# Image Processing

# Computer Vision with Python

- Section Goals
  - Learn various image processing operations.
  - Perform image operations such as Smoothing, Blurring, Morphological Operations.
  - Grab properties such as color spaces and histograms.

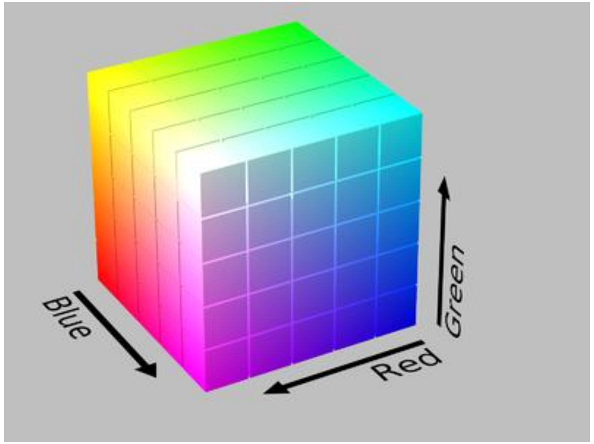
# Colorspaces

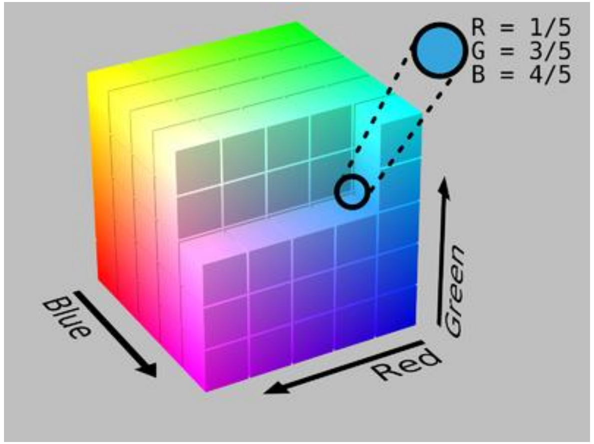
- So far we've only worked with RGB color spaces, in RGB coding, colors are modeled as a combination of Red, Green, and Blue.
- In the 1970s HSL (hue, saturation, lightness) and HSV (hue, saturation, value) were developed as alternative color models.

- HSL and HSV are more closely aligned with the way human vision actually perceives color.
- While in the course we will deal almost exclusively with RGB images, its a good idea to understand how to convert to HSL and HSV colorspaces.

# Computer Vision with Python

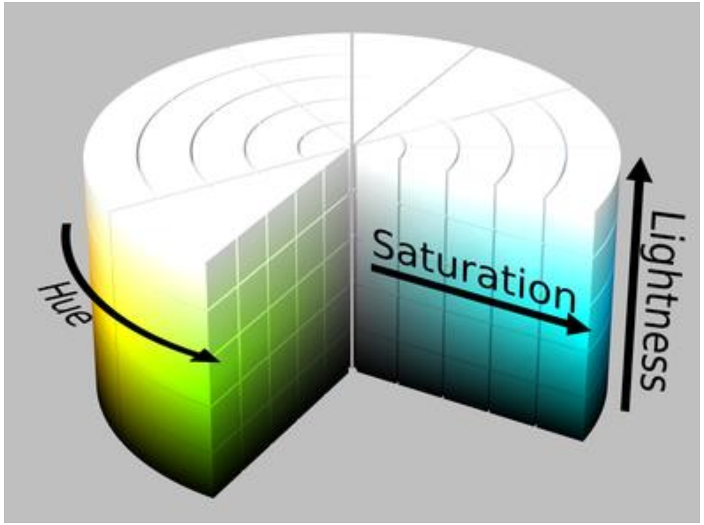
- RGB Model Representation of Colors



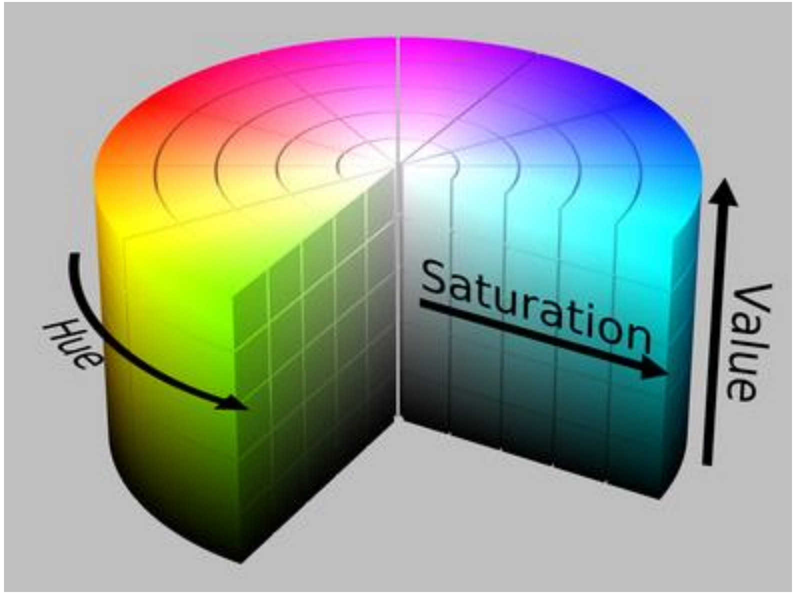
Computer Vision with Python

- HSL Cylinder Model



# Computer Vision with Python

- HSV Cylinder Model



* This lecture will be a quick review on using the **cvtColor** function to change colorspaces.
* We won't have to deal with HSL or HSV based color images in the rest of the course.

# Blending and Pasting Images

* Often we will be working with multiple images.
* OpenCV has many programmatic methods of blending images together and pasting images on top of each other.

