

**NAME:** MAHAM AZAM

**ROLL NO:** 005

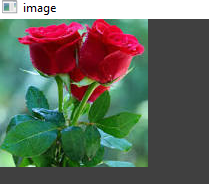
**SECTION:** 4A

**SUBJECT:** PROGRAMMING FOR AI LAB

**2.1:**

This code loads and shows a picture using a library called Open CV. It reads an image named img.jpeg and saves it in a variable called img. Then, it opens a window to display the image. The program waits until you press any key, and once you do, it closes the image window.

**Output:**



**2.1.2:**

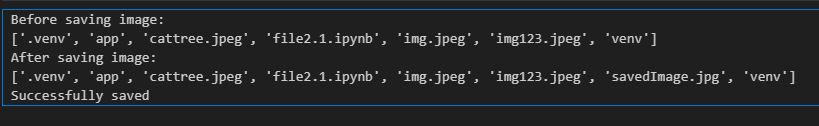
This program use the Open CV library to open and display an image. First, it sets the file path of an image named "img123.jpeg" and loads it using the cv2.imread () function. The image is stored in a variable called image. Then, a window is created with the name "image", and the image is displayed using cv2.imshow (). The program waits until the user presses any key before closing the image window with cv2.destroyAllWindows (). This helps in viewing the image properly before closing it. Here its output.



**2.1.3**

This code uses Open CV to read and save an image. First of all its sets the path of an image which is stored on the desktop with the file name cattree.jpeg. Then, it reads the image using cv2.imread (). The program changes the current working directory to the folder where the image is stored. Before saving the image, it prints a list of files in that folder. The image is then saved with a new name "savedImage.jpg", using cv2.imwrite (). After saving it prints the updated list of files to show that the new image has been added. At the end it displays a message "Successfully saved".

