Aim: To get familiar with OpenCV operations

Note: Use matplotlib or OpenCV to display images and write the code after the #Answer comment

Grade points = 20%

Please submit the notebook as the submission and make sure output for each cell is displayed and all cells are executed

Important Note: Please submit your original work and don't share your work with others

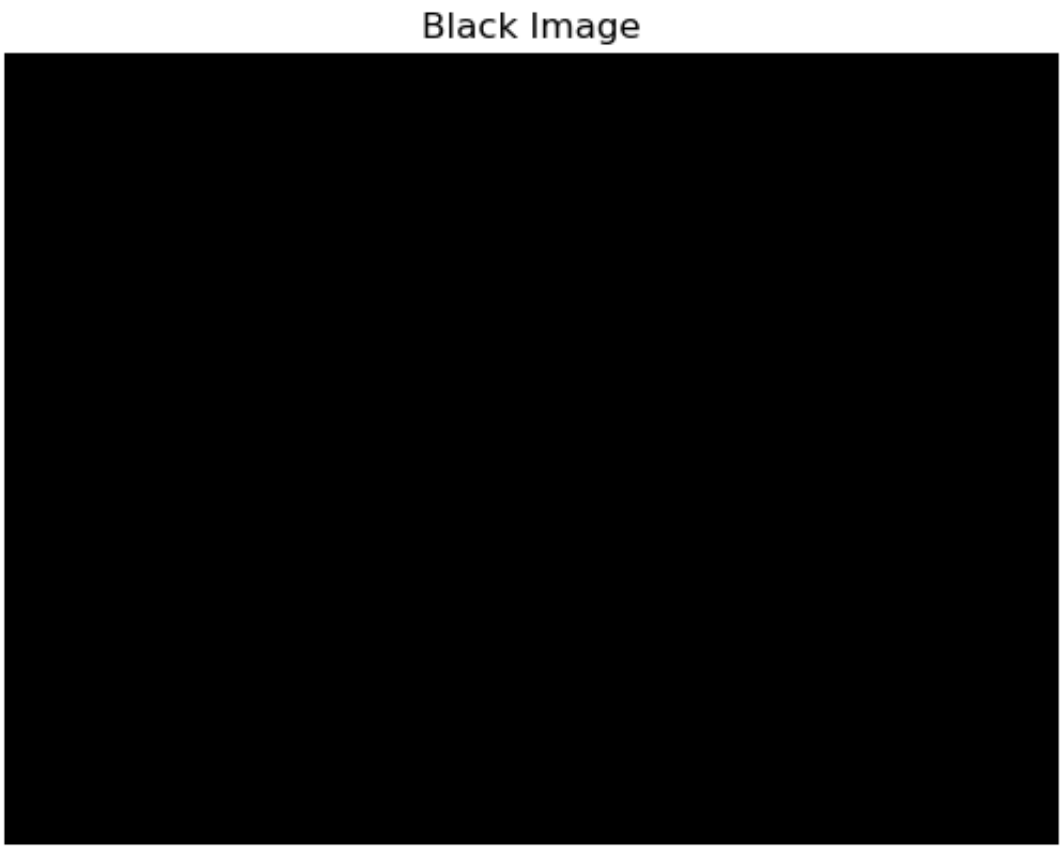
Question 1

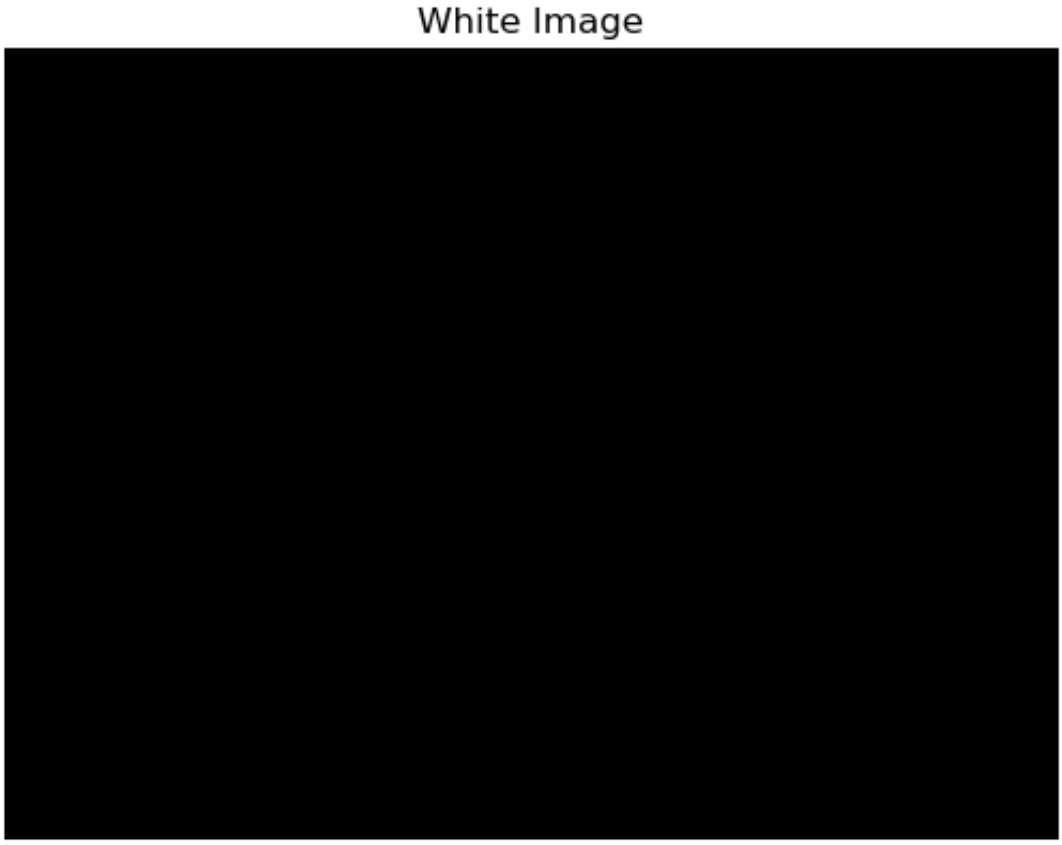
In [3]:
```
#read image opencv-logo
#display shape and size of the image
import cv2

# Load the image
image = cv2.imread('opencv-logo.png')

# Display the shape of the image
print("Shape of the image:", image.shape)

# Display the size of the image
print("Size of the image:", image.size)

#Answer
```

```
Shape of the image: (600, 487, 3)
Size of the image: 876600
```

Question 2

In [5]:
```
#create a numpy array of zeroes that is 150 pixels tall, 200 pixels wide
#display this black image
#Similarly create a numpy array on ones with above dimensions and display
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Create a black image (all zeros) of 150 pixels height and 200 pixels wi
black_image = np.zeros((150, 200), dtype=np.uint8)

# Create a white image (all ones multiplied by 255) of the same dimension
white_image = np.ones((150, 200), dtype=np.uint8) * 255

# Display the black image
plt.imshow(black_image, cmap='gray')
plt.title("Black Image")
plt.axis('off')
plt.show()

# Display the white image
plt.imshow(white_image, cmap='gray')
plt.title("White Image")
plt.axis('off')
plt.show()

#Answer
```





