Image processing using OpenCV

VORTEX ACADEMY

THE ART OF THINKING

Eng. Mai Allam

What will we discuss today?

- Introduction to Images
- Reading Images & Video
- Resizing and Rescaling Frames
- Drawing Shapes & Putting Text
- Image Transformations
- BITWISE operations & Masking
- Assignments & Questions

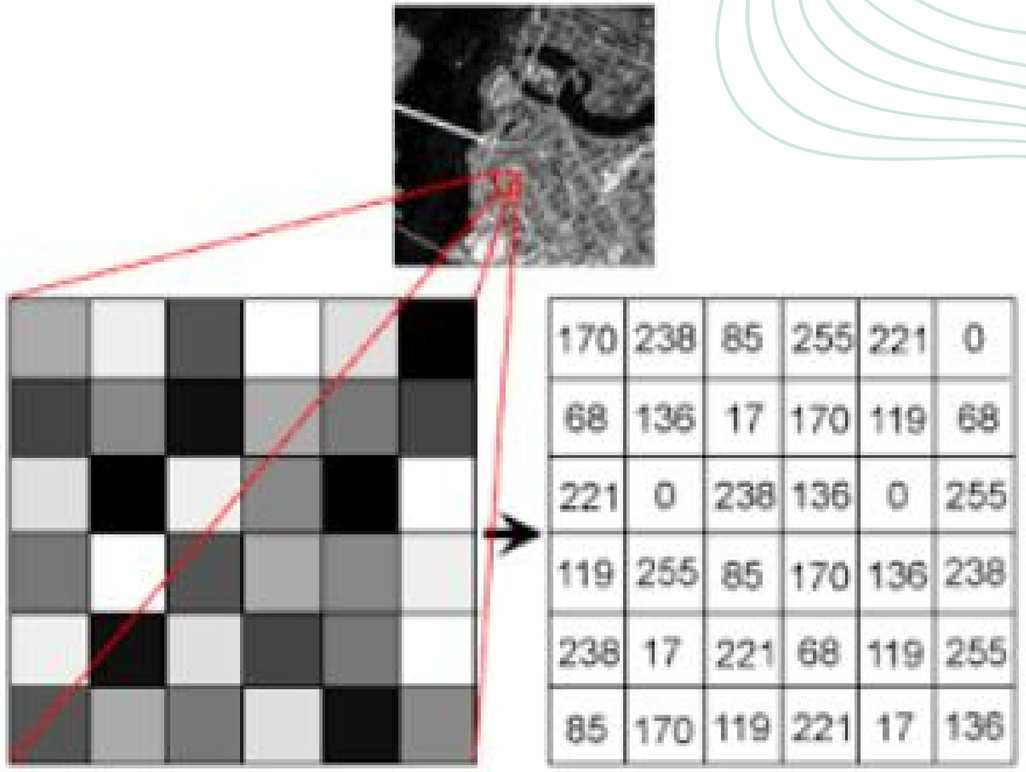
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Introduction to Images

Image Representation [Grayscale]

Every number represents the pixel intensity at that particular location. In the above image, I have shown the pixel values for a grayscale image where every pixel contains only one value. the intensity of the black color at that location.



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

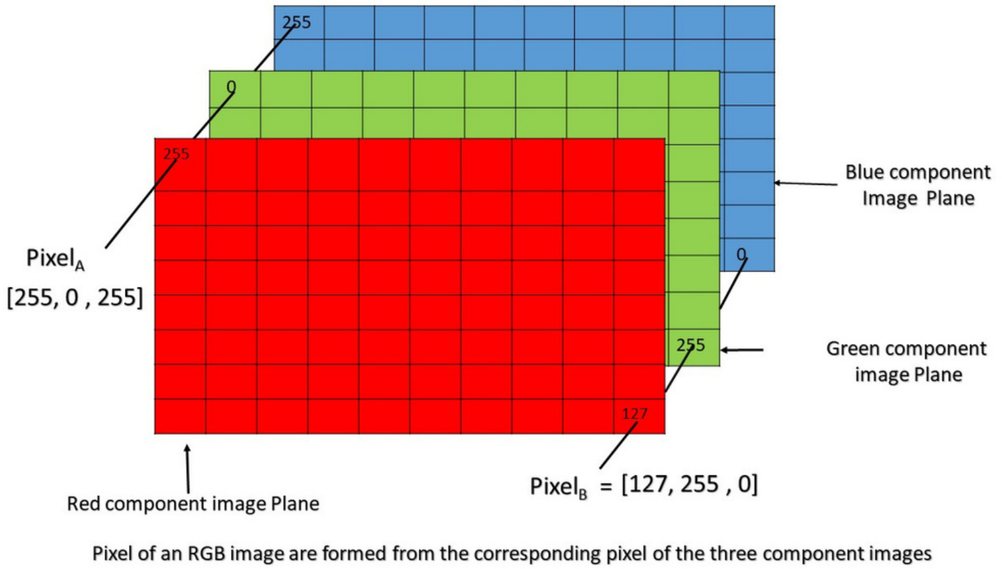
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Introduction to Images

Image Representation [RGB]

An RGB image can be viewed as three different images(a red scale image, a green scale image, and a blue scale image) stacked on top of each other, and when fed into the red, green, and blue inputs of a color monitor, it produces a color image on the screen.



Pixel of an RGB image are formed from the corresponding pixel of the three component images

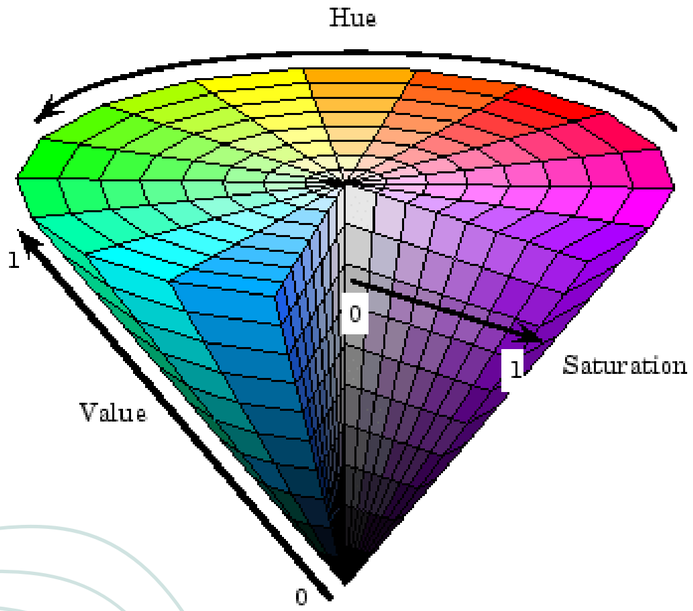
RGB[: , : , 1] represents the Red color plane of the RGB image
RGB[: , : , 2] represents the Green color plane of the RGB image
RGB[: , : , 3] represents the Blue color plane of the RGB image

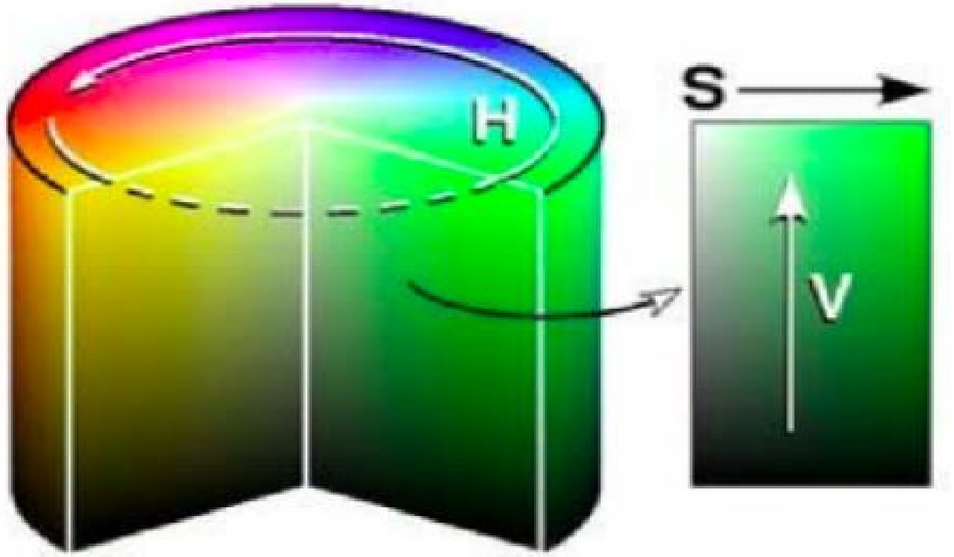
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Introduction to Images

Other Representation [BGR, HSV,..]



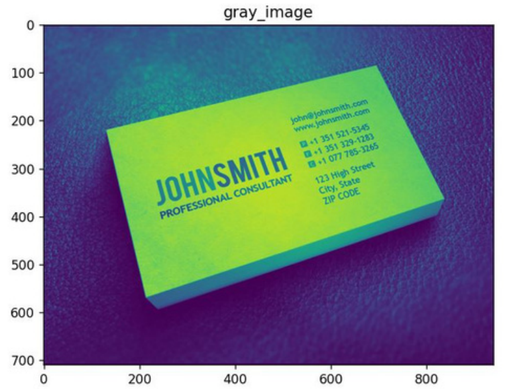


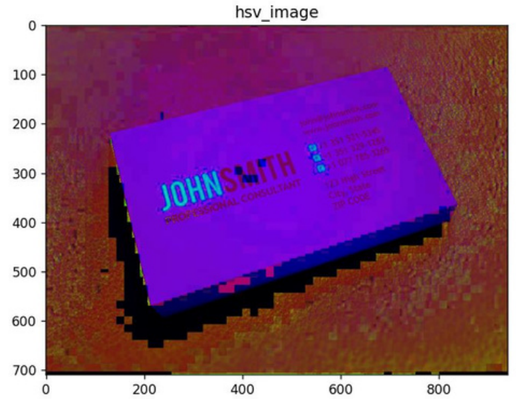
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Reading Images

```python
#import the required libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2


image = cv2.imread('D:/vortex/MATE 2023/training phase/ms2/opencv/opencv/jhonsmith.jpg')

#converting image to Gray scale
gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

#plotting the grayscale image
plt.imshow(gray_image)
plt.title("gray_image")
plt.show()

#converting image to HSV format
hsv_image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)

#plotting the HSV image
plt.imshow(hsv_image)
plt.title("hsv_image")
plt.show()