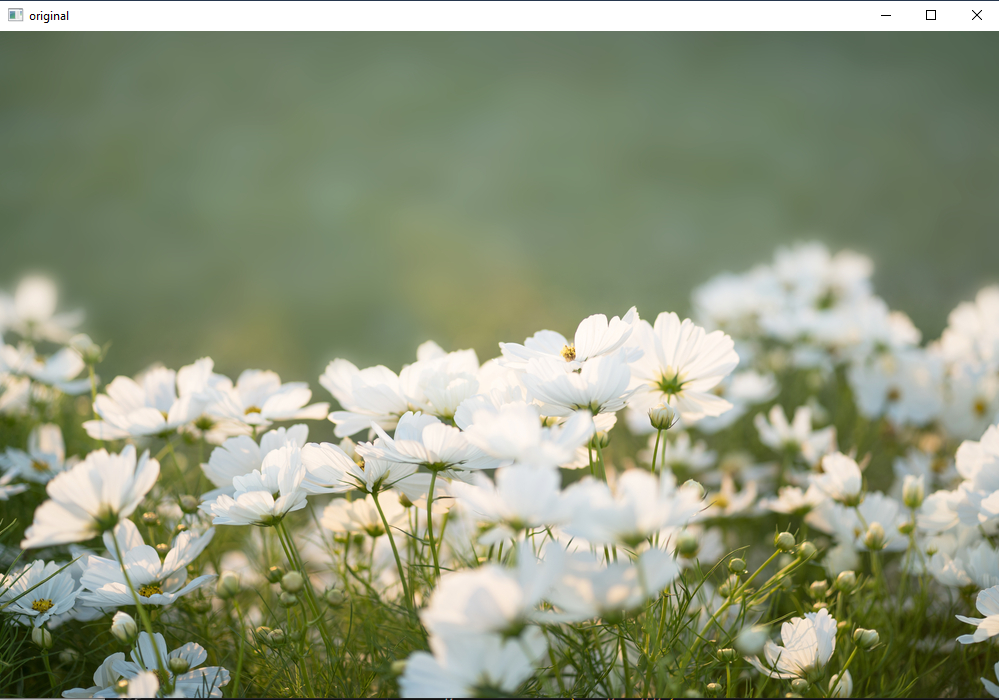
**Output:**

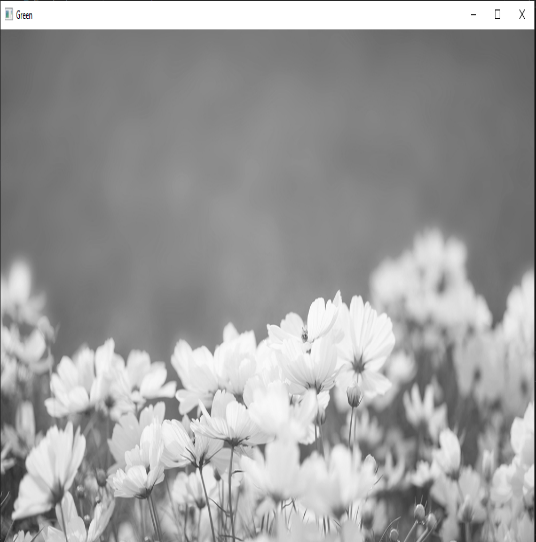


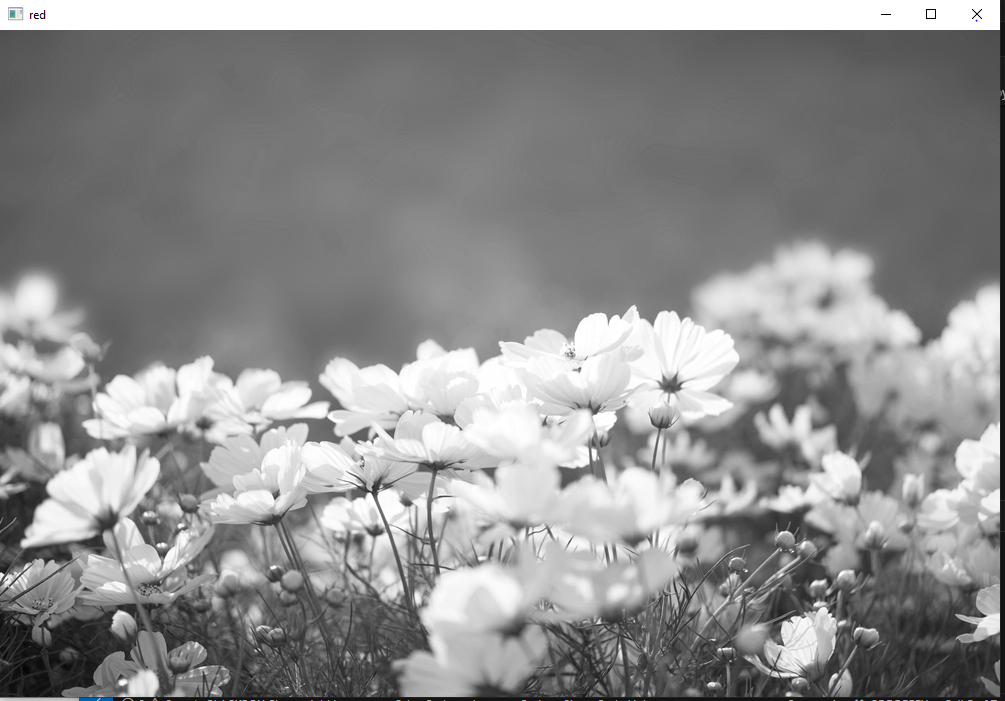
**2.1.4:**

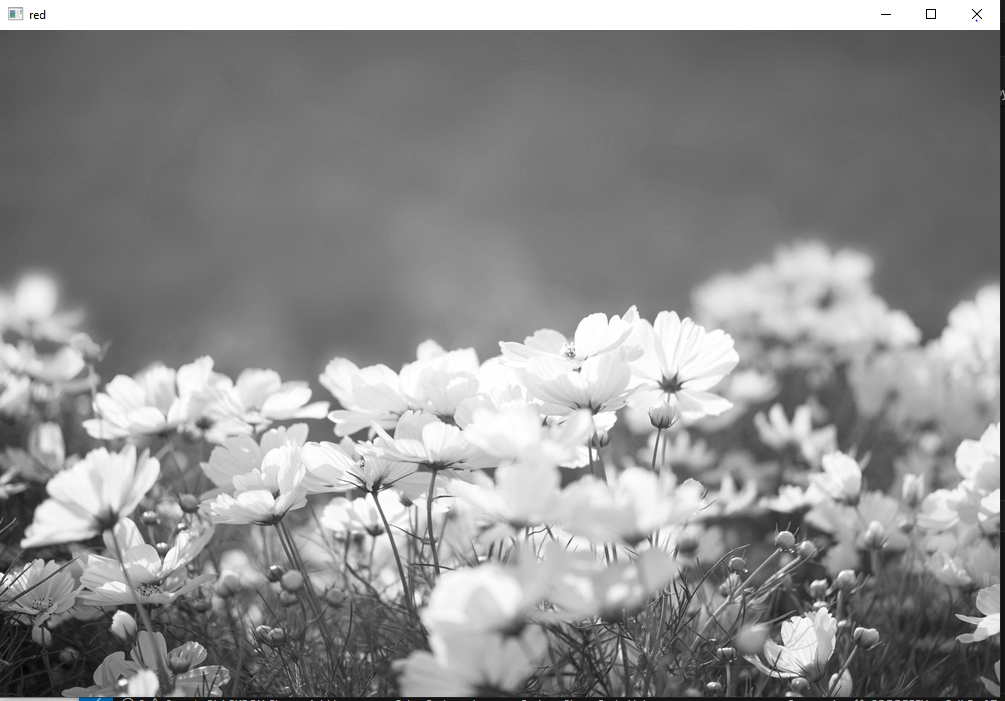
This code uses Open CV to open an image and separate its colors into blue, green, and red. It starts by reading an image file called white.jpg using cv2.imread ().The program first displays the original image, after pressing a key, it shows only the blue part of the image, then the green, and finally the red, each in different windows and waits once more after pressing a key each time all windows are closed using cv2 destroy all windows. Here its output.

**Output:**



****

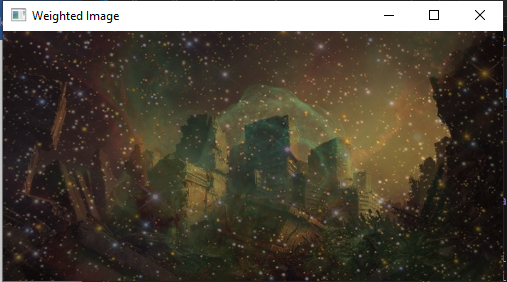




**2.1.5:**

This code uses Open CV to blend two images together. First of all, it loads two images image1 and image2, using cv2.imread (). Then combines these images using cv2.addWeighted (), Where the first image has the weight of 0.5 and the second image weight is 0.4 with the final parameters of 0. That’s controls brightness the blended image is then display a window using cv2 imshow the program waits for the user to press a key if the escape key is pressed it closes all windows using cv2destroyAllWindows to free up memory. Below is the output.

**Output:**



**2.1.6**

This program uses Open CV to apply a bitwise AND operation on two images. First of all it loads two images, img1.png and img2.png, using cv2.imread (). Then, it uses cv2.bitwise\_and () to compare the pixels of both images and keep only the parts where both images have the same pixel values and the result is stored in dest\_ and and displayed in a window using Cv2 imshow The program waits for the user to press a key and then Escape key is pressed, it closes the image window using cv2 destroy all windows.

