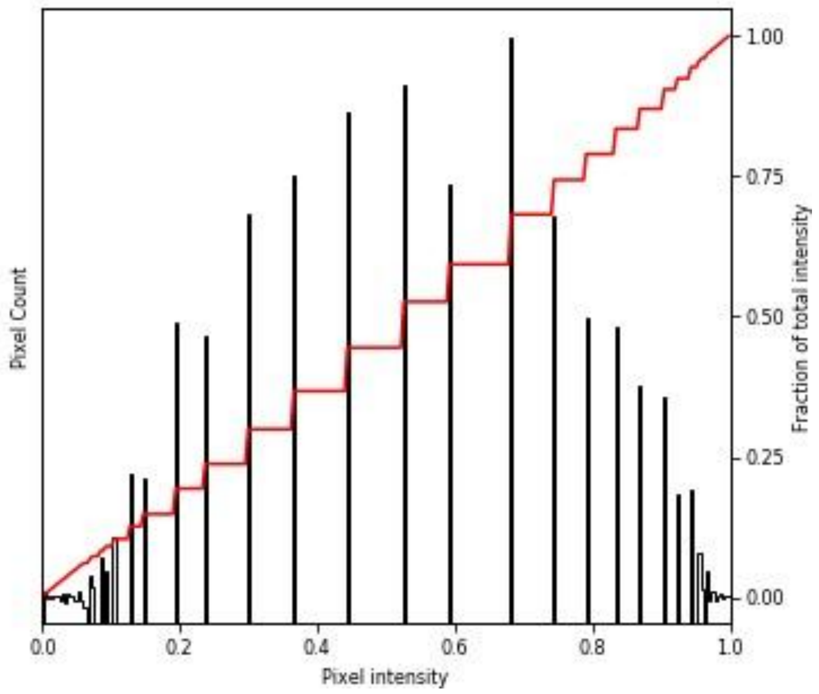


Histogram Equalization (Contrast)

- Spreads out the most frequent intensity values of the of an image.
  - Cumulative distribution function now nearly linear.

```
img_eq = exposure.equalize_hist(img)
```