```



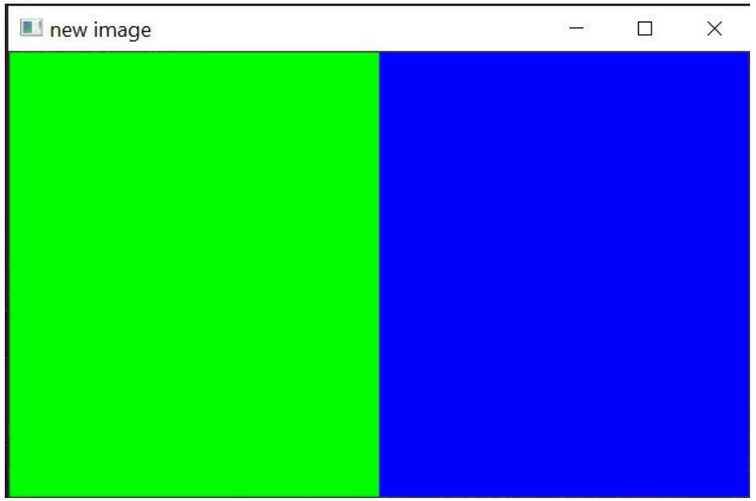


VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Creating Images

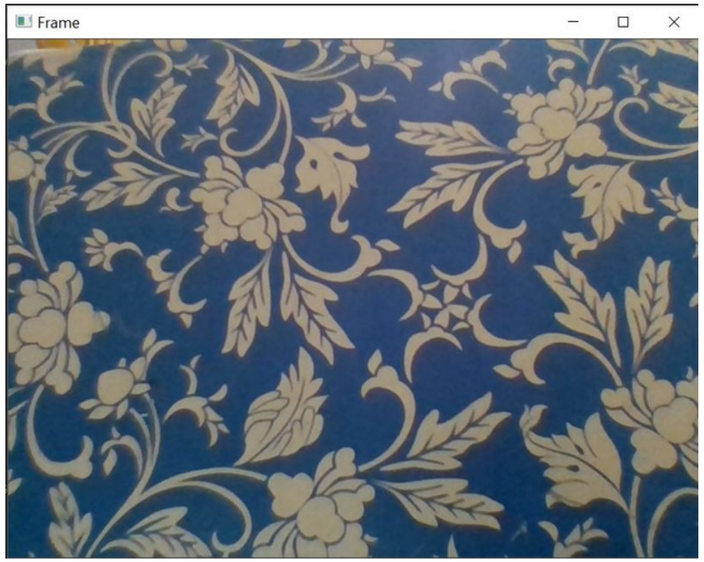
```python
import cv2

import numpy as np
nw_img = np.ones((300,500,3), dtype=np.uint8)
nw_img[:,0:250] = (0,255,0)
nw_img[:,250:500] = (255,0,0)
cv2.imshow("new image", nw_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

Reading Video

```python
import cv2
# Create an object to read from camera
video = cv2.VideoCapture(0)
# check if camera is opened previously or not
if (video.isOpened() == False):
    print("Error reading video file")

# set resolutions. so, convert them from float to integer.
frame_width = int(video.get(3))
frame_height = int(video.get(4))
size = (frame_width, frame_height)

# Below VideoWriter object will create a frame of above defined The output is stored in 'filename.avi' file.
result = cv2.VideoWriter('filename.avi',cv2.VideoWriter_fourcc(*'MJPG'),10, size)

while(True):
    ret, frame = video.read()

    if ret == True:
        # Write the frame into the file 'filename.avi'
        result.write(frame)
        # Display the frame saved in the file
        cv2.imshow('Frame', frame)
        # Press S on keyboard to stop the process
        if cv2.waitKey(1) & 0xFF == ord('s'):
            break
    # Break the loop
    else:
        break

# When everything done, release the video capture and video write objects
video.release()
result.release()
# Closes all the frames
cv2.destroyAllWindows()
```

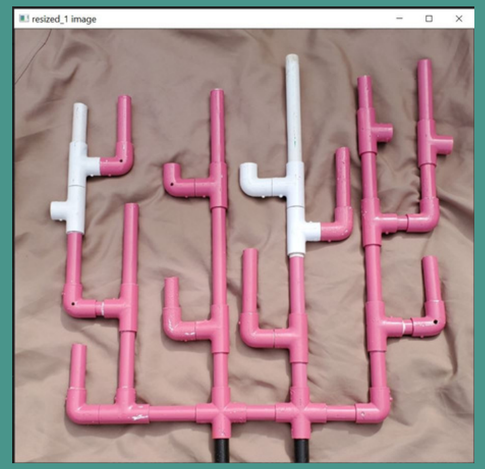


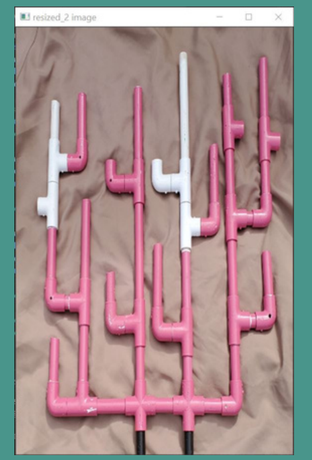
VORTEX ACADEMY
THE ART OF THINKING

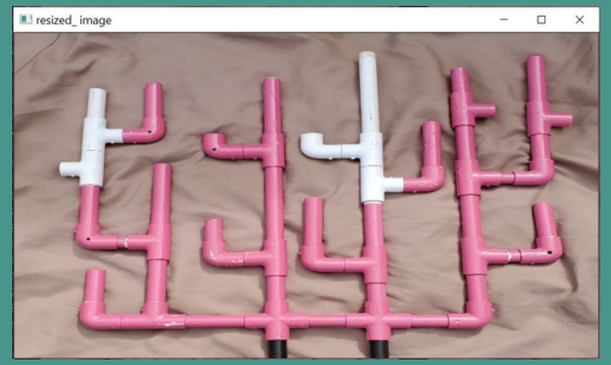
Eng. Mai Allam

Resizing and Rescaling Frames

```python
import cv2

img = cv2.imread('D:/vortex/MATE 2023/training phase/ms2/opencv/opencv/coral3.jpg')

scale_percent = 60 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)

# resize image
resized_1 = cv2.resize(img, dim)
cv2.imshow("resized_1 image", resized_1)

width = 440
dim = (width, height) # keep original height
resized_2 = cv2.resize(img, dim)

cv2.imshow("resized_2 image", resized_2)

width = int(img.shape[1] * scale_percent / 100)
height = 400
dim = (width, height) # keep original width
resized_3 = cv2.resize(img, dim)

cv2.imshow("resized_ image", resized_3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```





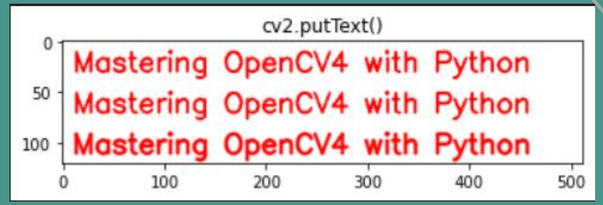


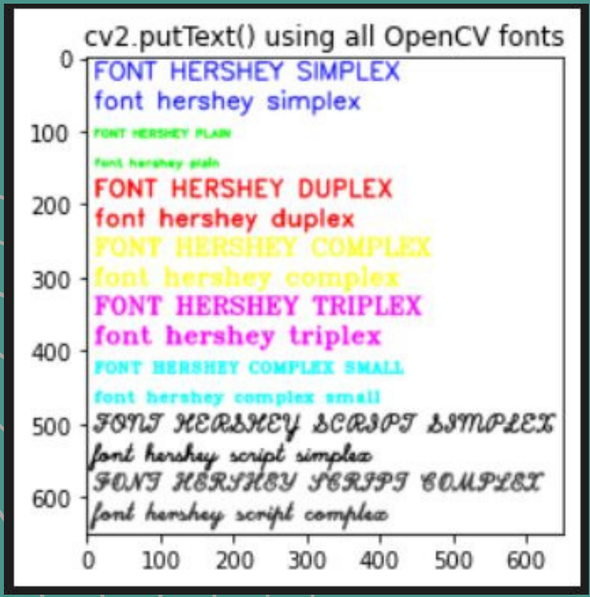
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Putting Text

```
# We draw some text in the image:
cv2.putText(image, 'Mastering OpenCV4 with Python', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, colors['red'], 2, cv2.LINE_4)
cv2.putText(image, 'Mastering OpenCV4 with Python', (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 0.9, colors['red'], 2, cv2.LINE_8)
cv2.putText(image, 'Mastering OpenCV4 with Python', (10, 110), cv2.FONT_HERSHEY_SIMPLEX, 0.9, colors['red'], 2, cv2.LINE_AA)
```






```
# Dictionary containing some strings to output:
fonts = {0: "FONT HERSHEY SIMPLEX", 1: "FONT HERSHEY PLAIN", 2: "FONT HERSHEY DUPLEX", 3: "FONT HERSHEY COMPLEX",
         4: "FONT HERSHEY TRIPLEX", 5: "FONT HERSHEY COMPLEX SMALL ", 6: "FONT HERSHEY SCRIPT SIMPLEX",
         7: "FONT HERSHEY SCRIPT COMPLEX"}
```

VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Drawing Shapes with mouse events

VORTEX ACADEMY
THE ART OF THINKING

```python
# mouse callback function
def draw_circle(event, x, y, flags, param):
    """Mouse callback function"""

    global circles
    if event == cv2.EVENT_LBUTTONDBLCLK:
        # Add the circle with coordinates x,y
        print("event: EVENT_LBUTTONDBLCLK")
        circles.append((x, y))
    if event == cv2.EVENT_RBUTTONDBLCLK:
        # Delete all circles (clean the screen)
        print("event: EVENT_RBUTTONDBLCLK")
        circles[:] = []
    elif event == cv2.EVENT_RBUTTONDOWN:
        # Delete last added circle
        print("event: EVENT_RBUTTONDOWN")
        try:
            circles.pop()
        except (IndexError):
            print("no circles to delete")
```