Question 3

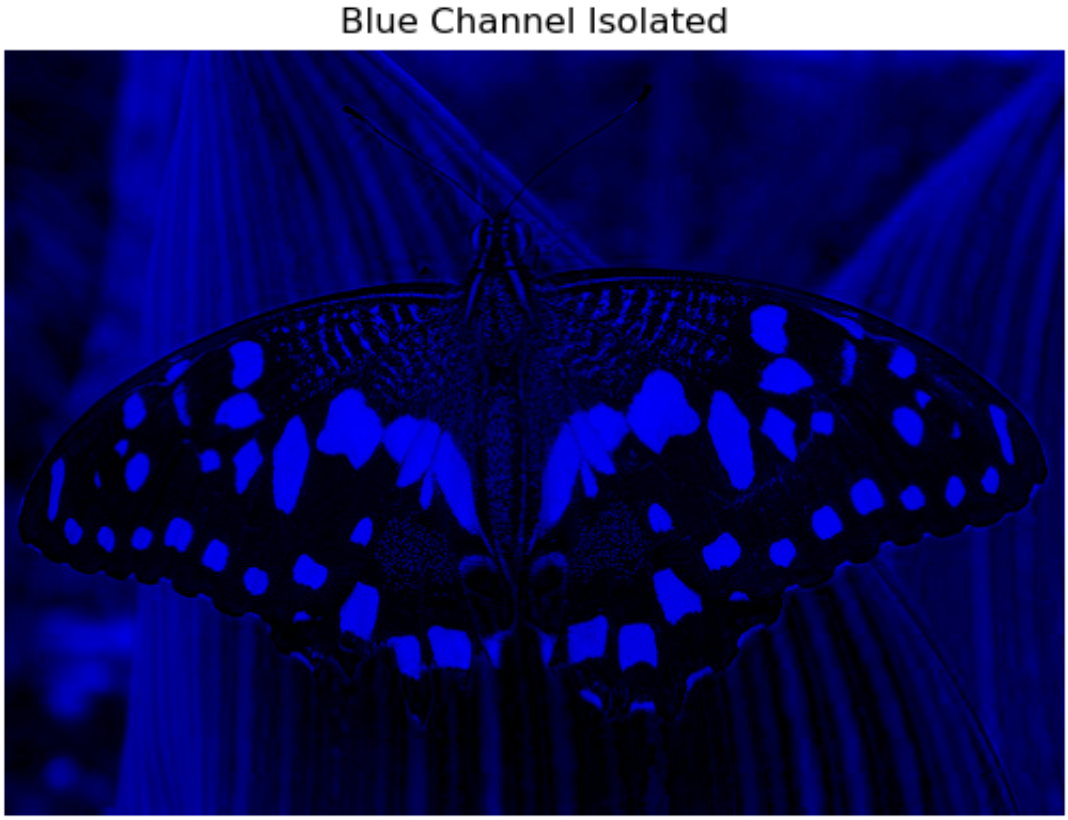
In [9]:
```
#read the butterfly image and display the r,g,b channels of the image
#Then display the butterfly image in the HSV color space
import cv2
import matplotlib.pyplot as plt

# Load the butterfly image
image = cv2.imread('butterfly.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert to RGB for

# Split the image into R, G, B channels
R, G, B = image_rgb[:, :, 0], image_rgb[:, :, 1], image_rgb[:, :, 2]

# Display the R, G, and B channels separately
plt.figure(figsize=(10, 7))

plt.subplot(1, 3, 1)
plt.imshow(R, cmap='gray')
plt.title("Red Channel")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(G, cmap='gray')
plt.title("Green Channel")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(B, cmap='gray')
plt.title("Blue Channel")
plt.axis('off')

plt.show()

# Convert the image from BGR to HSV color space
image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Display the image in HSV color space
plt.imshow(cv2.cvtColor(image_hsv, cv2.COLOR_HSV2RGB))
plt.title("Butterfly Image in HSV Color Space")
plt.axis('off')
plt.show()

#Answer
```

Question 4

In [11]:
```
#create a 5 by 5 array where every number is a 10
#run the cell below to create an array of random numbers and see if you c
#what are the largest and smalled values in this array?
#use PIL and matplotlib to read and display the any image of your choice
#convert the image to a NumPy Array
#use slicing to set the RED and GREEN channels of the picture to 0, then

#Answer
import numpy as np

# Create a 5x5 array with all elements set to 10
array_10 = np.full((5, 5), 10)
print("5x5 Array with all elements as 10:\n", array_10)

# Create an array of random numbers (for example, 5x5)
random_array = np.random.rand(5, 5)
print("\nRandom Array:\n", random_array)

# Find the largest and smallest values in the array
max_value = np.max(random_array)
min_value = np.min(random_array)
print("\nLargest Value:", max_value)
print("Smallest Value:", min_value)

from PIL import Image
import matplotlib.pyplot as plt

# Load the image (replace 'butterfly.jpg' with the path to your image)
image_path = 'butterfly.jpg'
image = Image.open(image_path)

# Display the image using Matplotlib
plt.imshow(image)
plt.title("Original Image")
plt.axis('off')
plt.show()

# Convert the image to a NumPy array
image_array = np.array(image)

# Set the Red and Green channels to 0
# Note: The order is RGB when using PIL, so indices are 0, 1, 2 for R, G,
image_array[:, :, 0] = 0  # Red channel
image_array[:, :, 1] = 0  # Green channel

# Display the image with only the Blue channel
plt.imshow(image_array)
plt.title("Blue Channel Isolated")
plt.axis('off')
plt.show()
```

```
5x5 Array with all elements as 10:
 [[10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]]

Random Array:
 [[0.80521422 0.10944249 0.76974232 0.27312134 0.50696854]
 [0.77120371 0.88330481 0.67174382 0.45842736 0.42091676]
 [0.04694811 0.78937777 0.80145285 0.06988986 0.84213921]
 [0.5237598  0.12237482 0.24235416 0.3734558  0.67970359]
 [0.76252861 0.52465347 0.6027501  0.98354803 0.92243723]]

Largest Value: 0.9835480332933183
Smallest Value: 0.046948108206913974
```