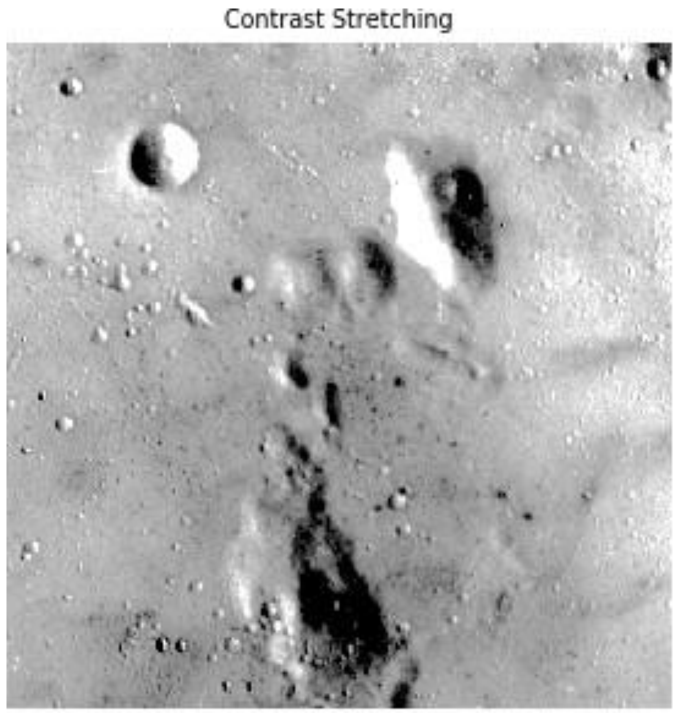


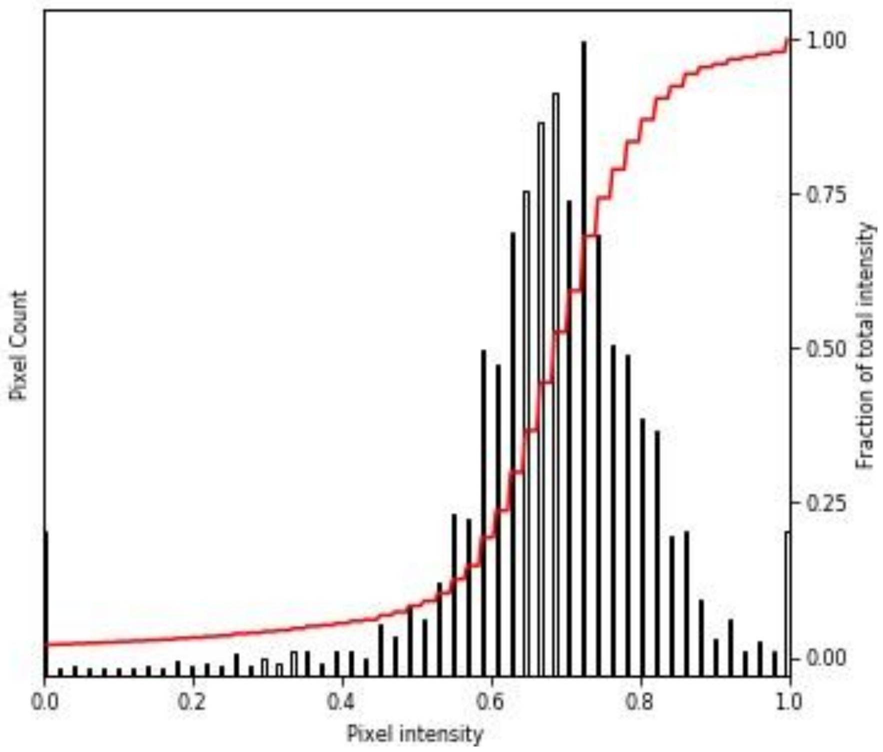
Notice the small craters and that parts of the surface are darker!

Contrast stretching

- Alternative technique that "stretches" the intensities so that they fall within the 2nd and 98th percentiles.

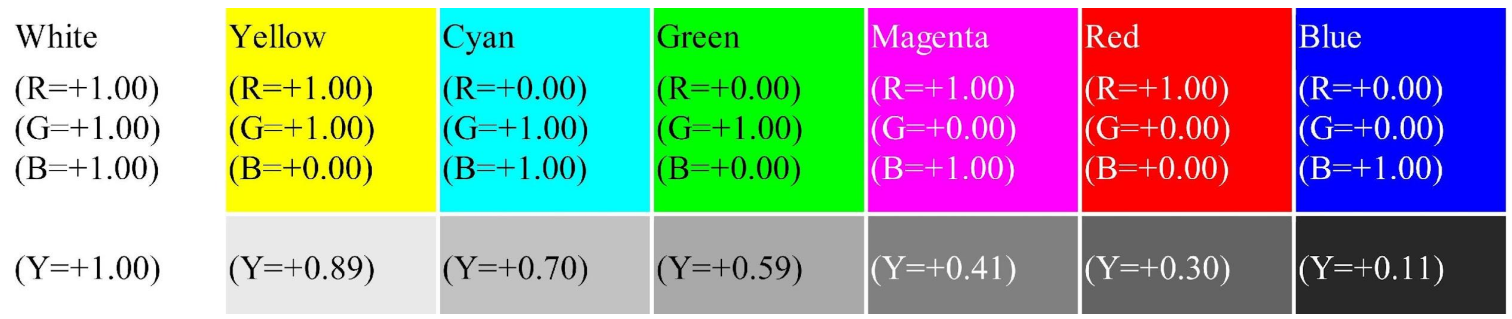
```
p2, p98 = np.percentile(img, (2, 98))
img_rescale = exposure.rescale_intensity(img, in_range=(p2, p98))
```