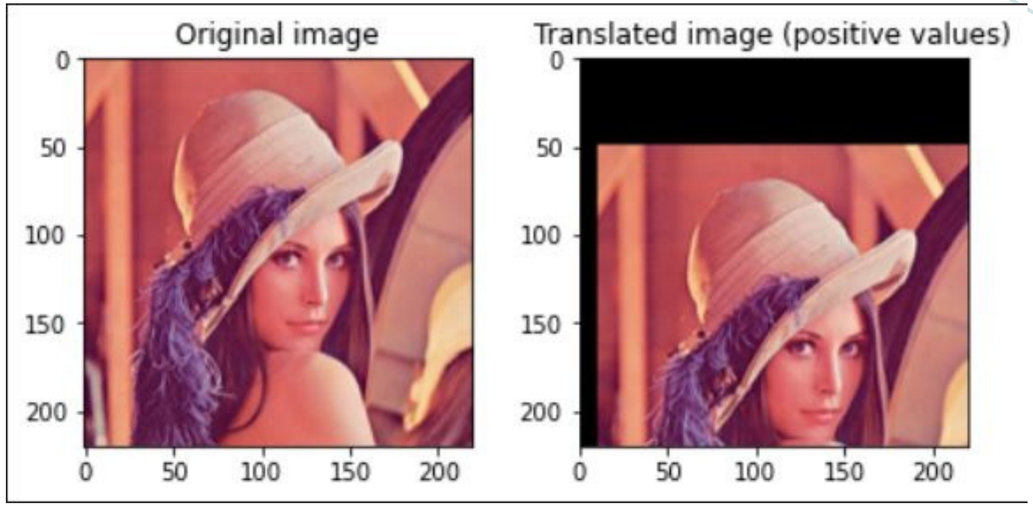
```python
# Structure to hold the created circles:
circles = []
# We create the canvas to draw: 600 x 600 pixels, 3 channels, uint8 (8-bit unsigned integers)
# We set the background to black using np.zeros():
image = np.zeros((600, 600, 3), dtype="uint8")
# We create a named window where the mouse callback will be established:
cv2.namedWindow('Image mouse')
# We set the mouse callback function to 'draw_circle':
cv2.setMouseCallback('Image mouse', draw_circle)
# We draw the menu:
draw_text()
# We get a copy with only the text printed in it:
clone = image.copy()

while True:
    # We 'reset' the image (to get only the image with the printed text):
    image = clone.copy()
    # We draw now only the current circles:
    for pos in circles:
        # We print the circle (filled) with a  fixed radius (30):
        cv2.circle(image, pos, 30, colors['blue'], -1)
    # Show image 'Image mouse':
    cv2.imshow('Image mouse', image)
    key = cv2.waitKey(1)
    # Continue until 'q' is pressed:
    if key == ord('q') or key == ord("Q"):
        break
# Destroy all generated windows:
cv2.destroyAllWindows()
```

Eng. Mai Allam

Geometric transformations of images

Translation

```python
# Import required packages:
import cv2
import matplotlib.pyplot as plt
import numpy as np


def show_with_matplotlib(color_img, title, pos):
    """Shows an image using matplotlib capabilities"""

    # Convert BGR image to RGB
    img_RGB = color_img[:, :, ::-1]
    plt.subplot(1, 2, pos)
    plt.imshow(img_RGB)
    plt.title(title)
    plt.tight_layout(w_pad=2)
    #plt.axis('off')


# Read the input image:
image = cv2.imread('assets\images\lenna.png')

# Show loaded image:
show_with_matplotlib(image, 'Original image', 1)
height, width = image.shape[:2]

M = np.float32([[1, 0, 10], [0, 1, 50]])

# Once this transformation Matrix is created, we can pass it to the function cv2.warpAffine():
dst_image = cv2.warpAffine(image, M, (width, height))

# Show the translated image:
show_with_matplotlib(dst_image, 'Translated image (positive values)', 2)
```



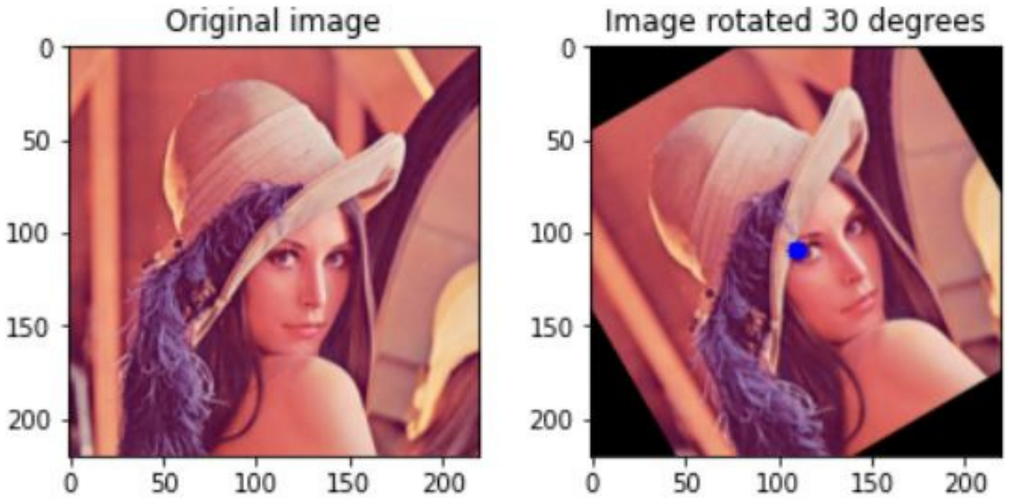
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Geometric transformations of images

Rotating

```python
# Import required packages:
import cv2
import matplotlib.pyplot as plt
import numpy as np


def show_with_matplotlib(color_img, title, pos):
    """Shows an image using matplotlib capabilities"""

    # Convert BGR image to RGB
    img_RGB = color_img[:, :, ::-1]
    plt.subplot(1, 2, pos)
    plt.imshow(img_RGB)
    plt.title(title)
    plt.tight_layout(w_pad=2)
    #plt.axis('off')


# Read the input image:
image = cv2.imread('assets\images\lenna.png')
height, width = image.shape[:2]
# Show loaded image:
show_with_matplotlib(image, 'Original image', 1)

M = cv2.getRotationMatrix2D((width / 2.0, height / 2.0), 30, 1)
dst_image = cv2.warpAffine(image, M, (width, height))

# Show the center of rotation and the rotated image:
cv2.circle(dst_image, (round(width / 2.0), round(height / 2.0)), 5, (255, 0, 0), -1)
show_with_matplotlib(dst_image, 'Image rotated 30 degrees', 2)
```



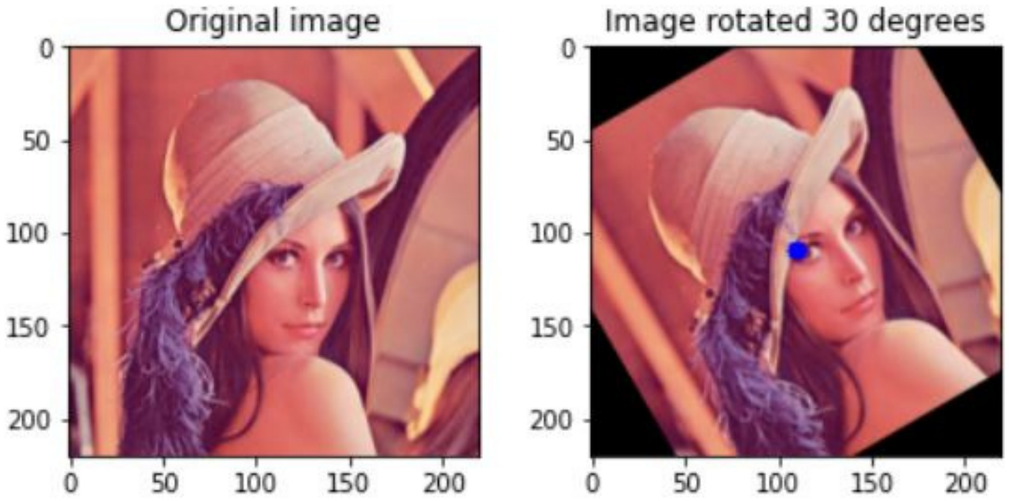
VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Geometric transformations of images

Perespective Transform

```python
# Import required packages:
import cv2
import matplotlib.pyplot as plt
import numpy as np


def show_with_matplotlib(color_img, title, pos):
    """Shows an image using matplotlib capabilities"""

    # Convert BGR image to RGB
    img_RGB = color_img[:, :, ::-1]
    plt.subplot(1, 2, pos)
    plt.imshow(img_RGB)
    plt.title(title)
    plt.tight_layout(w_pad=2)
    #plt.axis('off')


# Read the input image:
image = cv2.imread('assets\images\lenna.png')
height, width = image.shape[:2]
# Show loaded image:
show_with_matplotlib(image, 'Original image', 1)

M = cv2.getRotationMatrix2D((width / 2.0, height / 2.0), 30, 1)
dst_image = cv2.warpAffine(image, M, (width, height))

# Show the center of rotation and the rotated image:
cv2.circle(dst_image, (round(width / 2.0), round(height / 2.0)), 5, (255, 0, 0), -1)
show_with_matplotlib(dst_image, 'Image rotated 30 degrees', 2)
```