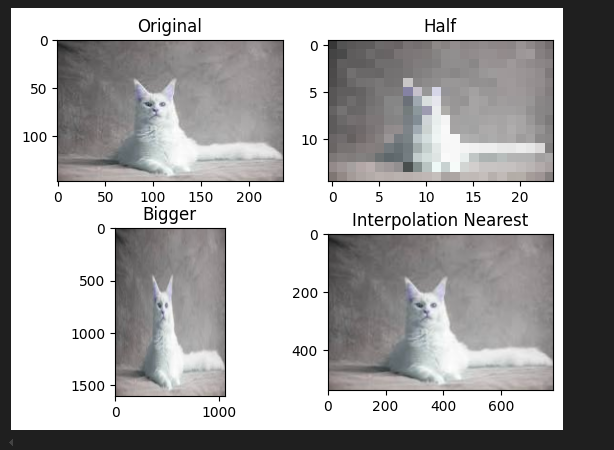
**Output:**



**2.2**

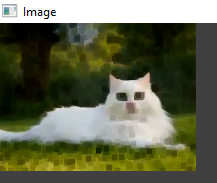
This code uses Open CV and Matplotlib to resize an image and display different versions of it first loads an image called whitecat.jpeg using cv2.imread (). Then, it creates three resized versions a smaller one, a larger one, and another resized version using interpolation for better quality. These images are stored in a list along with the original image and then uses Matplotlib to display all four images in a 2x2 grid, each labeled with a title. Finally it call the plt.show () to displays the images.

**Output:**



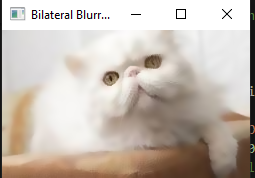
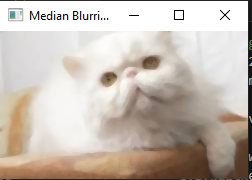
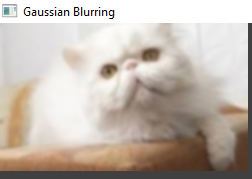
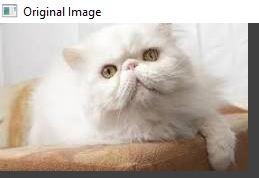
This code changes an image using a method called erosion. First of all, it loads an image then it creates a small square using numbers like 5x5, which helps in changing the image. The erosion effect makes the bright parts of the image shrink and helps remove small spots or noise. Finally, the program shows the changed image in a window using cv2.imshow.

**Output**



This code load an image using cv2.imread () and display it with cv2.imshow (). Then, it applies three different blurring techniques. First, it uses Gaussian blur with a 7x7 kernel to smooth the image then it applies median blur with a 5 size kernel and at the end it uses Bilateral Blur, which makes the image smoother while preventing edges. After applying each effect, the modified image is displayed one by one. Then it waits for a key press before showing the next image, and at the end, it closes all image windows using cv2.destroyAllWindows (). Here the output of this code.

**Output**



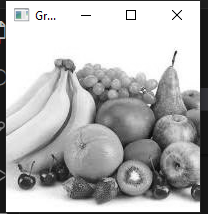
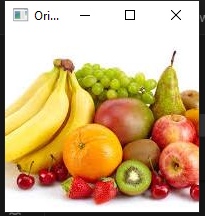
This code adds a border around an image using Open CV first of all it loads an image using cv2.imread () and then it uses cv2.copyMakeBorder () to create a border of 10 pixels on all sides of the frame. The border is a black color because the value is set to 0 and then at the end it will displays the modified image in a window. Here its output