Color Space Conversion

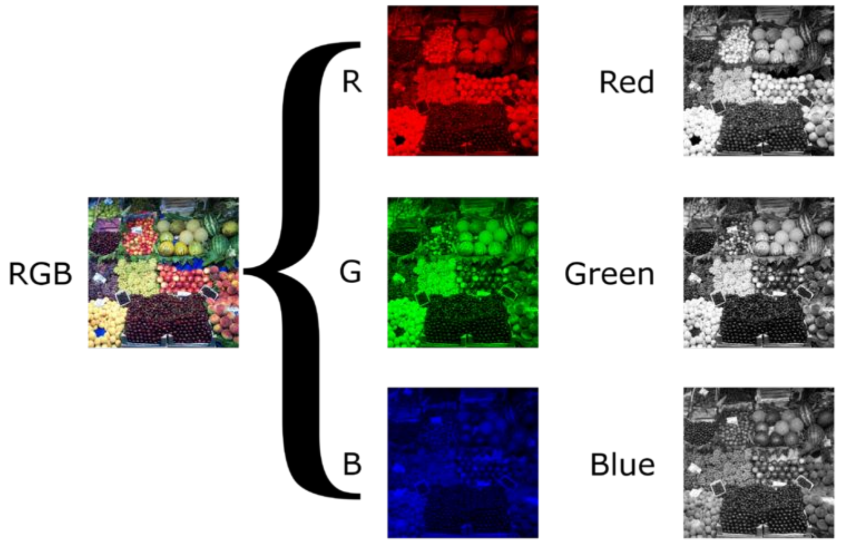
- **RGB to Grayscale:**
  - **Weighted according to the eye's sensitivity to colors (NTSC)**
  - `G = 0.2989 * R + 0.5870 * G + 0.1140 * B`



White	Yellow	Cyan	Green	Magenta	Red	Blue
(R=+1.00) (G=+1.00) (B=+1.00)	(R=+1.00) (G=+1.00) (B=+0.00)	(R=+0.00) (G=+1.00) (B=+1.00)	(R=+0.00) (G=+1.00) (B=+0.00)	(R=+1.00) (G=+0.00) (B=+1.00)	(R=+1.00) (G=+0.00) (B=+0.00)	(R=+0.00) (G=+0.00) (B=+1.00)
(Y=+1.00)	(Y=+0.89)	(Y=+0.70)	(Y=+0.59)	(Y=+0.41)	(Y=+0.30)	(Y=+0.11)

- **A color image can be thought of as a mixture of grayscale intensities in three colors**

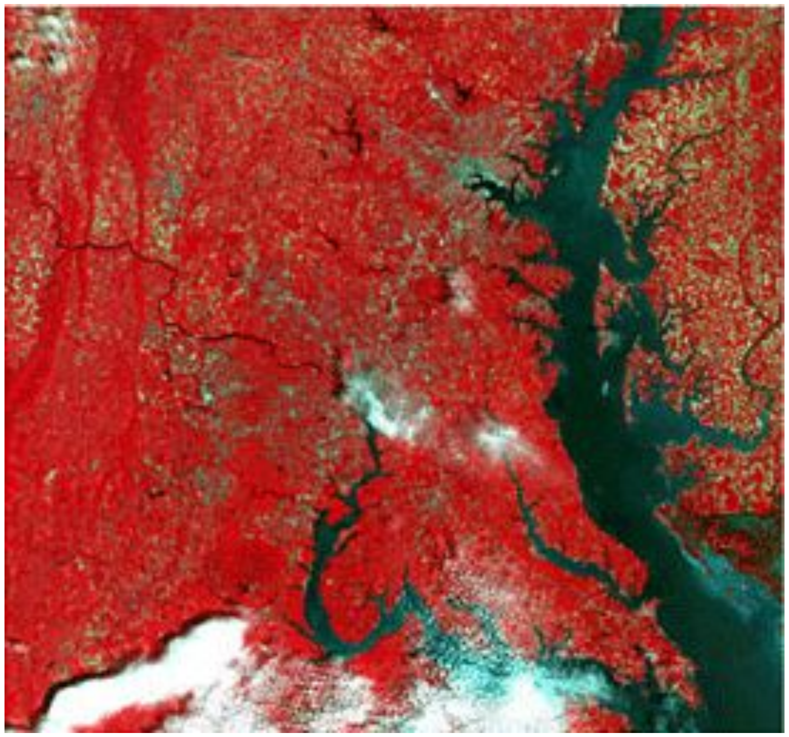
- **Mixing in different proportions or "stretching" the intensities, to create a "false color" image to highlight particular features.**