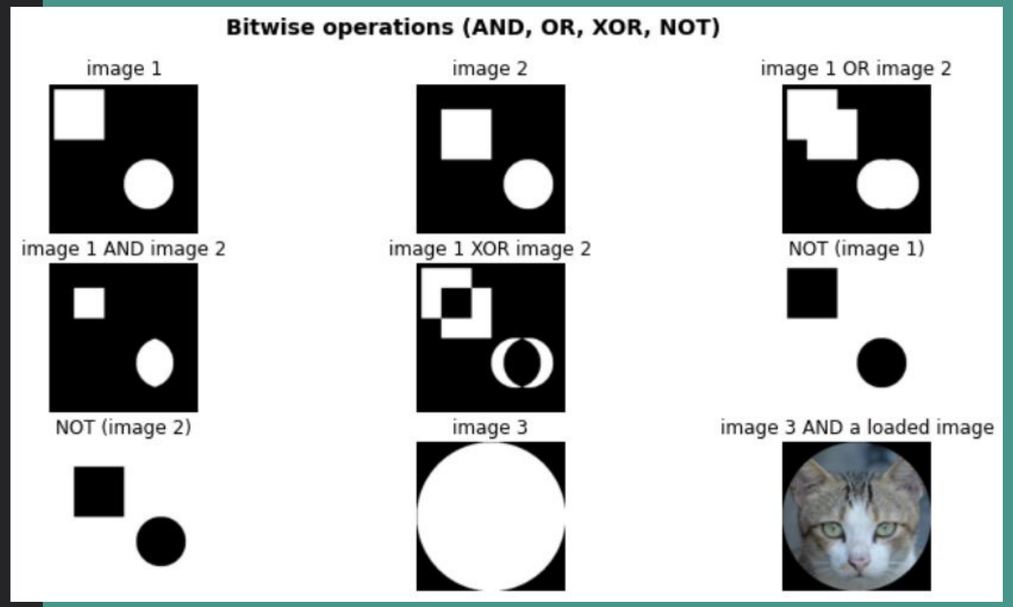


VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

BITWISE operations GrayScale Images

```python
img_1 = np.zeros((300, 300), dtype="uint8")# Create the first image:
cv2.rectangle(img_1, (10, 10), (110, 110), (255, 255, 255), -1)
cv2.circle(img_1, (200, 200), 50, (255, 255, 255), -1)

img_2 = np.zeros((300, 300), dtype="uint8")# Create the second image:
cv2.rectangle(img_2, (50, 50), (150, 150), (255, 255, 255), -1)
cv2.circle(img_2, (225, 200), 50, (255, 255, 255), -1)

bitwise_or = cv2.bitwise_or(img_1, img_2)# Bitwise OR
bitwise_and = cv2.bitwise_and(img_1, img_2)# Bitwise AND
bitwise_xor = cv2.bitwise_xor(img_1, img_2)# Bitwise XOR
bitwise_not_1 = cv2.bitwise_not(img_1)# Bitwise NOT
bitwise_not_2 = cv2.bitwise_not(img_2)# Bitwise NOT

# Display all the resulting images:
show_with_matplotlib(cv2.cvtColor(img_1, cv2.COLOR_GRAY2BGR), "image 1", 1)
show_with_matplotlib(cv2.cvtColor(img_2, cv2.COLOR_GRAY2BGR), "image 2", 2)
show_with_matplotlib(cv2.cvtColor(bitwise_or, cv2.COLOR_GRAY2BGR), "image 1 OR image 2", 3)
show_with_matplotlib(cv2.cvtColor(bitwise_and, cv2.COLOR_GRAY2BGR), "image 1 AND image 2", 4)
show_with_matplotlib(cv2.cvtColor(bitwise_xor, cv2.COLOR_GRAY2BGR), "image 1 XOR image 2", 5)
show_with_matplotlib(cv2.cvtColor(bitwise_not_1, cv2.COLOR_GRAY2BGR), "NOT (image 1)", 6)
show_with_matplotlib(cv2.cvtColor(bitwise_not_2, cv2.COLOR_GRAY2BGR), "NOT (image 2)", 7)

# See how these operations can be used to work with images:
image = cv2.imread('assets\images\cat.png')# Load the original image:

img_3 = np.zeros((300, 300), dtype="uint8")# Create the image to be used as mask:
cv2.circle(img_3, (150, 150), 150, (255, 255, 255), -1)

bitwise_and_example = cv2.bitwise_and(image, image, mask=img_3)# Bitwise AND using the img_3 as mask:

# Show these two images:
show_with_matplotlib(cv2.cvtColor(img_3, cv2.COLOR_GRAY2BGR), "image 3", 8)
show_with_matplotlib(bitwise_and_example, "image 3 AND a loaded image", 9)
```

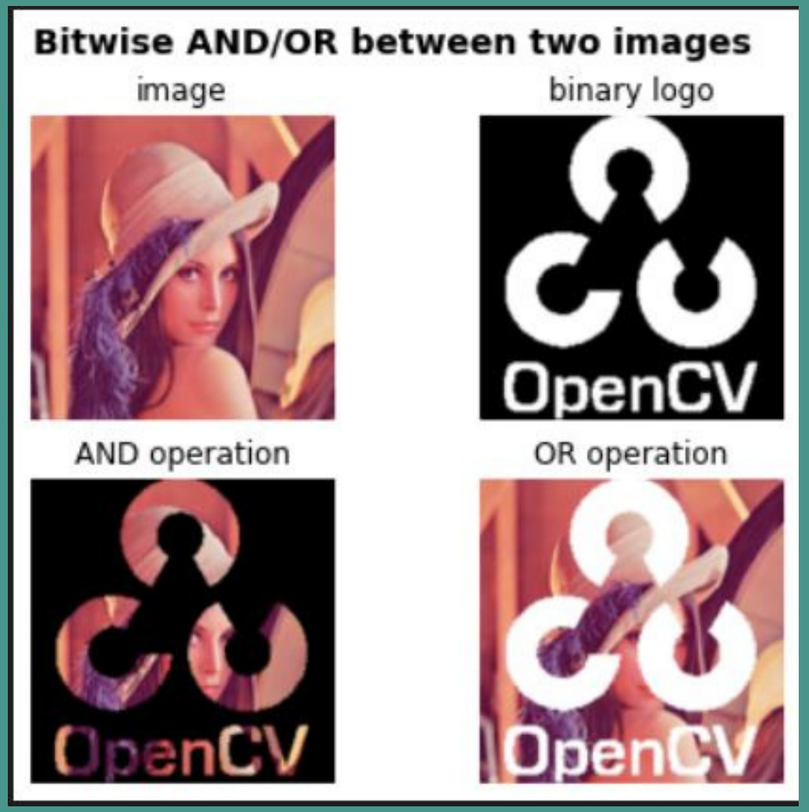


VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

BITWISE operations Colored Images

```
# Create the dimensions of the figure and set title:
plt.figure(figsize=(6, 5))
plt.suptitle("Bitwise AND/OR between two images", fontsize=14, fontweight='bold')

# Load the original image (250x250):
image = cv2.imread('assets\images\lenna.png')
image = cv2.resize(image, (250, 250))
# Load the binary image (but as a GBR color image - with 3 channels) (250x250):
binary_image = cv2.imread('assets\images\opencv_binary.png')

# Bitwise AND
bitwise_and = cv2.bitwise_and(image, binary_image)

# Bitwise OR
bitwise_or = cv2.bitwise_or(image, binary_image)

# Display all the resulting images:
show_with_matplotlib(image, "image", 1)
show_with_matplotlib(binary_image, "binary logo", 2)
show_with_matplotlib(bitwise_and, "AND operation", 3)
show_with_matplotlib(bitwise_or, "OR operation", 4)

# Show the Figure:
plt.show()
```

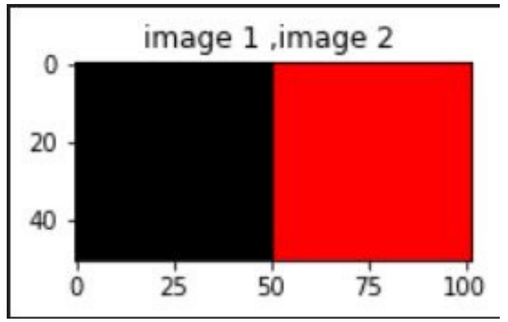


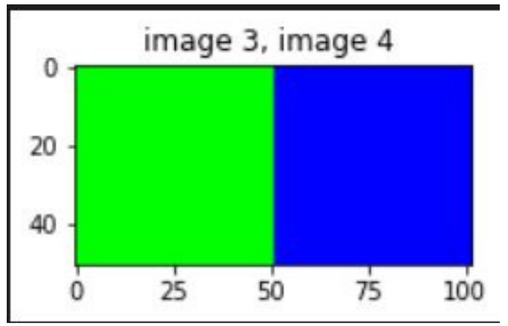
VORTEX ACADEMY
THE ART OF THINKING

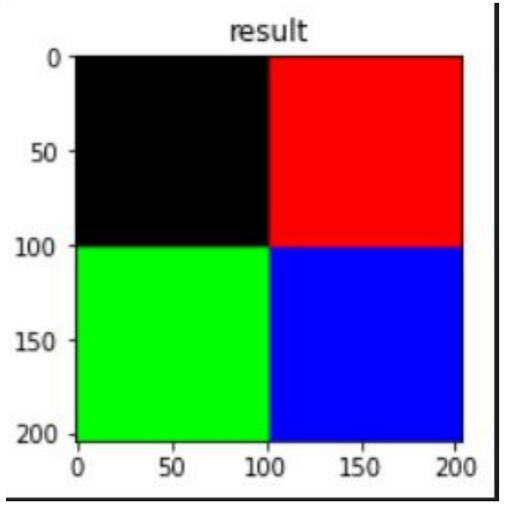
Eng. Mai Allam

Assignments Task 1

```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

def show_with_matplotlib(color_img, title):
    plt.figure(figsize=(3, 3))
    img_RGB = color_img[:, :, ::-1]
    plt.imshow(color_img)
    plt.title(title)
    plt.show()

image_1 = np.full((51, 51, 3), (0, 0, 0), dtype= np.uint8)
image_2 = np.full((51, 51, 3), (255, 0, 0), dtype= np.uint8)
image_3 = np.full((51, 51, 3), (0, 255, 0), dtype= np.uint8)
image_4 = np.full((51, 51, 3), (0, 0, 255), dtype= np.uint8)

vert1 = np.concatenate((image_1, image_2), axis= 1)
vert2 = np.concatenate((image_3, image_4), axis= 1)

result = np.concatenate((vert1, vert2), axis= 0)

width, height = result.shape[:2]

result = cv2.resize(result, (width * 2, height * 2), interpolation=cv2.INTER_LINEAR)

show_with_matplotlib(vert1, "image 1 ,image 2 ")
show_with_matplotlib(vert2, "image 3, image 4 ")
show_with_matplotlib(result, "result")
```







VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Assignments

Task 2

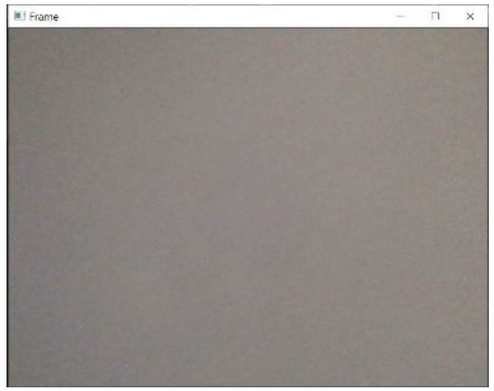
VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

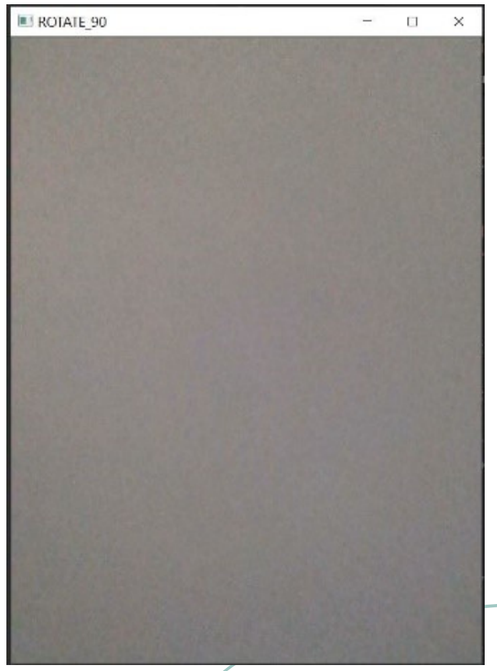
```python
import cv2
video = cv2.VideoCapture(0)# Create an object to read from camera

if (video.isOpened() == False):# check if camera is opened previously or not
    print("Error reading video file")

frame_width = int(video.get(3))# set resolutions. so, convert them from float to integer.
frame_height = int(video.get(4))
size_video = (frame_width, frame_height)

# Below VideoWriter object will create a frame of above defined The output is stored in 'filename.avi' file.
result = cv2.VideoWriter('savedVideo.avi',cv2.VideoWriter_fourcc(*'MJPG'),10, size_video)

switch, count = "z", 0
while(True):
    ret, frame = video.read()

    frame_rotated = cv2.rotate(frame,cv2.ROTATE_90_COUNTERCLOCKWISE)
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
    frame_hsv = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)

    if ret == True:
            key = cv2.waitKey(1)

            if switch == "z": cv2.imshow('Frame', frame)

            if key == ord("q") or key == ord('Q'): break

            if key == ord("r") or key == ord("R"):
                switch = "r"
                cv2.destroyAllWindows()

            if key == ord("c") or key == ord("C"):
                switch = "c"
            if key == ord("S") or key == ord("s"):
                switch = "s"
                cv2.destroyAllWindows()
            if key == ord("h") or key == ord("H"):
                switch = "h"
                cv2.destroyAllWindows()
            if key == ord("g") or key == ord("G"):
                switch = "g"
                cv2.destroyAllWindows()
            if key == ord("z") or key == ord("Z"):
                switch = "z"
                cv2.destroyAllWindows()
            if key == ord("x") or key == ord("X"):
                switch = "x"
                cv2.destroyAllWindows()

            if switch == "r" :cv2.imshow("ROTATE_90", frame_rotated)

            if switch == "c":
                count += 1
                path = 'saved frames/'+ str(count) + '.png'
                cv2.imwrite(path, frame)
                switch = 'z'

            if switch == "h": cv2.imshow('frame_hsv', frame_hsv)

            if switch == "x":
                cv2.imshow('Frame', frame)
                cv2.imshow('frame_rotated', frame_rotated)
                cv2.imshow('frame_hsv', frame_hsv)
                cv2.imshow('Frame_gray', frame_gray)

            if switch == "g": cv2.imshow('Frame_gray', frame_gray)

            if switch == "s":
                result.write(frame) # Write the frame into the file 'filename.avi'
                cv2.imshow('Frame', frame)

# When everything done, release the video capture and video write objects
video.release()
result.release()
cv2.destroyAllWindows() # Closes all the frames
```