**Output**



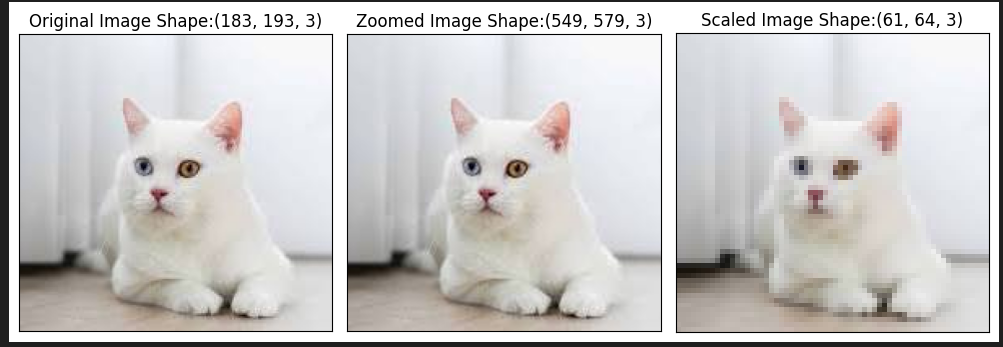
This code converts a colored image to grayscale image using Open CV first it loads an image from the desktop and then displays it and after that it uses cv2.cvtColor () to change the image from original color to black and white. After that the grayscale image is shown in a new window then it waits for the user to press a key before moving to the next step and in the end it closes all image windows.

**Output**

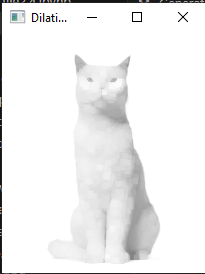


This code loads and image using Open CV and convert into bgr to rgb format using cv2.cvtcolor and then used Matplotlib to display it properly and then it creates two scale factor one is zoom in by 3 times and the other is decreasing it by 3 times. The resizing is done using cv2.resize (), where a special method is used to keep the image clear and then all three images are displayed side by side using Matplotlib, with their sizes labeled. The program also removes extra details like x and y axis labels to keep the display neat.

**Output**

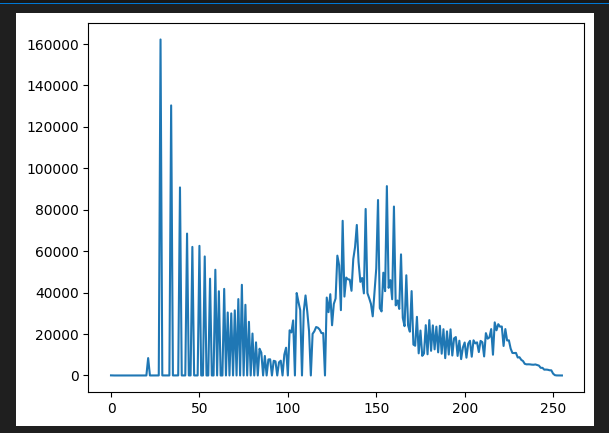


This code loads an image in black and white using cv2.imread with the flag 0 then it creates a small kernel of size 5x5 to process the image and then applies erosion using cv2.erode which shrinks the bright area of the image and dilation using cv2.dilate which expands the bright area both operations are performed once as specified by the iterations parameter the original image along with the eroded and dilated images are displayed using cv2.imshow and the program waits for a key press before closing the windows.



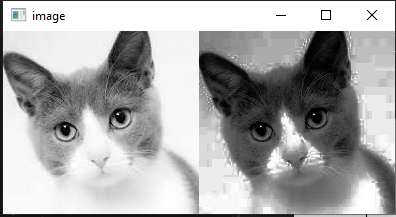
This code load a black and white image using Open CV and then display its using Matplotlib. And then it calculates how many pixels in the image have each brightness level from 0 to 255 where 0 is black and 255 is white. This information is stored in a histogram, which is a graph showing how bright or dark the image is. Finally, the code plots the histogram using Matplotlib and displays it as a graph.

**Output**



This code loads a black and white image using cv2.imread and then it applies histogram equalization using cv2.equalize hist to improve the contrast of the image the original image and the processed image are stacked side-by-side using numpy. Np. And then compared the both images side by side so they can be comparessed and then display the resulting image using cv2.inshow and then program waits for a key press before closing the window.

**Output**



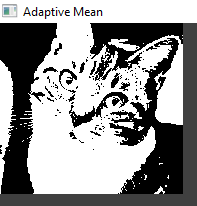
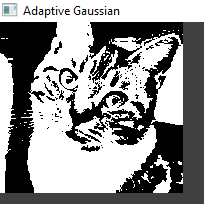
This code load an image and converts it into grayscale using cv2.cvt and then applies five different thresholding methods such as **Binary, Inverted Binary, Truncated, To Zero, and To Zero Inverted** thresholding each thresholding method has its effect on the pixel values based on the specified threshold of 120 the result shown in the separate window using cv2.imshow the program waits for the key to press and then closes the window.

**Output**



This code loads an image from the desktop and converts it to grayscale. Then, it applies two methods, Adaptive Mean and Adaptive Gaussian Thresholding, which gives more importance to closer pixels. Both processed images are displayed, and the program waits for a key press before closing the windows.

**Output**

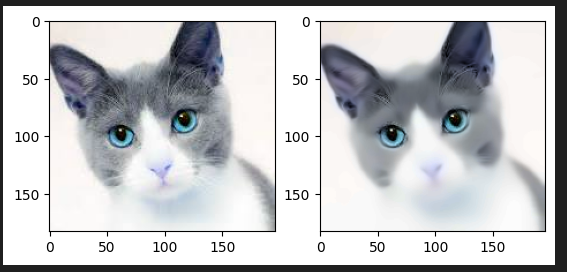


This code makes an image black and white using Otsu's thresholding first it loads and image and converts it to grayscale and then it applies Otsu’s method, which automatically finds the best brightness level to separate dark and light areas. The processed image is displayed using cv2.imshow, and the program waits for a key press before closing the window.

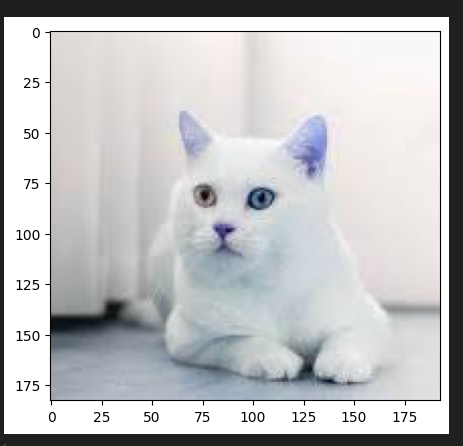


This code reads an image from the specified path and applies a denoising filter using the cv2 fast NL means denoising colored function to reduce noise in the image the denoised image is saved in the variable dst then it uses matplotlib to display the original and denoised images side by side using plt.subplot and plt.imshow finally it shows the images with plt.show.

**Output**

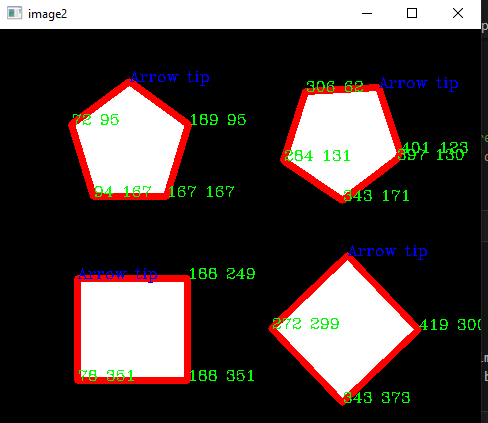


This code load an image using open CV and then display it using Matplotlib.pyplot.imshow, it shown in BGR format, but Matplotlib expects RGB format. The code reads the image using cv2.imread () , then plt.imshow() is used to display the image correctly but we you need to convert the image to bgr to rgb using cv2.cvtcolor img.cv2 .This helps in processing and analyzing images in Python.



This code loads the image in both color and grayscale and then it converts the grayscale image into a black and white image using a threshold. it detects the outline and draws red outlines around them. The program also finds the corner points of the outlines and labels them with their coordinates in green text. If the point is at the tip of an arrow, it is labeled separately in blue. Finally, the processed image is displayed.