Original Image

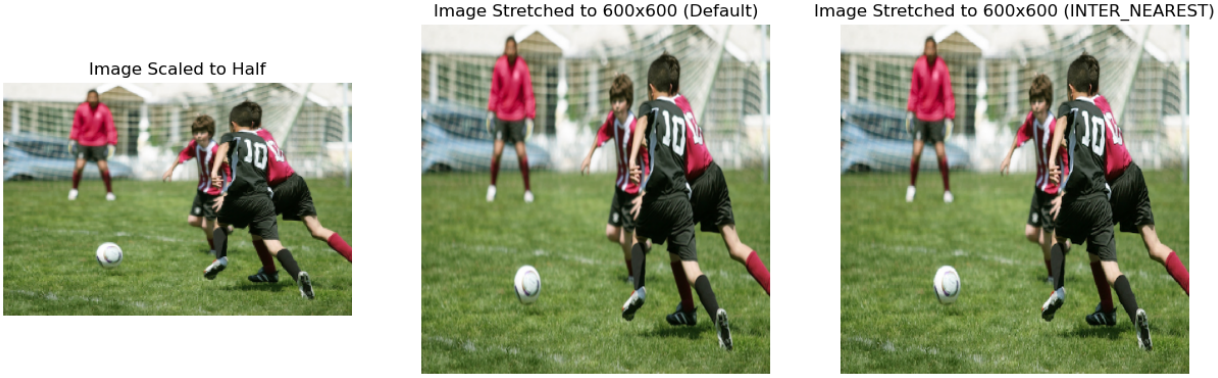


Question 5

In [ ]:
```
#read in the players image
#Perform scaling using resize method
#a) half the image using dim=(0,0), fx=0.5, fy=0.5
#b) stretch the image to dim = (600,600)
#c) stretch the image to dim = (600,600 using interpolation=cv2.INTER_NEA

#Answer
```

Question 6

In [13]:
```
#read in the detect_blob image
#then draw a box shape around the blue solid box in the image that is you
#crop your region of interest
#and then rotate it 45 degrees so that it is not clipped

#Answer

import cv2
import matplotlib.pyplot as plt

# Load the players image
image = cv2.imread('players.jpg')

# a) Scale the image to half its size
image_half = cv2.resize(image, (0, 0), fx=0.5, fy=0.5)

# b) Stretch the image to 600x600 with default interpolation (INTER_LINEA
image_stretch_default = cv2.resize(image, (600, 600))

# c) Stretch the image to 600x600 using INTER_NEAREST interpolation
image_stretch_nearest = cv2.resize(image, (600, 600), interpolation=cv2.I

# Display the images
plt.figure(figsize=(15, 5))

# Original Image
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(image_half, cv2.COLOR_BGR2RGB))
plt.title("Image Scaled to Half")
plt.axis('off')

# Stretched Image with Default Interpolation
plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(image_stretch_default, cv2.COLOR_BGR2RGB))
plt.title("Image Stretched to 600x600 (Default)")
plt.axis('off')

# Stretched Image with INTER_NEAREST Interpolation
plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(image_stretch_nearest, cv2.COLOR_BGR2RGB))
plt.title("Image Stretched to 600x600 (INTER_NEAREST)")
plt.axis('off')

plt.show()
```