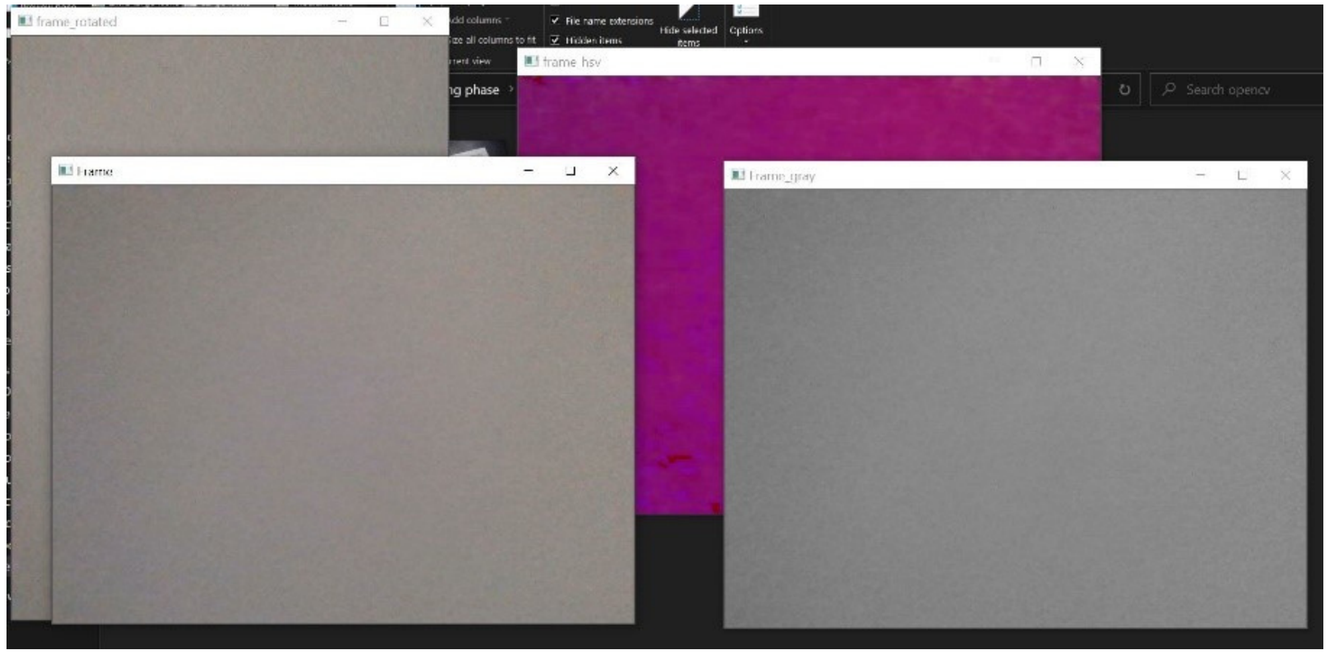
VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

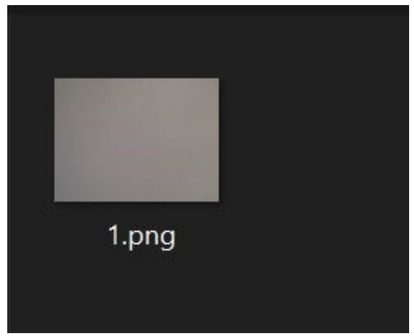
Assignments

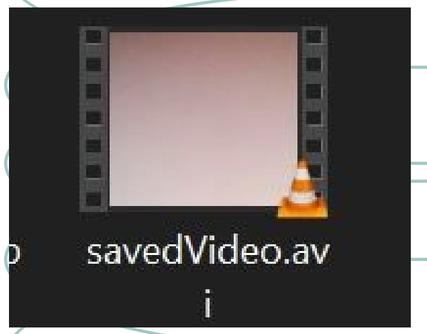
Task 2











Assignments

Task 3

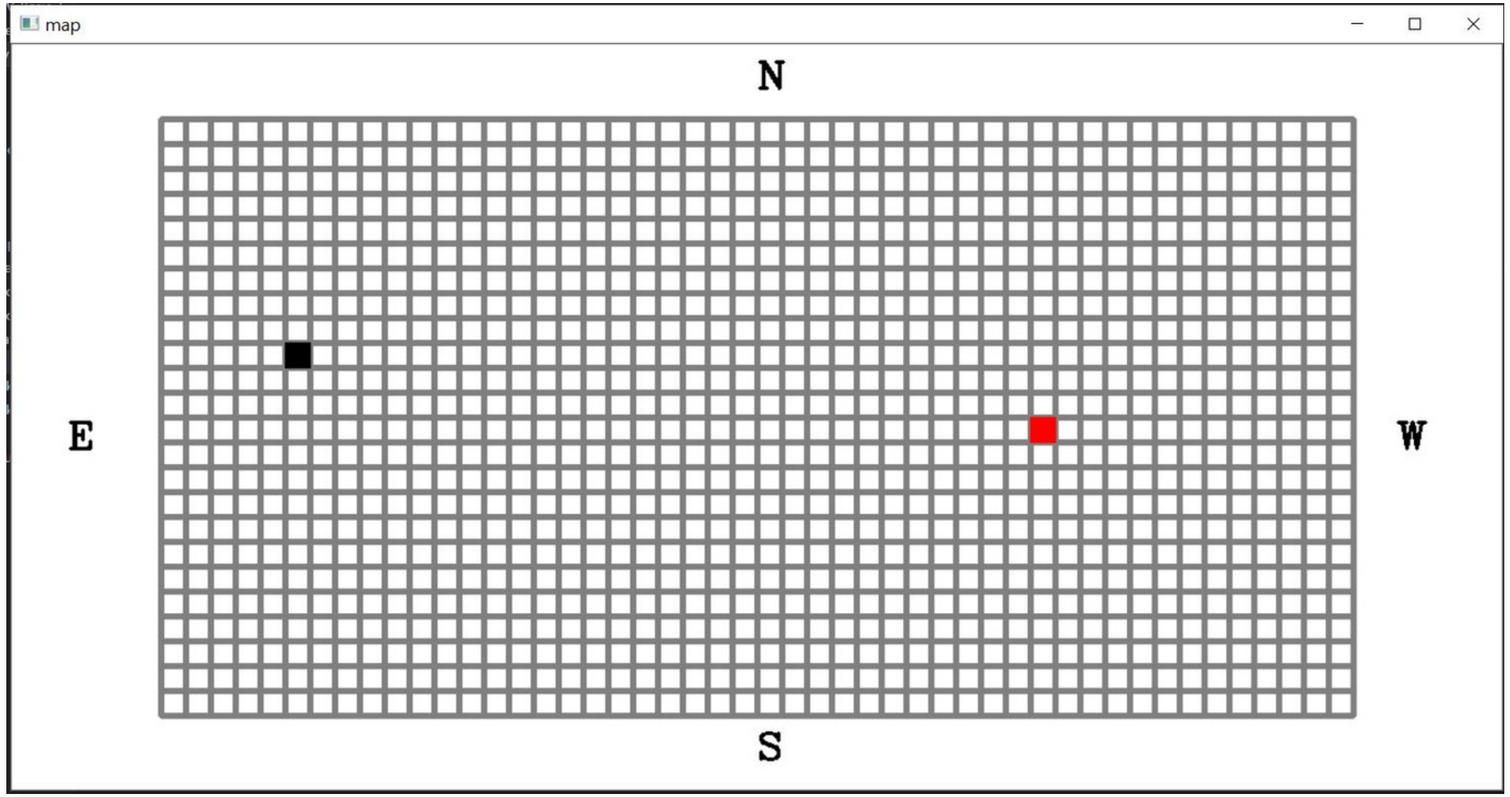
VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

```python
import cv2
import numpy as np

font=cv2.FONT_HERSHEY_COMPLEX

class FloatMap:
    def __init__(self) -> None:
        self.map = self.draw_map()
        self.cells_locations = []
        self.get_cells_locations()
        cv2.rectangle(self.map, *self.cells_locations[9][5], (0, 0, 0), -1)

    def draw_map(self):
        grid_image = np.ones((600, 1200, 3), dtype=np.uint8)*255

        for j in range(60, 560, 20):
            cv2.line(grid_image, (120, j), (1080, j), (128, 128, 128), 3)

        for i in range(120, 1100, 20):
            cv2.line(grid_image, (i, 60), (i,540), (128, 128, 128), 3)

        cv2.putText(grid_image,"W",(1115,325),font,1,0,2)
        cv2.putText(grid_image,"E",(45,325),font,1,0,2)
        cv2.putText(grid_image,"N",(600,35),font,1,0,2)
        cv2.putText(grid_image,"S",(600,575),font,1,0,2)

        return grid_image

    def get_cells_locations(self):
        for i in range(60, 540, 20):
            current_locations = []
            for j in range(120, 1080, 20):
                current_locations.append([(j, i), (j+20, i+20)])
            self.cells_locations.append(current_locations)