- Blending images is done through the **addWeighted** function that uses both images and combines them.
- To blend images we use a simple formula:
    - new_pixel = $\boldsymbol{\alpha} \times$ pixel_1 $+ \boldsymbol{\beta} \times$ pixel_2 $+ \boldsymbol{\gamma}$

# Computer Vision with Python

- Let’s explore the syntax in this lecture!

# Blending and Pasting Images - Part Two

# Computer Vision with Python

- So far we’ve seen how to “overlay” images on top of each other by simply replacing values of the larger image with values of the smaller image for a desired ROI.
- But what if we only want to blend or replace part of the image?

# Computer Vision with Python

- Operations we've done so far:

# Computer Vision with Python

- But what if we want to mask part of the smaller image?


Img 1 Mask Img 2 with masked Img1

# Computer Vision with Python

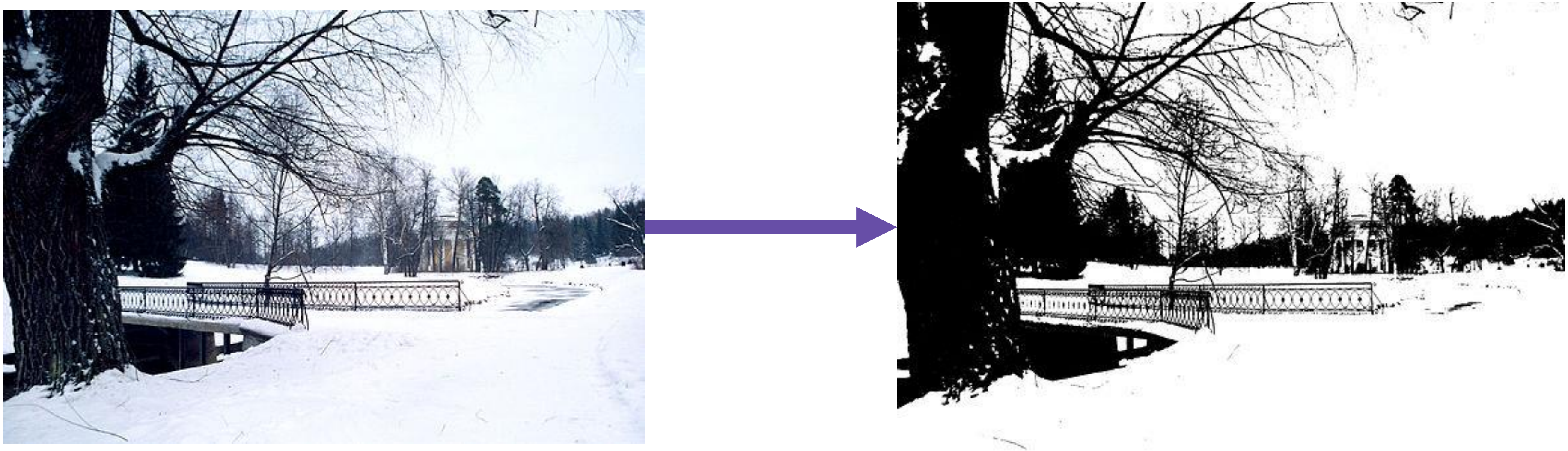
- Let's explore the syntax for these steps!
- Keep in mind, it is quite complicated!
- There are 3 really good supplemental links at the bottom of the lecture notebook for you to explore for other use cases.

# Image Thresholding

- In some CV Applications it is often necessary to convert color images to grayscale, since only edges and shapes end up being important.
- Similarly, some applications only require a binary image showing general shapes.

- Thresholding is fundamentally a very simple method of segmenting an image into different parts.
- Thresholding will convert an image to consist of only two values, white or black.

# Converting a color image to binary.



# Computer Vision with Python

- Let’s explore the syntax and options for thresholding with OpenCV!

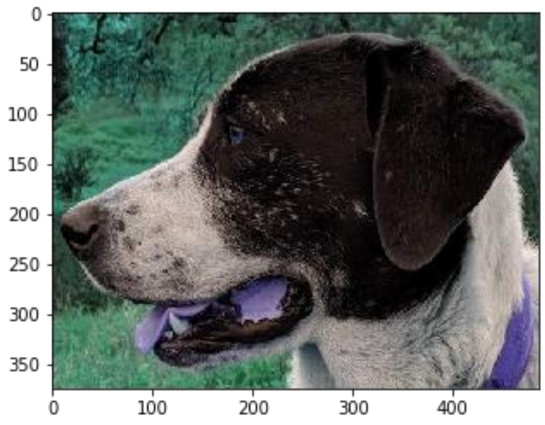
# Blurring and Smoothing

- A common operation for image processing is blurring or smoothing an image.
- Smoothing an image can help get rid of noise, or help an application focus on general details.
- There are many methods of blurring and smoothing.

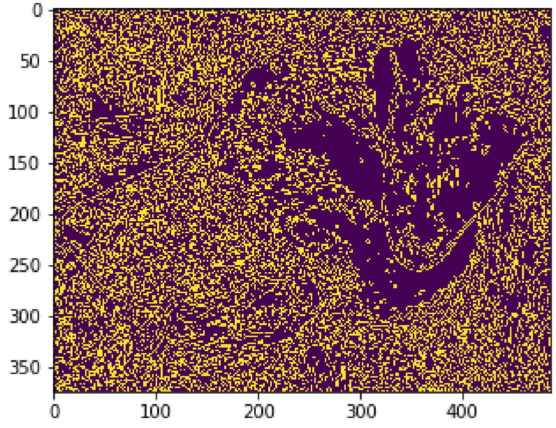
- Often blurring or smoothing is combined with edge detection.
- Edge detection algorithms detect too many edges when shown a high resolution image without any blurring.