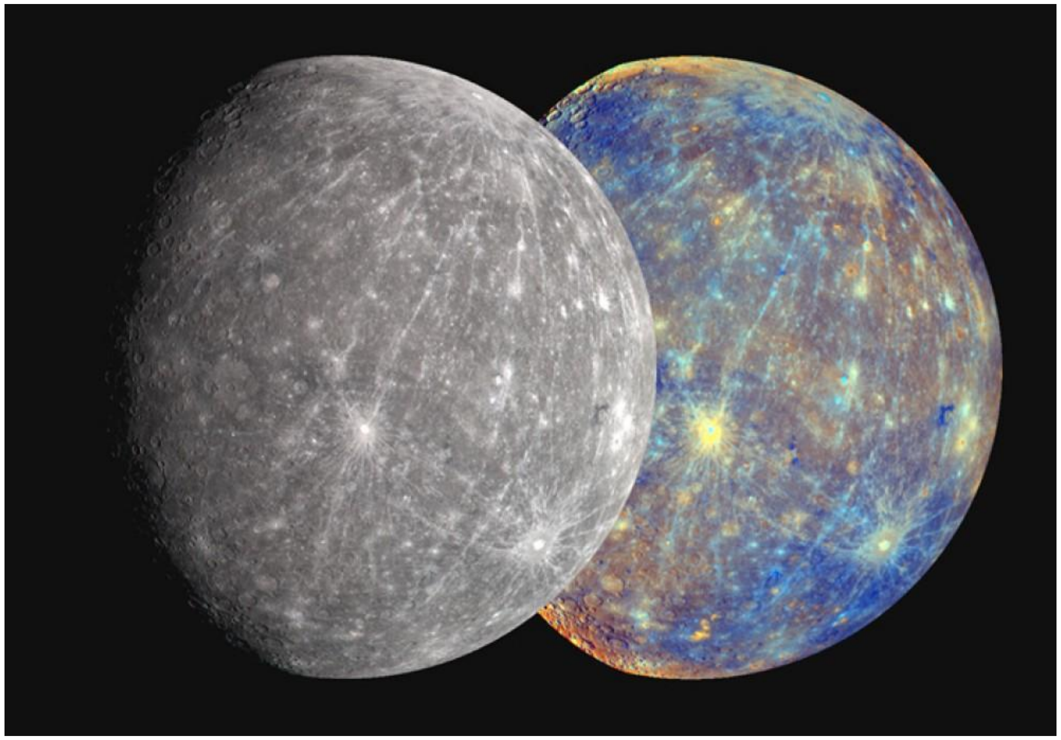


Figures: Wikipe

False Color images

- False color images are RGB images, where one or more of the channels is representing something other than the intensity at the usual wavelength, (part of the EM spectrum outside the visible range).
- Even within the visible range the RGB bands can be manipulated to enhance the the ability to see subtle differences





Figures: Wikipedia. NASA – MESSENGER orbit

Satellite Imagery from LANDSAT

- Continuous satellite imagery program co-managed by the USGS and NASA
  - first satellite launched in 1972, Data publically available
  - https://landsat.usgs.gov/landsat-data-access
  - Latest satellite (LANDASAT 8) has 11 bands
  - Each band is a grayscale image corresponding to the intensity within a given wavelength range

- Use any 3 bands as the R,G,B channel to create a false color
- Certain bands emphasize certain features
- Bands can be "stretched" to tune the image