```

```python
class FloatDestination:
    def __init__(self, speed, direction, time) -> None:
        self.distance = speed*time*60*60
        self.y = self.distance*np.cos(direction*np.pi/180.)
        self.x = self.distance*np.sin(direction*np.pi/180.)

        self.y_direction = "north" if self.y > 0 else "south"
        self.x_direction = "east" if self.x > 0 else "west"

        self.distance = np.round(self.distance/1000, 3)
        self.y = np.round(abs(self.y)/1000, 2)
        self.x = np.round(abs(self.x)/1000, 2)


        self.y_delta = int(np.round(self.y/2))
        self.x_delta = int(np.round(self.x/2))

        self.new_y = 9 - self.y_delta if self.y_direction == "north" else 9 + self.y_delta
        self.new_x = 5 + self.x_delta if self.x_direction == "east" else 5 - self.x_delta

    def get_destination(self):
        return self.new_x, self.new_y


float_map = FloatMap()
float_map_image = float_map.map

speed = float(input("Enter the speed: "))
angle = float(input("Enter the angle: "))
time = float(input("Enter the time: "))

float_map_cells = float_map.cells_locations
x, y = FloatDestination(speed, angle, time).get_destination()

cv2.rectangle(float_map_image, *float_map_cells[y][x], (0, 0, 255), -1)

cv2.imshow("map", float_map_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Assignments Task 3

```
PS D:\vortex\MATE 2023\training phase\ms2\OpenCV2> python -u "d:\vortex\MATE 2023\training phase\ms2\OpenCV2\float.py"
Enter the speed: 0.117
Enter the angle: 96
Enter the time: 144
```



VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Assignments Task 4

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

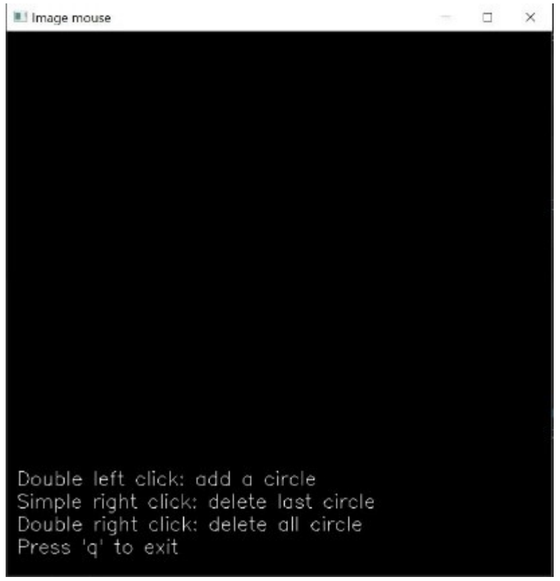
```python
import cv2
import numpy as np

colors = {'blue': (255, 0, 0), 'green': (0, 255, 0), 'red': (0, 0, 255), 'yellow': (0, 255, 255),
          'magenta': (255, 0, 255), 'cyan': (255, 255, 0), 'white': (255, 255, 255), 'black': (0, 0, 0),
          'gray': (125, 125, 125), 'rand': np.random.randint(0, high=256, size=(3,)).tolist(),
          'dark_gray': (50, 50, 50), 'light_gray': (220, 220, 220)}

def draw_text():
    menu_pos = (10, 500)
    menu_pos2 = (10, 525)
    menu_pos3 = (10, 550)
    menu_pos4 = (10, 575)

    cv2.putText(image, 'Double left click: add a circle', menu_pos, cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255))
    cv2.putText(image, 'Simple right click: delete last circle', menu_pos2, cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255))
    cv2.putText(image, 'Double right click: delete all circle', menu_pos3, cv2.FONT_HERSHEY_SIMPLEX, 0.7,(255, 255, 255))
    cv2.putText(image, 'Press \'q\' to exit', menu_pos4, cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255))

# mouse callback function
def draw_circle(event, x, y, flags, param):
    """Mouse callback function"""

    global circles
    if event == cv2.EVENT_LBUTTONDBLCLK:
        circles.append((x, y))
    if event == cv2.EVENT_RBUTTONDBLCLK:
        circles[:] = []
    elif event == cv2.EVENT_RBUTTONDOWN:
        try:
            circles.pop()
        except (IndexError):
            print("no circles to delete")
    # if event == cv2.EVENT_MOUSEMOVE:
    #     print("event: EVENT_MOUSEMOVE")
    # if event == cv2.EVENT_LBUTTONUP:
    #     print("event: EVENT_LBUTTONUP")
    # if event == cv2.EVENT_LBUTTONDOWN:
    #     print("event: EVENT_LBUTTONDOWN")
```

```python
circles = []

image = np.zeros((600, 600, 3), dtype="uint8")

cv2.namedWindow('Image mouse')

cv2.setMouseCallback('Image mouse', draw_circle)

draw_text()

clone = image.copy()

while True:

    image = clone.copy()

    for pos in circles:
        cv2.circle(image, pos, 30, colors['blue'], -1)

    cv2.imshow('Image mouse', image)

    key = cv2.waitKey(1)
    if key == ord('q') or key == ord("Q"):
        break

cv2.destroyAllWindows()
```

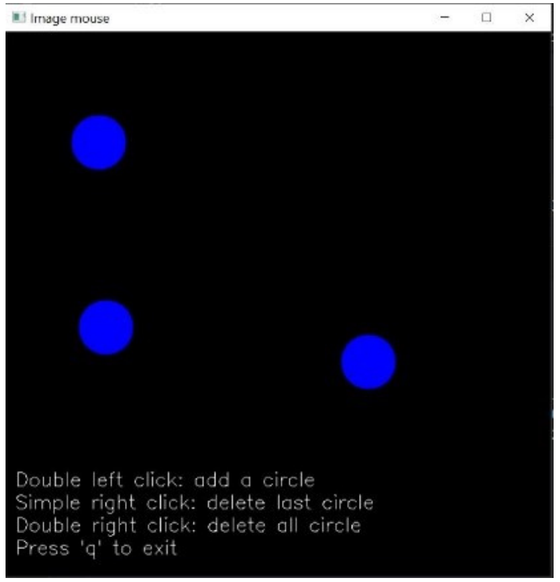
Assignments Task 4

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

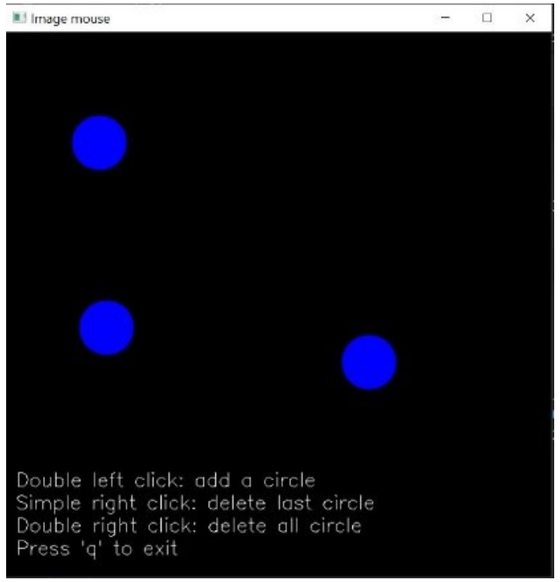
Output



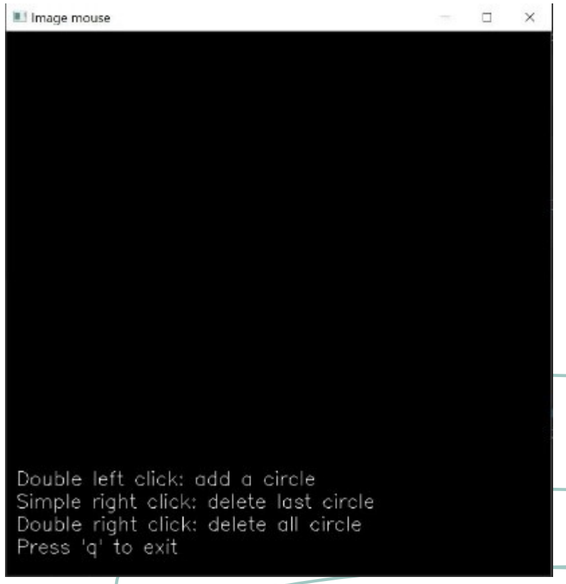
Double left click 3 times



Simple right click 1 time



double right click



Assignments

Task 5 [Order is a must]

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

```python
import numpy as np
import cv2

jhonsmith=cv2.imread("jhonsmith.jpg")
jhonsmith = cv2.resize(jhonsmith, (450,250))
h, w, c = jhonsmith.shape

circles=[]
counter=0
text=""

def draw_circle(event, x, y, flags, param):

    global circles, counter
    try:
        if event==cv2.EVENT_LBUTTONDOWN:
            circles.append((x, y))
            counter+=1
    except IndexError:
        print("only 4 points")

    if event == cv2.EVENT_RBUTTONDBLCLK: # Delete all circles
        circles[:] = []
        counter = 0

    elif event == cv2.EVENT_RBUTTONDOWN: # Delete last added circle
        try:
            circles.pop()
            counter-=1
        except (IndexError):
            print("no circles to delete")

dst_image = clone = jhonsmith.copy()

cv2.namedWindow('jhonsmith')
cv2.setMouseCallback('jhonsmith', draw_circle)
```

```python
while True:
    image = clone.copy()
    key=cv2.waitKey(1)
    for pos in circles:
        cv2.circle(image,pos,3,(0,0,255),-1)

    if counter==4:
        points1=np.float32([circles[0],circles[1],circles[2],circles[3]])
        points2=np.float32([[0,0],[w,0],[0,h],[w,h]])
        matrix=cv2.getPerspectiveTransform(points1,points2)
        dst_image=cv2.warpPerspective(jhonsmith,matrix,(w,h))

    if counter==0:
            text="choose upper left point"
            cv2.putText(image,text,(10,20),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),1)

    if counter==1:
        text="choose upper right point"
        cv2.putText(image,text,(10,20),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),1)

    if counter==2:
        text="choose lower left point"
        cv2.putText(image,text,(10,20),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),1)

    if counter==3:
        text="choose lower right point"
        cv2.putText(image,text,(10,20),cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,255),1)

    im_v = cv2.vconcat([image, dst_image])
    cv2.imshow("jhonsmith", im_v)