# Computer Vision with Python

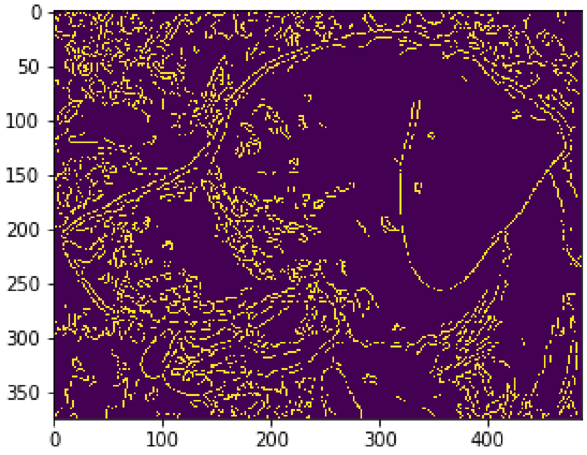
- Often blurring or smoothing is combined with edge detection.




**Original**




**Detected Edges (no blur)**




**Detected Edges (with blur)**

Computer Vision with Python

- **Methods we'll be exploring**
  - Gamma Correction
    - Gamma correction can be applied to an image to make it appear brighter or darker depending on the Gamma value chosen.


PIERIAN DATA

- **Methods we'll be exploring**
  - Kernel Based Filters
  - Kernels can be applied over an image to produce a variety of effects.
  - The best way to explain this is through an interactive visualization.
  - Go to: **http://setosa.io/ev/image-kernels/**

# Blurring and Smoothing

Part Two

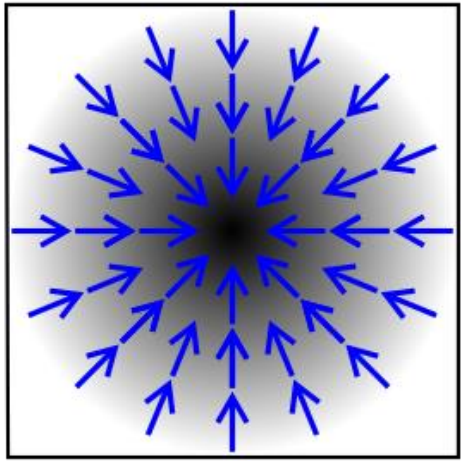
# Morphological Operators

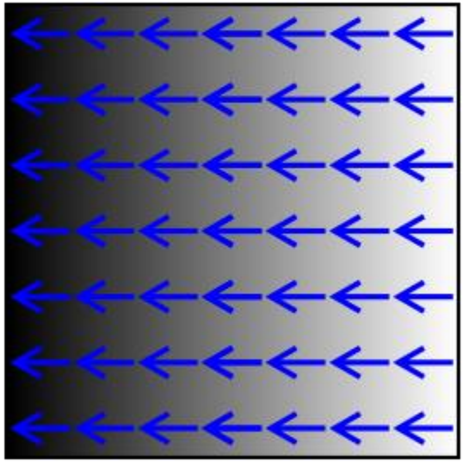
- Morphological Operators are sets of Kernels that can achieve a variety of effects, such as reducing noise.
- Certain operators are very good at reducing black points on a white background (and vice versa)

- Certain operators can also achieve an erosion and dilation effect that can add or erode from an existing image.
- This effect is most easily seen on text data, so we will practice various morphological operators on some simple white text on a black background.
- Let’s get started!

# Gradients

- An image gradient is a directional change in the intensity or color in an image.



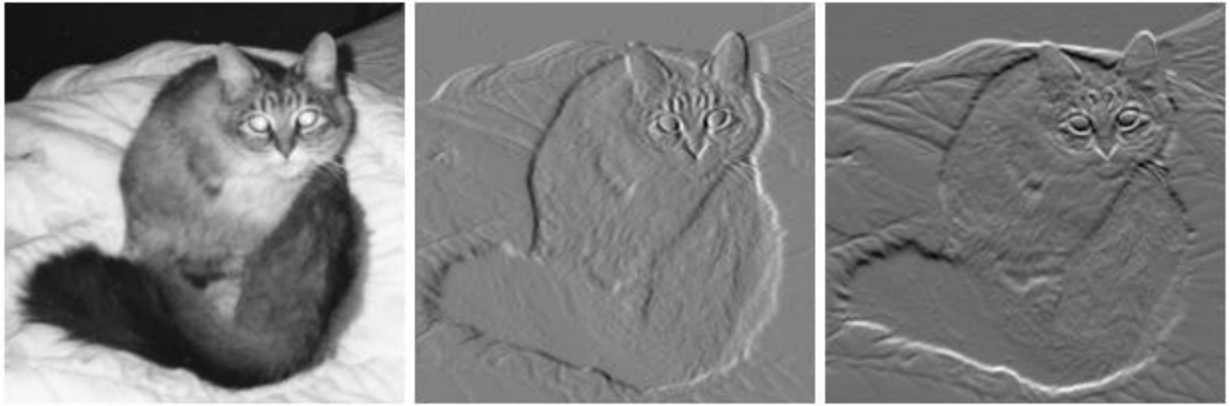


# Computer Vision with Python

- In this lecture we will mainly be exploring basic Sobel-Feldman Operators (often called Sobel for short)
- Later on in the course we will expand on this operator for general edge detection.

# Computer Vision with Python

- Gradients can be calculated in a specific direction.