**Output**

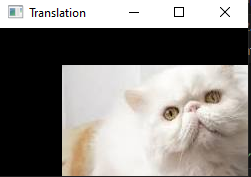
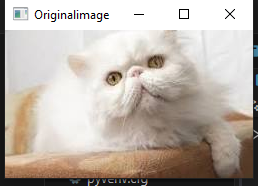


This code loads an image taj.jpg using Open CV and then read the image using cv2.imshow and then it applies bilateral filtering, which removes noise but keeps important details like edges clear. Finally, the filtered image is saved as "taj\_bilateral.jpg".



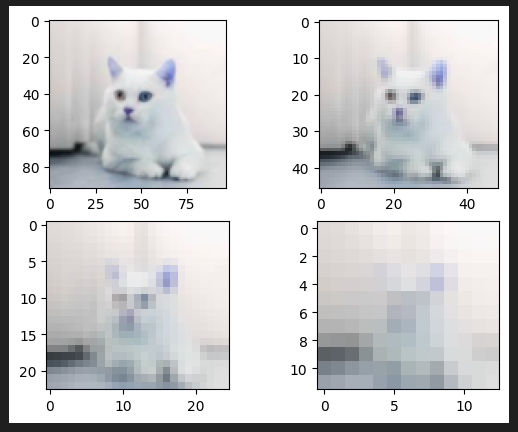
This code load an image using open CV and then reads an image with cv2.imshow and stores its height and width it calculates a translation matrix T and then shifts the image by one-fourth of its height and width then it uses cv2.warpAffine to apply the translation matrix to the image resulting in a shifted image the original and translated images are displayed using cv2.imshow and the program waits for a key press before closing the windows with cv2.destroyallwindows.

**Output**



This code load an image "image18.jpeg" and makes a copy it reduce the size of the image step by step and then it uses a loop to shrink the image 4 times using cv2.pyrDown(), which makes the image smaller while keeping its details. Each smaller version is shown in a window and a plot.the program waits for a key press with cv2.waitKey before proceeding and then close the window.

**Output**



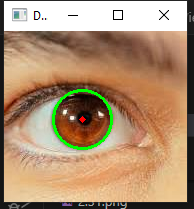
**2.3**

This code load an image and then convert into grayscaled and draws straight lines in an image and then it detects edges using the Canny edge detector after that, it applies the Hough Transform to find straight lines in the image. For each detected line, it calculates the start and end points and draws red lines over them and then it saves the new image as "linesDetected.jpg" with the detected lines.



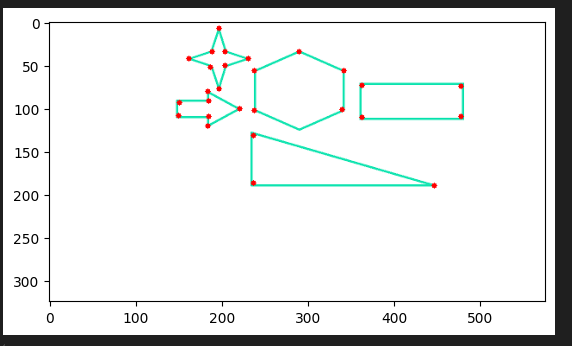
This code load an image "eyes123.jpeg" and converts it to grayscale and then detect the circle in the image and then it applies a slight blur to remove noise after that it uses the Hough Circle Transform to find circles in the image. If circles are detected it draws green circles around them and marks their centers with a small red dot. It displays the image with the detected circles.

**Output**

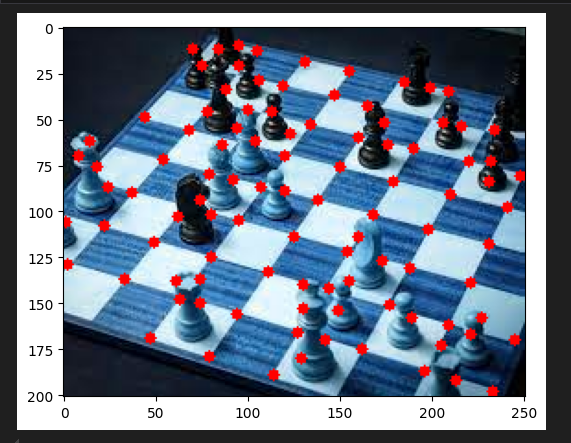


This code loads an image and converts it into grayscale and then it detects corners using the good Features To Track function which finds 27 corners with a quality level of0.01 and a minimum distance of 10 pixels between corners the detected corners are then drawn on the original image as small white circles and the result is displayed the image with the detected corners using matplotlib.