    if key== ord('Q') or key== ord('q'):
        break

cv2.destroyAllWindows()
```

Assignments

Task 5 [Order is a must]

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

Output

Simple left click 1 time

Simple left click 2 times

Simple left click 3 times

Simple left click 4 times

Simple right click 1 time

Simple right click 2 times

Simple left click 4 times

Double right click

Simple left click 4 times

Assignments

Task 5 [Order doesn't matter]

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

myPoints, c =[], 0

def mouse(event,x,y,flag,params):
    global c
    if event == cv2.EVENT_LBUTTONDOWN:
        myPoints.append((x,y))
        c+=1
    if event == cv2.EVENT_RBUTTONDBLCLK: # Delete all circles
        myPoints[:] = []
        c=0
    elif event == cv2.EVENT_RBUTTONDOWN: # Delete last added circle
        try:
            myPoints.pop()
            c-=1
        except (IndexError):
            print("no points to delete")

def reorder():
    x,y=0,0
    mx=1e9
    newPoints = []
    for i in myPoints:
        if i[0]+i[1]<mx:
            mx = i[0]+i[1]
            x,y = i[0],i[1]
    newPoints.append((x,y))

    mx=0
    for i in myPoints:
        if i[0]-i[1]>mx:
            mx = i[0]-i[1]
            x,y = i[0],i[1]
    newPoints.append((x,y))
    mx=1e9
    for i in myPoints:
        if i[0]-i[1]<mx:
            mx = i[0]-i[1]
            x,y = i[0],i[1]
    newPoints.append((x,y))

    mx=0
    for i in myPoints:
        if i[0]+i[1]>mx:
            mx=i[0]+i[1]
            x,y=i[0],i[1]
    newPoints.append((x,y))
    return newPoints

jhonsmith = cv2.imread("jhonsmith.jpg")
jhonsmith = cv2.resize(jhonsmith, (450,250))

width = jhonsmith.shape[1]
height = jhonsmith.shape[0]

cv2.namedWindow('jhonsmith')
cv2.setMouseCallback("jhonsmith", mouse)
warp = copy = jhonsmith.copy()
while True:
    copy = jhonsmith.copy()
    key = cv2.waitKey(1)
    for pos in myPoints: cv2.circle(copy, pos, 3 ,(0,0,255),-1)

    im_v = cv2.vconcat([copy, warp])
    cv2.imshow("jhonsmith", im_v)
    try:
        if c==4:
            myPoints = reorder()
            pts1 = np.float32(myPoints)
            pts2 = np.float32([[0, 0],[width, 0], [0, height],[width, height]])
            matrix = cv2.getPerspectiveTransform(pts1, pts2)
            warp = cv2.warpPerspective(jhonsmith, matrix, (width, height))
    except: pass
    if key ==ord('q') or key ==ord('Q'):
        cv2.destroyAllWindows()
        break
```

Assignments

Task 5 [Order doesn't matter]

Output

Simple left click 1 time

Simple left click 2 times

Simple left click 3 times

Simple left click 4 times

Double right click

VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Assignments Task 6 [Manual]

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

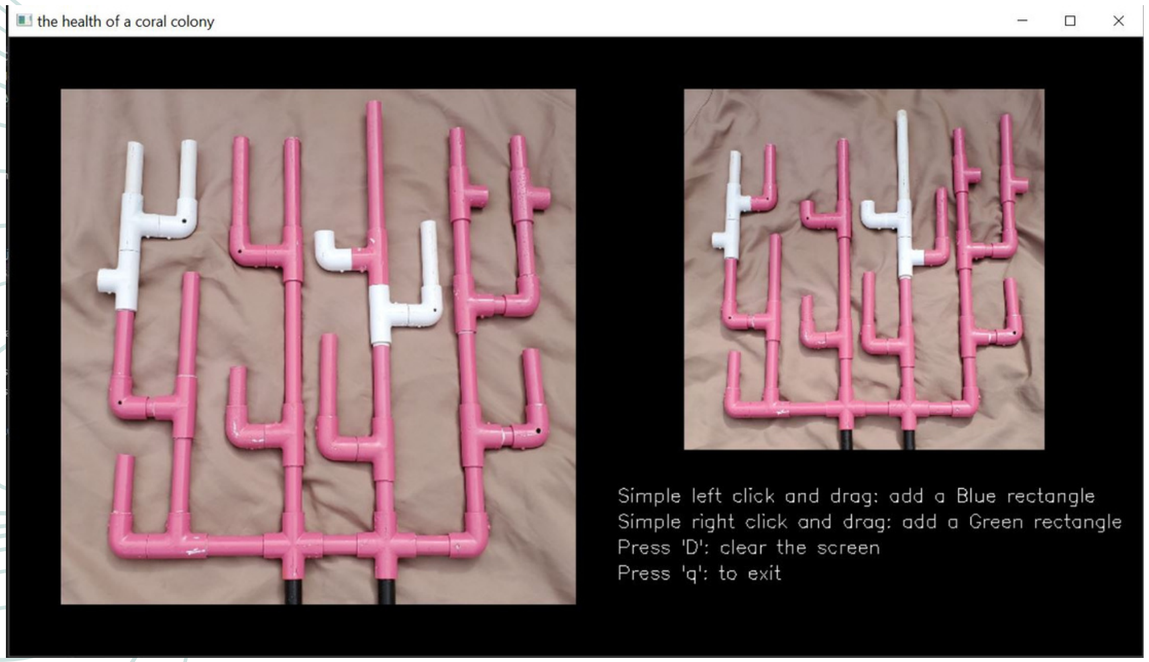
```python
import cv2
import numpy as np

original = cv2.imread("coral3.jpg")
original = cv2.resize(original, (350,350))

img_to_compare = cv2.imread("coral4.jpg")
img_to_compare = cv2.resize(img_to_compare, (500,500))

image = np.zeros_like(original)
image =cv2.resize(image,(1100,600))

img = img_to_compare[:, :, ]
top_corner = img[0:1000, 0:1000]
image[50:550, 50:550] = top_corner

img = original[:, :, ]
down_corner = img[0:1000, 0:1000]
image[50:400, 655:1005] = down_corner

def draw_text(image):
    menu_pos1 = (590, 450)
    menu_pos2 = (590, 475)
    menu_pos3 = (590, 500)
    menu_pos4 = (590, 525)

    cv2.putText(image, 'Simple left click and drag: add a Blue rectangle', menu_pos1, cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255))
    cv2.putText(image, 'Simple right click and drag: add a Green rectangle', menu_pos2, cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255))
    cv2.putText(image, 'Press \'D\': clear the screen', menu_pos3, cv2.FONT_HERSHEY_SIMPLEX, 0.6,(255, 255, 255))
    cv2.putText(image, 'Press \'q\': to exit', menu_pos4, cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255))


draw = False
ix, iy = -1, -1
```

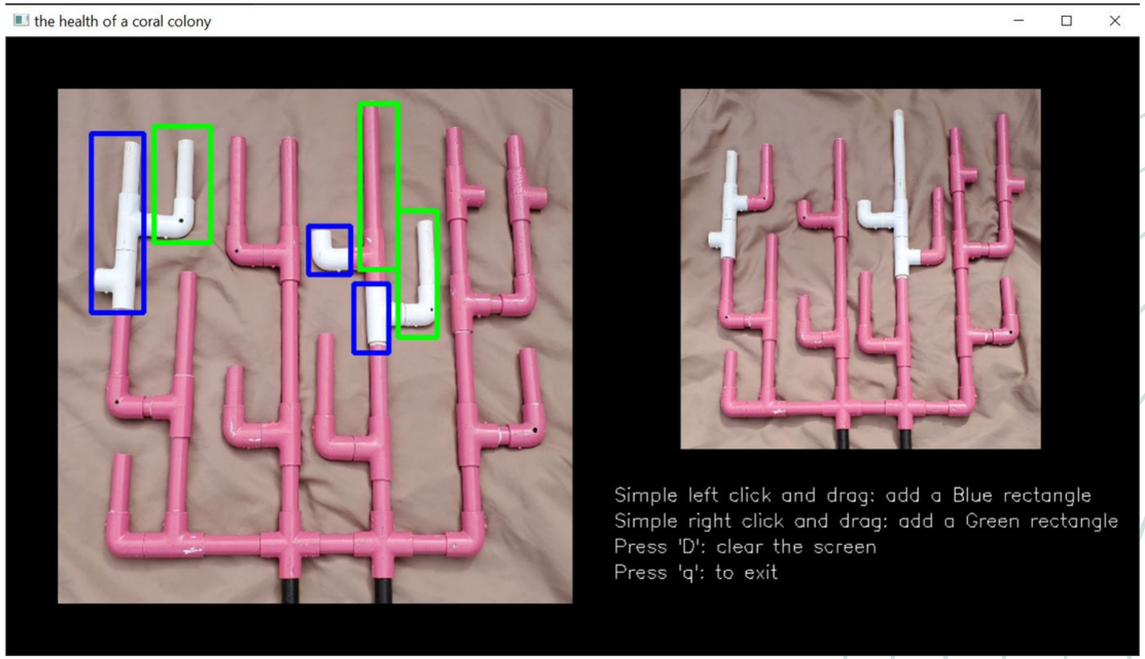
```python
def rectangle_shape(event, x, y, flags, param):
    global ix,iy,draw

    if event == cv2.EVENT_LBUTTONDOWN:
        draw =True
        ix, iy = x, y

    elif event == cv2.EVENT_LBUTTONUP:
        draw =False
        color = (255, 0, 0)
        cv2.rectangle(image, (ix, iy), (x, y), color, 3)

    if event == cv2.EVENT_RBUTTONDOWN:
        draw =True
        ix, iy = x, y

    elif event == cv2.EVENT_RBUTTONUP:
        draw =False
        color = (0, 255, 0)
        cv2.rectangle(image, (ix, iy), (x, y), color, 3)


draw_text(image)
clone = image.copy()

cv2.namedWindow("the health of a coral colony")
cv2.setMouseCallback("the health of a coral colony", rectangle_shape)


while True:
    cv2.imshow("the health of a coral colony", image)

    key =cv2.waitKey(1)

    if key == ord('D') or key == ord("d"):
        image = clone.copy()

    elif key == ord('q') or key == ord("Q"):
        cv2.destroyAllWindows()
        break
```

Assignments Task 6 [Manual]

VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam





Assignments Task 6 [Autonomous]