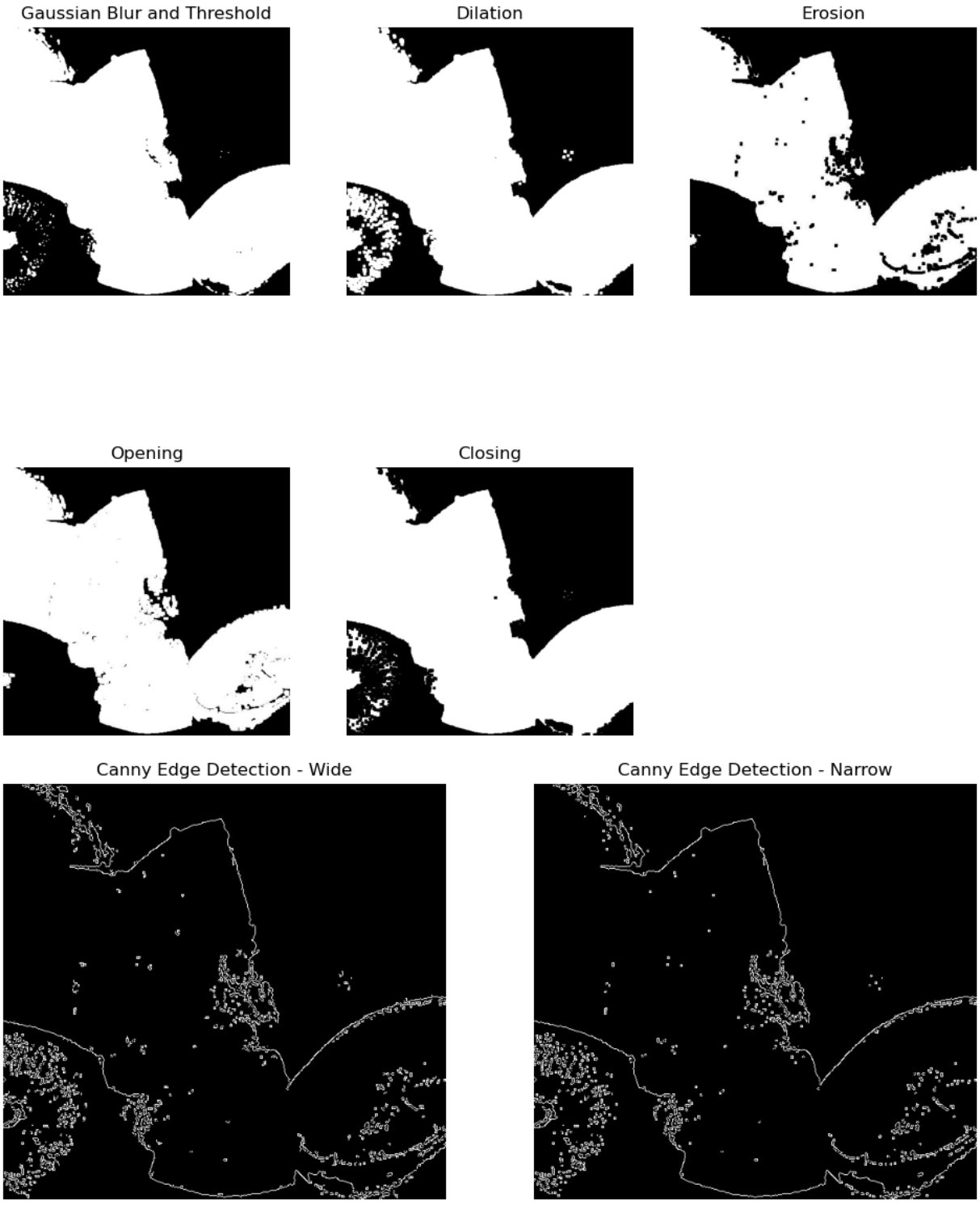
Question 7

In [ ]:
```
#read in the thresh image, perform thresholding by using gaussian blur wi
#then perform dilation, erosion, opening & closing using a 5x5 kernel wit
#Finally apply canny edge detection - experiment using wide and narrow th
#(so just 2 images in total one wide and one narrow)

#Answer
```

