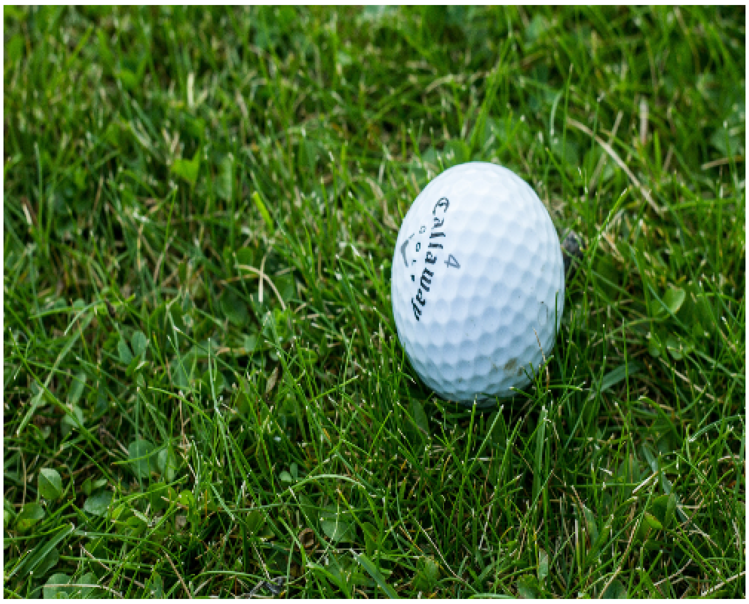
IMAGE ENHANCEMENT AND MANIPULATION

1. Change an RGB image to HSV color space and Gray
2. Detect a specific color in an image
3. Detect an object based on a specific color

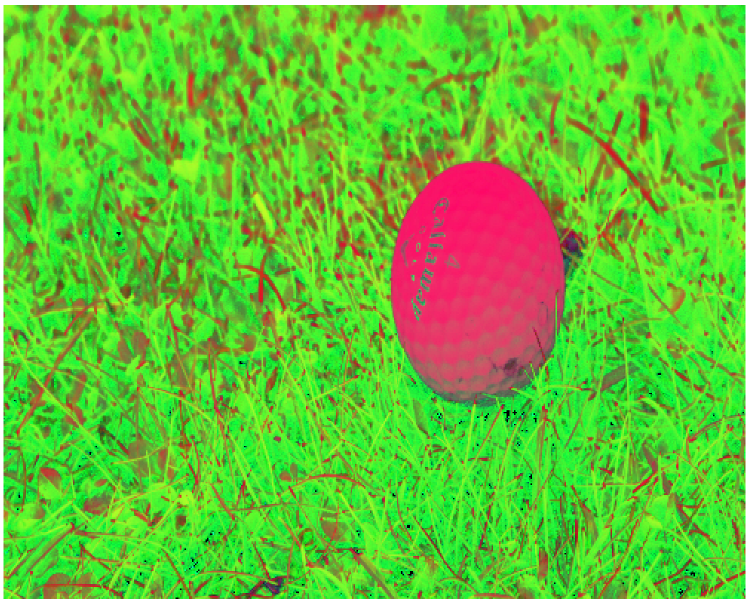
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

```
golfImage = cv2.resize(cv2.imread('/content/drive/MyDrive/Computer Vision And image Processing/Images/golf.jpg', cv2.IMREAD_COLO
```

```
cv2_imshow(golfImage)
```