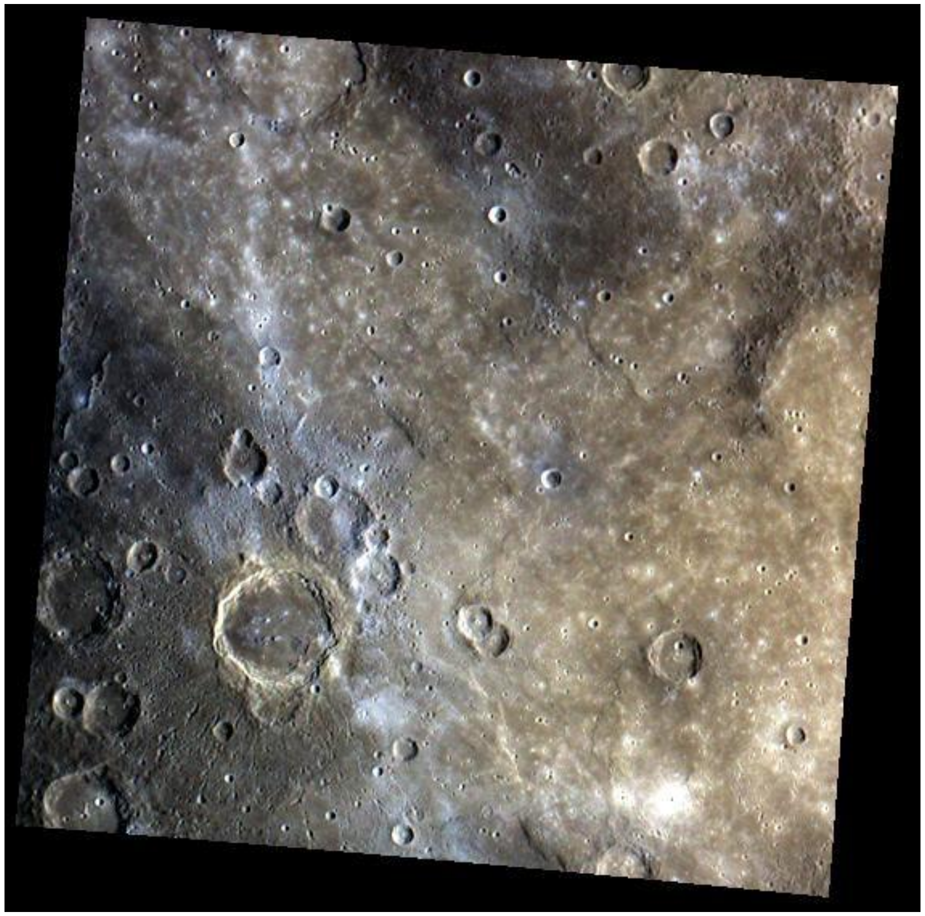
- Visualize: http://landsatexplorer.esri.com

	Bands	Wavelength (micrometers)	Resolution (meters)
Landsat 8 Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) **Launched February 11, 2013**	Band 1 - Coastal aerosol	0.43 - 0.45	30
	Band 2 - Blue	0.45 - 0.51	30
	Band 3 - Green	0.53 - 0.59	30
	Band 4 - Red	0.64 - 0.67	30
	Band 5 - Near Infrared (NIR)	0.85 - 0.88	30
	Band 6 - SWIR 1	1.57 - 1.65	30
	Band 7 - SWIR 2	2.11 - 2.29	30
	Band 8 - Panchromatic	0.50 - 0.68	15
	Band 9 - Cirrus	1.36 - 1.38	30
	Band 10 - Thermal Infrared (TIRS) 1	10.60 - 11.19	100
	Band 11 - Thermal Infrared (TIRS) 2	11.50 - 12.51	100

Creating an “enhanced color” image of Mercury’s surface

Enhanced (stretched) Color

- Color image of Mercury’s surface (MESSENGER), with subtle color differences
- How can we modify the image to emphasize these?