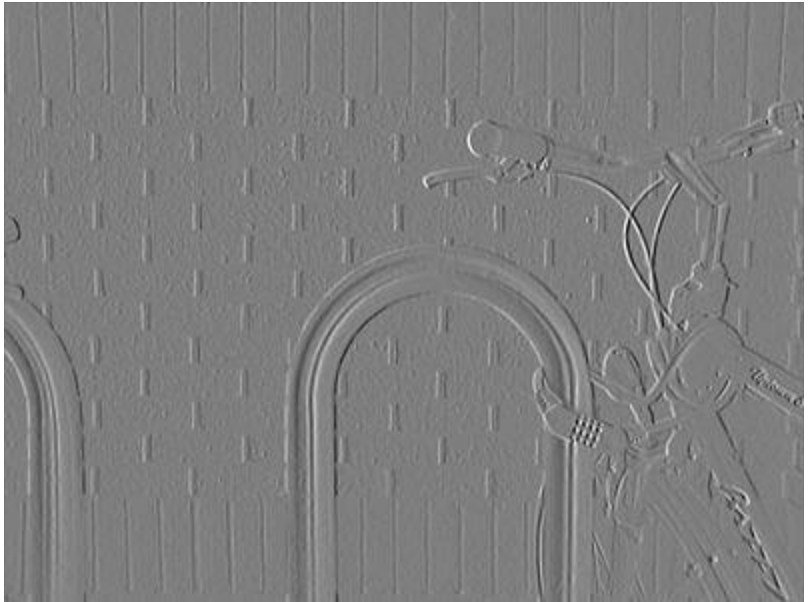


# Computer Vision with Python

- Here we see an image of a bike.
- Let’s calculate some gradients!

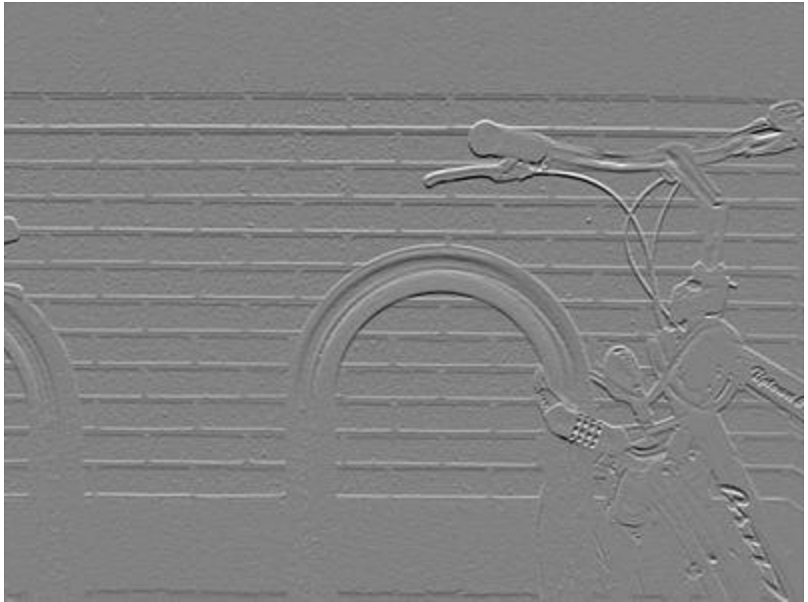
# Computer Vision with Python

- Normalized **x**-gradient from Sobel-Feldman Operator



# Computer Vision with Python

- Normalized **y**-gradient from Sobel-Feldman Operator



# Computer Vision with Python

- Normalized gradient magnitude from Sobel–Feldman operator

# Computer Vision with Python

- We will explore this sort of general edge detection in a future lecture.

- The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

- Check out the wikipedia article on the Sobel Operator for full math details.
- For our use case, we will focus on understanding the syntax of using Sobel with OpenCV.

# Computer Vision with Python

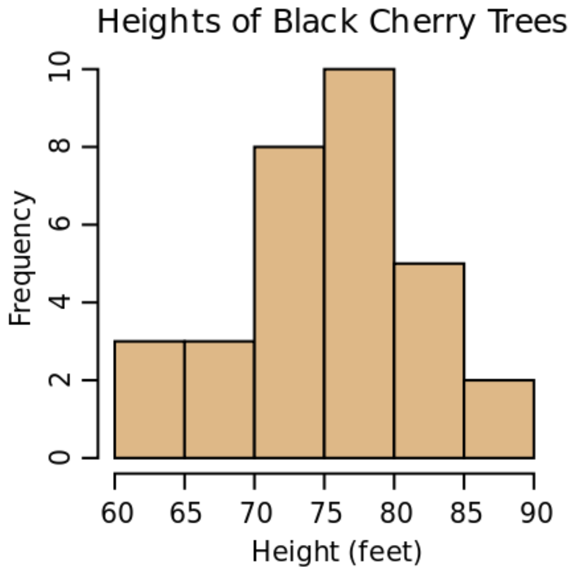
- Let's explore various gradient operators with OpenCV
- We'll also combine these concepts with a few other image processing techniques we've learned.

# Histograms

- Let's first understand what a regular histogram is, then we'll explain what an image histogram means.
- A histogram is a visual representation of the distribution of a continuous feature.

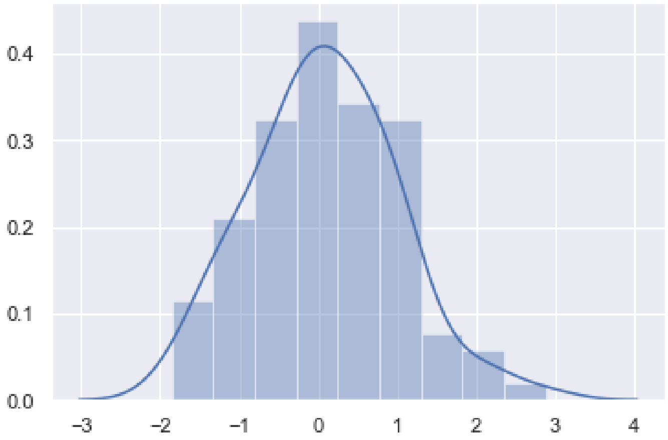
# Computer Vision with Python

- Simple example



# Computer Vision with Python

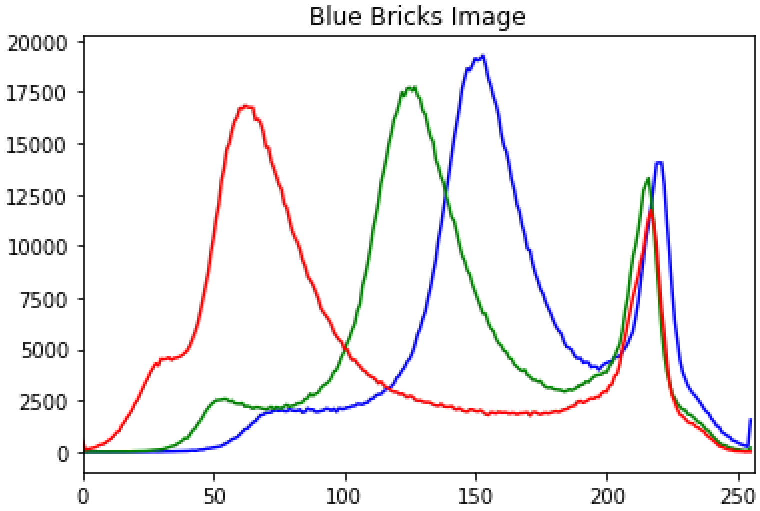
- We can also display the general trend of the frequency.



- For images, we can display the frequency of values for colors.
- Each of the three RGB channels has values between 0-255.
- We can plot these as 3 histograms on top of each other to see how much of each channel there is.

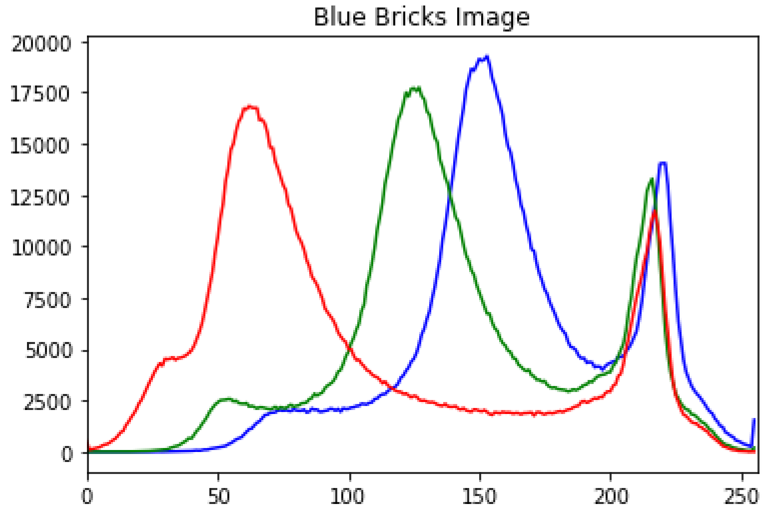
# Computer Vision with Python

- Example Image Histogram



# Computer Vision with Python

- Recall that 0 means pure black



- Let’s explore how to create image histograms with matplotlib and OpenCV!

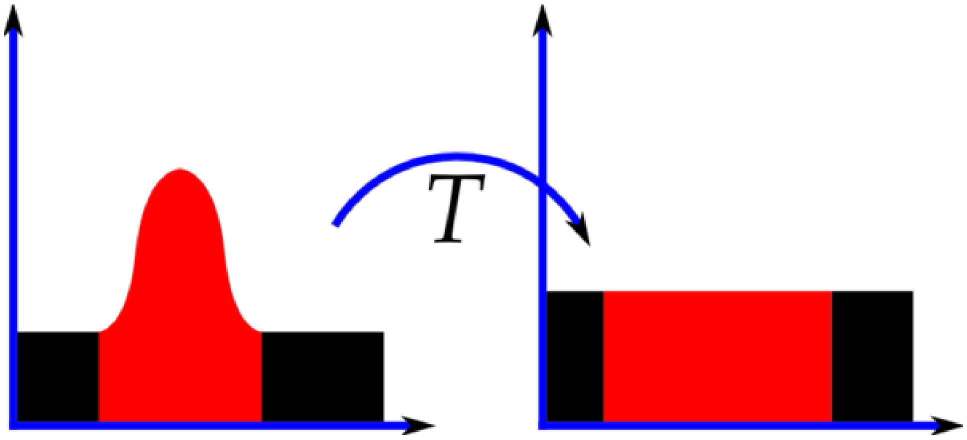
# Histograms - Part Two

- Let's continue our discussion of histograms with two additional topics:
  - Histograms on a masked portion of the image
  - Histogram Equalization

## Computer Vision with Python

- As mentioned in the previous lecture, we can select an ROI and only calculate the color histogram of that masked section.
- We'll show how to create a mask and achieve this effect.

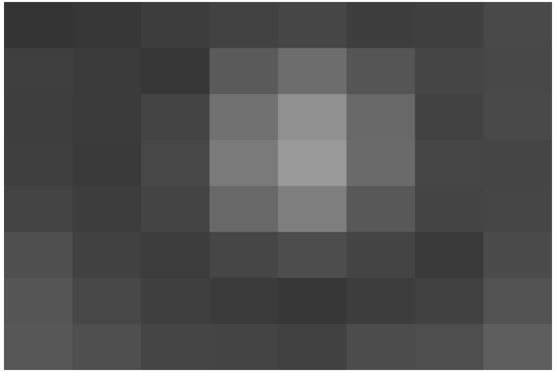
- Histogram Equalization is a method of contrast adjustment based on the image's histogram.

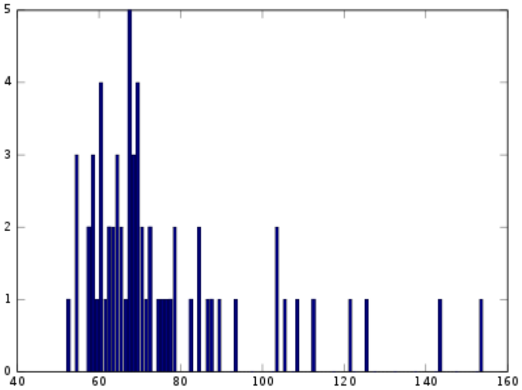


# Computer Vision with Python

- Example of an original image.