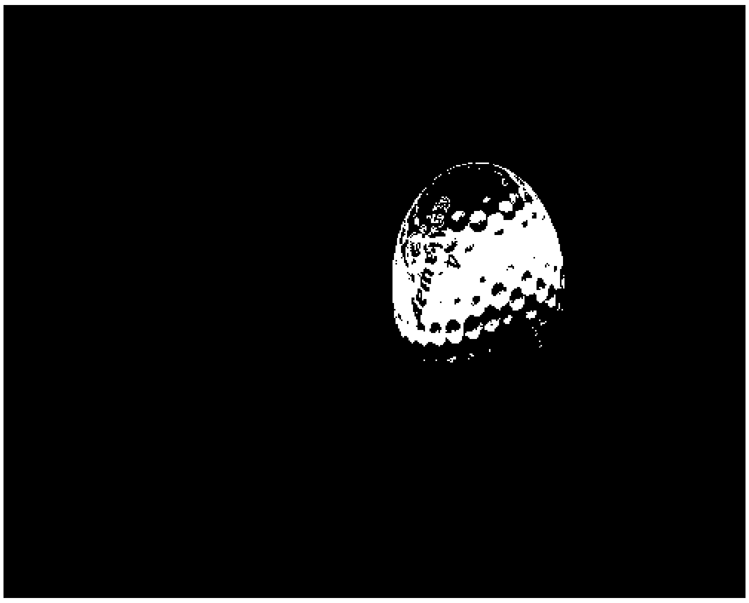
```
golfImageHSV = cv2.cvtColor(golfImage, cv2.COLOR_BGR2HSV)
cv2_imshow(golfImageHSV)
```

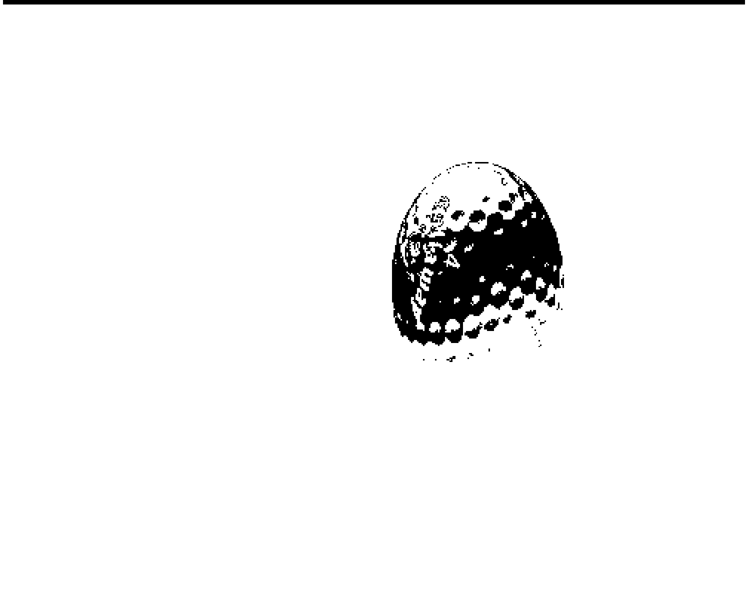


This code snippet will convert the BGR image 'golfImage' to the HSV color space and display the HSV image in a window titled 'HSV Image'

```
lower_red = np.array([95, 25, 99])
upper_red = np.array([110, 60, 255])

mask = cv2.inRange(golfImageHSV, lower_red, upper_red)
maskInv = cv2.bitwise_not(mask)
cv2_imshow(mask)
cv2_imshow(maskInv)
```





This code snippet is used to create a mask to segment out a specific range of red color in an image using OpenCV in Python. Here's a breakdown of the code:

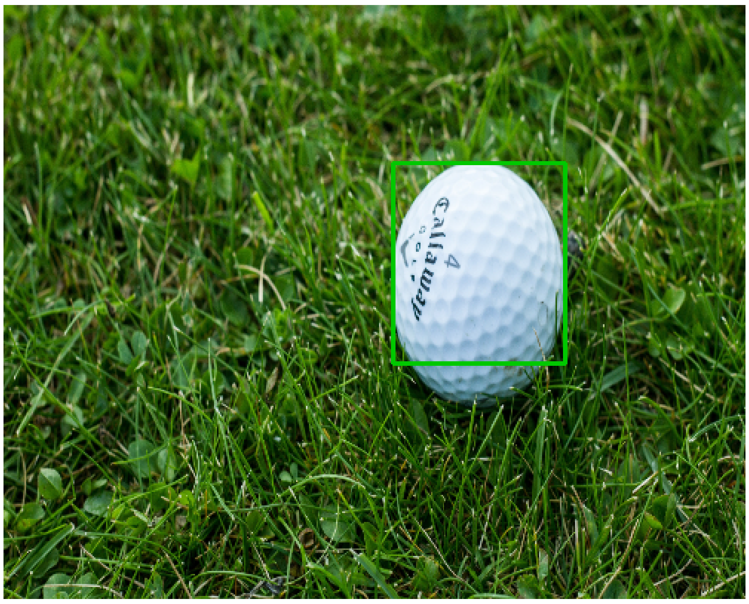
1. `lower_red = np.array([95, 25, 99])` and `upper_red = np.array([110, 60, 255])` : These lines define the lower and upper bounds of the red color range in the HSV color space. The lower bound represents the minimum values of (Hue, Saturation, Value) for the red color, and the upper bound represents the maximum values.
2. `mask = cv2.inRange(golfImageHSV, lower_red, upper_red)` : This line creates a binary mask by thresholding the input image `golfImageHSV` based on the specified lower and upper bounds of the red color range. Pixels within the specified range will be white (255) in the mask, and pixels outside the range will be black (0).
3. `maskInv = cv2.bitwise_not(mask)` : This line inverts the binary mask created in the previous step. It converts the white pixels to black and vice versa. This is useful when you want to extract the background instead of the specified color range.
4. `cv2_imshow(mask)` : This function displays the mask image, showing the segmented red color region as white and other regions as black.
5. `cv2_imshow(maskInv)` : This function displays the inverted mask image, showing the background region as white and the red color region as black.

Overall, this code segment helps in isolating the red color range in an image using HSV color space and visualizing the segmented regions through the generated masks.

```
boundingbox = cv2.boundingRect(mask)
if boundingbox is not None:
  print("Object detected \n")
  x, y, w, h = boundingbox
  cv2.rectangle(golfImage, (x, y), (x + w, y + h), (0, 200, 0), 2)
else:
  print("Object not detected")

cv2_imshow(golfImage)
```

```
Object detected
```



This code snippet is used to detect and draw a bounding box around an object in an image based on the mask created in the previous steps. Here's an explanation of the code:

1. `boundingbox = cv2.boundingRect(mask)` : This line calculates the bounding box that encloses the segmented object in the mask. The bounding box is represented by a tuple `(x, y, width, height)` where `(x, y)` are the coordinates of the top-left corner of the bounding box, and `width` and `height` are the dimensions of the box.
2. `if boundingbox is not None:` : This condition checks if a bounding box was successfully calculated. If a bounding box exists (i.e., object detected), the code inside the `if` block will be executed.
3. `print("Object detected \n")` : This line simply prints a message indicating that an object has been detected in the image.
4. `x, y, w, h = boundingbox` : This line unpacks the bounding box coordinates into individual variables for easier access.
5. `cv2.rectangle(golfImage, (x, y), (x + w, y + h), (0, 200, 0), 2)` : This line draws a green rectangle (specified by `(0, 200, 0)`) around the detected object in the original `golfImage` . The rectangle is defined by the top-left corner `(x, y)` and the bottom-right corner `(x + w, y + h)` of the bounding box. The `2` parameter specifies the thickness of the rectangle border.
6. `else:` : If no bounding box is found (i.e., object not detected), the code inside the `else` block will be executed.
7. `print("Object not detected")` : This line prints a message indicating that no object was detected in the image.
8. `cv2_imshow(golfImage)` : Finally, this function displays the original `golfImage` with the bounding box drawn around the detected object (if any).

Overall, this code segment helps in detecting objects based on the segmented mask and visualizing the detection results by drawing bounding boxes around the objects in the original image.