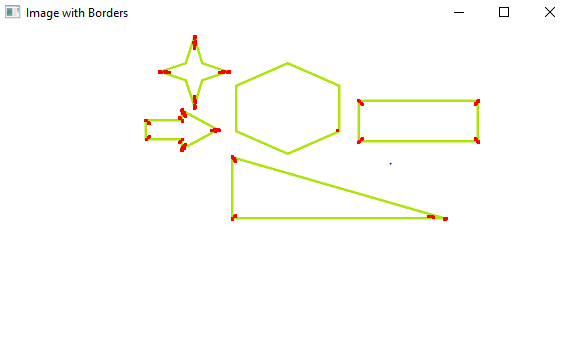
**Output**



This code load an image and converts it into grayscale to make detection easier and this code finds the corner in the image and then detect the corners using the good features to track the function which finds the 100 corners in the image with the minimum quality level of 0.01. After detecting the corner the program draws small blue circles on them to highlight their positions and then it displays the image with the detected corners using matplotlib. The code also include the wait key () function to close the open CV function.



This code load and image and then convert into grayscaled to make detection easier it finds corners in an image by using the Harris Corner Detection method then it applies a special function to detect strong corners after that, it highlights the detected corners by coloring them red you can press esc to close the window like it is good for detecting the important part of the image and then it displays the image with the marked corners.



This code is used to detect circular blobs in an image. The parameters ensure that only blobs that are sufficiently large, circular, convex, and with specific inertia are detected. After detecting the blobs, the code highlights them on the image in red circles, and it shows the number of circular blobs detected by adding the count as text on the image and then display the image with the detected circle.

**Output**



**2.4**

This code load an image from desktop using open CV and then read the image with cv2.imshow () it draws a green line on the image at the top-left corner and ends at (250,250) with the thick line of 9 pixels wide. After drawing the line, the program displays the image with the line on it.

**Output**

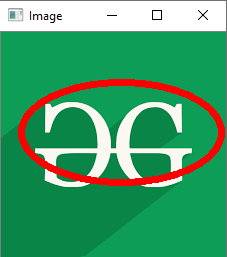


This code load an image from desktop using open CV and then read it using cv2.imshoe it draws a green arrow on the image starts from the top-left corner and points to (200,200) with the thick arrow of 9 pixels wide. After drawing, the program displays the image with the arrow on it.

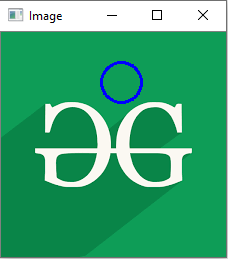


#system was not loading the output

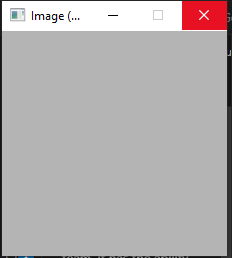
This code load an image from desktop using open CV and then drew an ellipse on it the ellipse was centered at coordinates 120 and 100 with axes lengths of 100 and 50 pixels and then set the angle to 0 and the ellipse goes from 0 to 360 degrees then choose red color for the ellipse and made the border 5 pixels thick once and added the ellipse to the image and then displayed it.



This code load an image from the desktop and then draws a blue circle on an image. The circle is centered at 120, 50 and has a radius of 20 pixels. It is outlined in blue 255 with a thickness of 2 pixels. After drawing, the program displays the image with the circle on it.



This code load an image from the desktop and then draw the blue rectangle on the image it start at 5, 5 and 220, 220 to make the square like shape and it has the blue border with the thickness of 2 pixels and then it display the image with the rectangle on it.



#the image was not responding in my pc