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

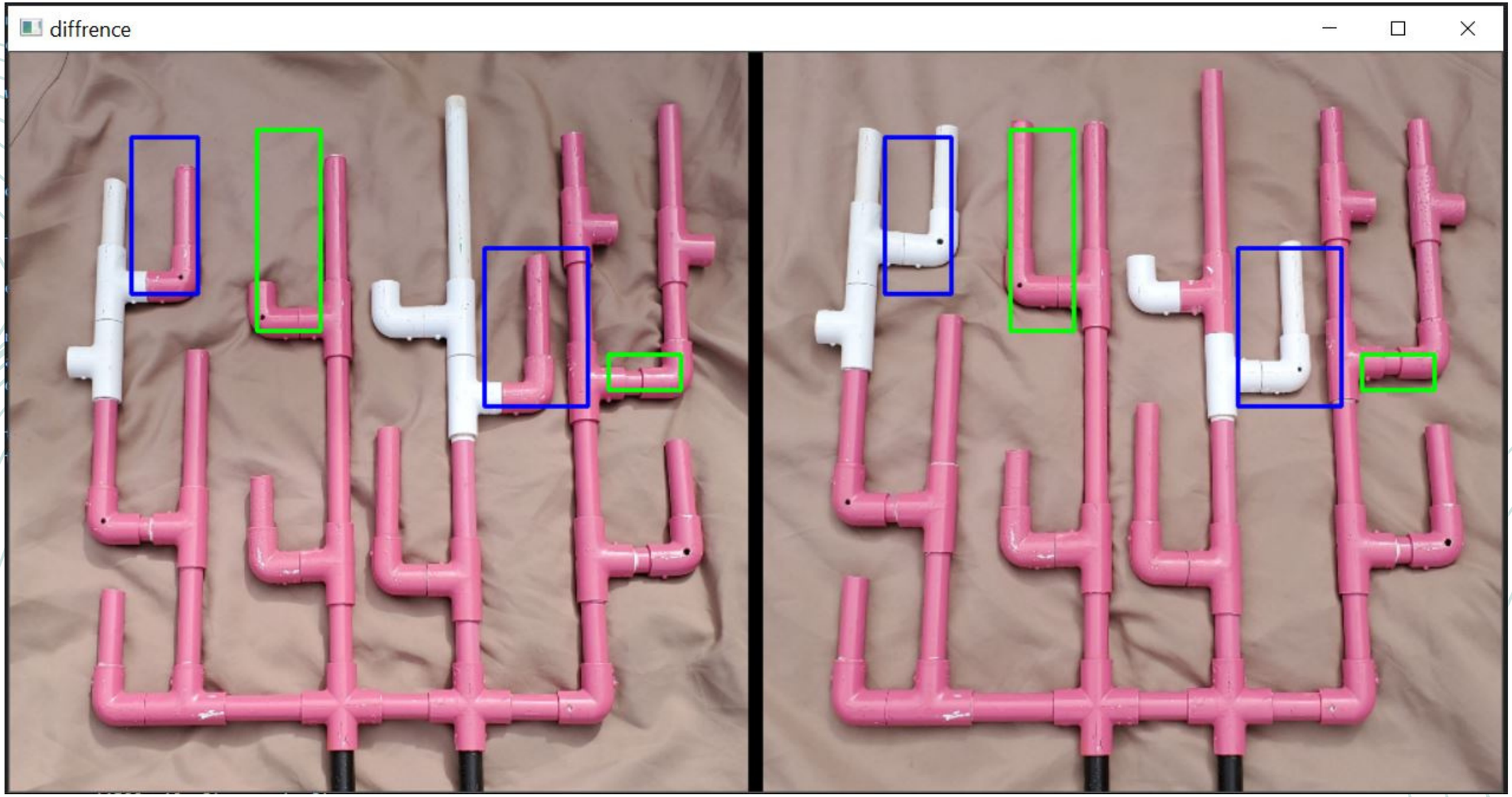
```python
import cv2
import numpy as np
import imutils
image = cv2.imread("coral3.jpg")
image = cv2.resize(image, (500, 500))
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
frame = cv2.imread("coral4.jpg")
frame = cv2.resize(frame, (500,500))
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

lower = np.array([16, 0, 0], dtype = "uint8")
upper = np.array([180, 255, 255], dtype = "uint8")

mask_image = cv2.inRange(hsv_image, lower, upper)
no_background_image = cv2.bitwise_and(hsv_image, hsv_image, mask = mask_image)
no_background_image = cv2.cvtColor(no_background_image, cv2.COLOR_HSV2BGR)

mask_frame = cv2.inRange(hsv_frame, lower, upper)
no_background_frame = cv2.bitwise_and(hsv_frame, hsv_frame, mask = mask_frame)
no_background_frame = cv2.cvtColor(no_background_frame, cv2.COLOR_HSV2BGR)

gray_image = cv2.cvtColor(no_background_image,cv2.COLOR_BGR2GRAY)
gray_frame = cv2.cvtColor(no_background_frame,cv2.COLOR_BGR2GRAY)

#=========================================================

sift = cv2.SIFT_create()
kp_image, desc_image = sift.detectAndCompute(gray_image,None)

FLANN_INDEX_KDTREE = 0
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=30)

kp_gray_frame, desc_frame = sift.detectAndCompute(gray_frame, None)

flann = cv2.FlannBasedMatcher(index_params,search_params)
matches = flann.knnMatch(desc_image,desc_frame,k=2)

good_points = []
for m, n in matches:
    if m.distance < 0.7*n.distance:
        good_points.append(m)
```

```python
img3 = cv2.drawMatches(gray_image, kp_image, gray_frame, kp_gray_frame, good_points, gray_frame)

query_pts = np.float32([kp_image[m.queryIdx].pt for m in good_points]).reshape(-1, 1, 2)
train_pts = np.float32([kp_gray_frame[m.trainIdx].pt for m in good_points]).reshape(-1, 1, 2)

matrix , mask = cv2.findHomography(query_pts, train_pts, cv2.RANSAC, 5.0)
matches_mask = mask.ravel().tolist()

h, w = gray_image.shape
pts = np.float32([[0, 0],[0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
dst = cv2.perspectiveTransform(pts, matrix)

homography = cv2.polylines(gray_frame, [np.int32(dst)], True, (255, 0, 0), 5)

diff = cv2.absdiff(gray_image, homography)

thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

kernel = np.ones((5, 5), np.uint8)
dilate = cv2.dilate(thresh, kernel, iterations=3)
erode = cv2.erode(dilate, kernel, iterations=6)

contours = cv2.findContours(erode.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)

for contour in contours:
    if cv2.contourArea(contour) >700:
        x, y, w1, h1 = cv2.boundingRect(contour)
        if w1 > 27 and w1 < 70:
            cv2.rectangle(image, (x, y), (x+w1, y+h1), (0, 255, 0), 2)
            cv2.rectangle(frame, (x, y), (x+w1, y+h1), (0, 255, 0), 2)

        if w1 > 27 and h1 >= 97 and h1 <= 130 and h1!=110:
            cv2.rectangle(image, (x, y), (x+w1, y+h1), (255, 0, 0), 2)
            cv2.rectangle(frame, (x, y), (x+w1, y+h1), (255, 0, 0), 2)


x = np.zeros((500, 10, 3), np.uint8)
cv2.imshow("diffrence", np.hstack([image, x, frame]))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Assignments Task 6 [Autonomous]

VORTEX ACADEMY
THE ART OF THINKING

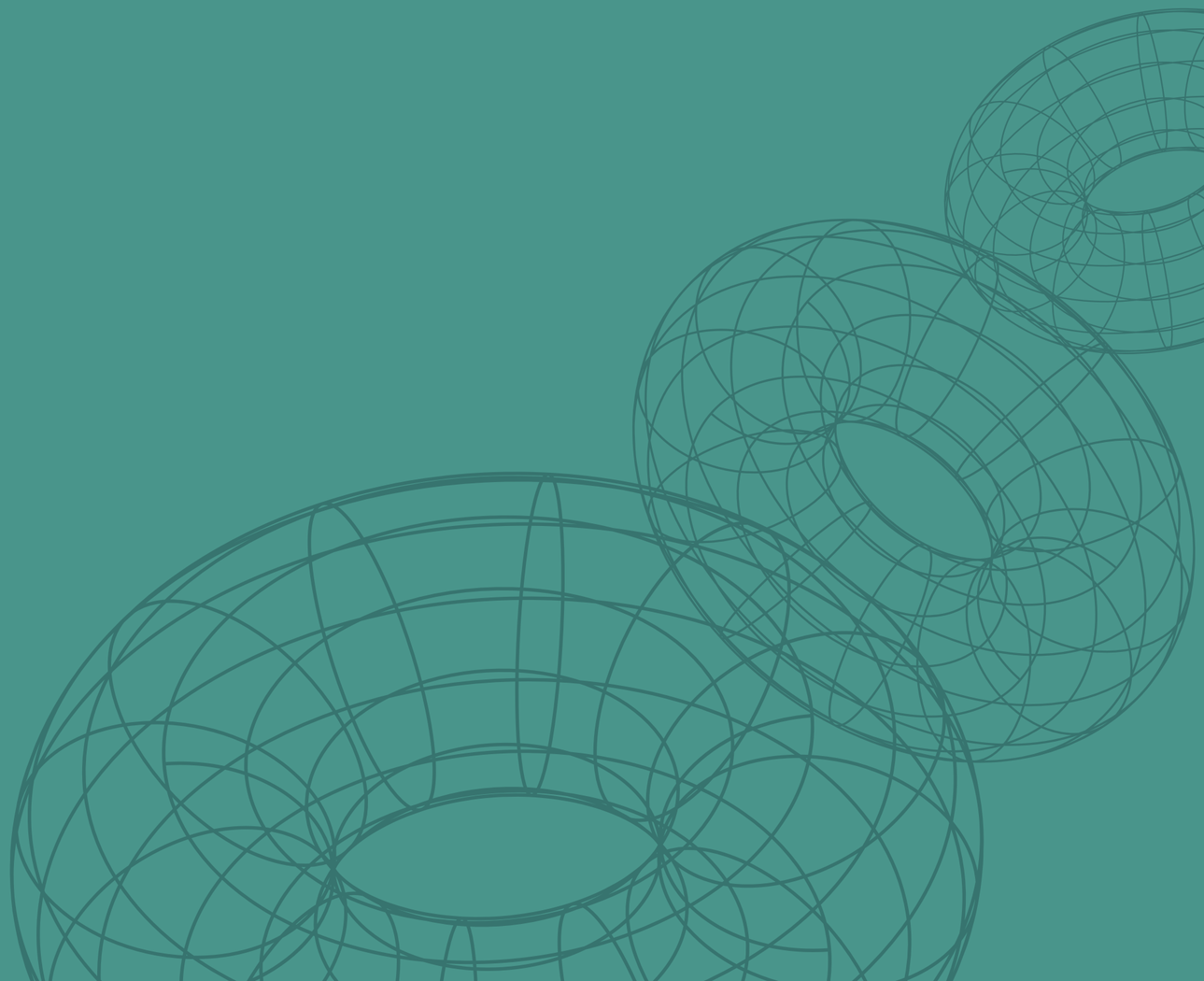
Eng. Mai Allam



VORTEX ACADEMY
THE ART OF THINKING

Eng. Mai Allam

Do you have any questions?



Free Resources

VORTEX ACADEMY
THE ART OF THINKING
Eng. Mai Allam

OpenCV Basics - Free Code Camp --> https://www.youtube.com/watch?v=oXlwWbU8l2o
OpenCV Basics - Murtaza --> https://www.youtube.com/watch?v=WQeoO7MI0Bs
Detecting Clicks --> https://www.youtube.com/watchv=DaQoorJQSZQ&list=PLMoSUbG1Q_r_sc0x7ndCsqdlkL7dwrmNF&index=8

Websites with useful Articles:
pyimagesearch --> https://pyimagesearch.com/category/image-processing/
LearnOpenCV --> https://learnopencv.com/getting-started-with-opencv/

Computer Vision Lectures:
First Principles of Computer Vision --> https://www.youtube.com/channel/UCf0WB91t8Ky6AuYcQV0CcLw
Introduction to Computer Vision --> https://www.udacity.com/course/introduction-to-computer-vision--ud810

Recommended OpenCV book (read till chapter 8)
Mastering OpenCV Book --> https://www.amazon.com/Mastering-OpenCV-Python-practical-processing/dp/1789344913