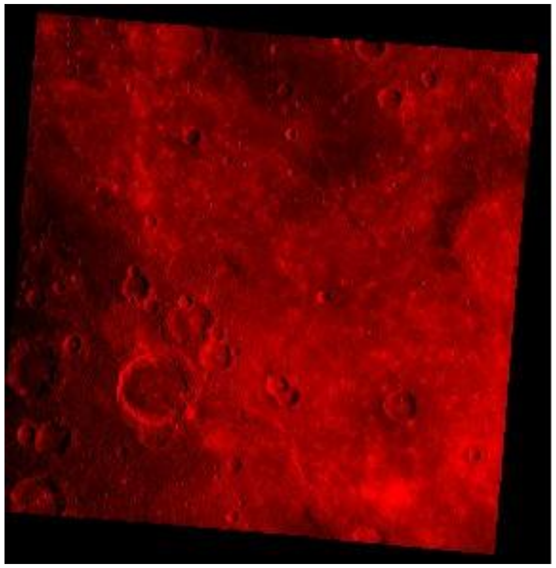


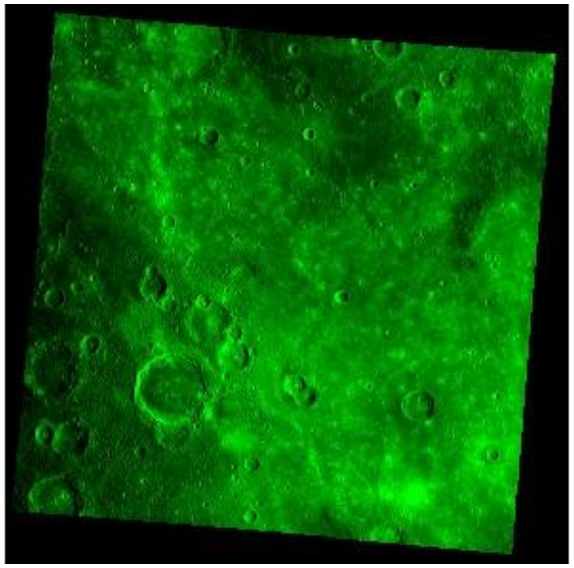
Figures: NASA, MESSNGER orbite

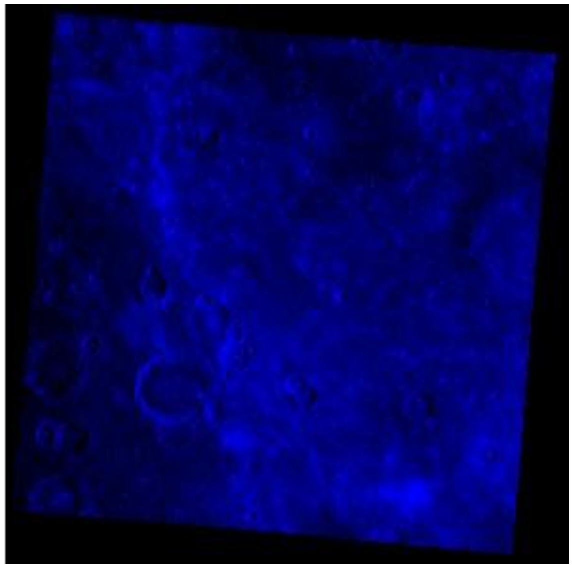
Enhanced (stretched) Color

- Begin by isolating the three bands

```
I_red = img[:,:,0].copy() # Intensity of each band
I_green = img[:,:,1].copy()
I_blue = img[:,:,2].copy()
```



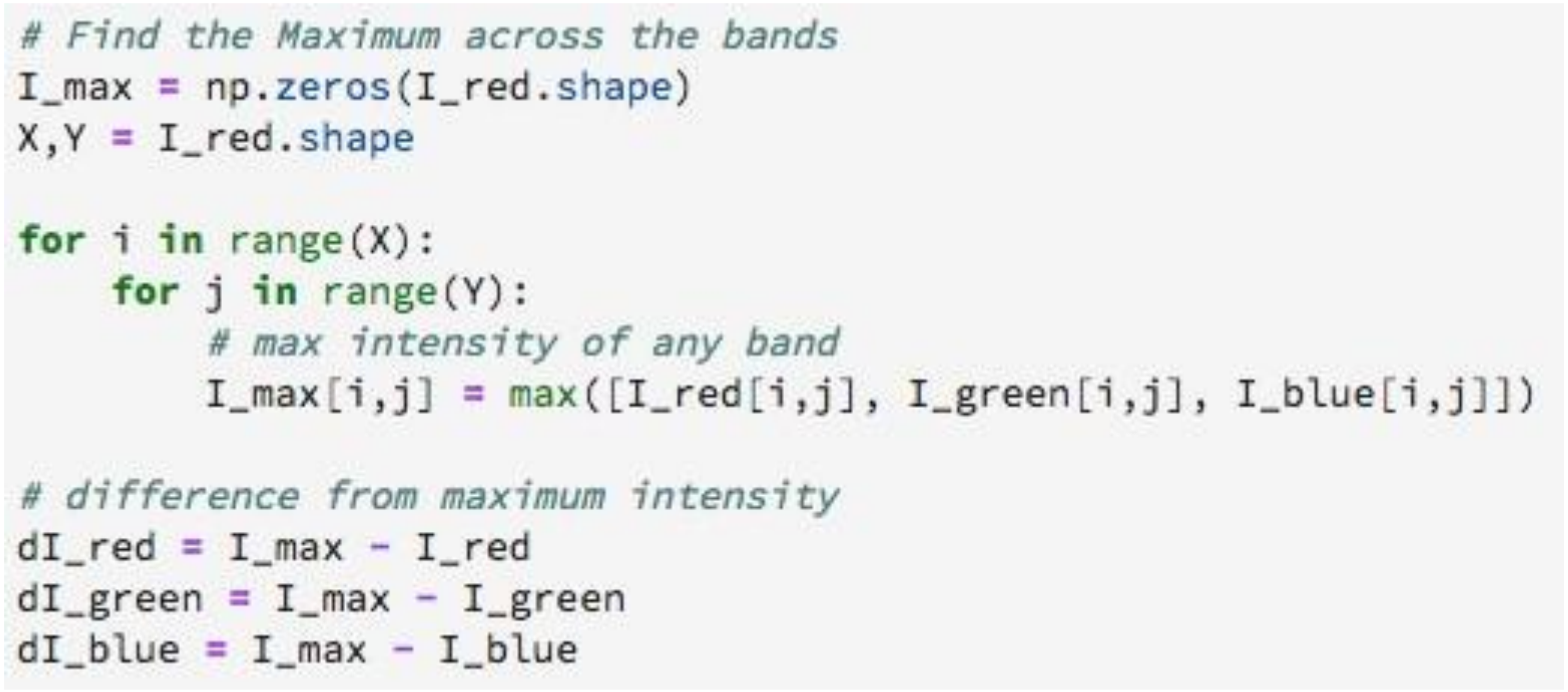




Enhanced (stretched) Color

Simple model for enhancing color:

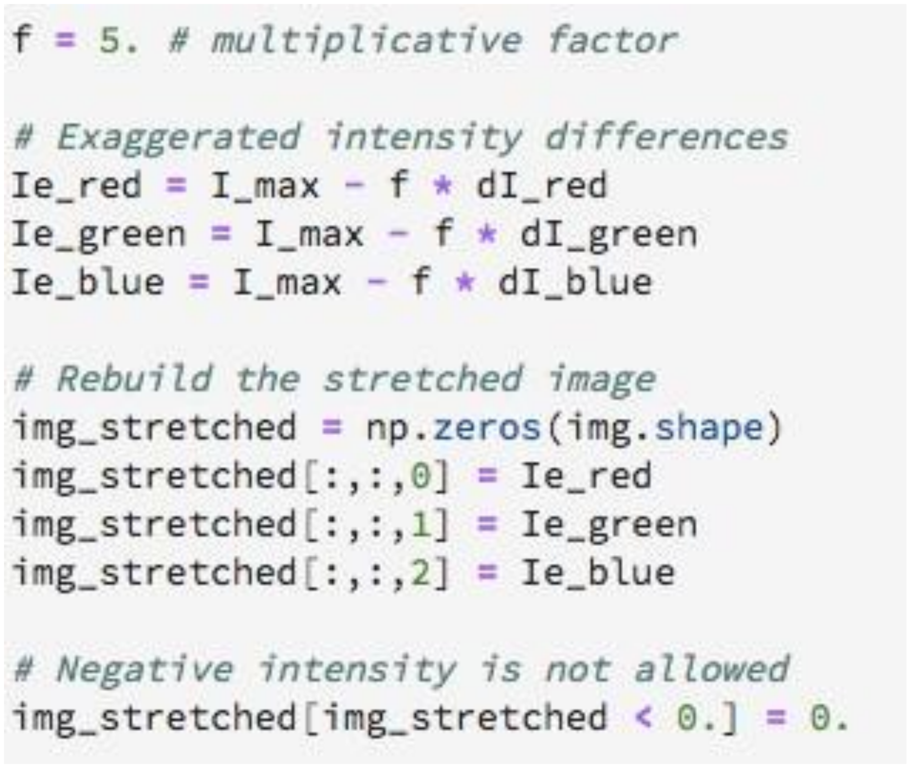
- For each pixel find the band with the maximum intensity and the difference in intensity for the other bands



```python
# Find the Maximum across the bands
I_max = np.zeros(I_red.shape)
X,Y = I_red.shape

for i in range(X):
    for j in range(Y):
        # max intensity of any band
        I_max[i,j] = max([I_red[i,j], I_green[i,j], I_blue[i,j]])

# difference from maximum intensity
dI_red = I_max - I_red
dI_green = I_max - I_green
dI_blue = I_max - I_blue
```

Enhanced (stretched) Color

- Scale the difference in intensities by some multiplicative factor
- Rebuild the image with the modified RBG channels

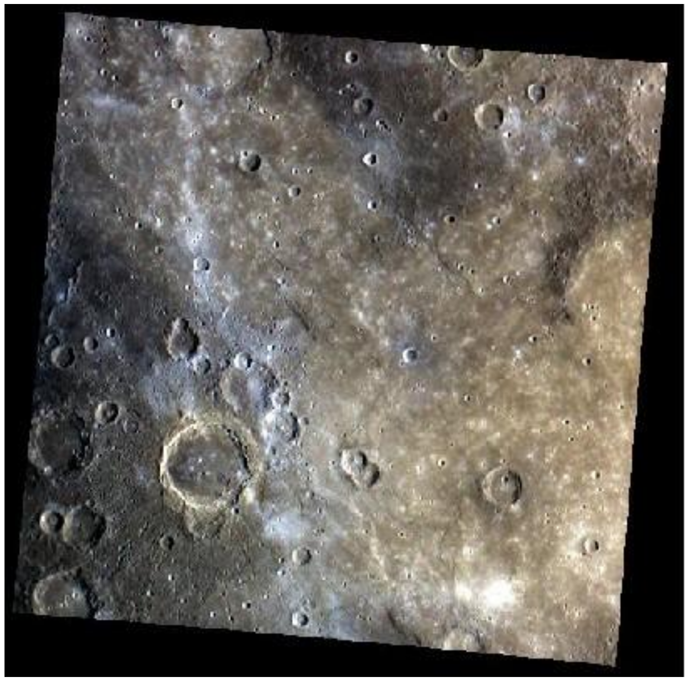
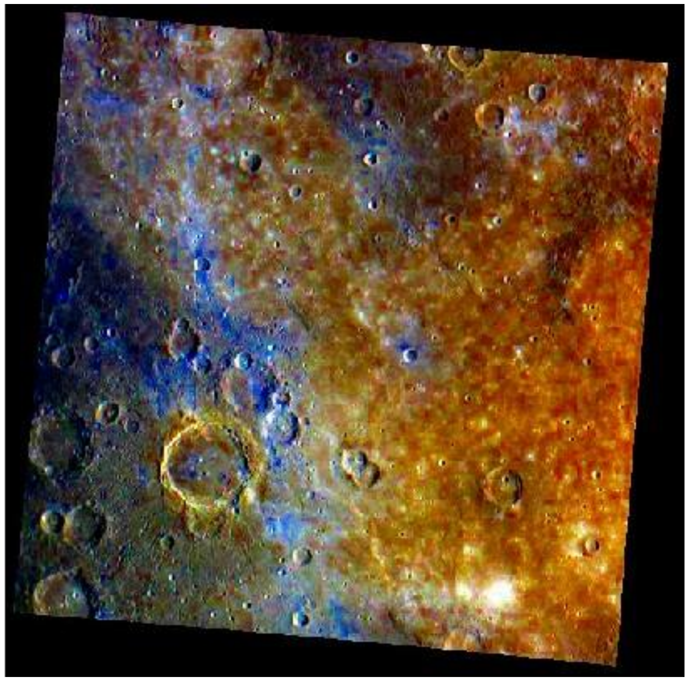


```
f = 5. # multiplicative factor

# Exaggerated intensity differences
Ie_red = I_max - f * dI_red
Ie_green = I_max - f * dI_green
Ie_blue = I_max - f * dI_blue

# Rebuild the stretched image
img_stretched = np.zeros(img.shape)
img_stretched[:,:,0] = Ie_red
img_stretched[:,:,1] = Ie_green
img_stretched[:,:,2] = Ie_blue

# Negative intensity is not allowed
img_stretched[img_stretched < 0.] = 0.
```

Remember: the RGB color space is additive, so subtracting intensity in red and green makes a pixel appear more blue.

Enhanced (stretched) Color

- We can now see that large regions of the surface a relatively redder or bluer than the average color

- May be indicative of differences in mineralogy or age (space weathering).