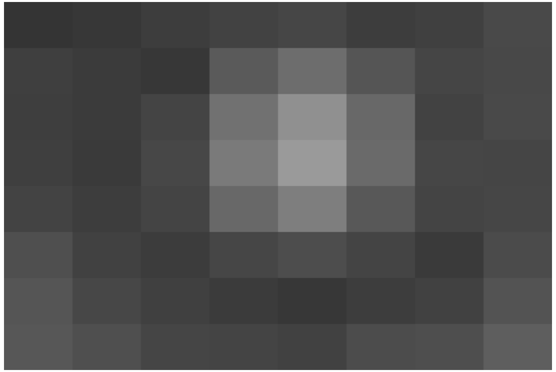
$$
\begin{bmatrix}
52 & 55 & 61 & 59 & 70 & 61 & 76 & 61 \\
62 & 59 & 55 & 104 & 94 & 85 & 59 & 71 \\
63 & 65 & 66 & 113 & 144 & 104 & 63 & 72 \\
64 & 70 & 70 & 126 & 154 & 109 & 71 & 69 \\
67 & 73 & 68 & 106 & 122 & 88 & 68 & 68 \\
68 & 79 & 60 & 79 & 77 & 66 & 58 & 75 \\
69 & 85 & 64 & 58 & 55 & 61 & 65 & 83 \\
70 & 87 & 69 & 68 & 65 & 73 & 78 & 90
\end{bmatrix}
$$

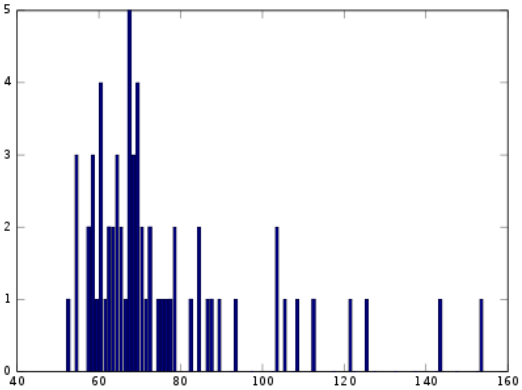




- Applying histogram equalization will reduce the color depth (shades of gray)

$$\begin{bmatrix} 52 & 55 & 61 & 59 & 70 & 61 & 76 & 61 \\ 62 & 59 & 55 & 104 & 94 & 85 & 59 & 71 \\ 63 & 65 & 66 & 113 & 144 & 104 & 63 & 72 \\ 64 & 70 & 70 & 126 & 154 & 109 & 71 & 69 \\ 67 & 73 & 68 & 106 & 122 & 88 & 68 & 68 \\ 68 & 79 & 60 & 79 & 77 & 66 & 58 & 75 \\ 69 & 85 & 64 & 58 & 55 & 61 & 65 & 83 \\ 70 & 87 & 69 & 68 & 65 & 73 & 78 & 90 \end{bmatrix}$$

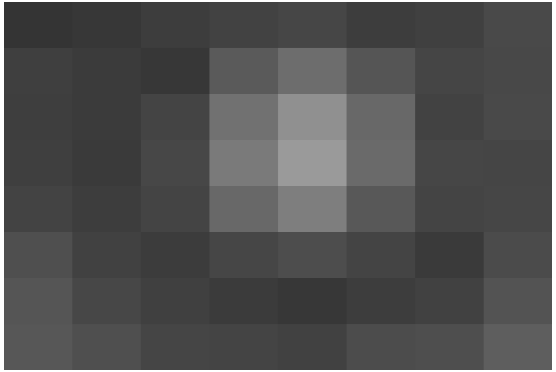


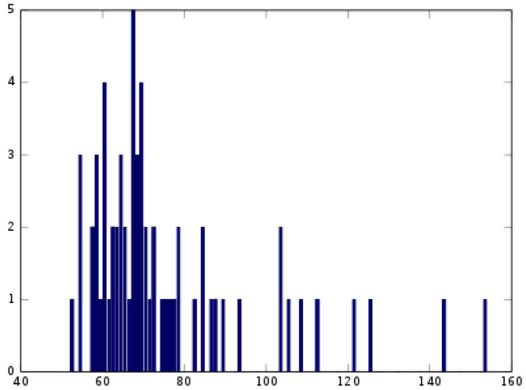


# Computer Vision with Python

- Let's take a look at the original min and max values

$$
\begin{bmatrix}
\boxed{52} & 55 & 61 & 59 & 70 & 61 & 76 & 61 \\
62 & 59 & 55 & 104 & 94 & 85 & 59 & 71 \\
63 & 65 & 66 & 113 & 144 & 104 & 63 & 72 \\
64 & 70 & 70 & 126 & \boxed{154} & 109 & 71 & 69 \\
67 & 73 & 68 & 106 & 122 & 88 & 68 & 68 \\
68 & 79 & 60 & 79 & 77 & 66 & 58 & 75 \\
69 & 85 & 64 & 58 & 55 & 61 & 65 & 83 \\
70 & 87 & 69 & 68 & 65 & 73 & 78 & 90
\end{bmatrix}
$$

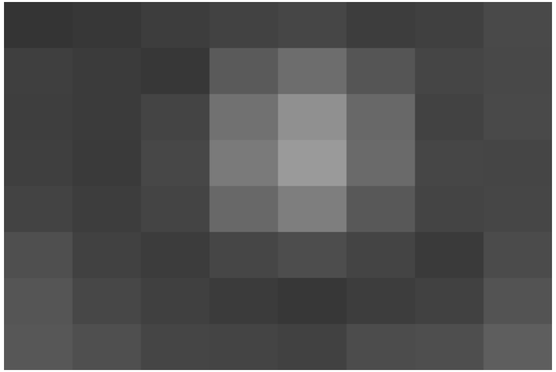


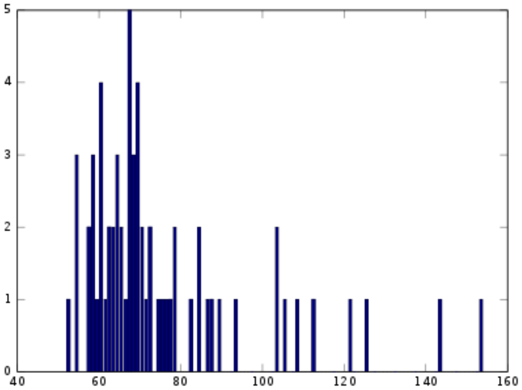


# Computer Vision with Python

- Let's now apply histogram equalization (check the resource link for full math)

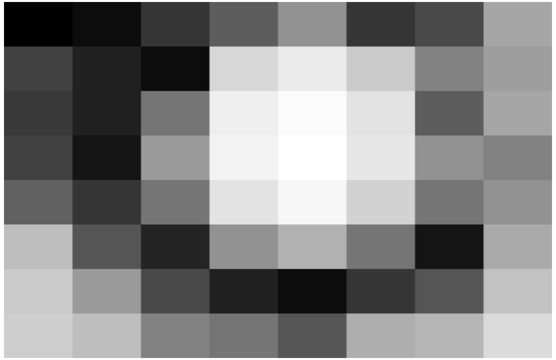
$$\begin{bmatrix} \boxed{52} & 55 & 61 & 59 & 70 & 61 & 76 & 61 \\ 62 & 59 & 55 & 104 & 94 & 85 & 59 & 71 \\ 63 & 65 & 66 & 113 & 144 & 104 & 63 & 72 \\ 64 & 70 & 70 & 126 & \boxed{154} & 109 & 71 & 69 \\ 67 & 73 & 68 & 106 & 122 & 88 & 68 & 68 \\ 68 & 79 & 60 & 79 & 77 & 66 & 58 & 75 \\ 69 & 85 & 64 & 58 & 55 & 61 & 65 & 83 \\ 70 & 87 & 69 & 68 & 65 & 73 & 78 & 90 \end{bmatrix}$$

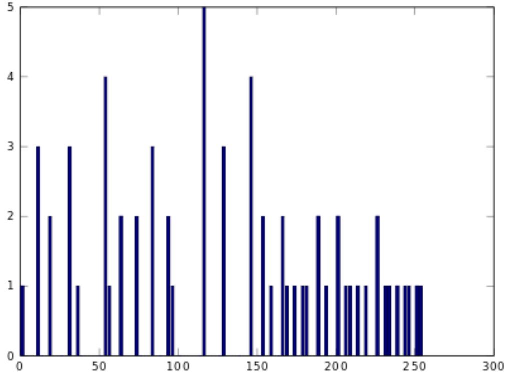




- Let's now apply histogram equalization (check the resource link for full math)

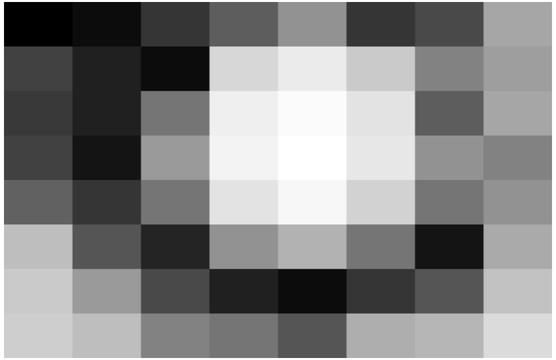
$$\begin{bmatrix} \boxed{0} & 12 & 53 & 32 & 146 & 53 & 174 & 53 \\ 57 & 32 & 12 & 227 & 219 & 202 & 32 & 154 \\ 65 & 85 & 93 & 239 & 251 & 227 & 65 & 158 \\ 73 & 146 & 146 & 247 & \boxed{255} & 235 & 154 & 130 \\ 97 & 166 & 117 & 231 & 243 & 210 & 117 & 117 \\ 117 & 190 & 36 & 190 & 178 & 93 & 20 & 170 \\ 130 & 202 & 73 & 20 & 12 & 53 & 85 & 194 \\ 146 & 206 & 130 & 117 & 85 & 166 & 182 & 215 \end{bmatrix}$$

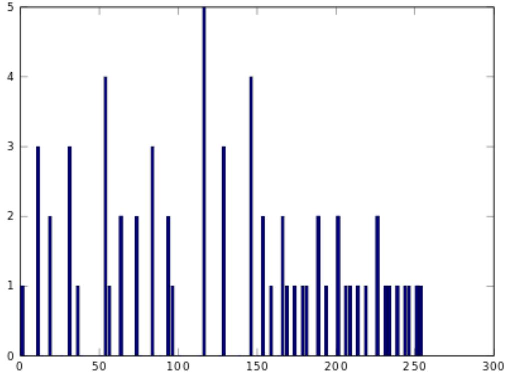




- Notice how the min and max values have now been equalized to be 0 and 255.

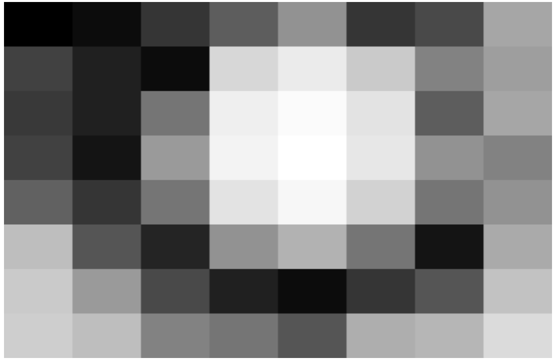
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 12 | 53 | 32 | 146 | 53 | 174 | 53 |
| 57 | 32 | 12 | 227 | 219 | 202 | 32 | 154 |
| 65 | 85 | 93 | 239 | 251 | 227 | 65 | 158 |
| 73 | 146 | 146 | 247 | 255 | 235 | 154 | 130 |
| 97 | 166 | 117 | 231 | 243 | 210 | 117 | 117 |
| 117 | 190 | 36 | 190 | 178 | 93 | 20 | 170 |
| 130 | 202 | 73 | 20 | 12 | 53 | 85 | 194 |
| 146 | 206 | 130 | 117 | 85 | 166 | 182 | 215 |

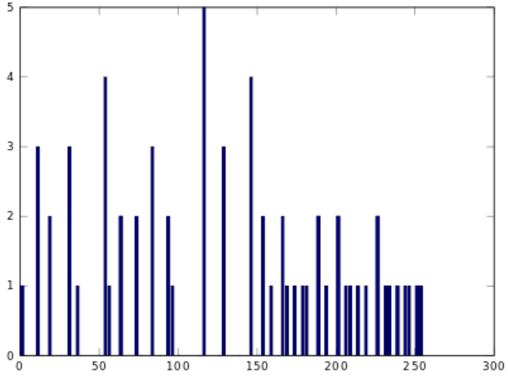




- We also now see less shades of gray (resulting in higher contrast)

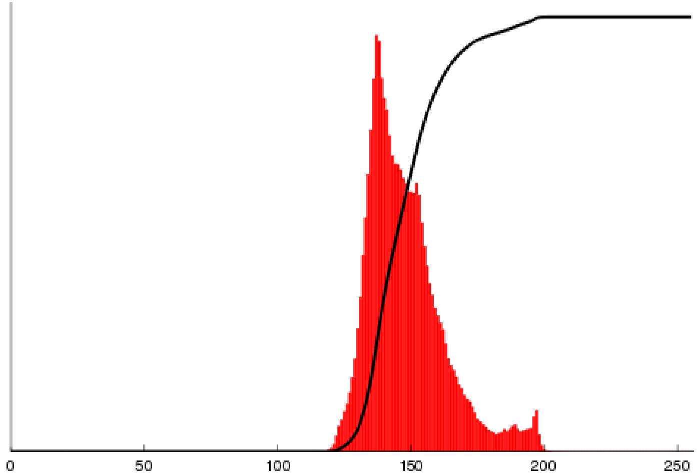
$$\begin{bmatrix} \boxed{0} & 12 & 53 & 32 & 146 & 53 & 174 & 53 \\ 57 & 32 & 12 & 227 & 219 & 202 & 32 & 154 \\ 65 & 85 & 93 & 239 & 251 & 227 & 65 & 158 \\ 73 & 146 & 146 & 247 & \boxed{255} & 235 & 154 & 130 \\ 97 & 166 & 117 & 231 & 243 & 210 & 117 & 117 \\ 117 & 190 & 36 & 190 & 178 & 93 & 20 & 170 \\ 130 & 202 & 73 & 20 & 12 & 53 & 85 & 194 \\ 146 & 206 & 130 & 117 & 85 & 166 & 182 & 215 \end{bmatrix}$$





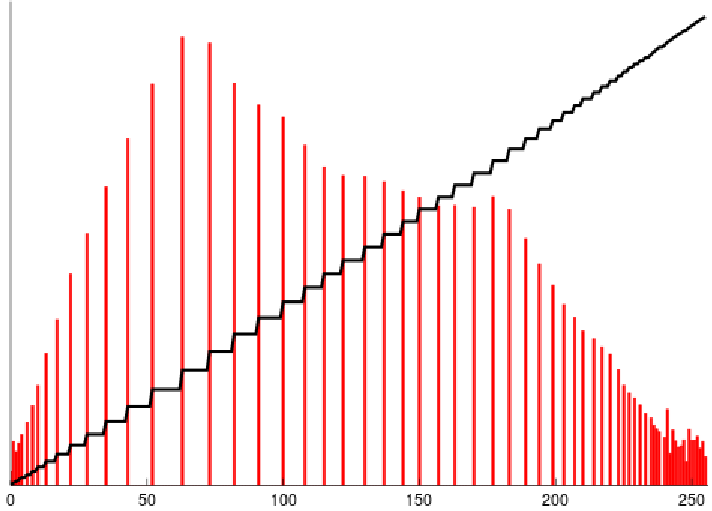
# Computer Vision with Python

- Let’s now see the results on a normal image

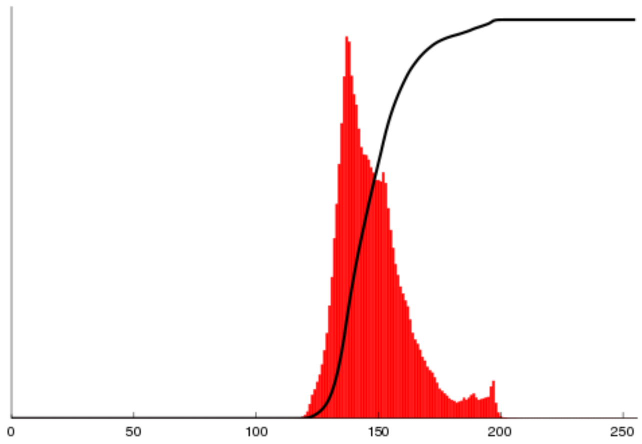


# Computer Vision with Python

- Let’s now see the results on a normal image



- Corresponding histogram (red) and cumulative histogram (black)




# Computer Vision with Python

- Let’s explore this with OpenCV!

# Image Processing Assessment Overview

# Image Processing Assessment Solutions