Question 8

In [1]:
```
#read in the butterfly image
#open the chess_football image and display it in the notebook. Make sure
#flip the image upside down and display it in the notebook.
#draw an empty RED rectangle around the butterfly and display the image i
#draw a BLUE TRIANGLE in the middle of the image. The size and angle is u
#now fill in this triangle
#display the original image as well as the resultant image

#Answer
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the thresh image
image = cv2.imread('thresh.jpg', cv2.IMREAD_GRAYSCALE)

# Step 1: Apply Gaussian blur and then thresholding
blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
_, thresh_blur = cv2.threshold(blurred_image, 55, 255, cv2.THRESH_BINARY)

# Step 2: Morphological operations (dilation, erosion, opening, closing)
kernel = np.ones((5, 5), np.uint8)

# Dilation
dilated = cv2.dilate(image, kernel, iterations=1)

# Erosion
eroded = cv2.erode(image, kernel, iterations=1)

# Opening (erosion followed by dilation)
opened = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

# Closing (dilation followed by erosion)
closed = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)

# Step 3: Canny edge detection with wide and narrow thresholds
# Wide threshold
edges_wide = cv2.Canny(image, 10, 200)

# Narrow threshold
edges_narrow = cv2.Canny(image, 100, 150)
```

```
# Display the results
plt.figure(figsize=(12, 10))

# Blurred and Thresholded Image
plt.subplot(2, 3, 1)
plt.imshow(thresh_blur, cmap='gray')
plt.title("Gaussian Blur and Threshold")
plt.axis('off')

# Dilation
plt.subplot(2, 3, 2)
plt.imshow(dilated, cmap='gray')
plt.title("Dilation")
plt.axis('off')

# Erosion
plt.subplot(2, 3, 3)
plt.imshow(eroded, cmap='gray')
plt.title("Erosion")
plt.axis('off')

# Opening
plt.subplot(2, 3, 4)
plt.imshow(opened, cmap='gray')
plt.title("Opening")
plt.axis('off')

# Closing
plt.subplot(2, 3, 5)
plt.imshow(closed, cmap='gray')
plt.title("Closing")
plt.axis('off')

# Display Canny Edge Detection results
plt.figure(figsize=(12, 6))

# Wide Threshold
plt.subplot(1, 2, 1)
plt.imshow(edges_wide, cmap='gray')
plt.title("Canny Edge Detection - Wide")
plt.axis('off')

# Narrow Threshold
plt.subplot(1, 2, 2)
plt.imshow(edges_narrow, cmap='gray')
plt.title("Canny Edge Detection - Narrow")
plt.axis('off')

plt.show()
```


Gaussian Blur and Threshold
Dilation
Erosion
Opening
Closing
Canny Edge Detection - Wide
Canny Edge Detection - Narrow

Question 9

In [5]:
```
import cv2

# Function to be called when a mouse event occurs
def draw_circle(event, x, y, flags, param):
    # Check if the right mouse button was clicked
    if event == cv2.EVENT_RBUTTONDOWN:
        # Draw a red circle with center at (x, y) and radius 20
        cv2.circle(img, (x, y), 20, (0, 0, 255), 2)  # Red color in BGR a

# Load the image (replace 'players.jpg' with the actual image file)
img = cv2.imread('players.jpg')
if img is None:
    print("Error loading image. Check the file path.")
    exit()

# Create a named window
cv2.namedWindow('Image')

# Set the mouse callback function to capture mouse events
cv2.setMouseCallback('Image', draw_circle)

# Display the image and wait for user interaction
while True:
    cv2.imshow('Image', img)
    if cv2.waitKey(1) & 0xFF == 27:  # Press 'ESC' to exit the loop
        break

# Save the modified image
cv2.imwrite('players_with_circles.jpg', img)
print("Image saved as 'players_with_circles.jpg'.")

# Display the saved image one last time
#cv2.imshow('Modified Image', img)
plt.imshow(img)

cv2.waitKey(0)  # Wait until any key is pressed
cv2.destroyAllWindows()
```

```
2024-10-13 20:25:20.223 python[12664:7196964] +[IMKClient subclass]: chos
e IMKClient_Legacy
2024-10-13 20:25:20.223 python[12664:7196964] +[IMKInputSession subclas
s]: chose IMKInputSession_Legacy
Image saved as 'players_with_circles.jpg'.
```

Question 10

In [1]:

```
#open and display the any image of your choice
#apply a binary threshold onto the image
#convert image colorspace to HSV and display the image
#create a low pass filter with a 4 by 4 Kernel filled with values of 1/10
#create a Horizontal Sobel Filter with a kernel size of 5 to the grayscal
#plot the color histograms for the RED, BLUE, and GREEN channel of the im

#Answer
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image (replace 'image.jpg' with the actual image file)
image = cv2.imread('players.jpg')
if image is None:
    print("Error loading image. Check the file path.")
    exit()

# Display the original image
plt.figure(figsize=(10, 5))
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis('off')
plt.show()

# Step 1: Apply a binary threshold onto the image
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_thresh = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
plt.imshow(binary_thresh, cmap='gray')
```

