## Lecture Notes on Computer Vision

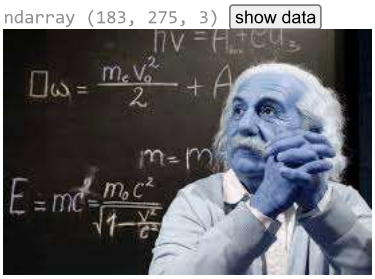
Prepared By: Chandan Chaudhari

Github: https://github.com/chandanc5525

```
1 import cv2
2 import numpy as np
3 import warnings
4 warnings.filterwarnings(action = 'ignore')
```

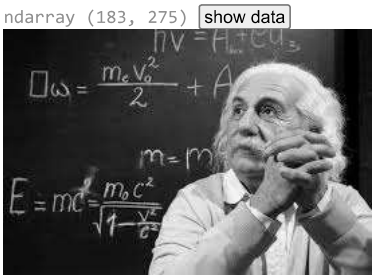
```
1 image = cv2.imread(r"/content/Albert.png")
2 image
```

ndarray (183, 275, 3) show data



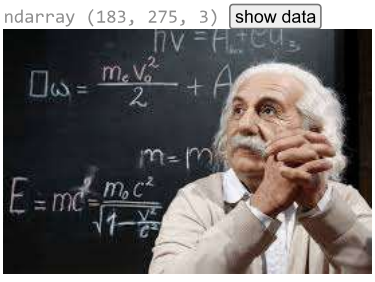
```
1 # Convert to Gray Scale Image
2
3 GrayScale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
4 GrayScale
```

ndarray (183, 275) show data



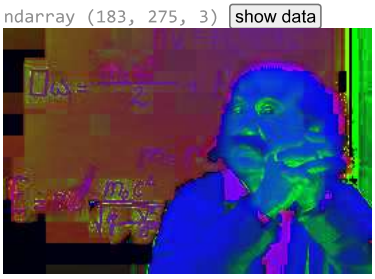
```
1 # Convert to RGB Scale
2
3 RGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
4 RGB
```

ndarray (183, 275, 3) show data



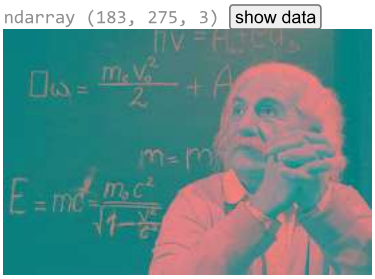
```
1 # Convert to HSV Scale (Hue Stauration Value)
2
3 HSV = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
4 HSV
```

ndarray (183, 275, 3) show data



```
# Convert to LAB Scale (Lightness, A (green-red), B (blue-yellow))

LAB = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
LAB
```

ndarray (183, 275, 3) show data

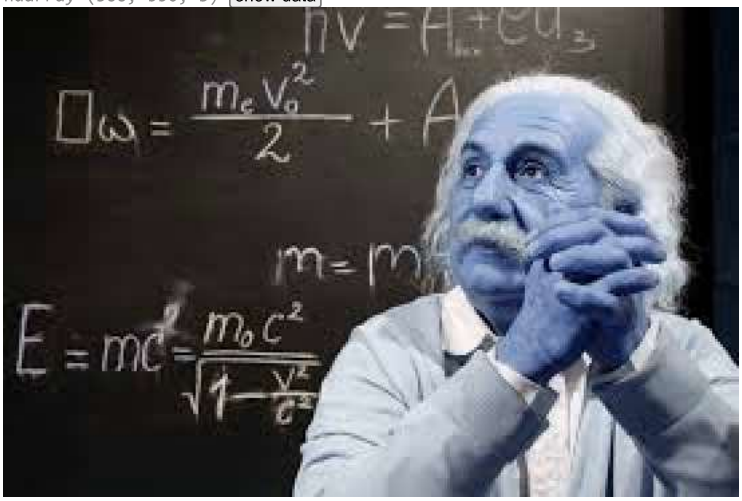


```
# Convert to YCrCb Scale (Luma, Chroma Red, Chroma Blue)

YCrCb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
YCrCb
```

ndarray (183, 275, 3) show data

```
# Resize the Image
'''
interpolation : It gives smoother results and reduces noise or aliasing.
'''
Resize = cv2.resize(image, (550,366), interpolation = cv2.INTER_AREA)
Resize
```

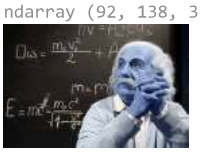
ndarray (366, 550, 3) show data



```
# Image Scaling

# Reduce size to 50%
smaller = cv2.resize(image, None, fx=0.5, fy=0.5)

# Double the size
bigger = cv2.resize(image, None, fx=2, fy=2)
```

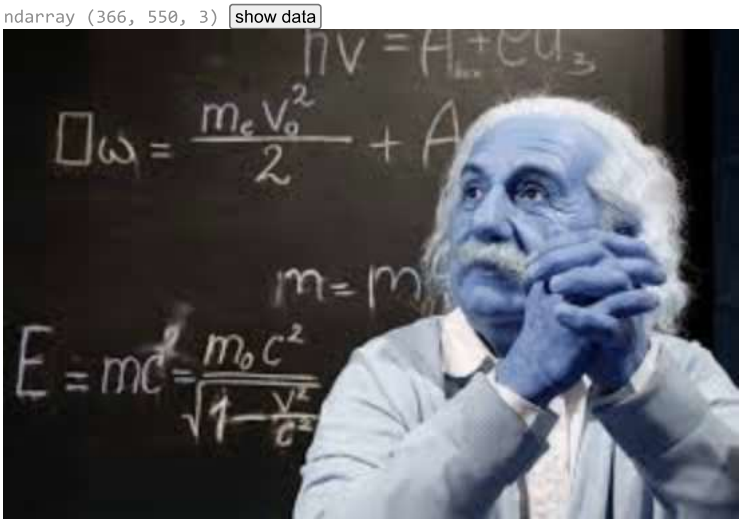
```
smaller
```

ndarray (92, 138, 3) show data

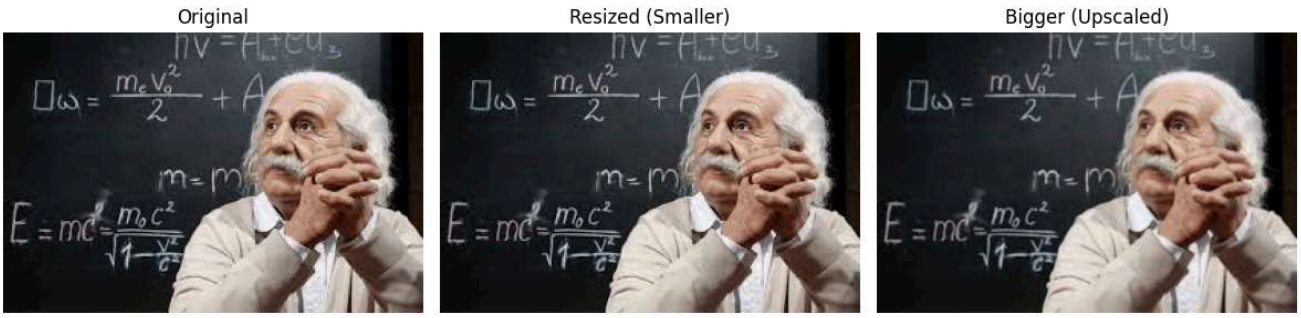


```
bigger
```

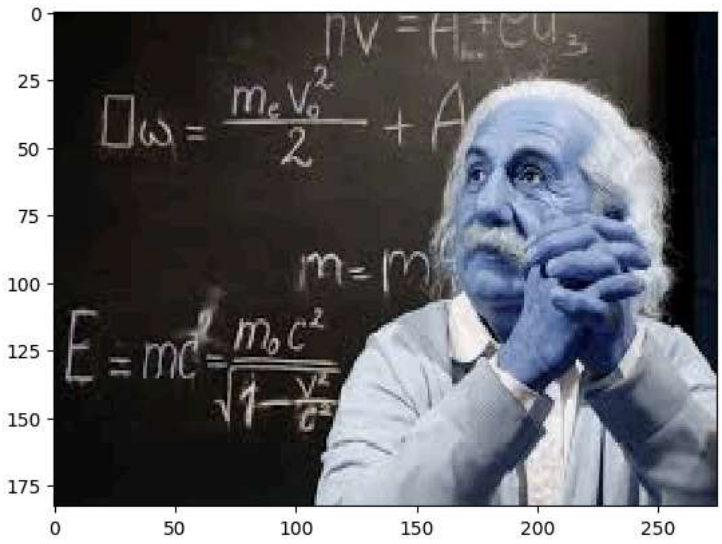
ndarray (366, 550, 3) show data



```
# Image Analysis

import matplotlib.pyplot as plt

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
resized = cv2.cvtColor(Resize, cv2.COLOR_BGR2RGB)
bigger = cv2.cvtColor(bigger, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.imshow(image_rgb)
plt.title('Original')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(resized)
plt.title('Resized (Smaller)')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(bigger)
plt.title('Bigger (Upscaled)')
plt.axis('off')

plt.tight_layout()
plt.show()
```




```
plt.imshow(image,cmap = 'Dark2', interpolation = 'nearest', aspect = 'auto')
plt.show()
```



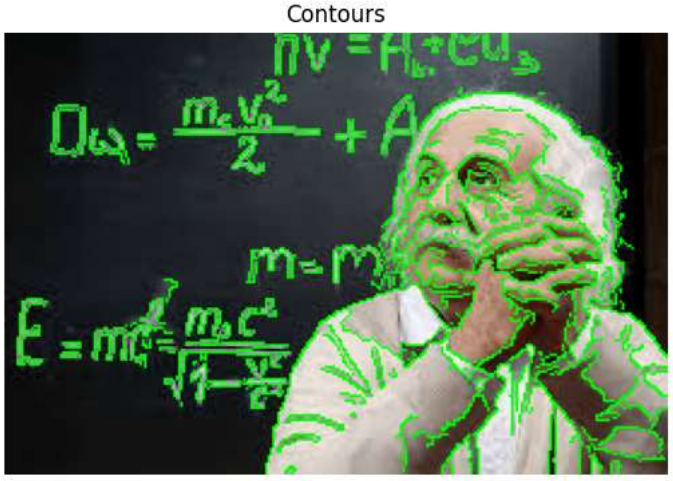

## Edge Detection Filter

```
# Sobel Filter:
'''
- Detects edges by calculating gradients in x and y directions.

- cv2.CV_64F:

Output image data type (64-bit float) to handle negative values 1, 0

- ksize=5: Uses a 5x5 kernel instead of default 3x3

'''

sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=5)
sobelxy = cv2.Sobel(image, cv2.CV_64F, 1, 1, ksize=5)

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.imshow(sobelx, cmap='gray')
plt.title('Sobel X')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(sobely, cmap='gray')
plt.title('Sobel Y')
```

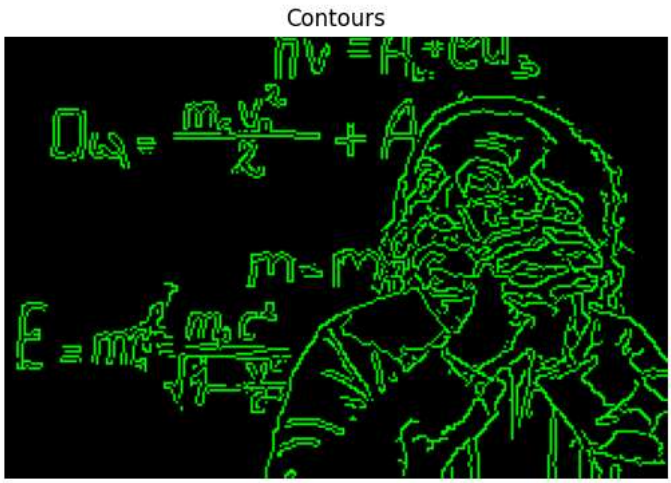
```
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(sobelxy, cmap='gray')
plt.title('Sobel XY')
plt.axis('off')

plt.tight_layout()
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for inte
```


```
# Using Canny edges
edges = cv2.Canny(image, 100, 200)

# Find contours
contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL,
                                       cv2.CHAIN_APPROX_SIMPLE)

# Create a copy for drawing contours
if len(image.shape) == 2 or image.shape[2] == 1:
    # Image is grayscale
    img_contours = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)
else:
    # Image is already BGR
    img_contours = image.copy()

# Draw contours
cv2.drawContours(img_contours, contours, -1, (0,255,0), 1)

plt.imshow(cv2.cvtColor(img_contours, cv2.COLOR_BGR2RGB))  # Convert for matplotlib
plt.title("Contours")
plt.axis('off')
plt.show()
```



```
# Using Canny edges
edges = cv2.Canny(image, 100, 200)

# Find contours
contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL,
                                       cv2.CHAIN_APPROX_SIMPLE)

# Create blank black canvas
img_contours = np.zeros((image.shape[0], image.shape[1], 3), dtype=np.uint8)

# Draw contours in green
cv2.drawContours(img_contours, contours, -1, (0,255,0), 1)

plt.imshow(img_contours)
plt.title("Contours")
plt.axis('off')
plt.show()
```




```
img = cv2.imread('/content/Albert.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Load pre-trained face detector
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_

# Detect faces
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# Draw rectangles around faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Face Detection")
plt.axis('off')
plt.show()

```

Face Detection