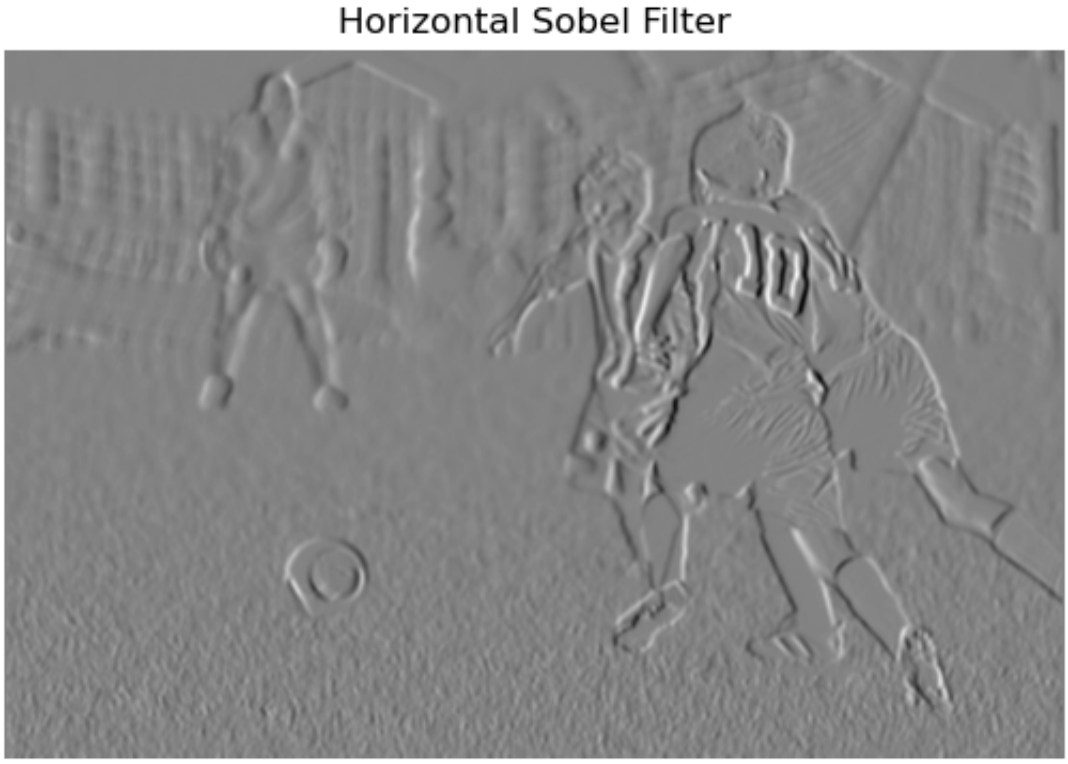
```
plt.title("Binary Threshold")
plt.axis('off')
plt.show()

# Step 2: Convert image to HSV color space and display
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
plt.imshow(cv2.cvtColor(hsv_image, cv2.COLOR_HSV2RGB))
plt.title("HSV Color Space")
plt.axis('off')
plt.show()

# Step 3: Create a low-pass filter (4x4 kernel with 1/10 value) and apply
kernel = np.ones((4, 4), np.float32) / 10
low_pass_image = cv2.filter2D(image, -1, kernel)
plt.imshow(cv2.cvtColor(low_pass_image, cv2.COLOR_BGR2RGB))
plt.title("Low-Pass Filtered Image (Blurring)")
plt.axis('off')
plt.show()

# Step 4: Apply a Horizontal Sobel Filter to the grayscale version of the
sobel_horizontal = cv2.Sobel(gray_image, cv2.CV_64F, 1, 0, ksize=5)
plt.imshow(sobel_horizontal, cmap='gray')
plt.title("Horizontal Sobel Filter")
plt.axis('off')
plt.show()

# Step 5: Plot color histograms for the Red, Green, and Blue channels
channels = cv2.split(image)
colors = ("b", "g", "r")  # OpenCV uses BGR format

plt.figure()
plt.title("Color Histogram")
plt.xlabel("Bins")
plt.ylabel("Number of Pixels")

# Loop over the channels and plot each histogram
for channel, color in zip(channels, colors):
    hist = cv2.calcHist([channel], [0], None, [256], [0, 256])
    plt.plot(hist, color=color)
    plt.xlim([0, 256])

plt.show()
```

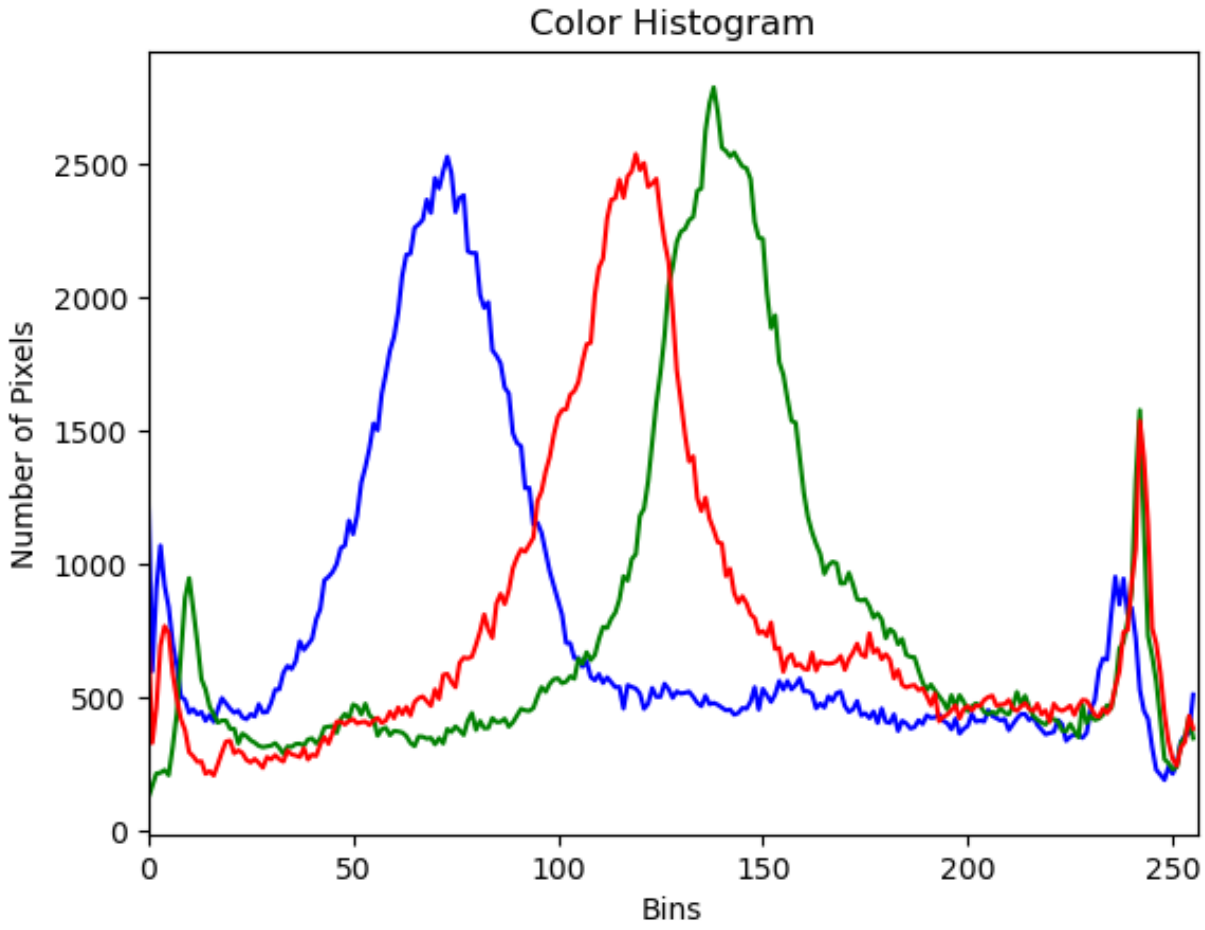
Original Image

Binary Threshold

HSV Color Space

Low-Pass Filtered Image (Blurring)


Horizontal Sobel Filter


Color Histogram
Number of Pixels
0
500
1000
1500
2000
2500
0
50
100
150
200
250
Bins

In [ ]: