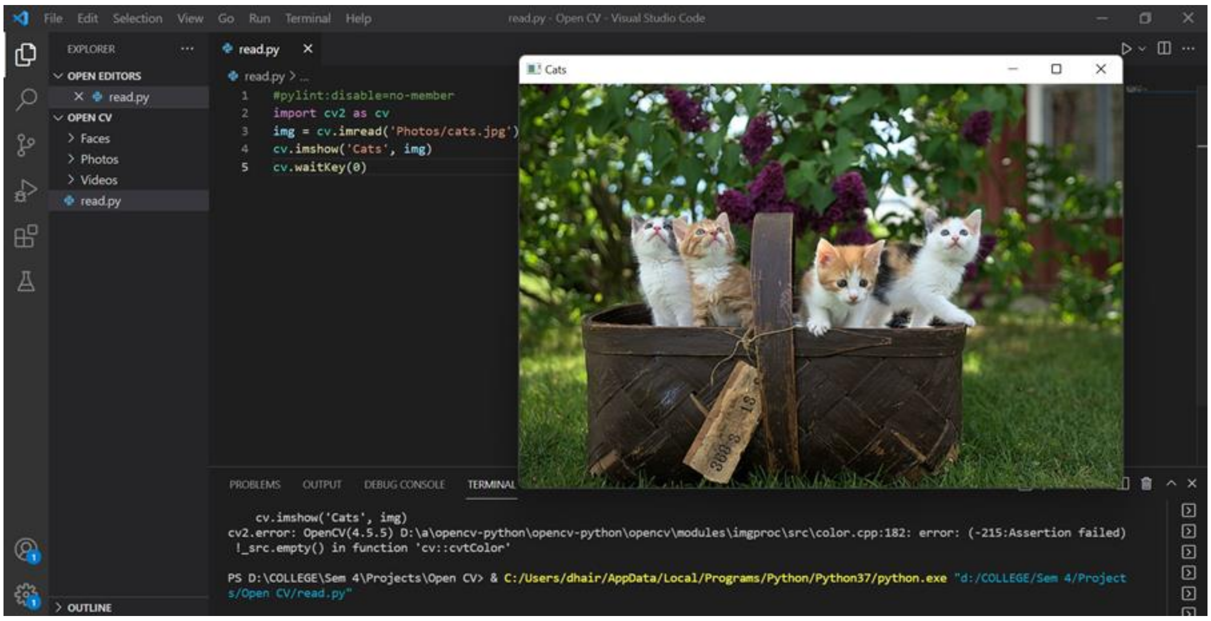
Open CV

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection

Reading an image in Open CV

import cv2 as cvimg = cv.imread('../Resources/Photos/cats.jpg')
#in the function arguments we have to specify the path of the imagecv.imshow('Cats', img)cv.waitKey(0)



```
#pylint:disable=no-member
import cv2 as cv
img = cv.imread('Photos/cats.jpg')
cv.imshow('Cats', img)
cv.waitKey(0)
```

```
    cv.imshow('Cats', img)
cv2.error: OpenCV(4.5.5) D:\a\opencv-python\opencv-python\opencv\modules\imgproc\src\color.cpp:182: error: (-215:Assertion failed)
 !_src.empty() in function 'cv::cvtColor'

PS D:\COLLEGE\Sem 4\Projects\Open CV> & C:/Users/dhair/AppData/Local/Programs/Python/Python37/python.exe "d:/COLLEGE/Sem 4/Project
s/Open CV/read.py"
```

reading an image from Open CV

Reading an video in Open CV

capture = cv.VideoCapture('../Resources/Videos/dog.mp4')while True:isTrue, frame = capture.read()# if cv.waitKey(20) & 0xFF==ord('d'):# This is the preferred way - if `isTrue` is false (the frame could# not be read, or we're at the end of the video), we immediately# break from the loop.if isTrue:cv.imshow('Video', frame)if cv.waitKey(20) & 0xFF==ord('d'):breakelse:breakcapture.release()cv.destroyAllWindows()

click here for video demonstration

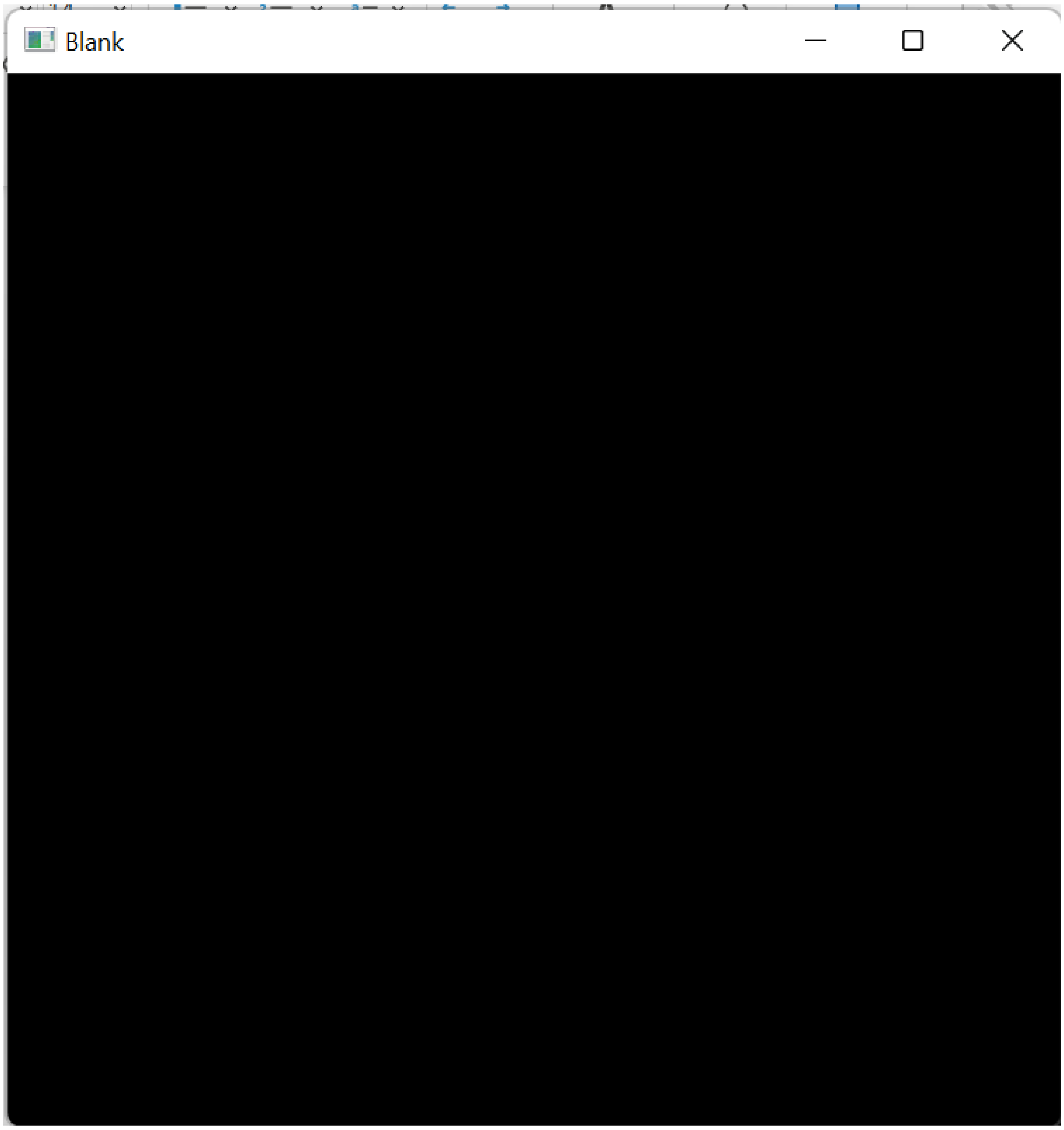
Rescaling a video file in Open CV

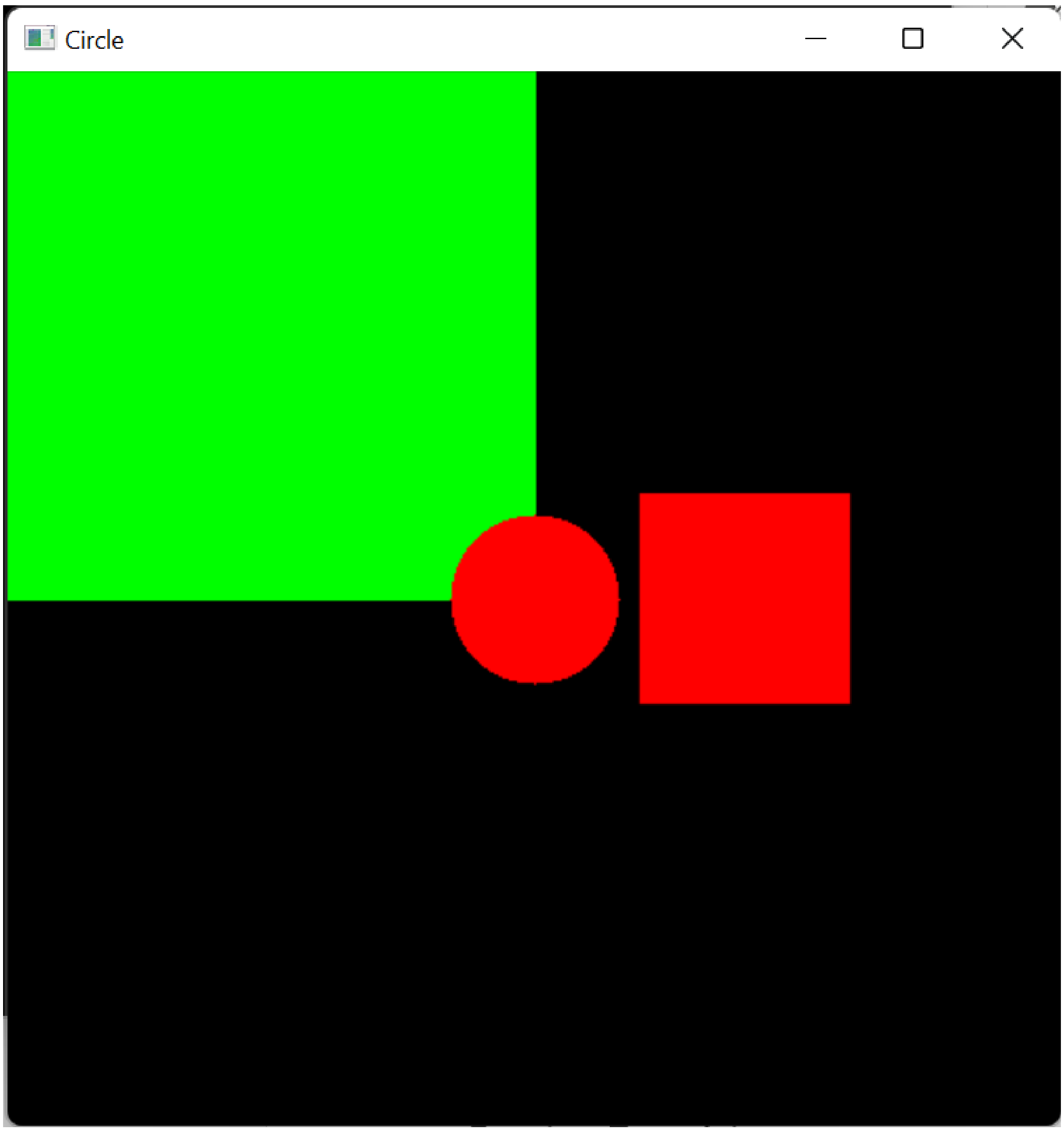
#pylint:disable=no-memberimport cv2 as cv# img = cv.imread('../Resources/Photos/cat.jpg')# cv.imshow('Cat', img)def rescaleFrame(frame, scale=0.75):# Images, Videos and Live Videowidth = int(frame.shape[1] * scale)height = int(frame.shape[0] * scale)dimensions = (width,height)return cv.resize(frame, dimensions, interpolation=cv.INTER_AREA)def changeRes(width,height):# Live videocapture.set(3,width)capture.set(4,height)# Reading Videoscapture = cv.VideoCapture('Videos/dog.mp4')while True:isTrue, frame = capture.read()frame_resized = rescaleFrame(frame, scale=.2)cv.imshow('Video', frame)cv.imshow('Video Resized', frame_resized)if cv.waitKey(20) & 0xFF==ord('d'):breakcapture.release()cv.destroyAllWindows()

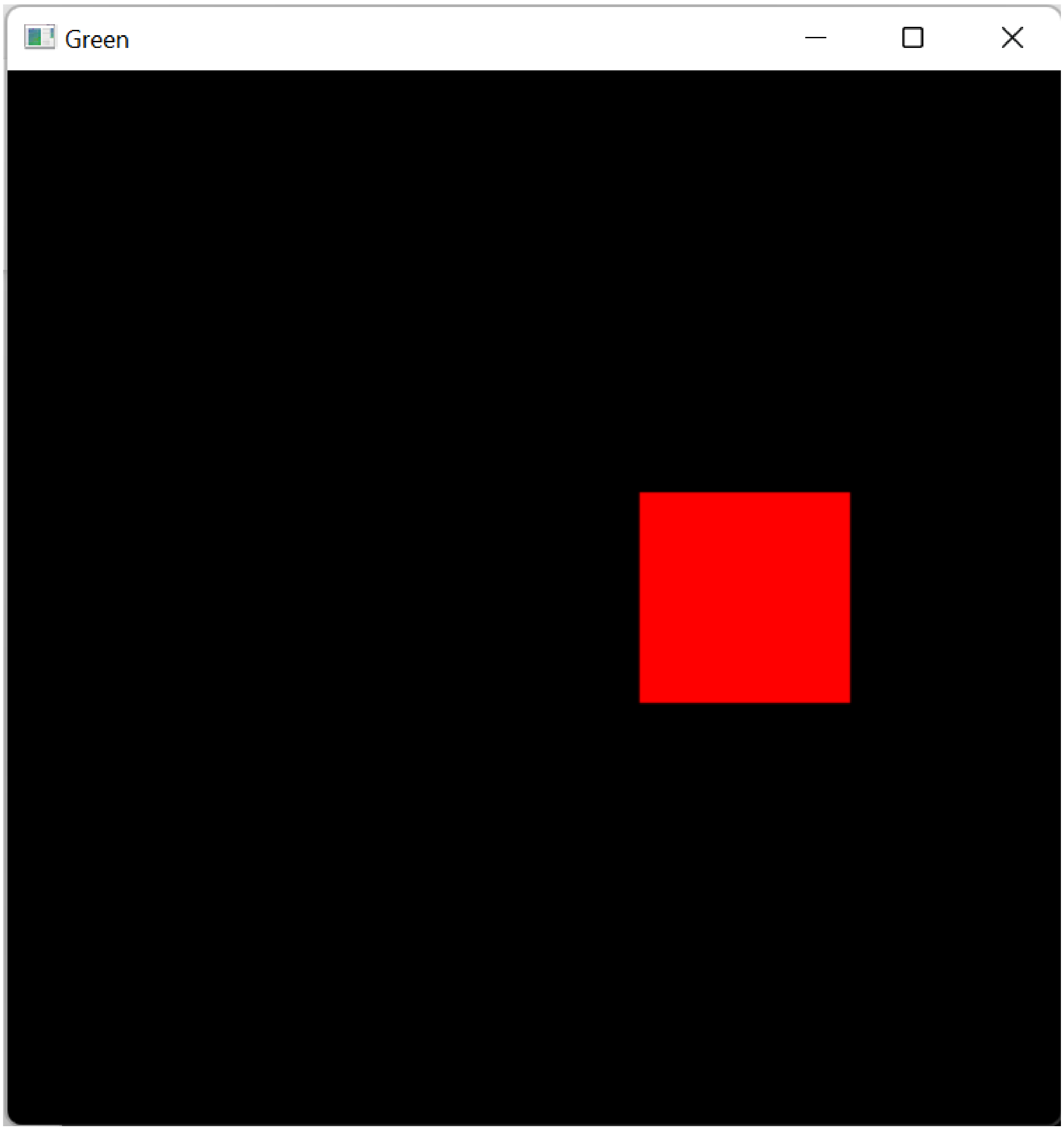
click here for video demonstration

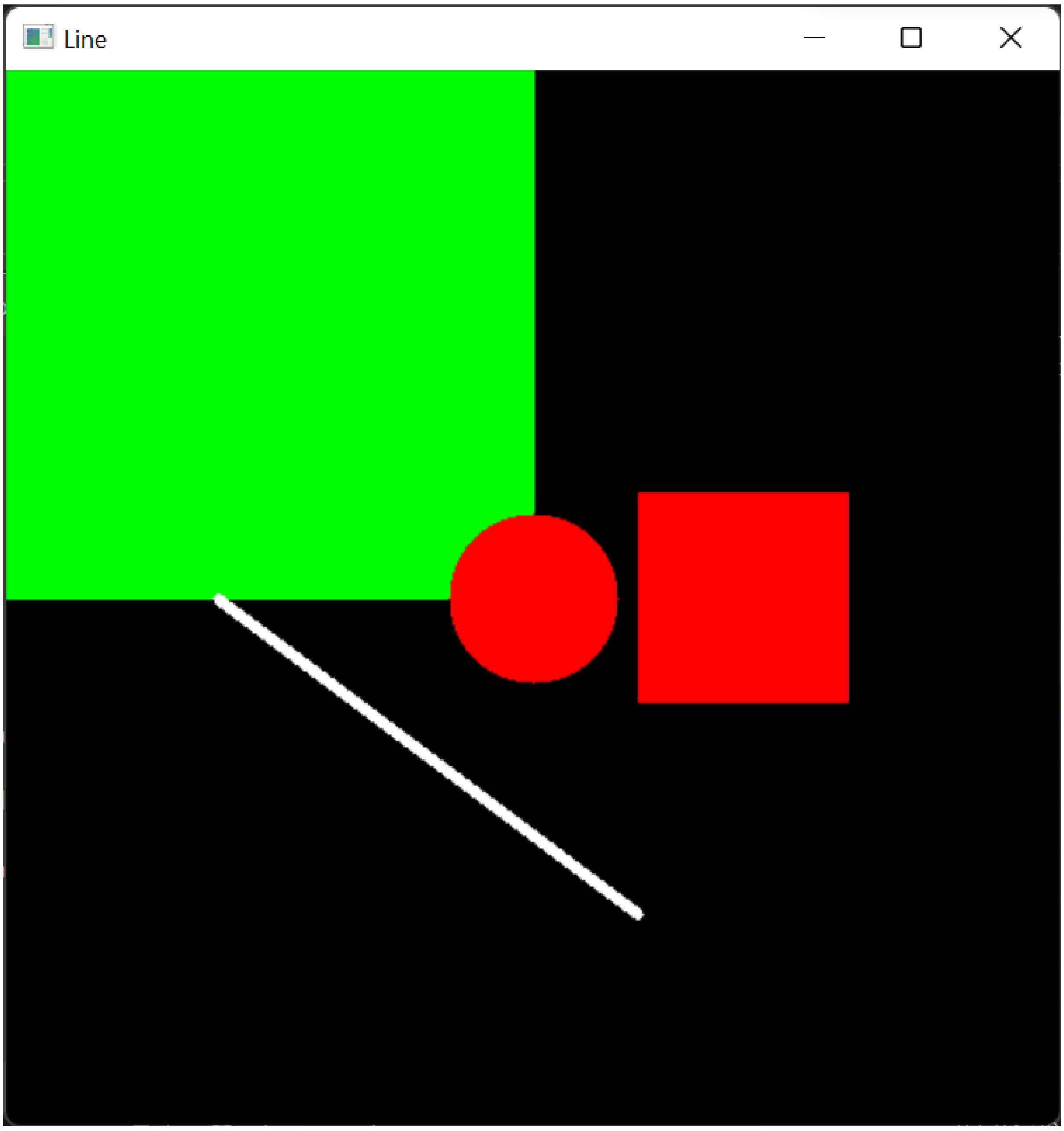
Drawing in Open CV

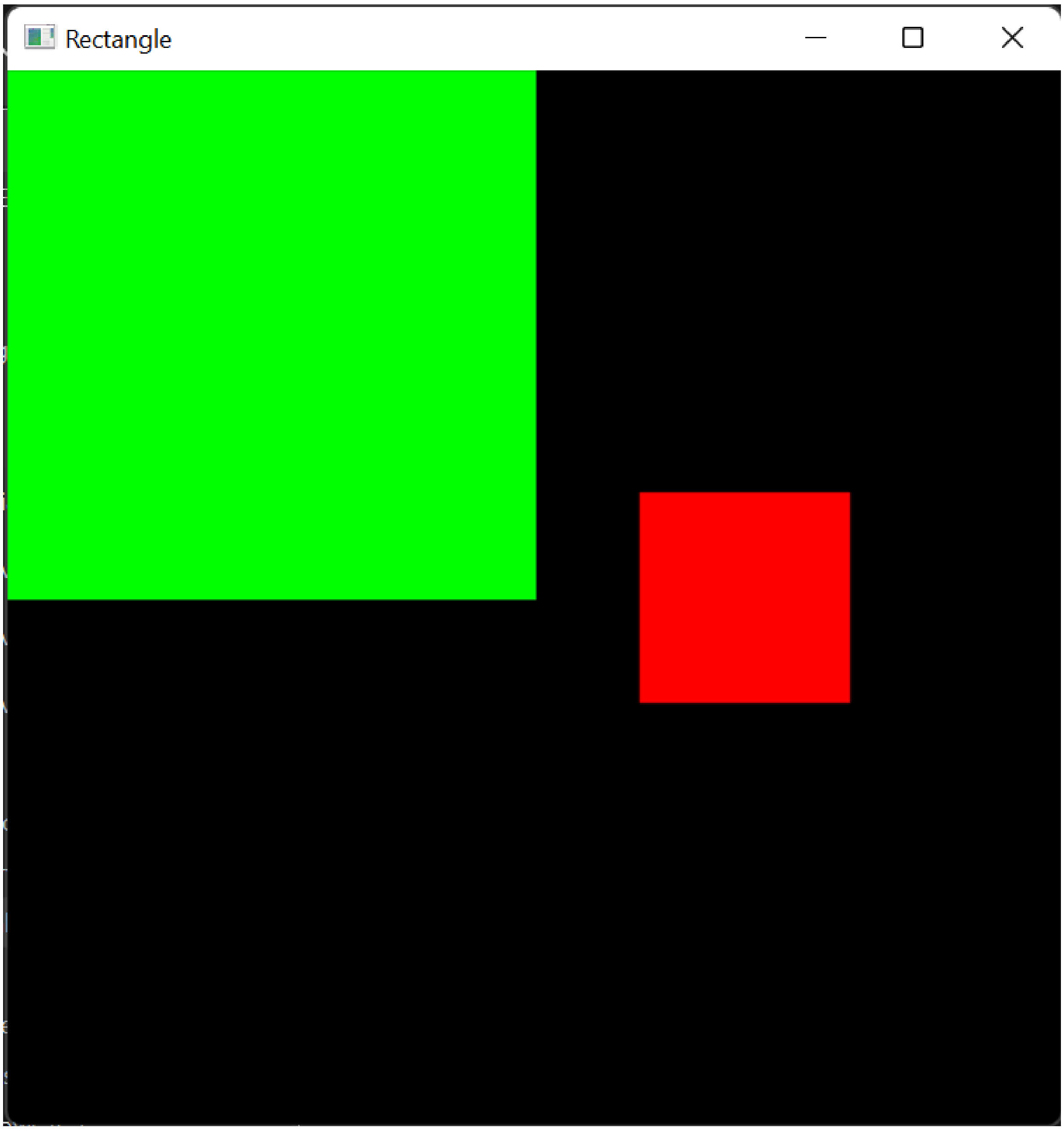
```
#pylint:disable=no-memberimport cv2 as cvimport numpy as npblank = np.zeros((500,500,3),
dtype='uint8')cv.imshow('Blank', blank)# 1. Paint the image a certain colourblank[200:300, 300:400]
= 0,0,255cv.imshow('Green', blank)# 2. Draw a Rectanglecv.rectangle(blank, (0,0),
(blank.shape[1]//2, blank.shape[0]//2), (0,255,0), thickness=-1)cv.imshow('Rectangle', blank)# 3.
Draw A circlecv.circle(blank, (blank.shape[1]//2, blank.shape[0]//2), 40, (0,0,255), thickness=-
1)cv.imshow('Circle', blank)# 4. Draw a linecv.line(blank, (100,250), (300,400), (255,255,255),
thickness=3)cv.imshow('Line', blank)# 5. Write textcv.putText(blank, 'Hello, my name is Jason!!!',
(0,225), cv.FONT_HERSHEY_TRIPLEX, 1.0, (0,255,0), 2)cv.imshow('Text', blank)cv.waitKey(0)
```

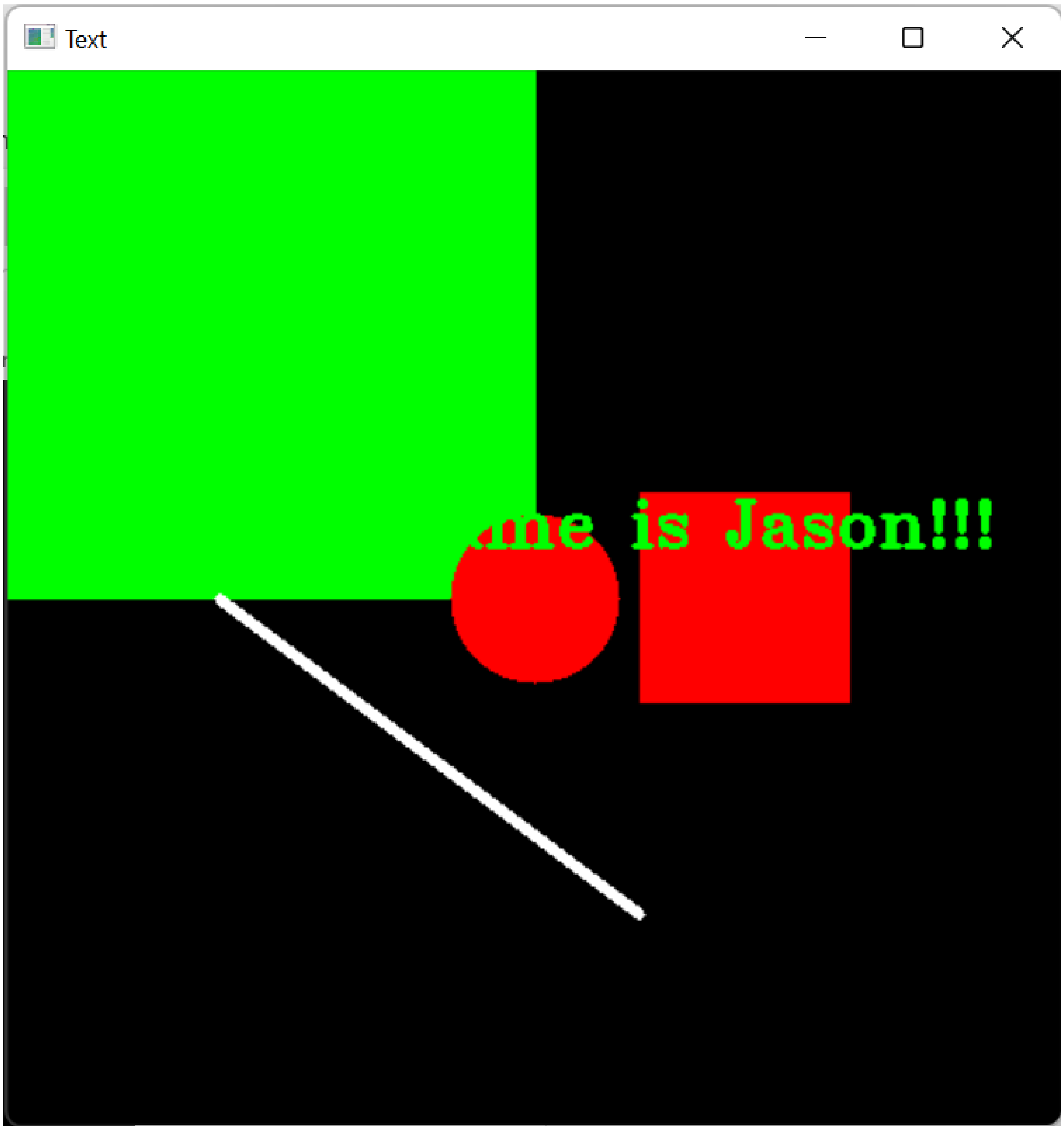



Circle


Green

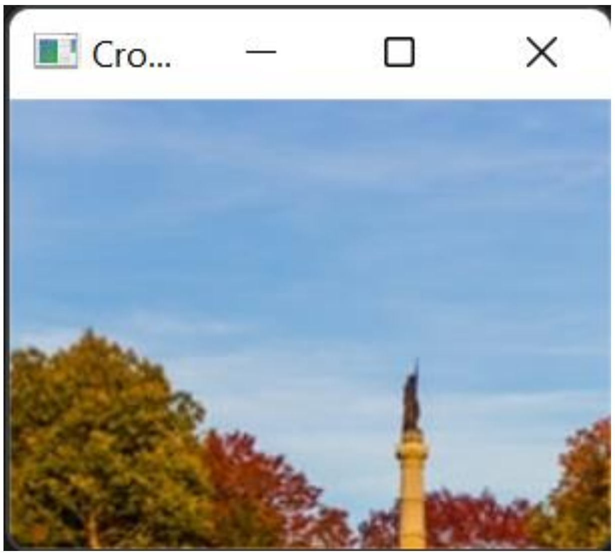

Line

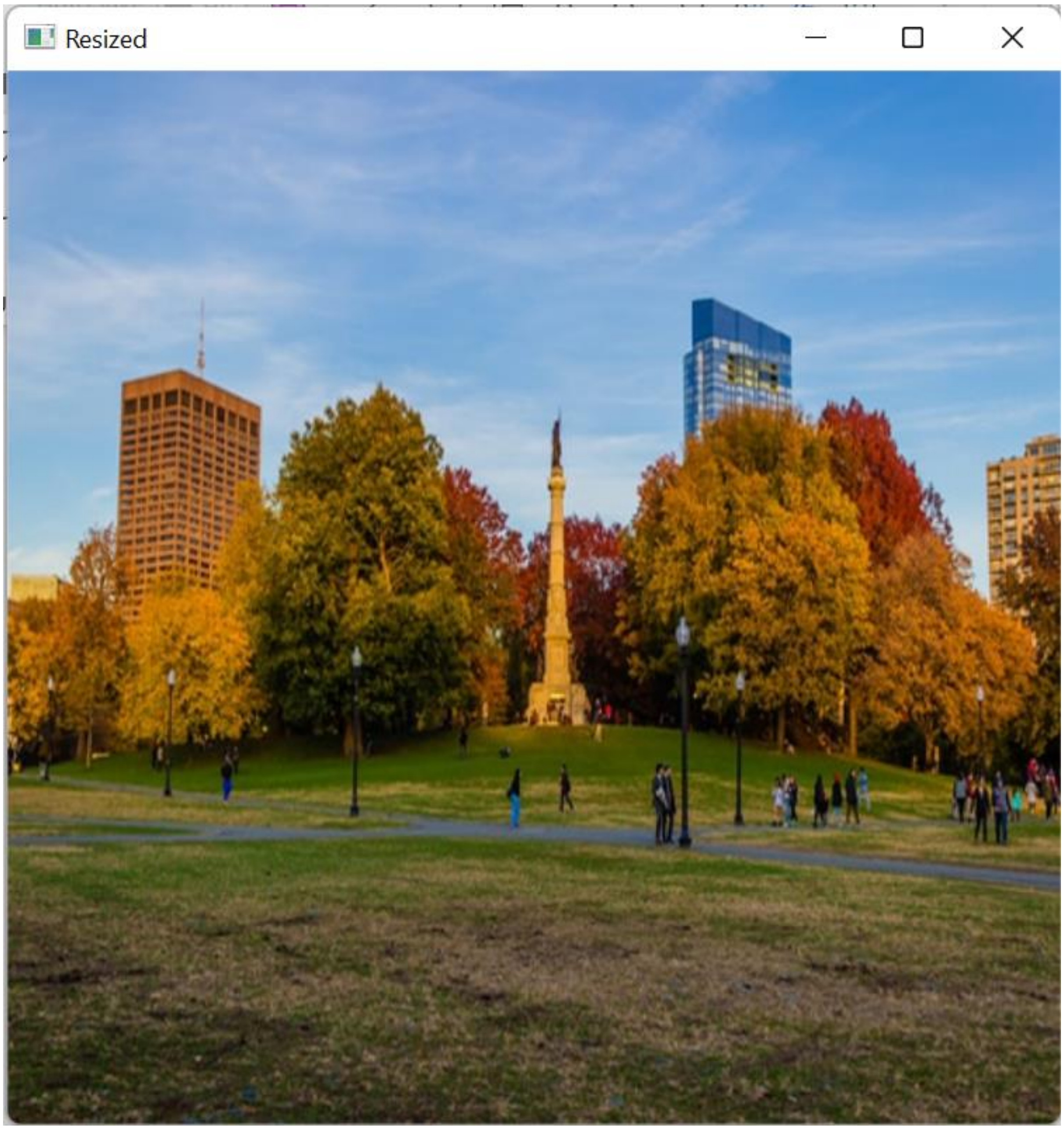

Rectangle

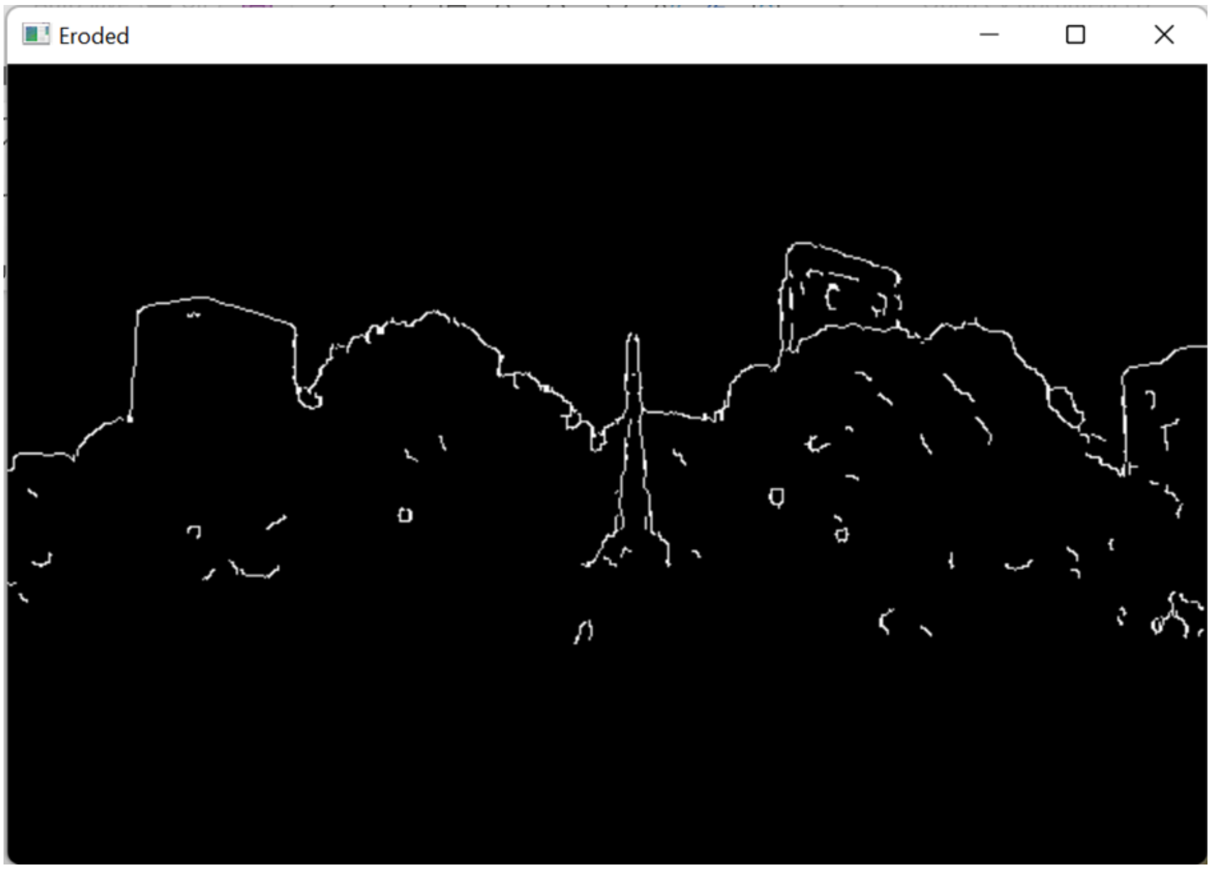


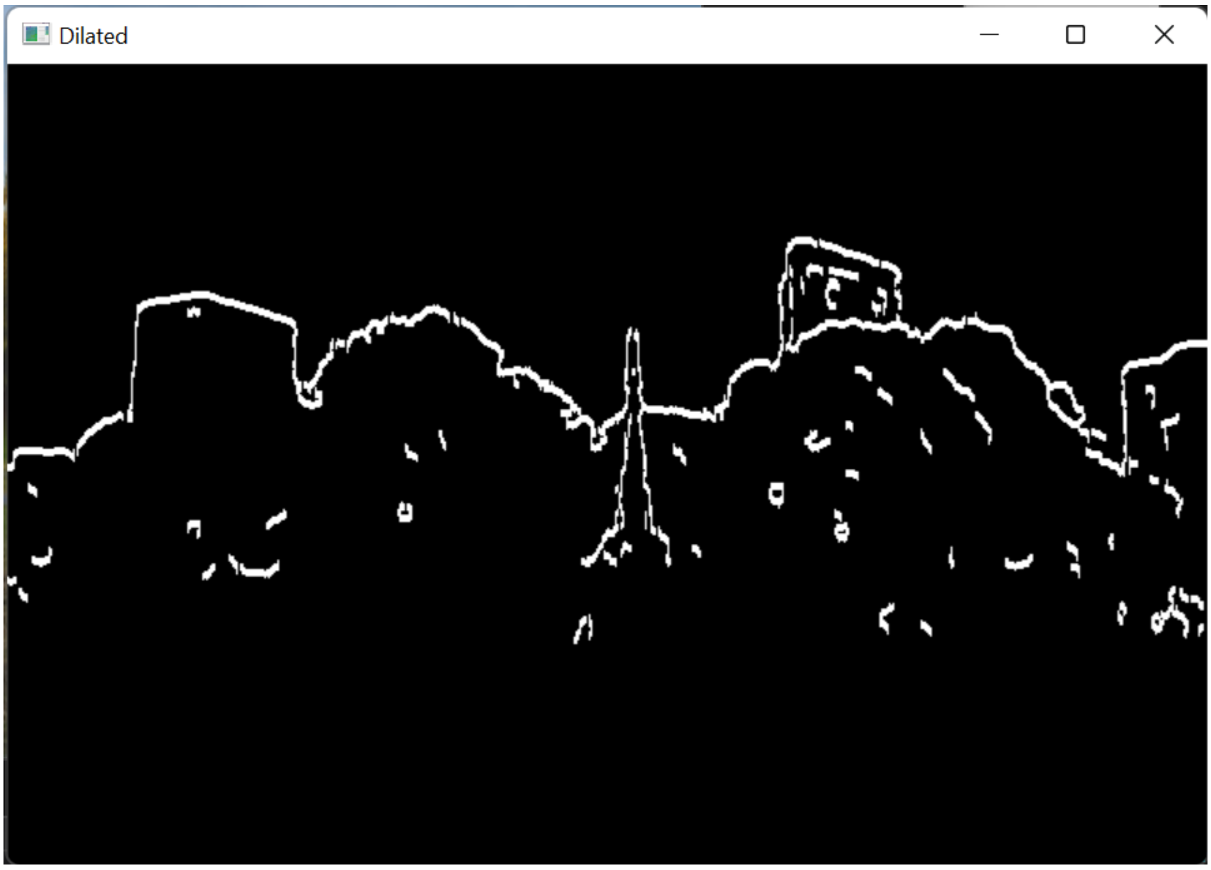
Basics Functions

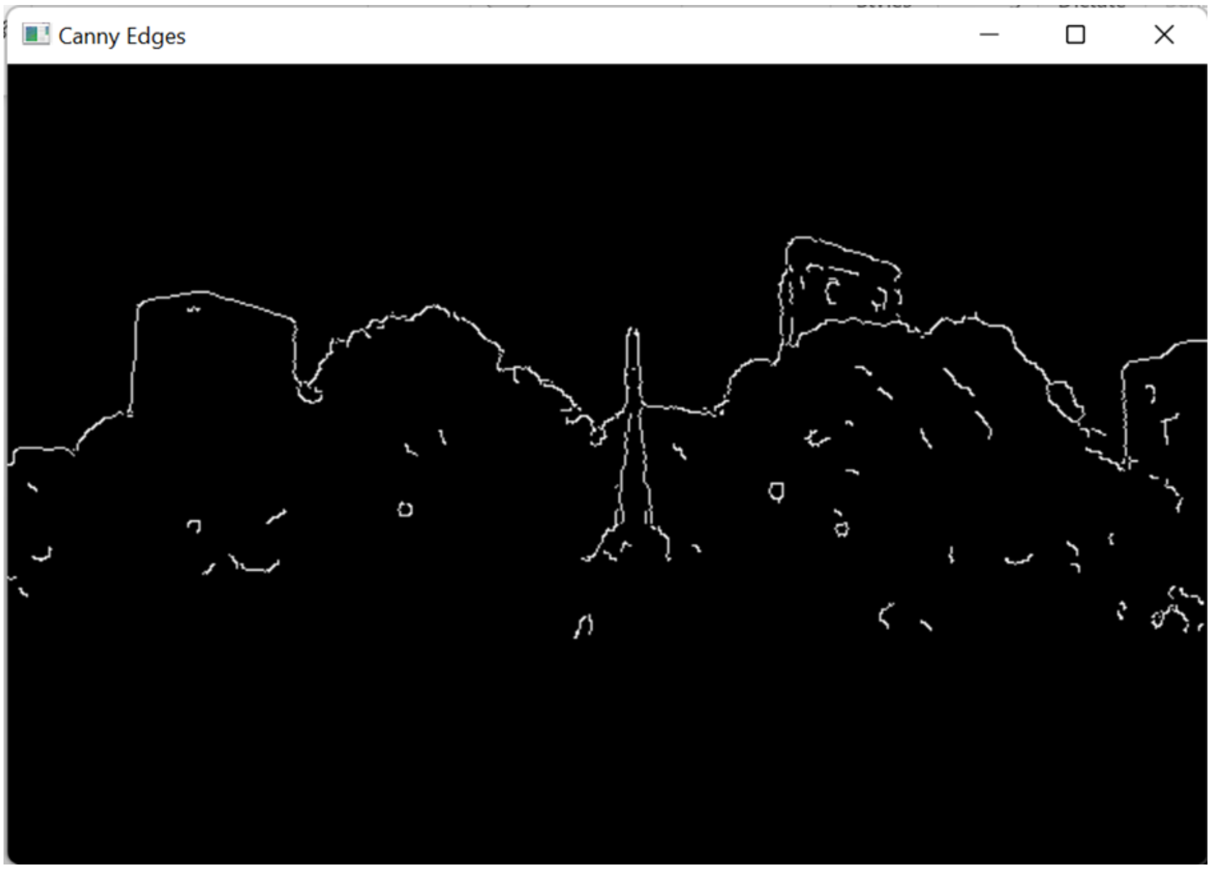
#pylint:disable=no-memberimport cv2 as cv# Read in an imageimg = cv.imread('../Resources/Photos/park.jpg')cv.imshow('Park', img)# Converting to grayscalegray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)cv.imshow('Gray', gray)# Blurblur = cv.GaussianBlur(img, (7,7), cv.BORDER_DEFAULT)cv.imshow('Blur', blur)# Edge Cascadecanny = cv.Canny(blur, 125, 175)cv.imshow('Canny Edges', canny)# Dilating the imagedilated = cv.dilate(canny, (7,7), iterations=3)cv.imshow('Dilated', dilated)# Erodingeroded = cv.erode(dilated, (7,7), iterations=3)cv.imshow('Eroded', eroded)# Resizeresized = cv.resize(img, (500,500), interpolation=cv.INTER_CUBIC)cv.imshow('Resized', resized)# Croppingcropped = img[50:200, 200:400]cv.imshow('Cropped', cropped)

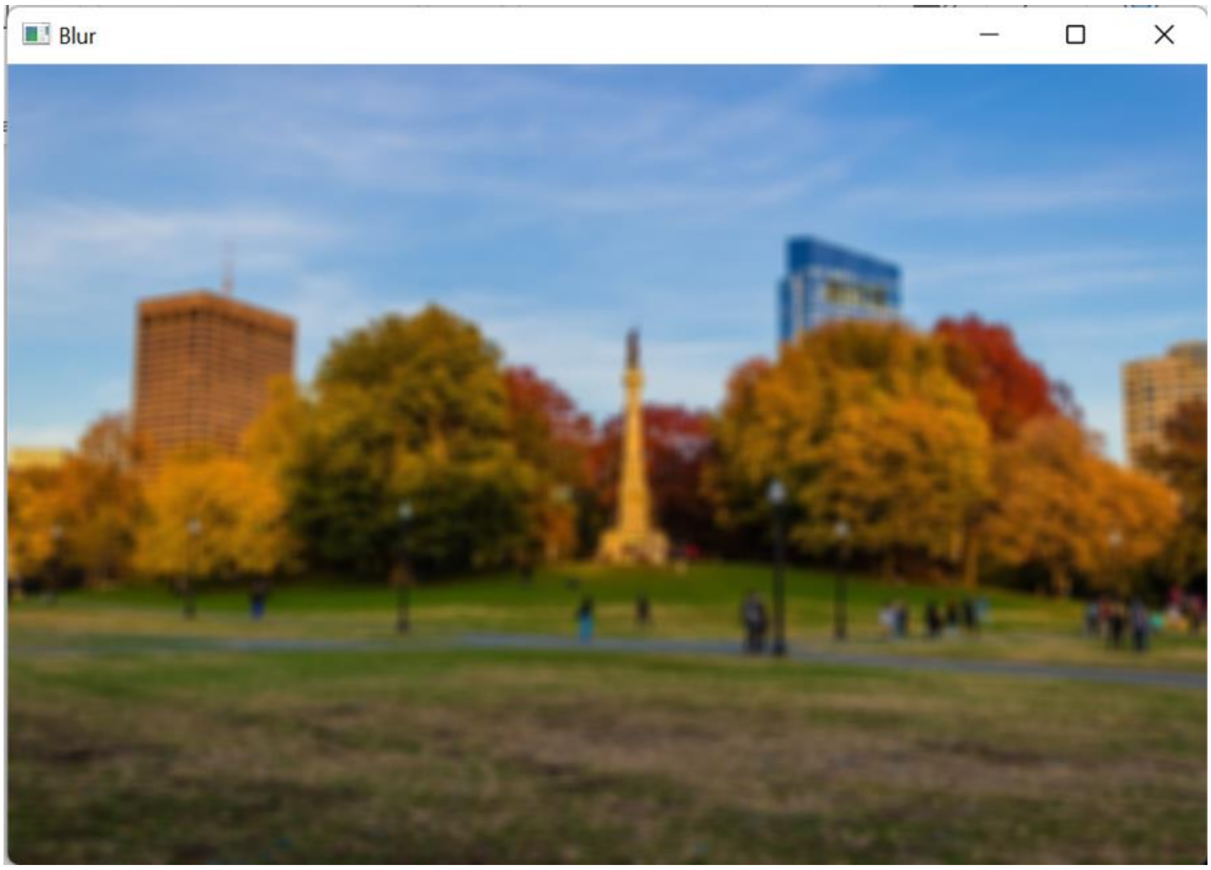

Cro...

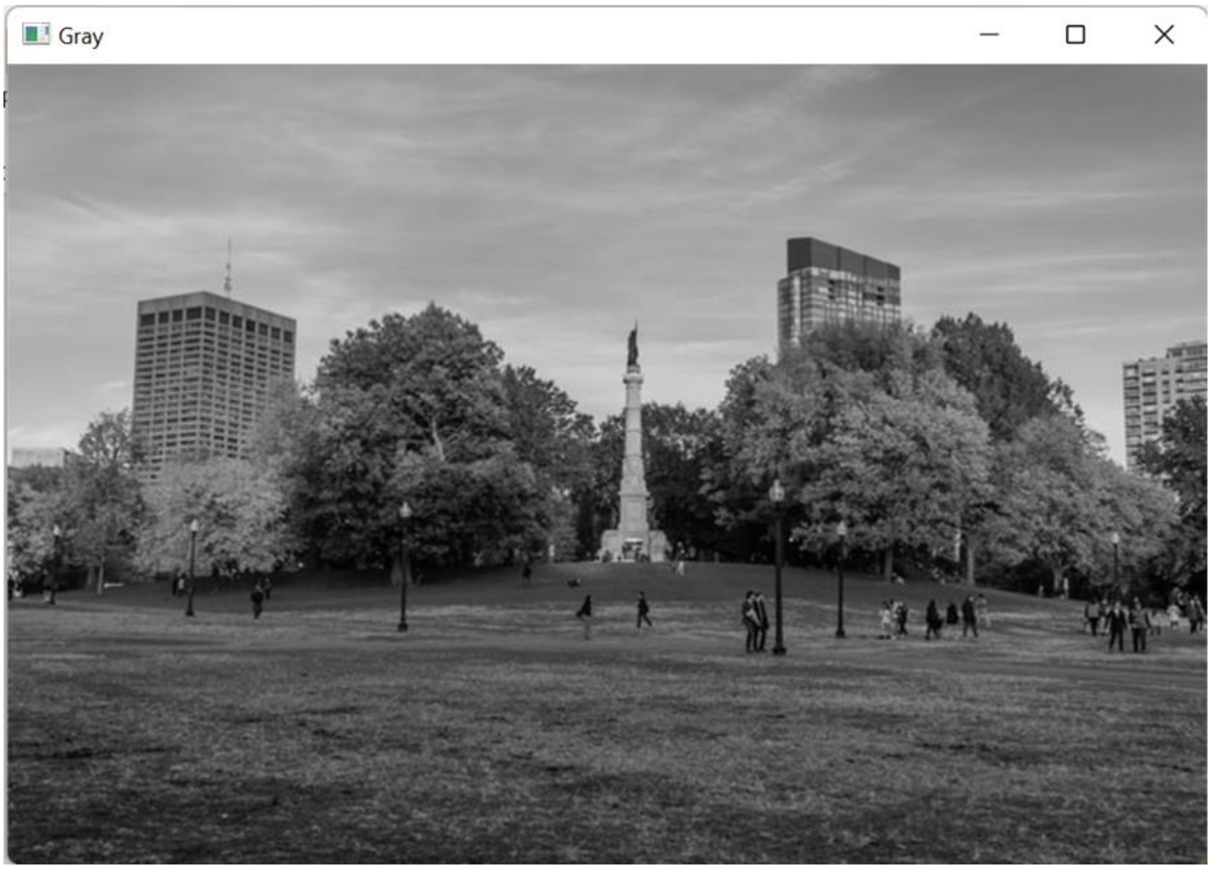

Resized

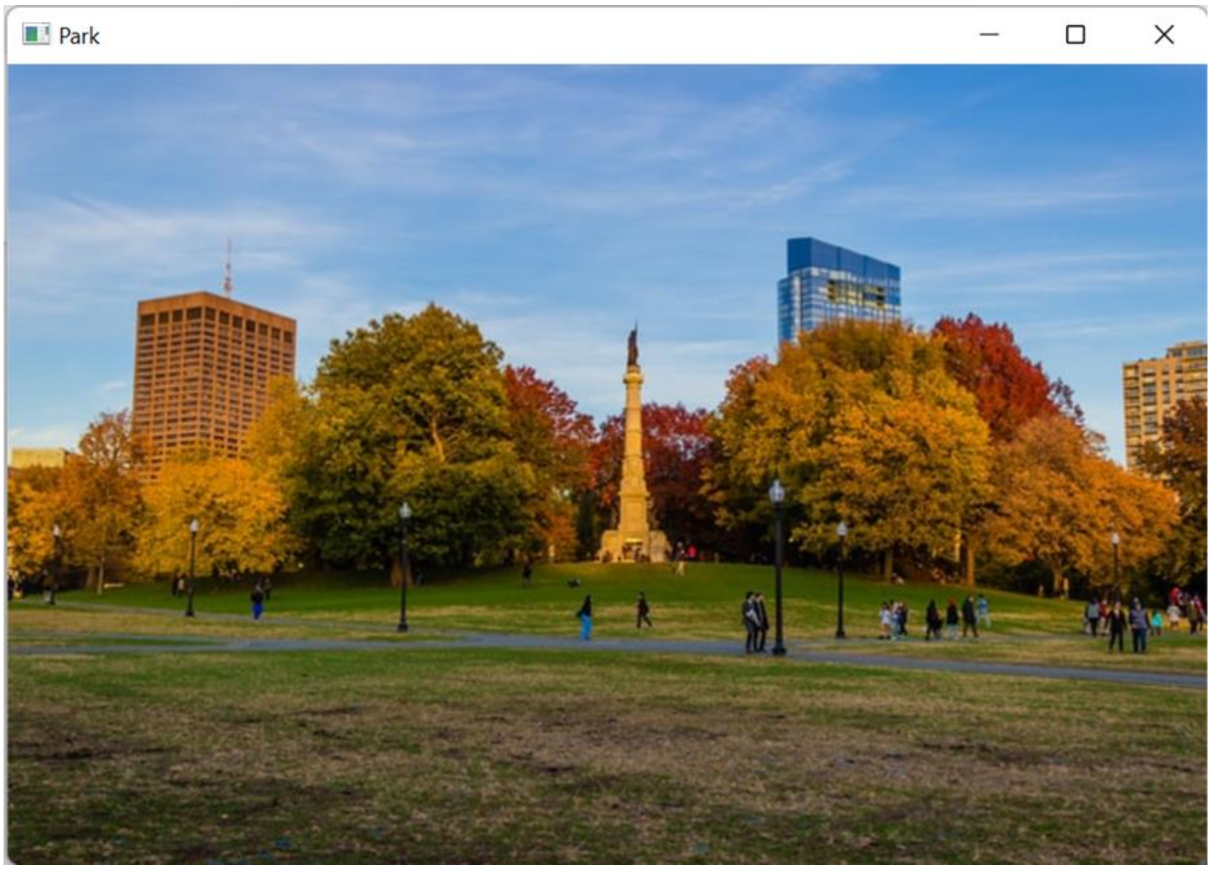

Eroded


Dilated


Canny Edges

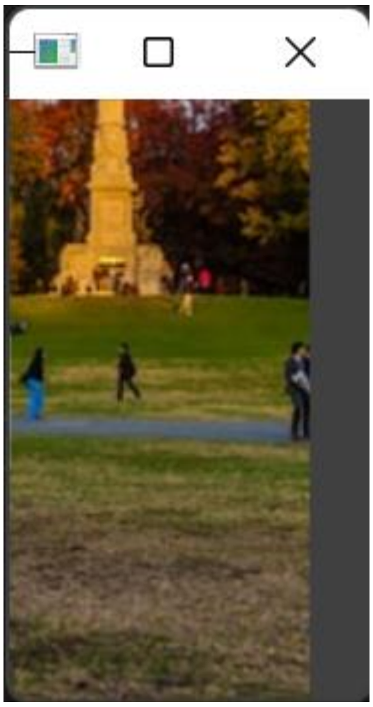

Blur

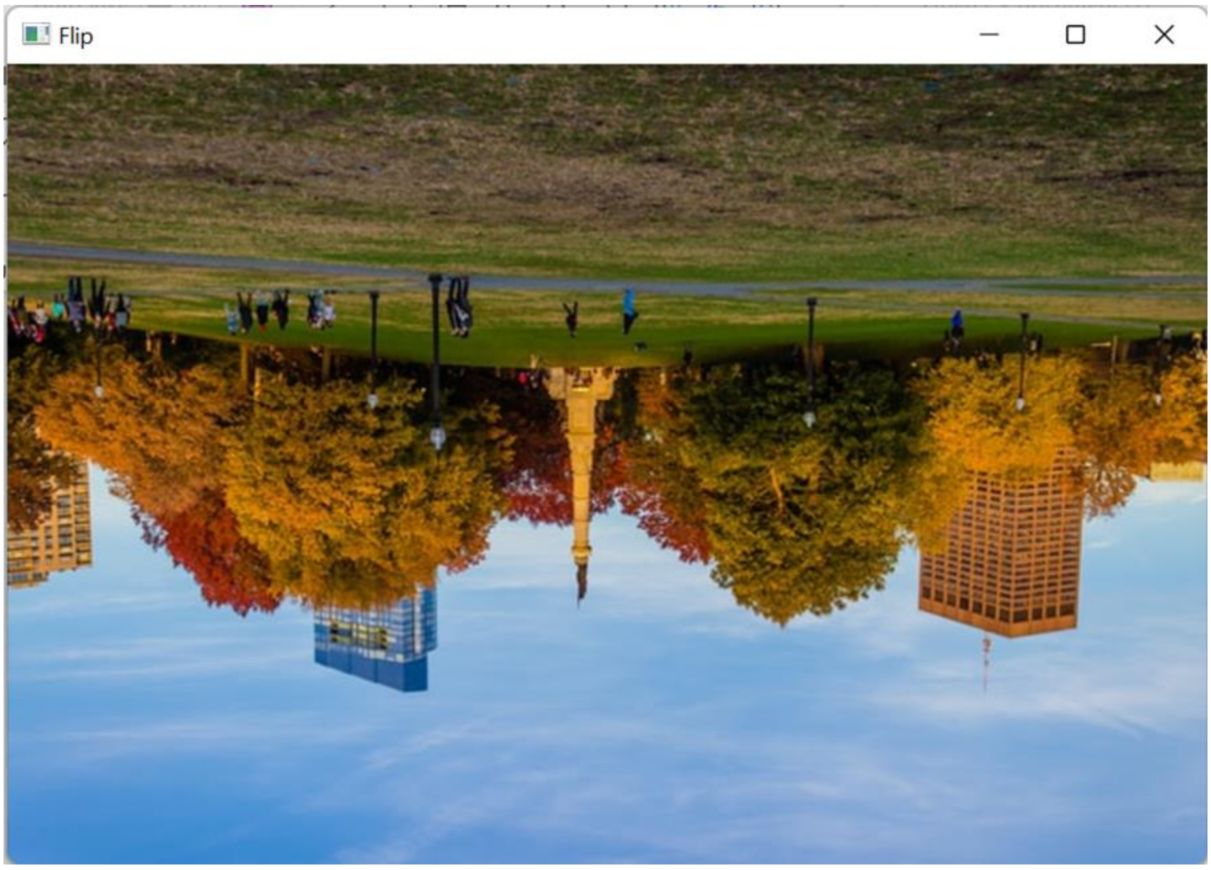


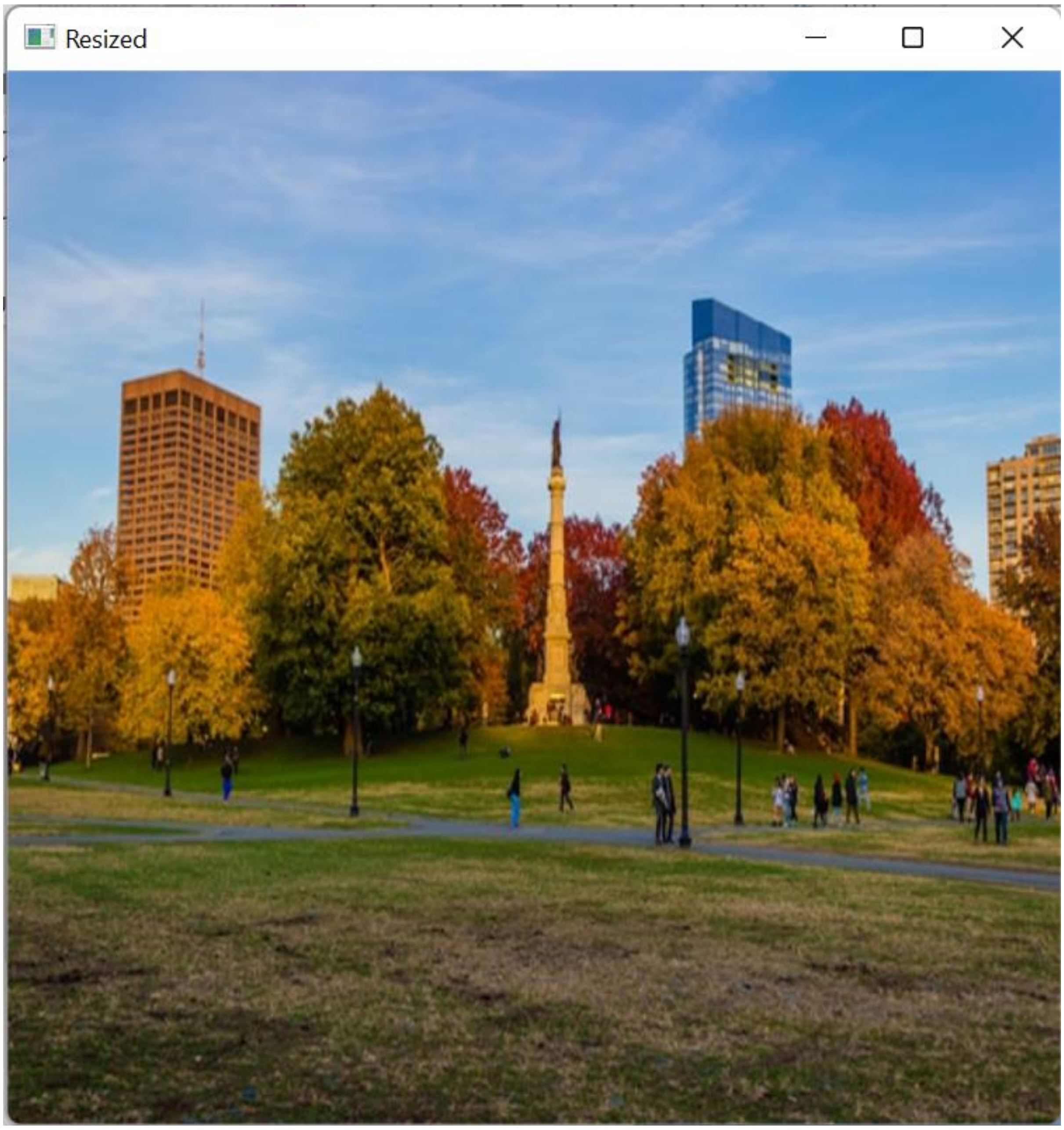


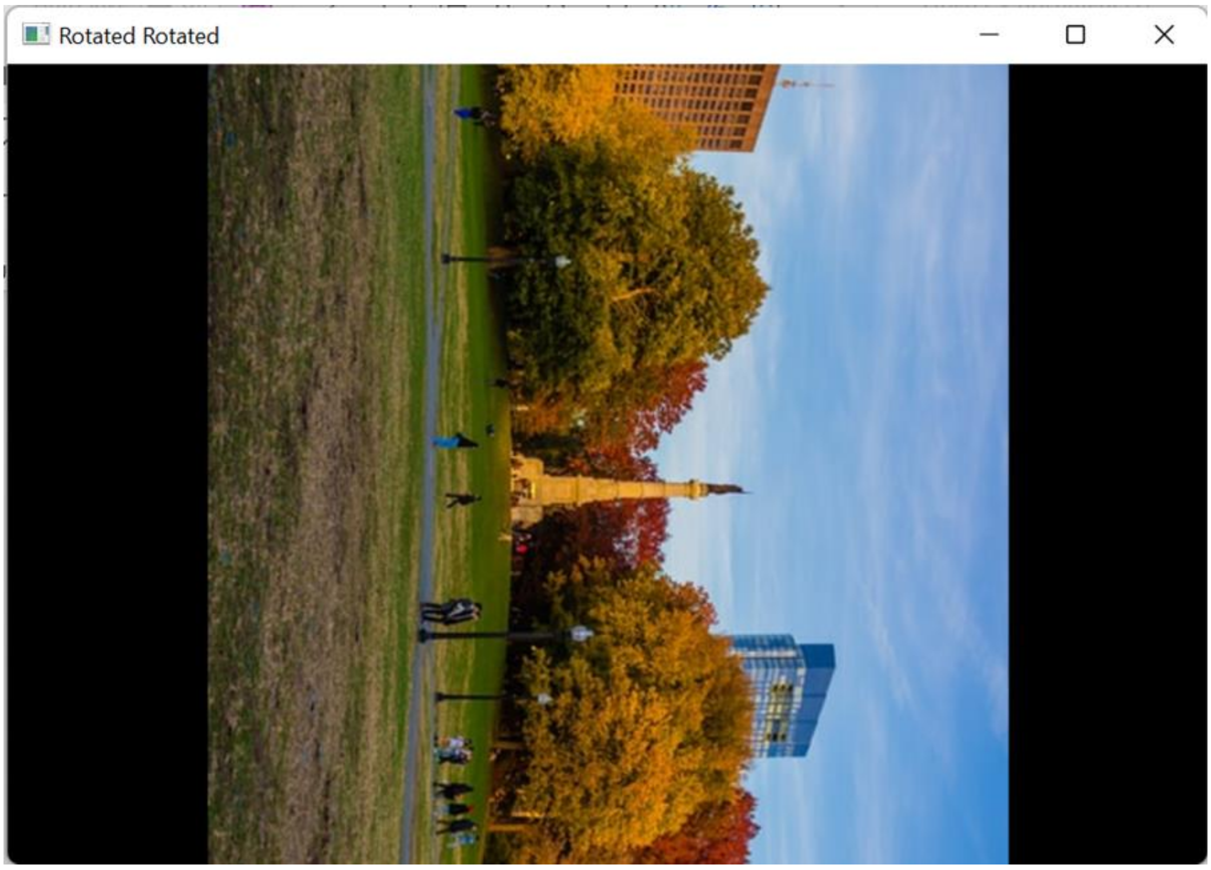
Transformations

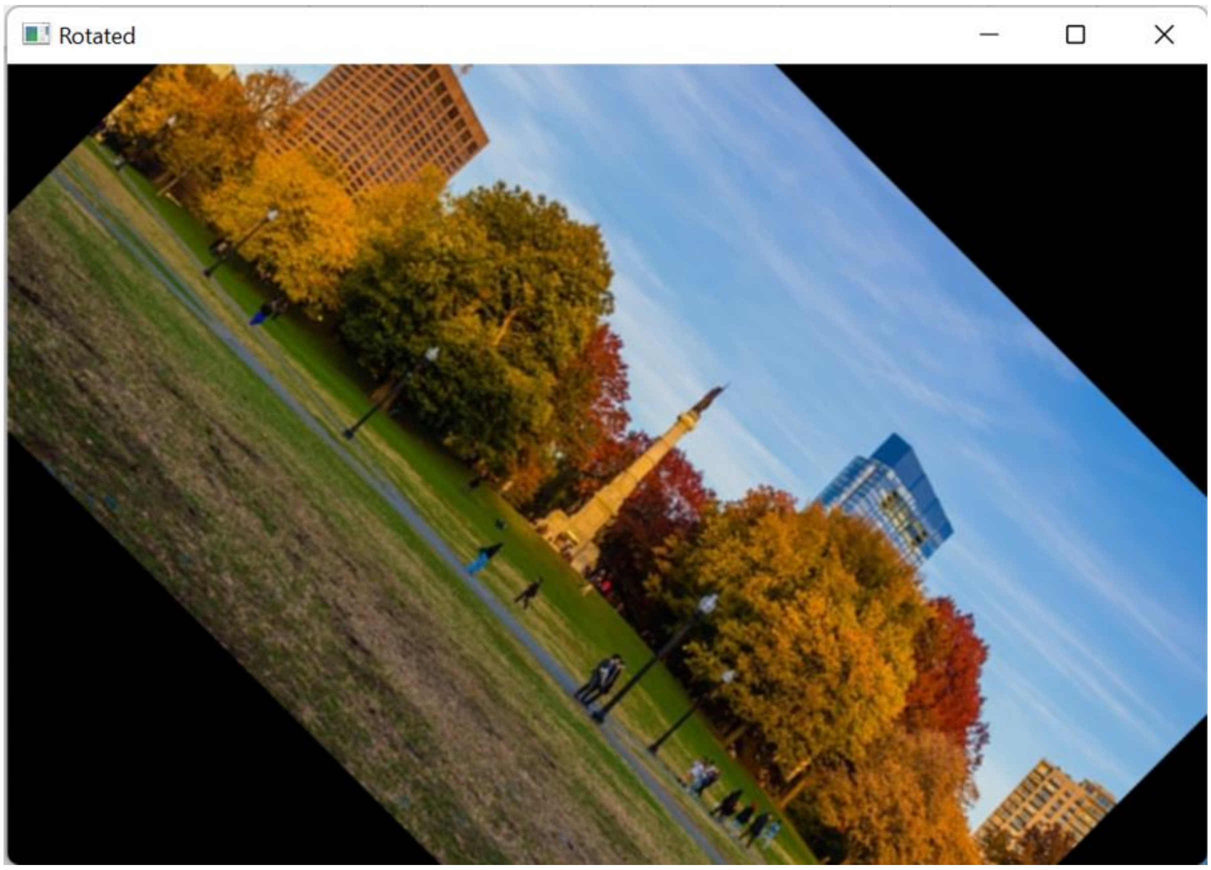
#pylint:disable=no-memberimport cv2 as cvimport numpy as npimg = cv.imread('../Resources/Photos/park.jpg')cv.imshow('Park', img)# Translationdef translate(img, x, y):transMat = np.float32([[1,0,x],[0,1,y]])dimensions = (img.shape[1], img.shape[0])return cv.warpAffine(img, transMat, dimensions)# -x --> Left# -y --> Up# x --> Right# y --> Downtranslated = translate(img, -100, 100)cv.imshow('Translated', translated)# Rotationdef rotate(img, angle, rotPoint=None):(height,width) = img.shape[:2]if rotPoint is None:rotPoint = (width//2,height//2)rotMat = cv.getRotationMatrix2D(rotPoint, angle, 1.0)dimensions = (width,height)return cv.warpAffine(img, rotMat, dimensions)rotated = rotate(img, -45)cv.imshow('Rotated', rotated)rotated_rotated = rotate(img, -90)cv.imshow('Rotated Rotated', rotated_rotated)# Resizingresized = cv.resize(img, (500,500), interpolation=cv.INTER_CUBIC)cv.imshow('Resized', resized)# Flippingflip = cv.flip(img, -1)cv.imshow('Flip', flip)# Croppingcropped = img[200:400, 300:400]cv.imshow('Cropped', cropped)cv.waitKey(0)

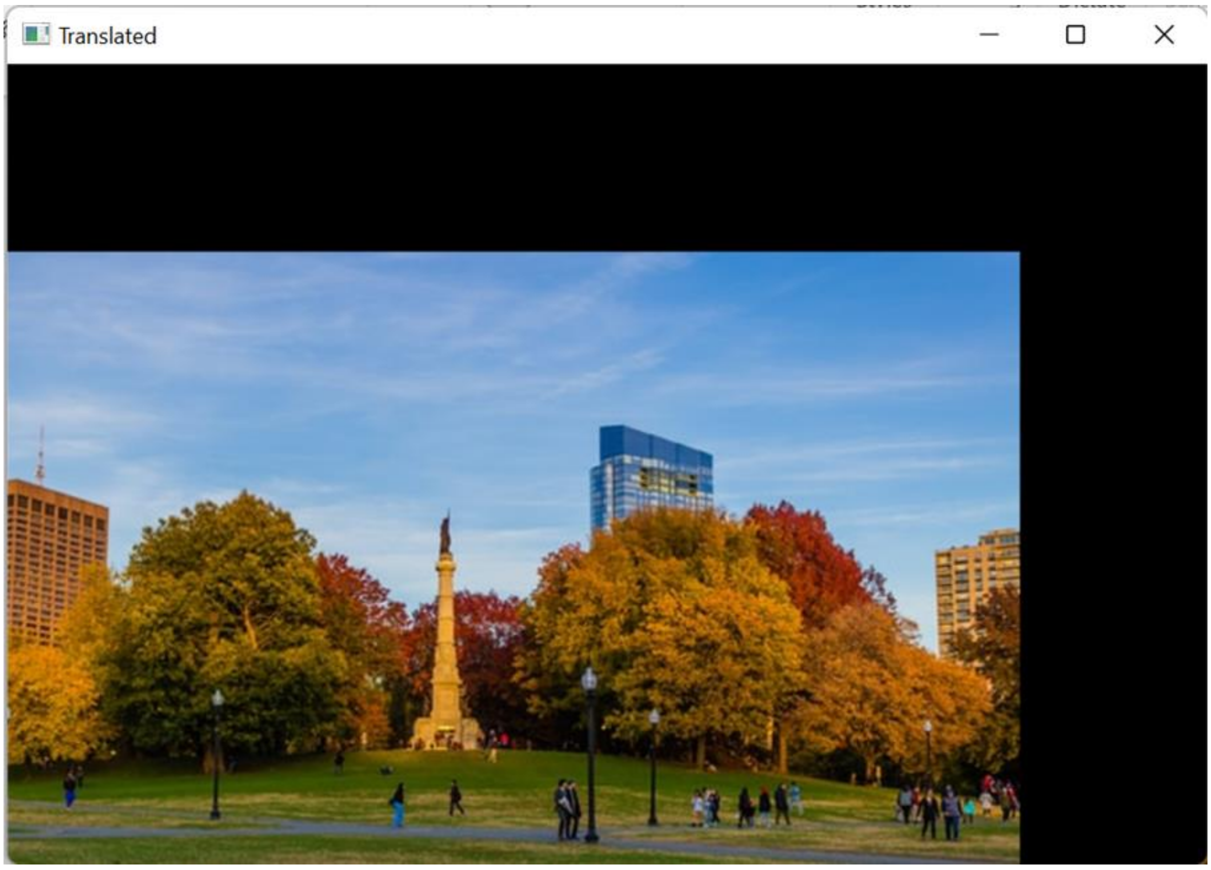


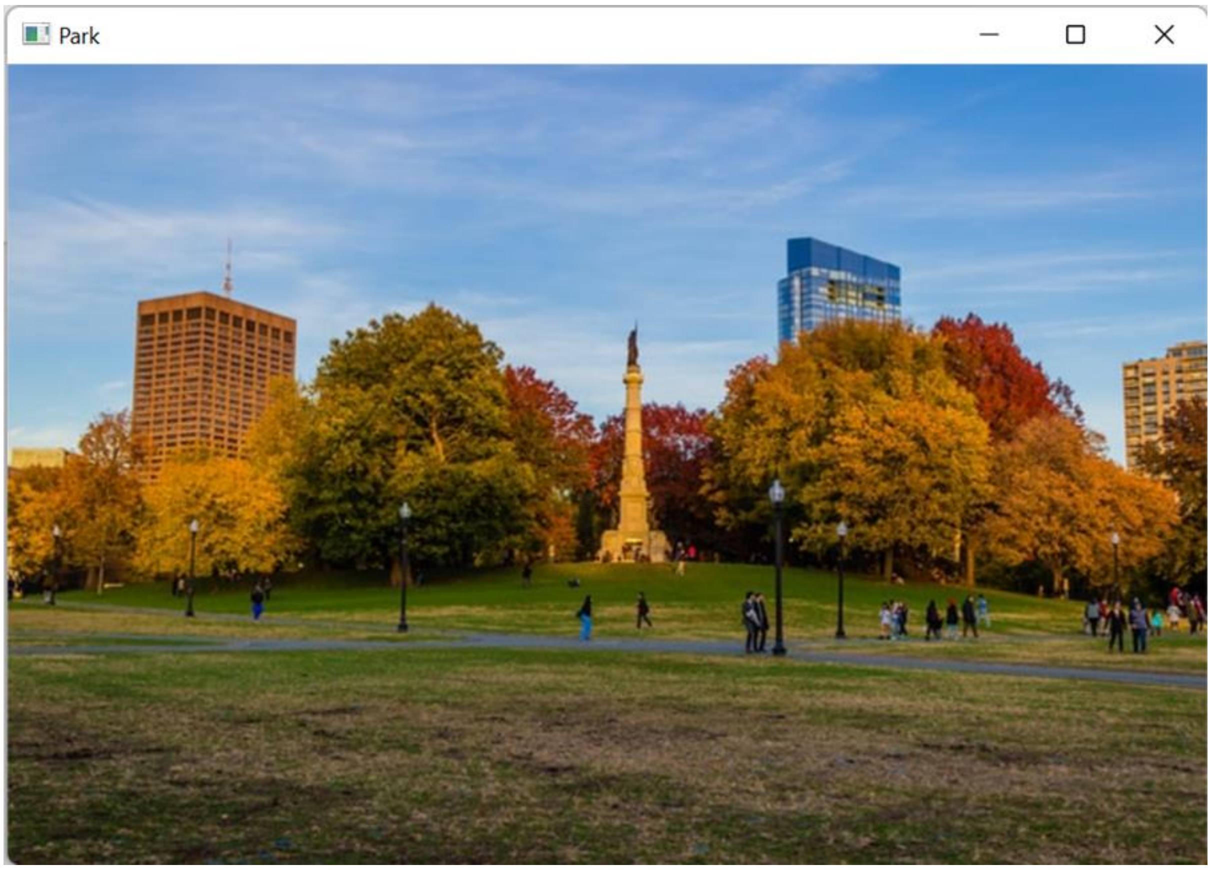

Flip


Resized


Rotated Rotated

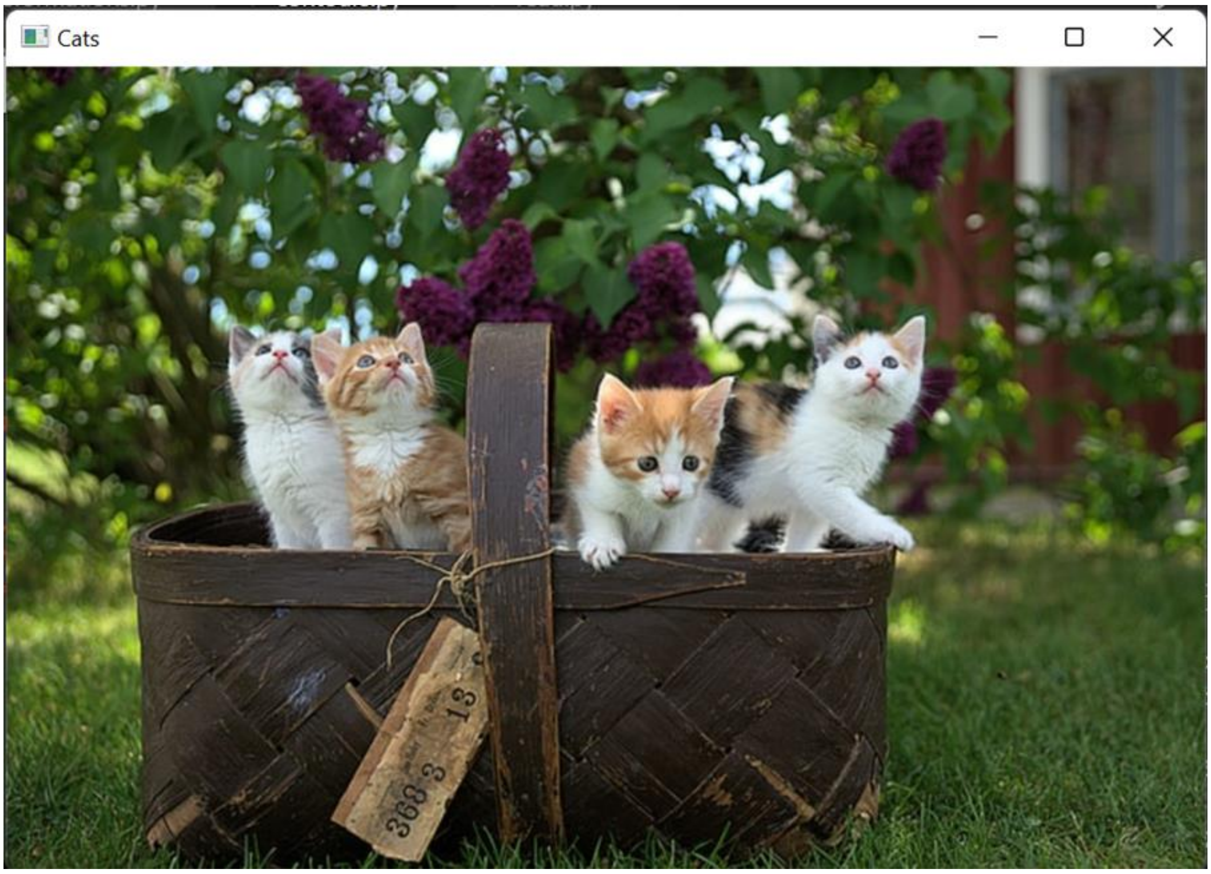

Rotated

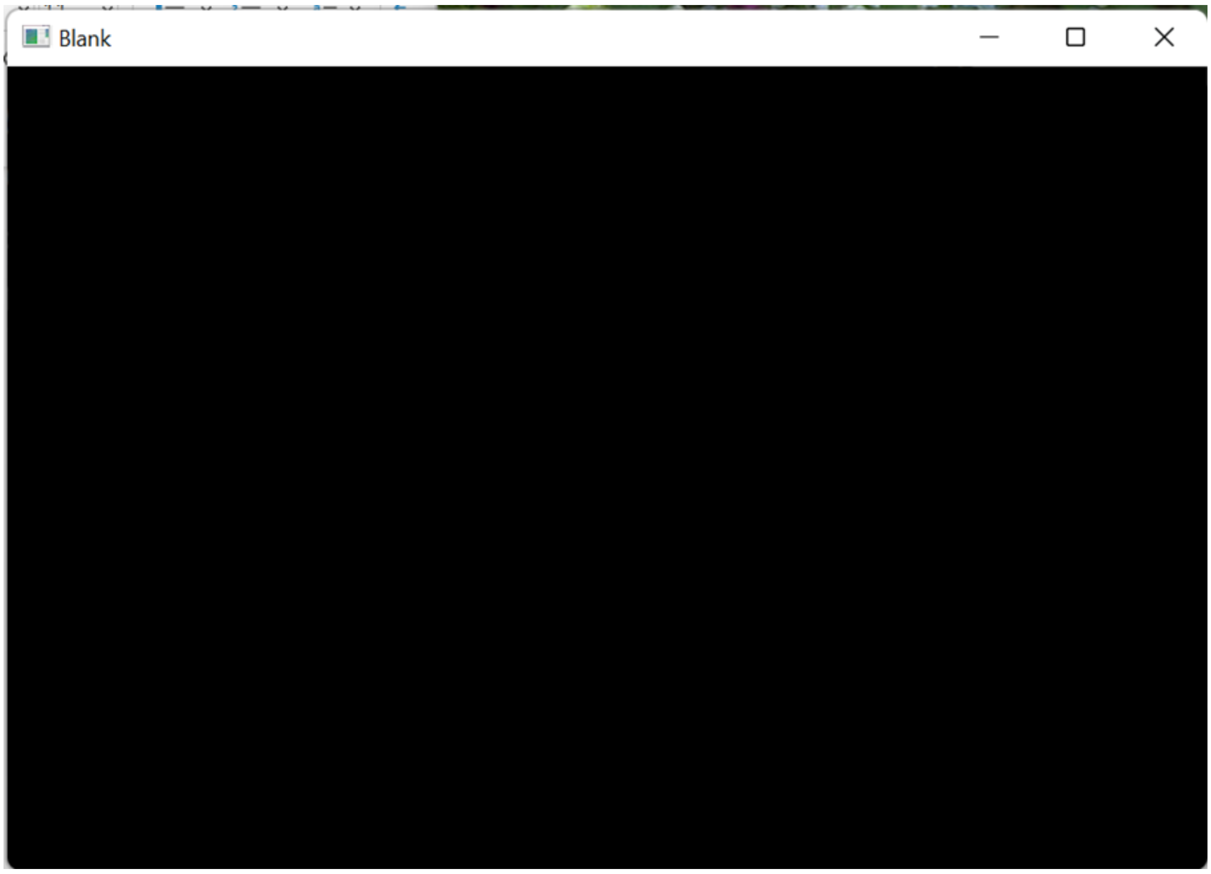


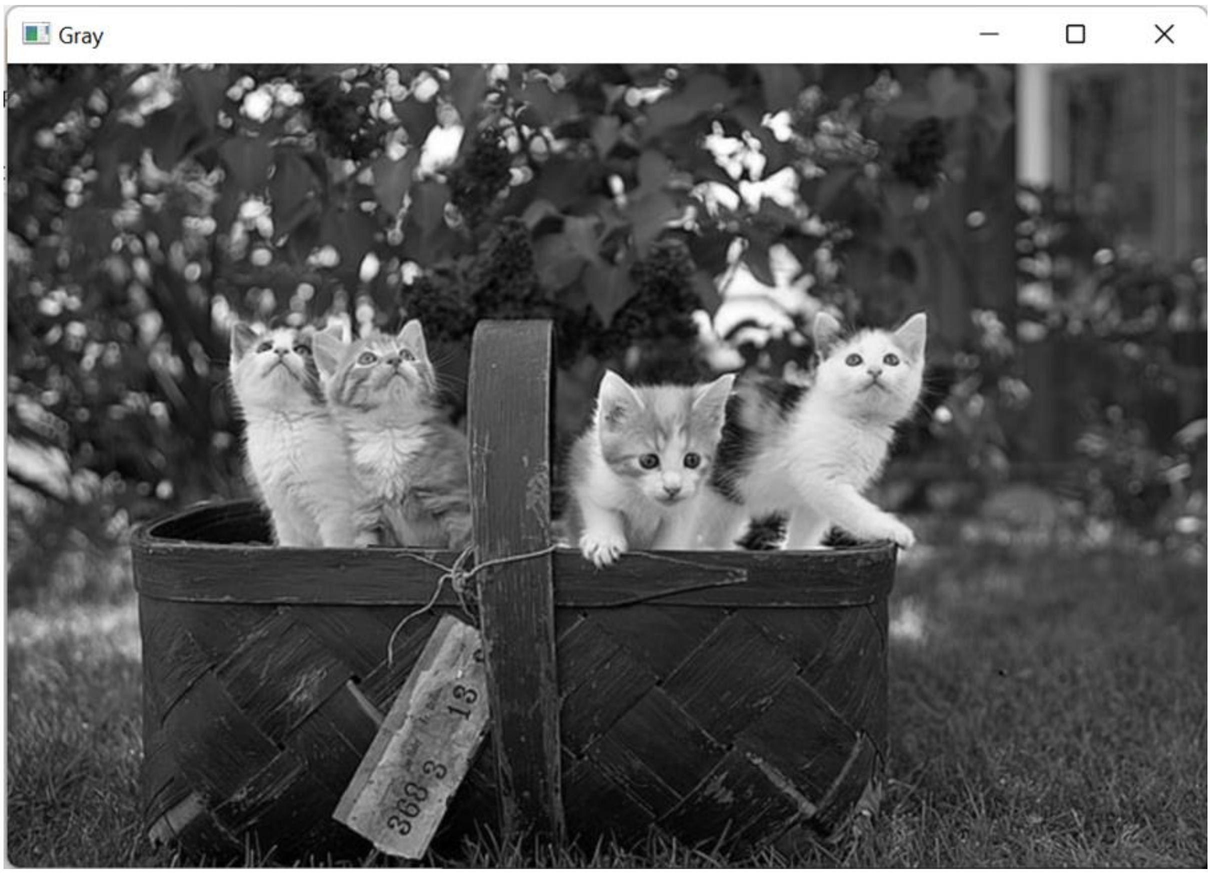


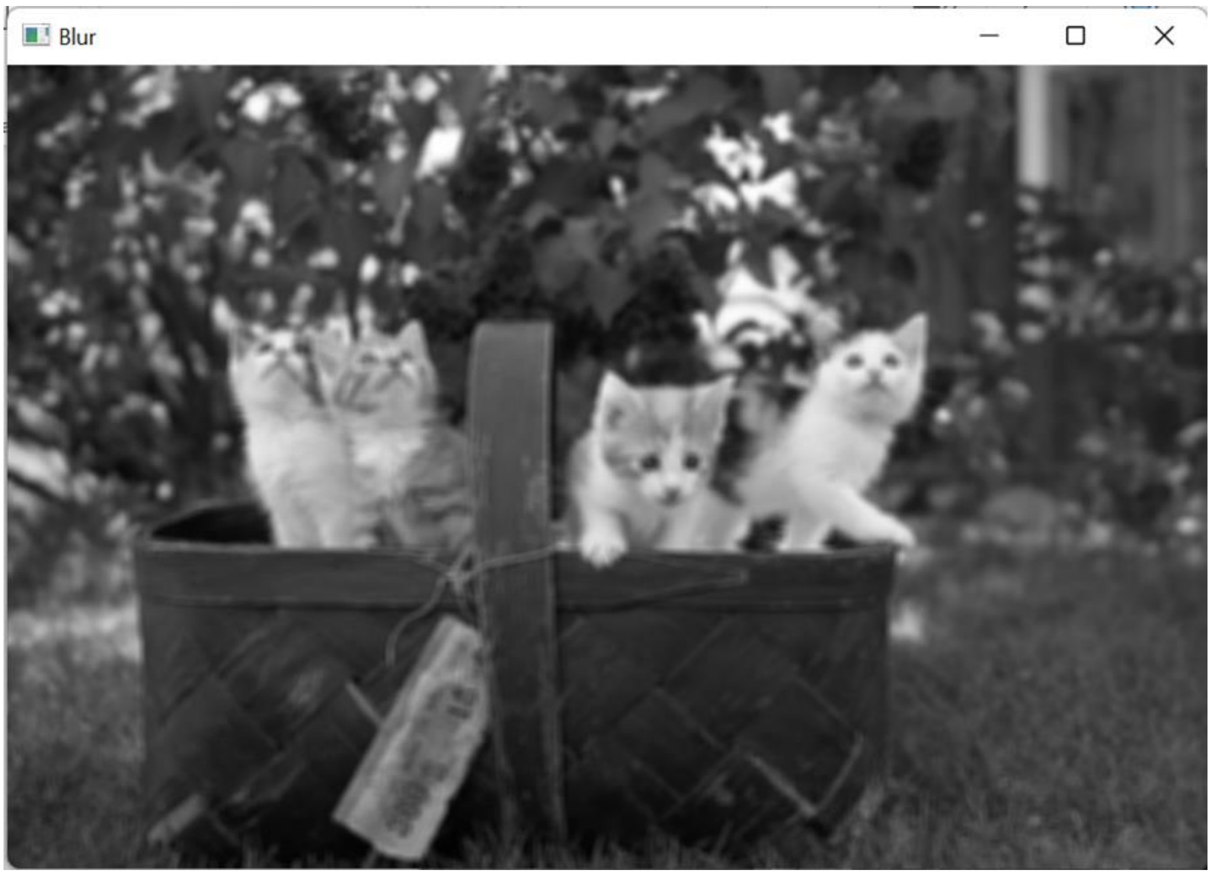
Contours

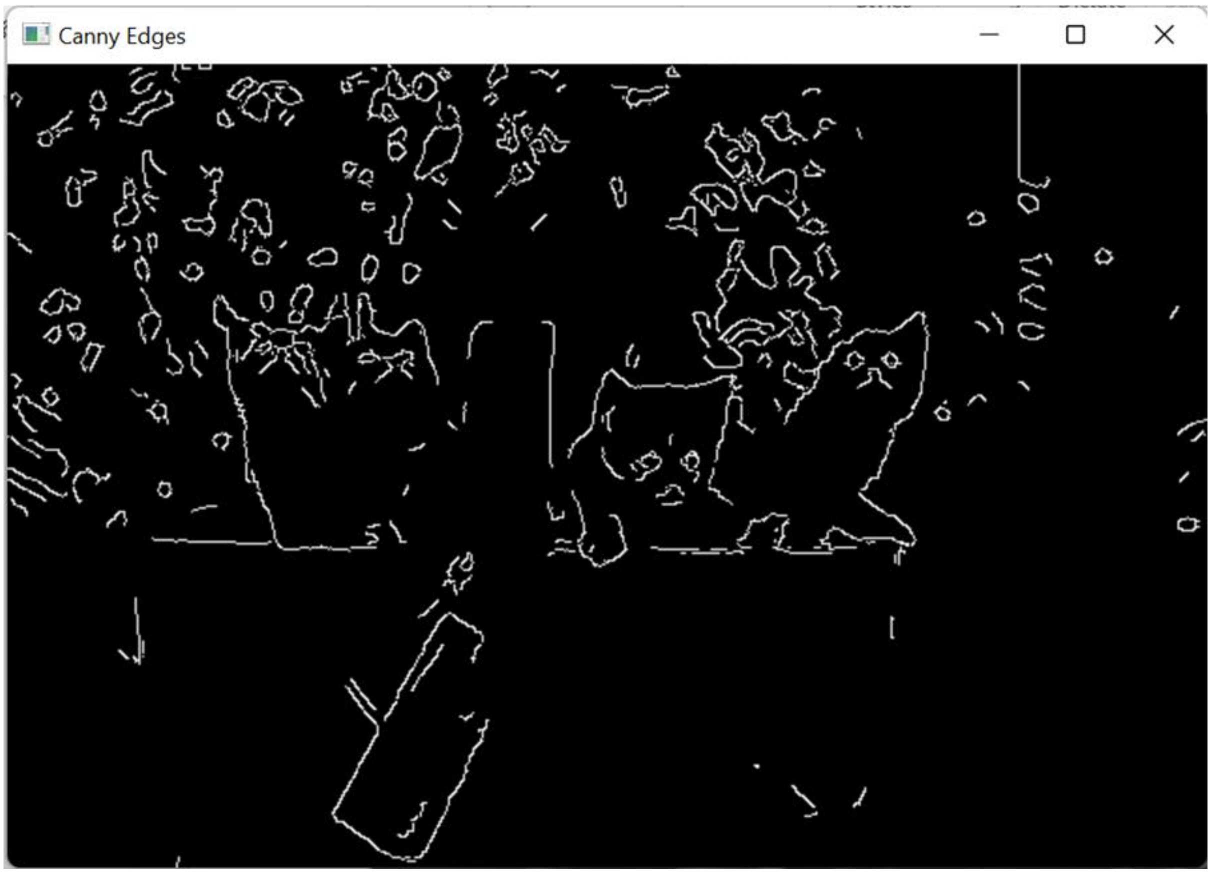
```
import cv2 as cvimport numpy as npimg =
cv.imread('../Resources/Photos/cats.jpg')cv.imshow('Cats', img)blank = np.zeros(img.shape,
dtype='uint8')cv.imshow('Blank', blank)gray = cv.cvtColor(img,
cv.COLOR_BGR2GRAY)cv.imshow('Gray', gray)blur = cv.GaussianBlur(gray, (5,5),
cv.BORDER_DEFAULT)cv.imshow('Blur', blur)canny = cv.Canny(blur, 125, 175)cv.imshow('Canny
Edges', canny)# ret, thresh = cv.threshold(gray, 125, 255, cv.THRESH_BINARY)# cv.imshow('Thresh',
thresh)contours, hierarchies = cv.findContours(canny, cv.RETR_LIST,
cv.CHAIN_APPROX_SIMPLE)print(f'{len(contours)} contour(s) found!')cv.drawContours(blank,
contours, -1, (0,0,255), 1)cv.imshow('Contours Drawn', blank)cv.waitKey(0)
```

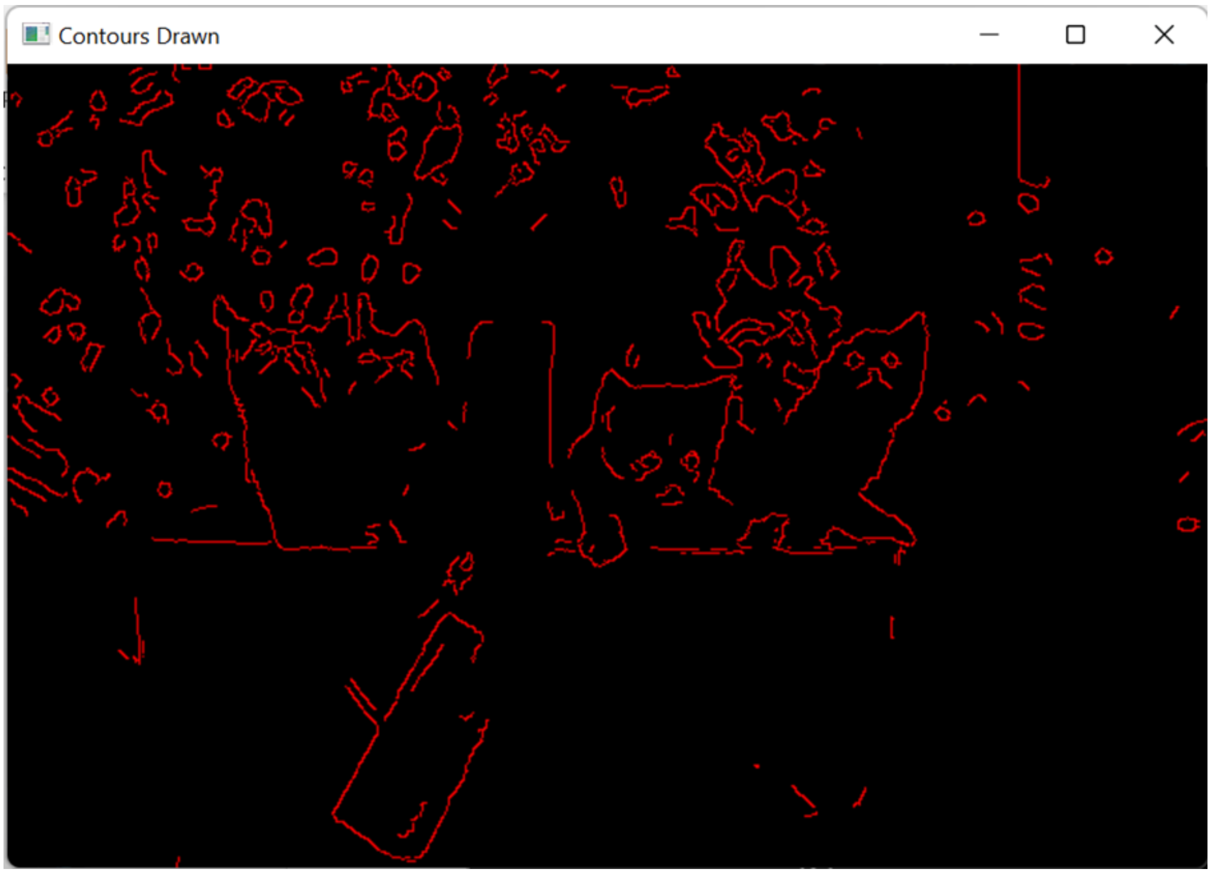



Blank


Gray

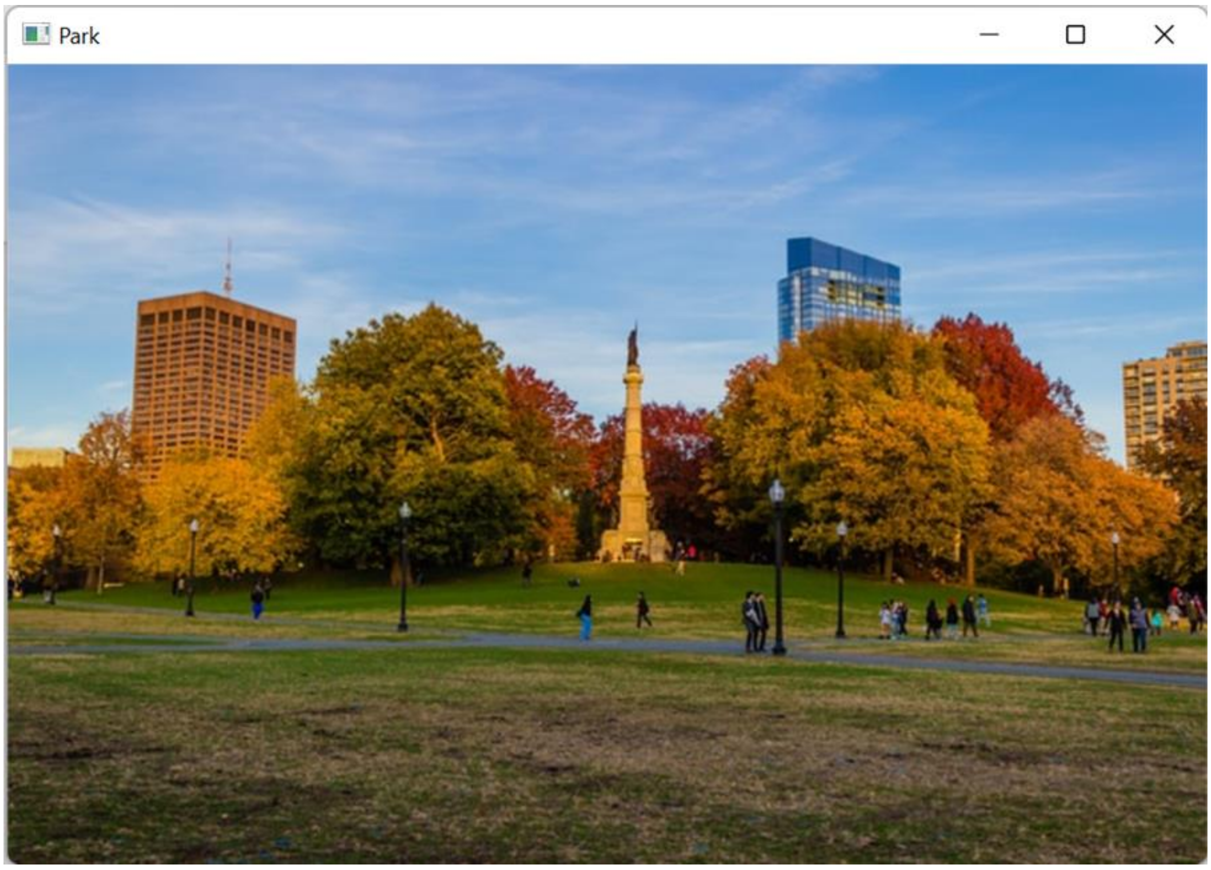

Blur

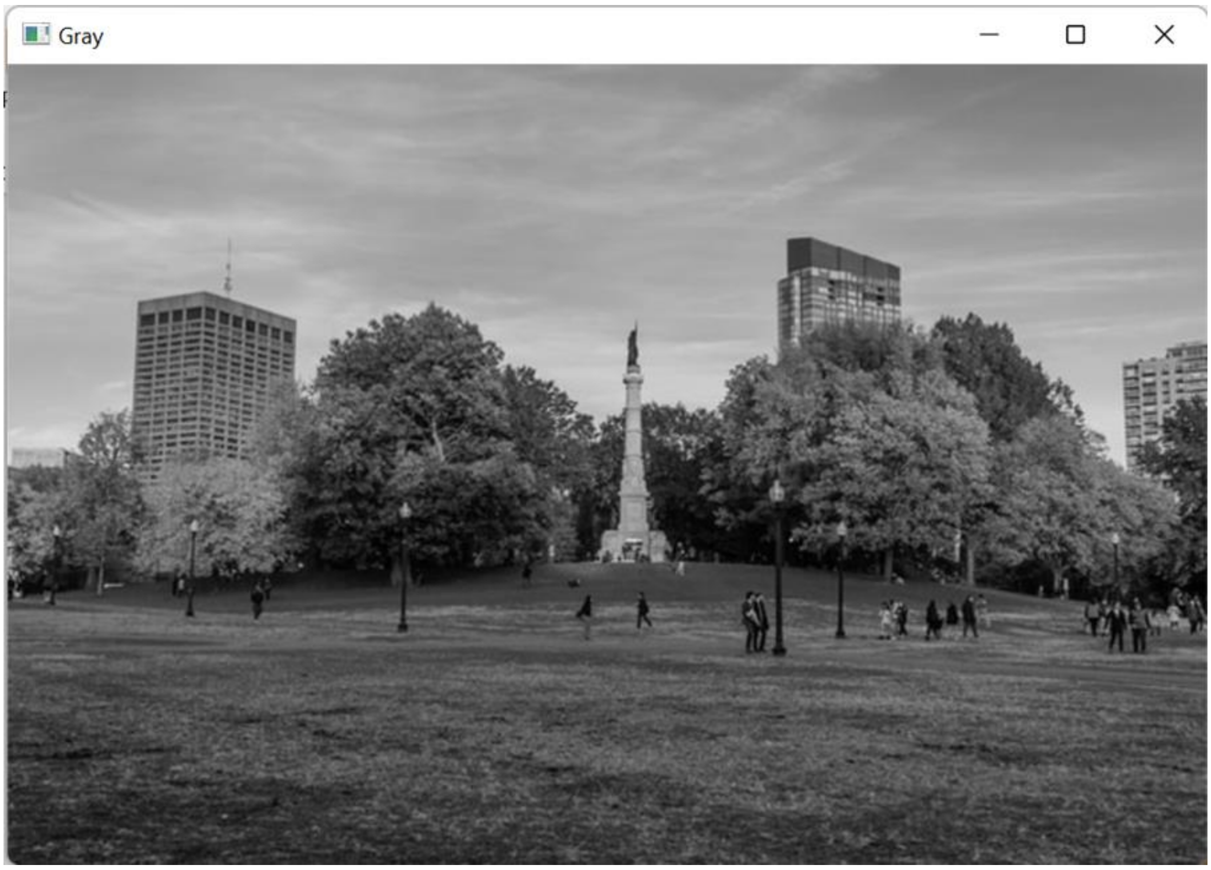

Canny Edges

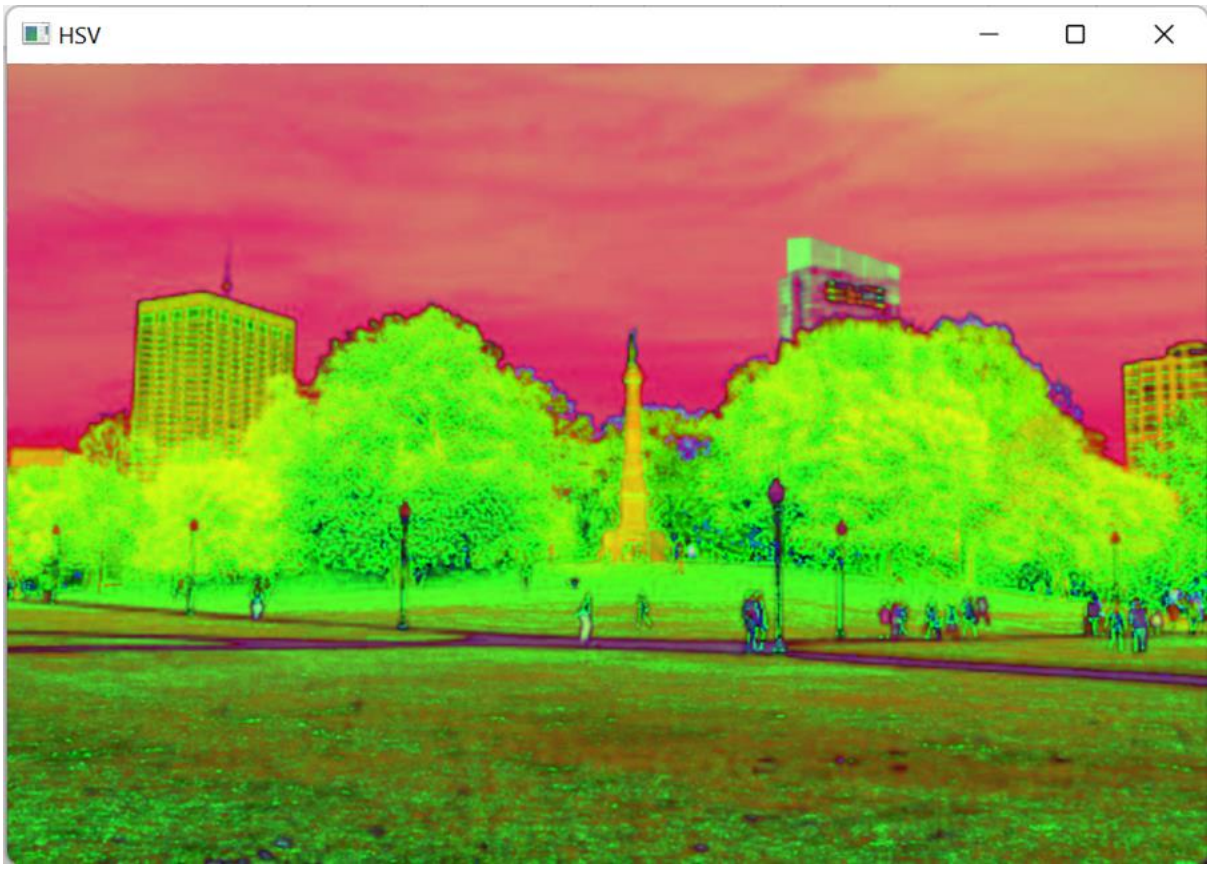


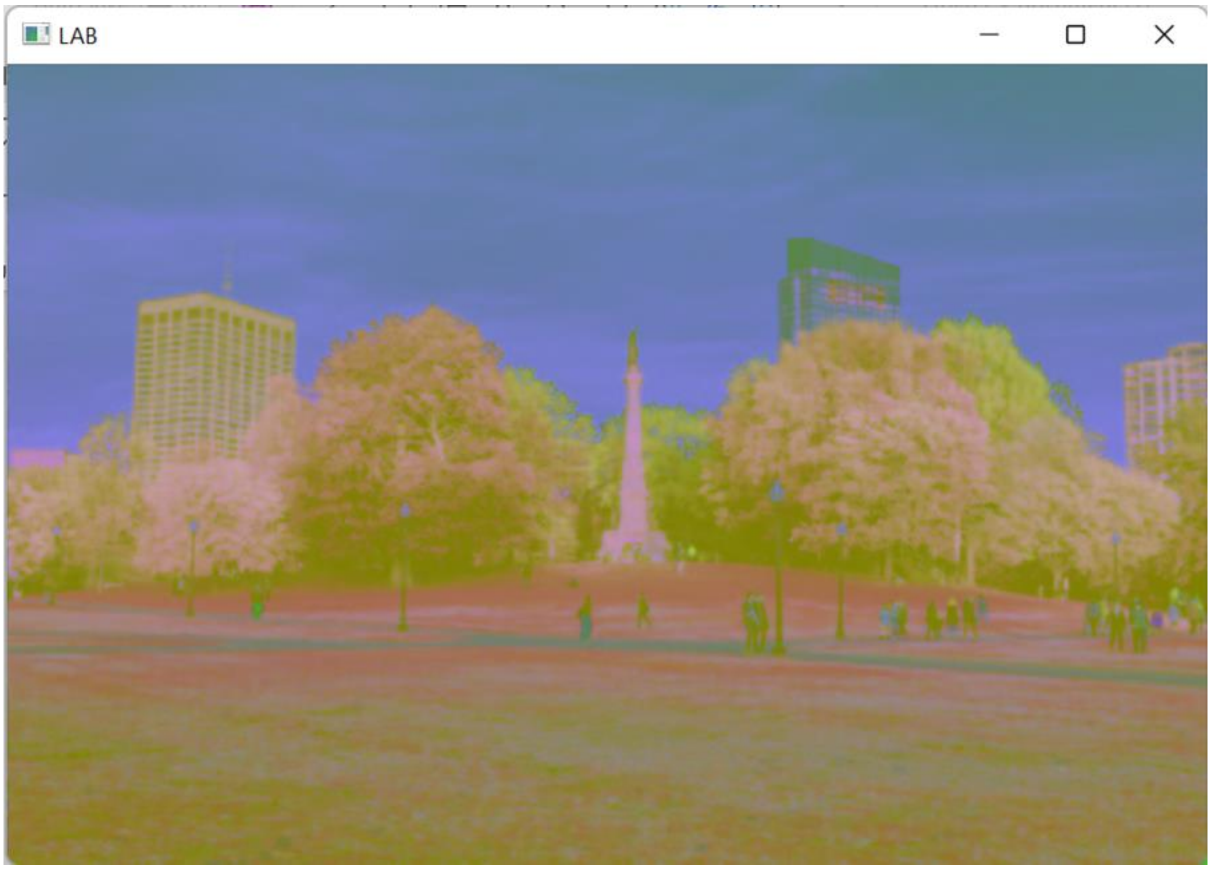
Colour Spacing

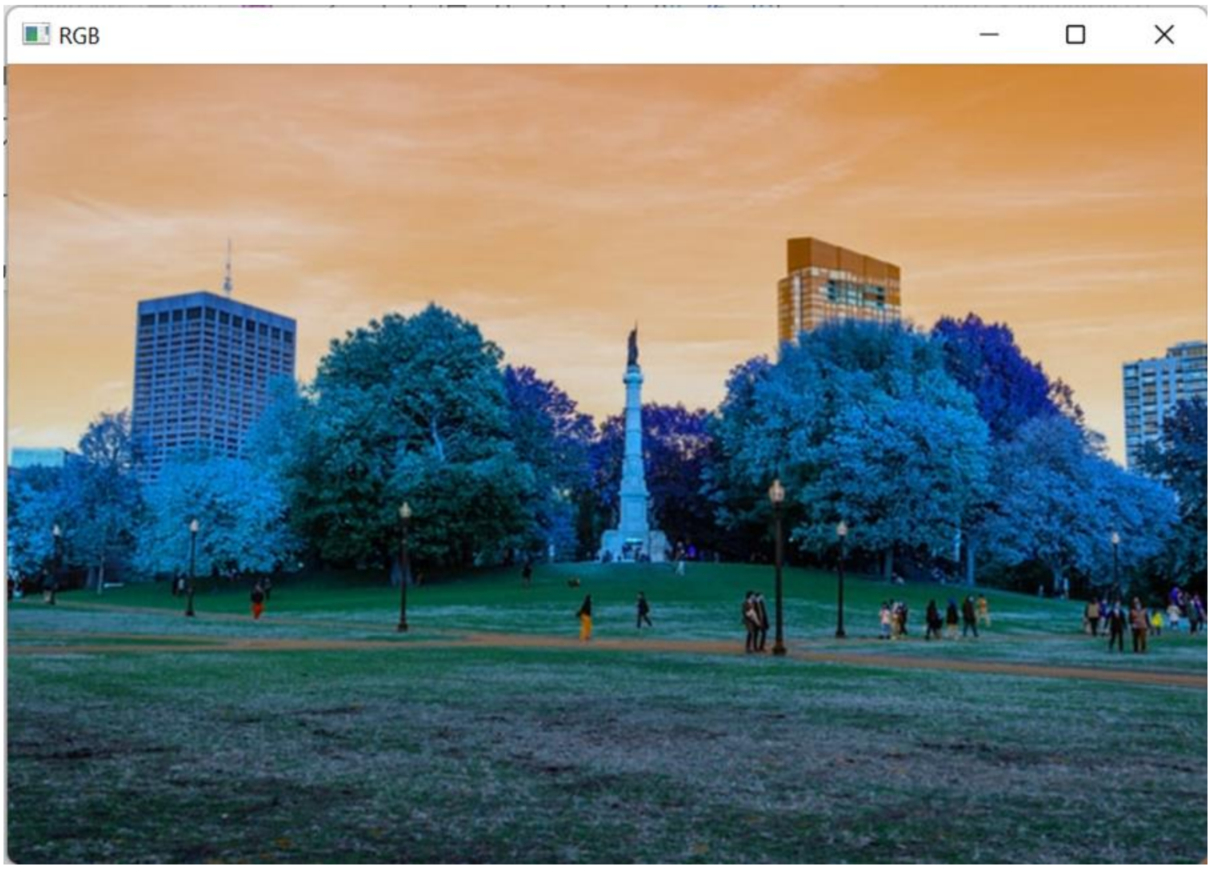
#pylint:disable=no-memberimport cv2 as cvimport matplotlib.pyplot as pltimg = cv.imread('../Resources/Photos/park.jpg')cv.imshow('Park', img)# plt.imshow(img)# plt.show()# BGR to Grayscalegray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)cv.imshow('Gray', gray)# BGR to HSVhsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)cv.imshow('HSV', hsv)# BGR to L*a*blab = cv.cvtColor(img, cv.COLOR_BGR2LAB)cv.imshow('LAB', lab)# BGR to RGBrgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)cv.imshow('RGB', rgb)# HSV to BGRlab_bgr = cv.cvtColor(lab, cv.COLOR_LAB2BGR)cv.imshow('LAB --> BGR', lab_bgr)cv.waitKey(0)

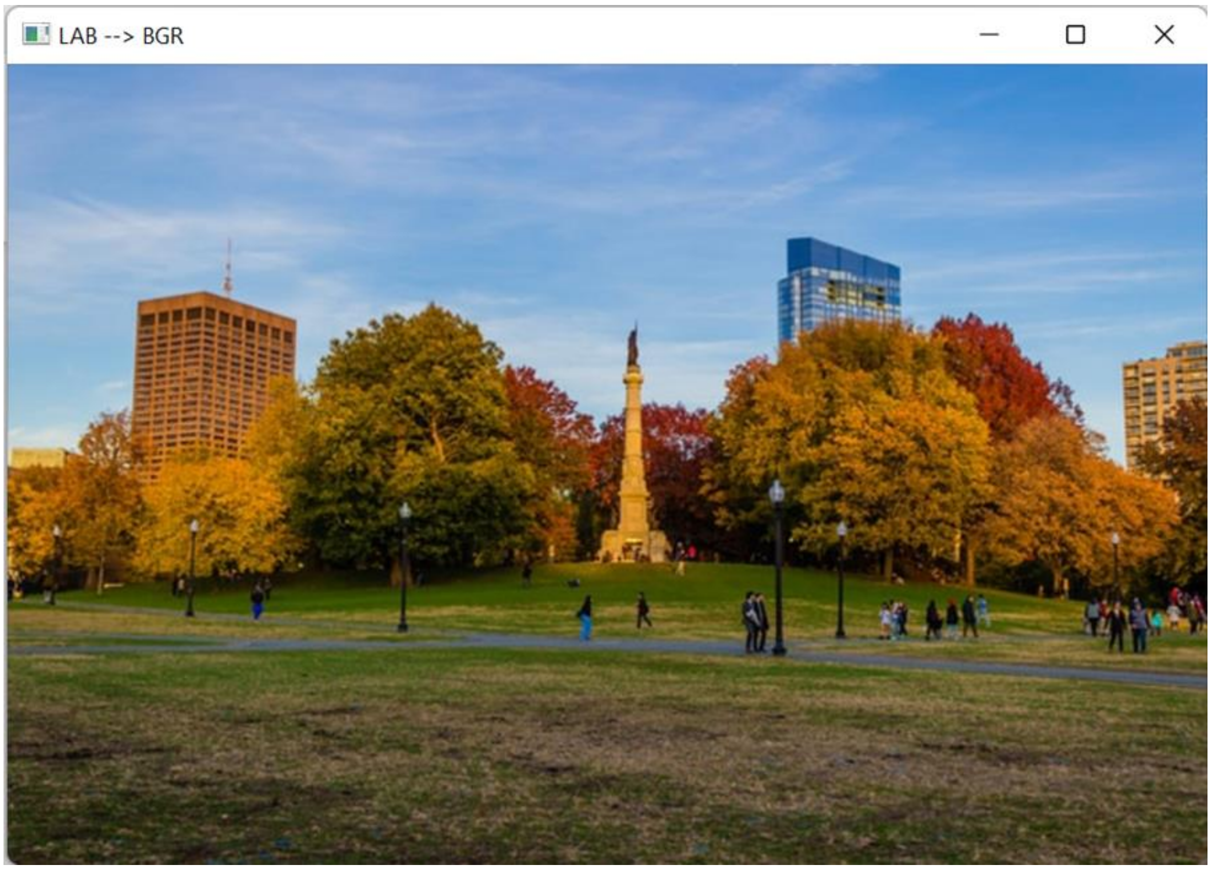

Park


Gray


HSV

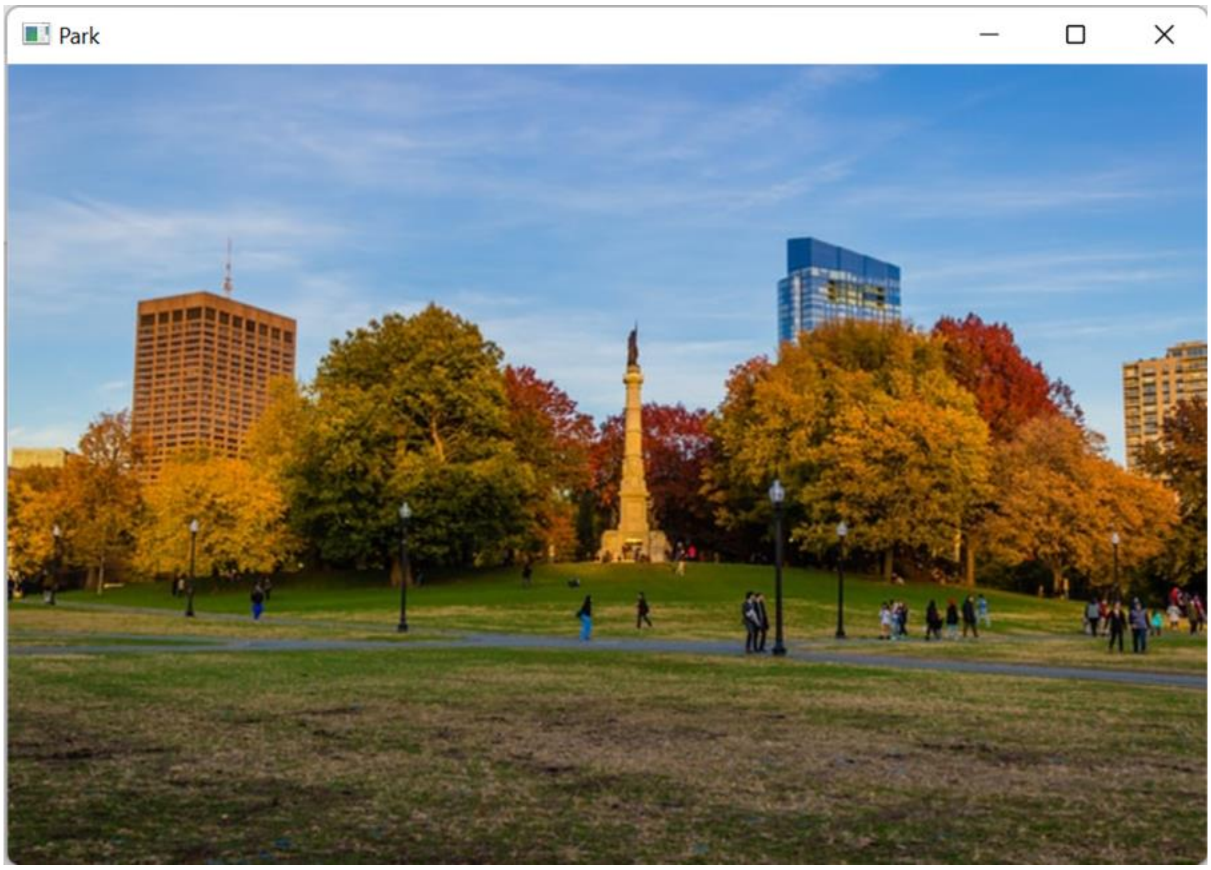

LAB

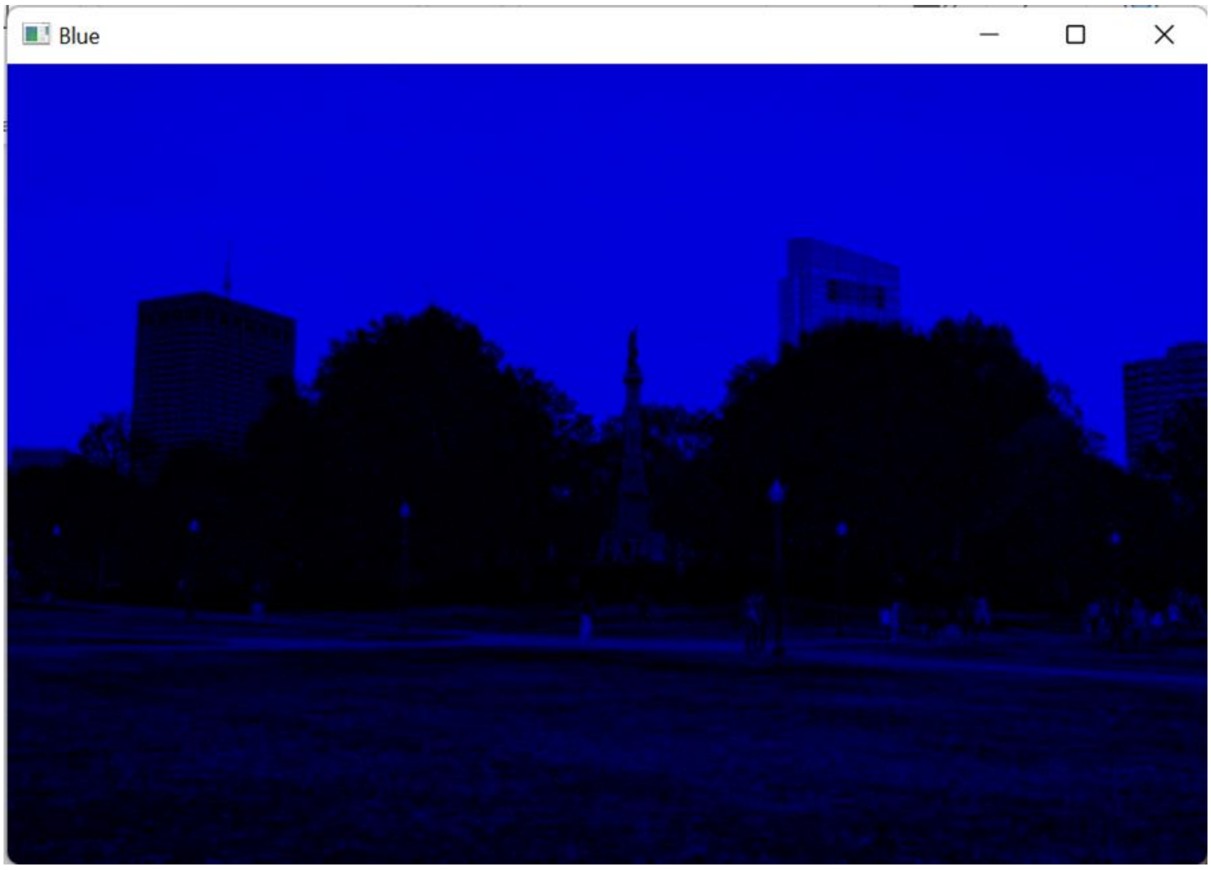


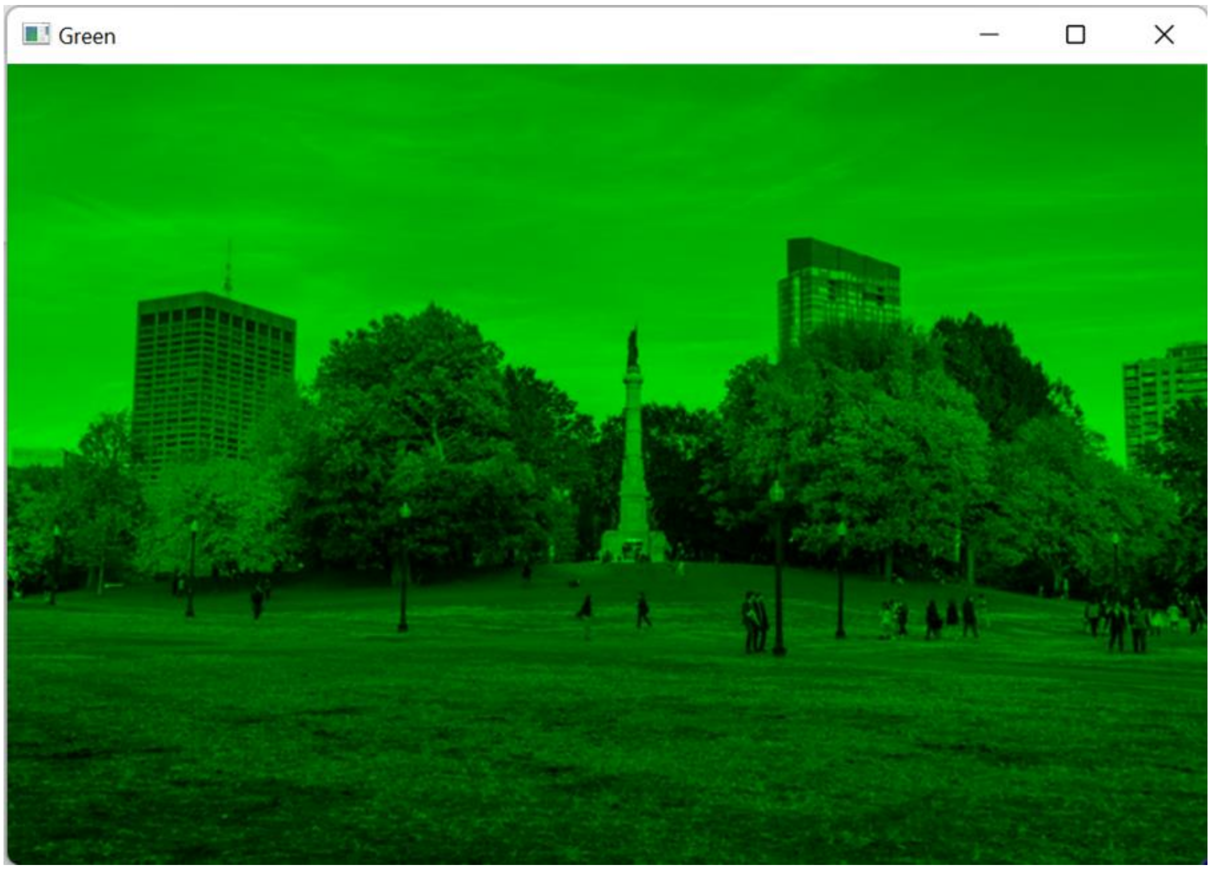


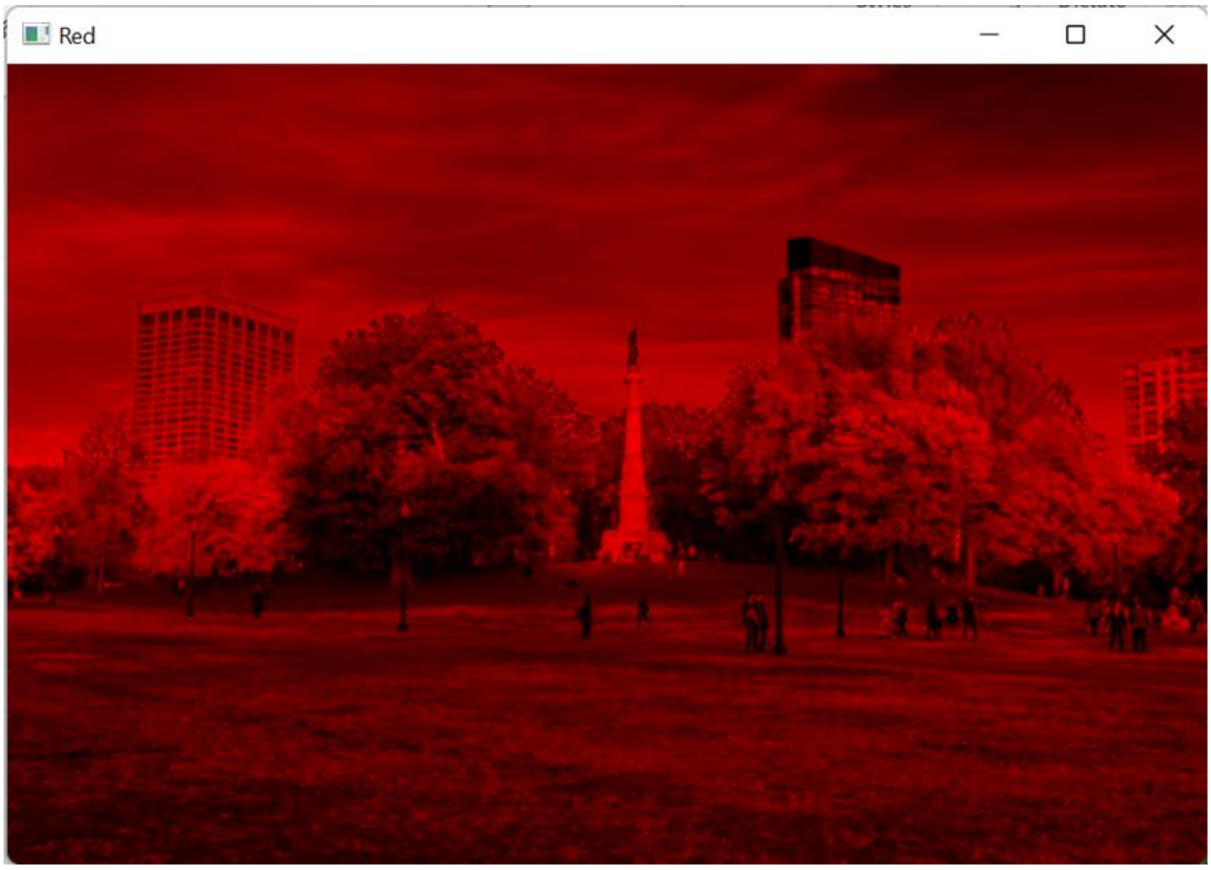
Split Merge

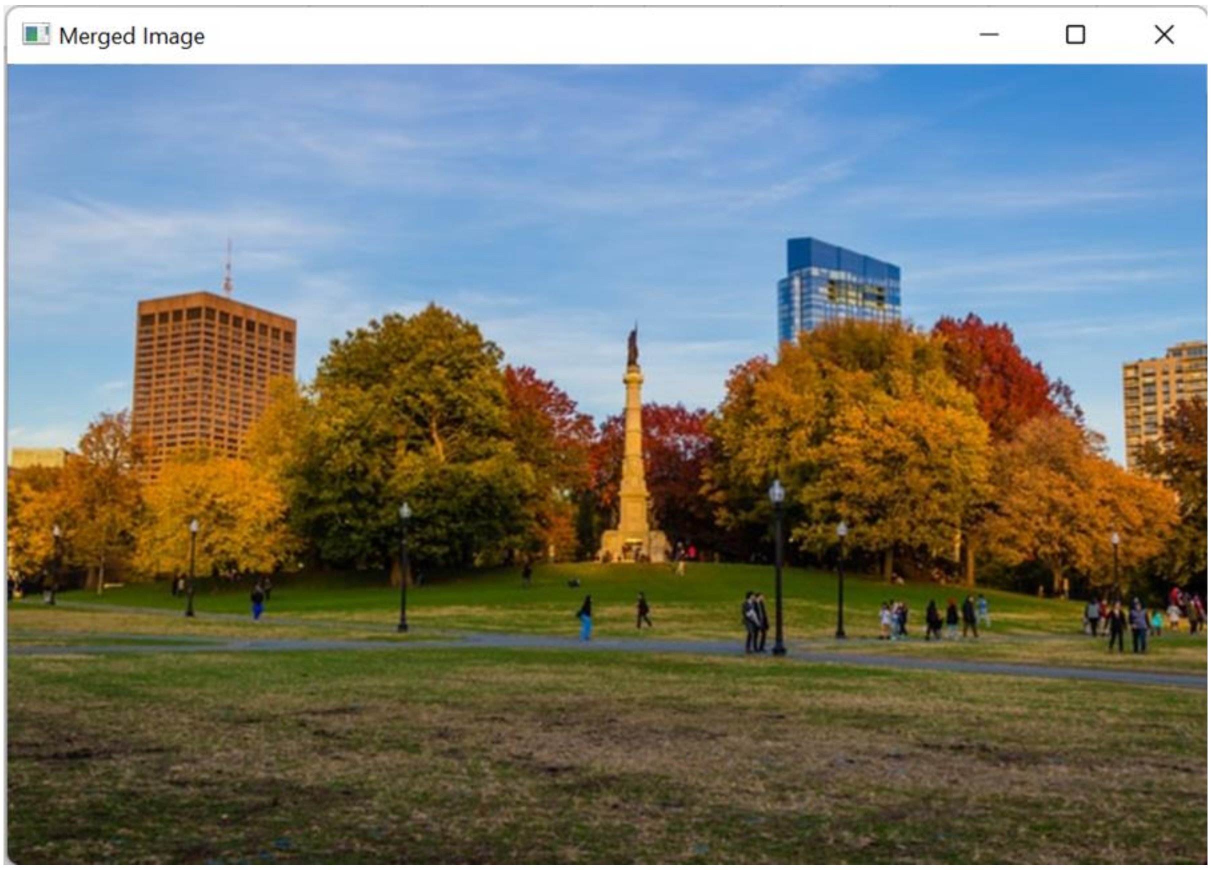
```
#pylint:disable=no-memberimport cv2 as cvimport numpy as npimg = cv.imread('../Resources/Photos/park.jpg')cv.imshow('Park', img)blank = np.zeros(img.shape[:2], dtype='uint8')b,g,r = cv.split(img)blue = cv.merge([b,blank,blank])green = cv.merge([blank,g,blank])red = cv.merge([blank,blank,r])cv.imshow('Blue', blue)cv.imshow('Green', green)cv.imshow('Red', red)print(img.shape)print(b.shape)print(g.shape)print(r.shape)merged = cv.merge([b,g,r])cv.imshow('Merged Image', merged)cv.waitKey(0)
```




Blue

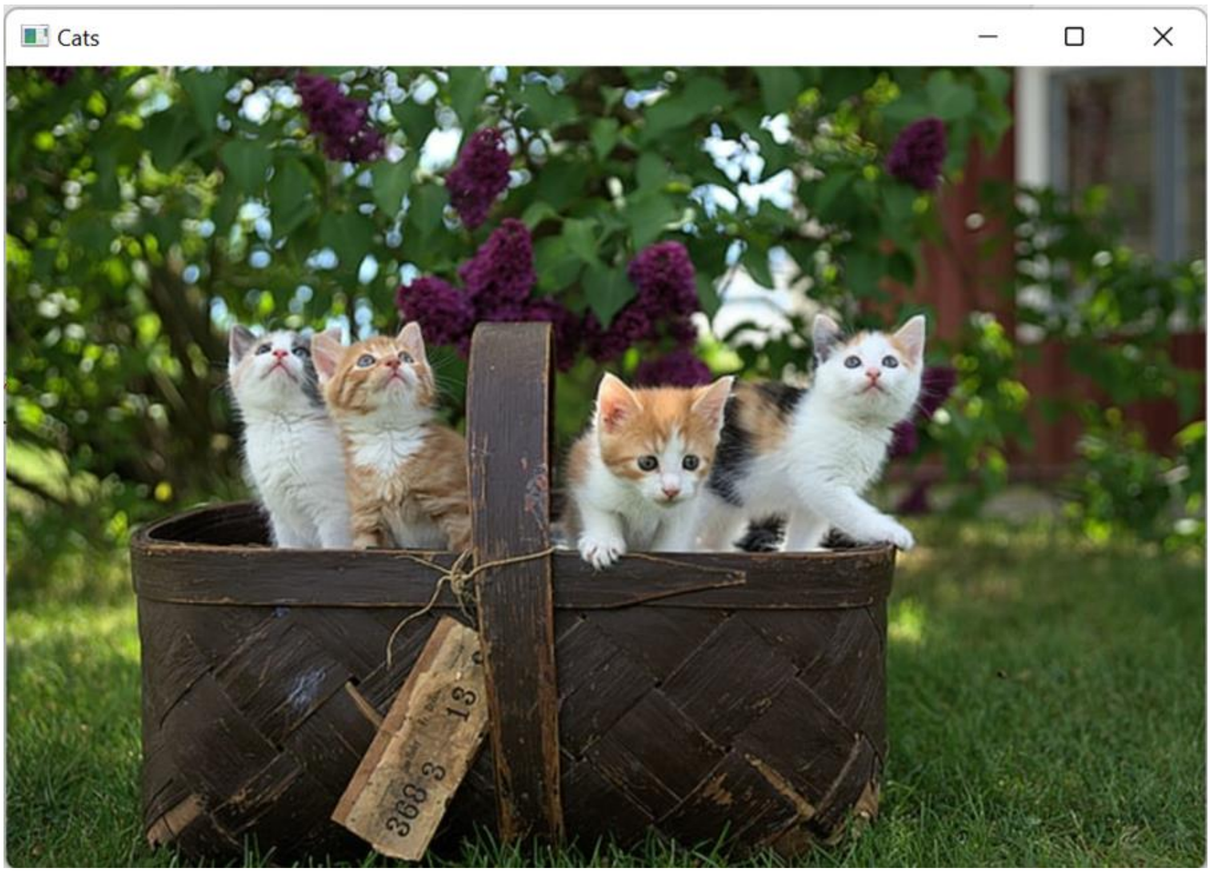

Green

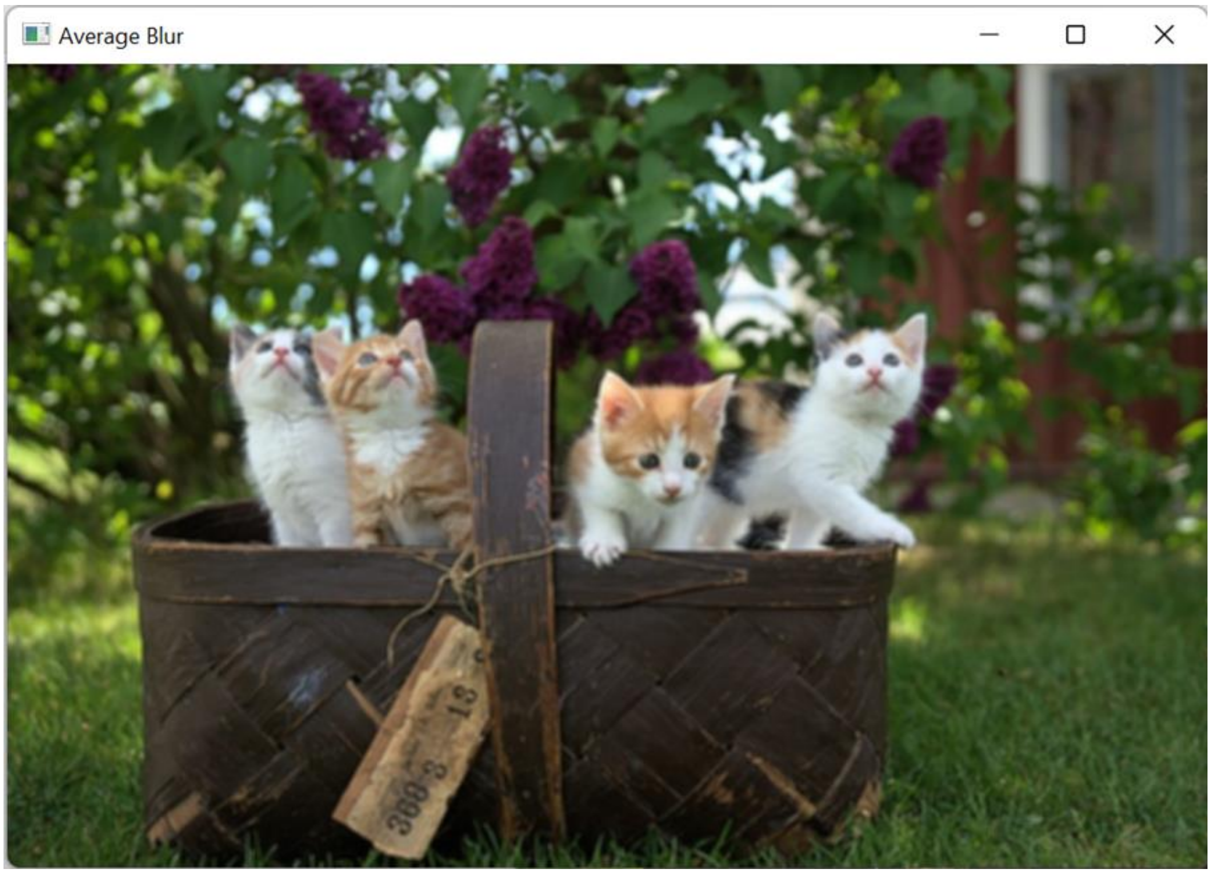


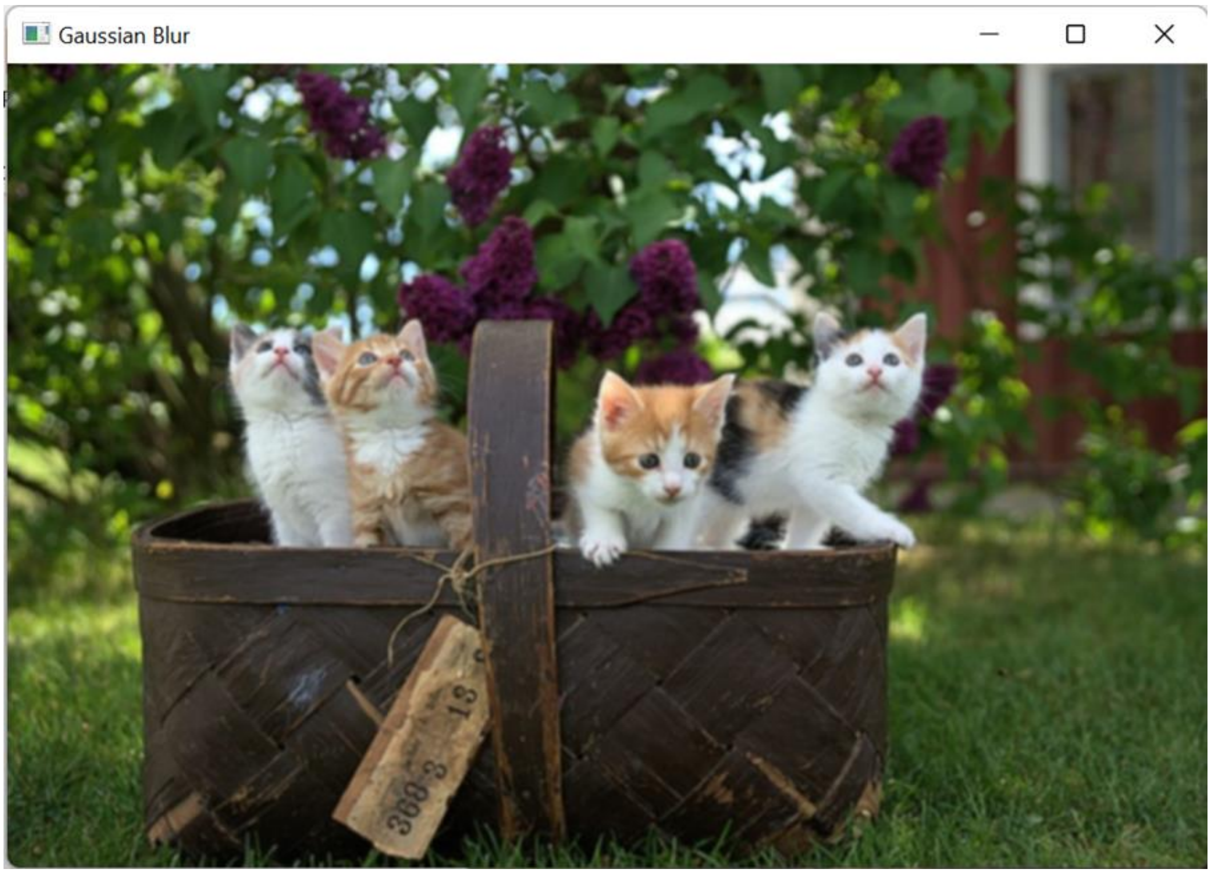


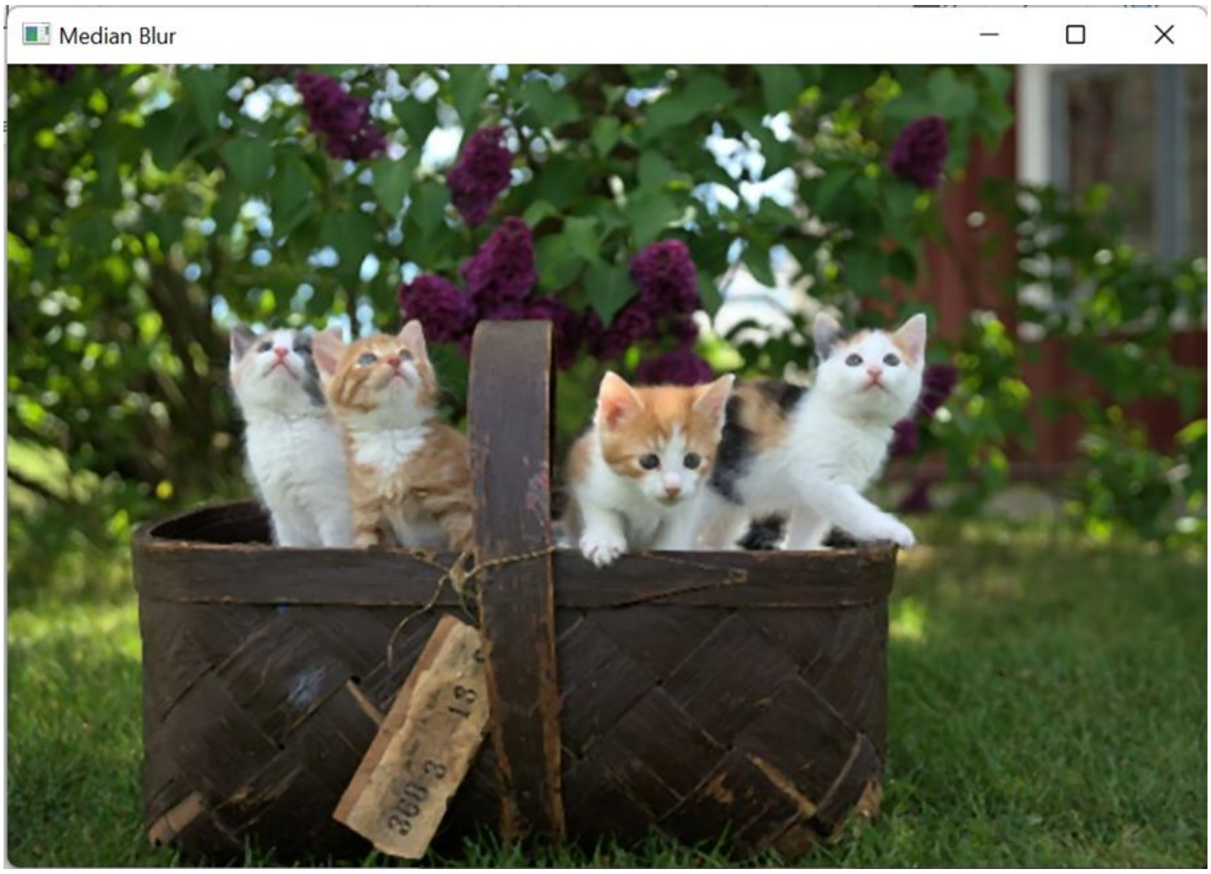
Smoothing and Bluring

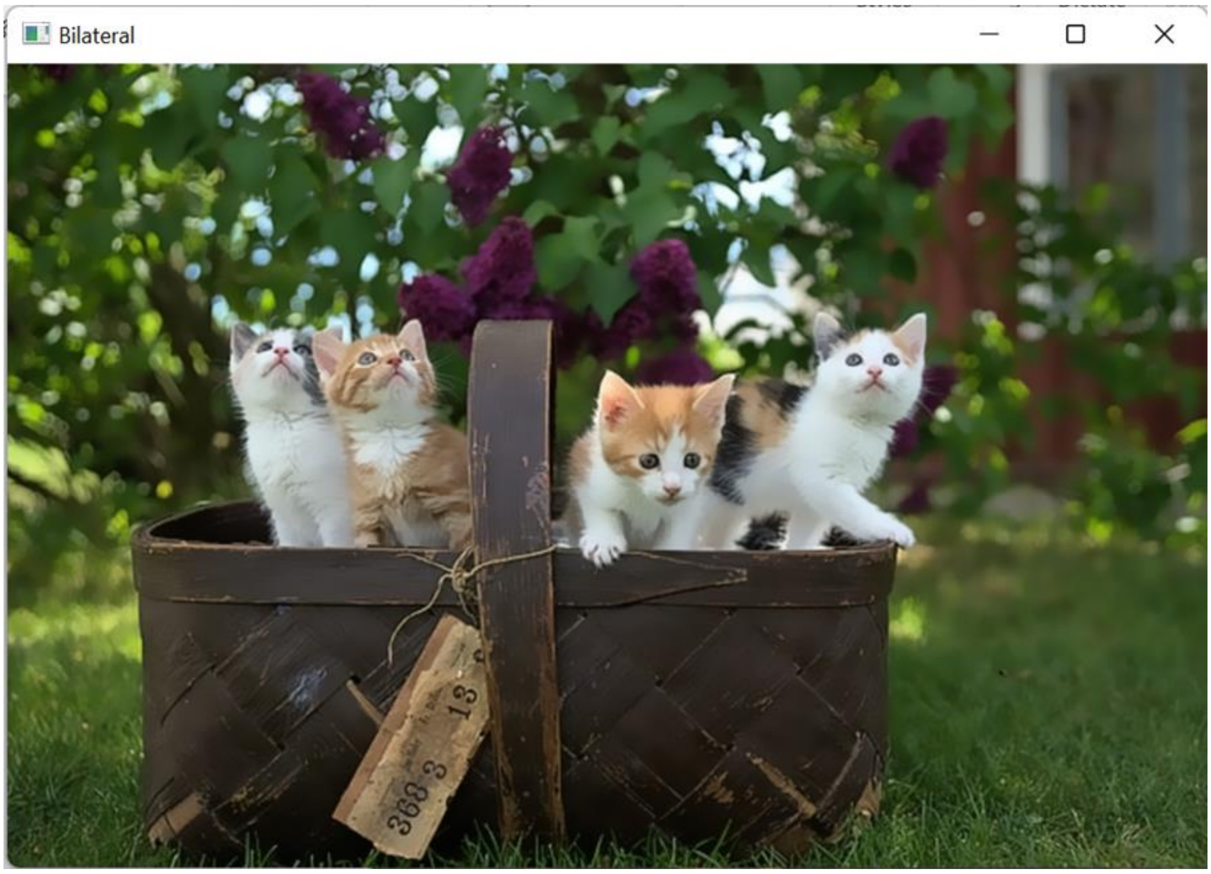
```
#pylint:disable=no-memberimport cv2 as cvimg =
cv.imread('../Resources/Photos/cats.jpg')cv.imshow('Cats', img)# Averagingaverage = cv.blur(img,
(3,3))cv.imshow('Average Blur', average)# Gaussian Blurgauss = cv.GaussianBlur(img, (3,3),
0)cv.imshow('Gaussian Blur', gauss)# Median Blurmedian = cv.medianBlur(img, 3)cv.imshow('Median
Blur', median)# Bilateralbilateral = cv.bilateralFilter(img, 10, 35, 25)cv.imshow('Bilateral',
bilateral)cv.waitKey(0)
```




Average Blur

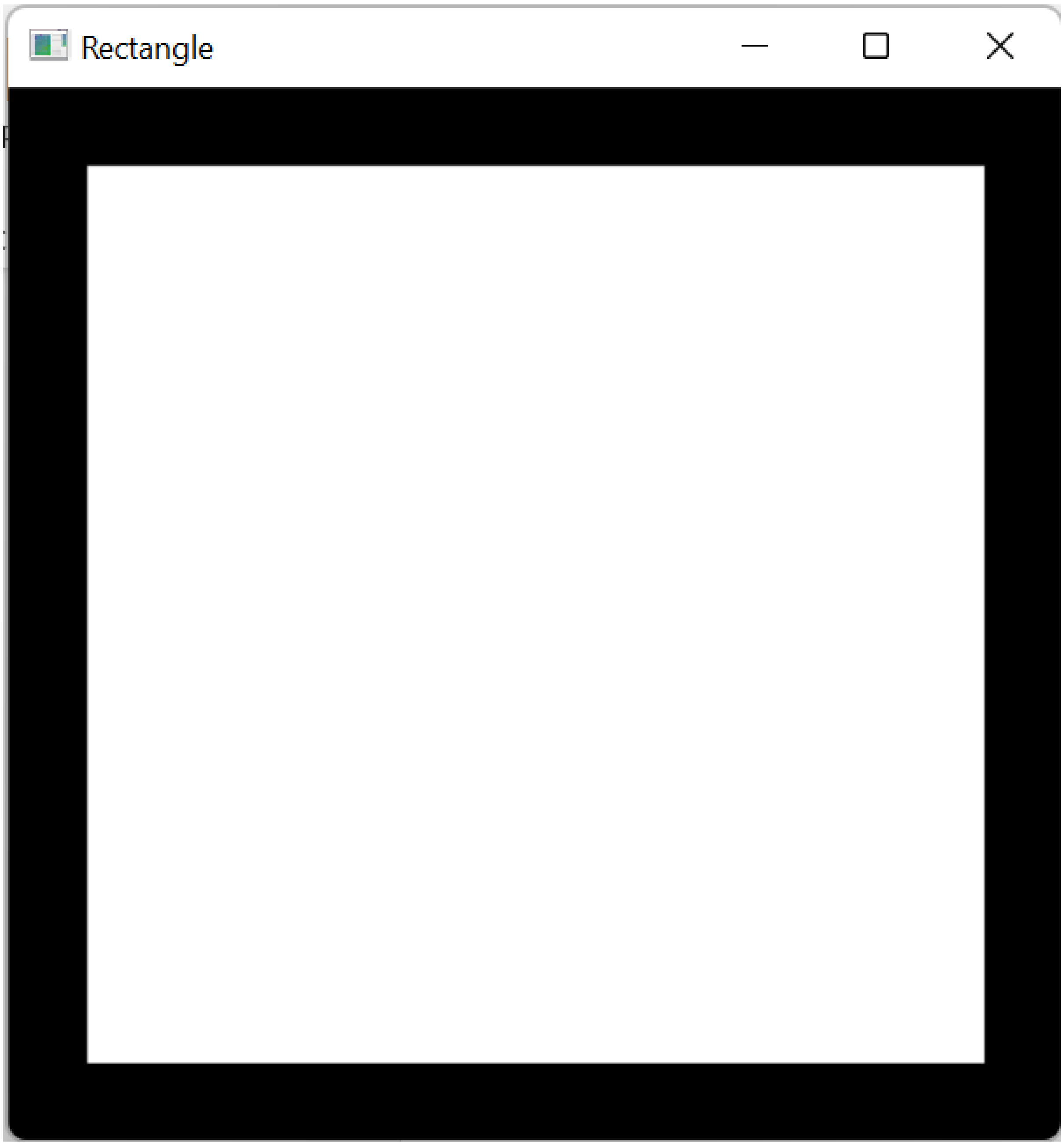

Gaussian Blur

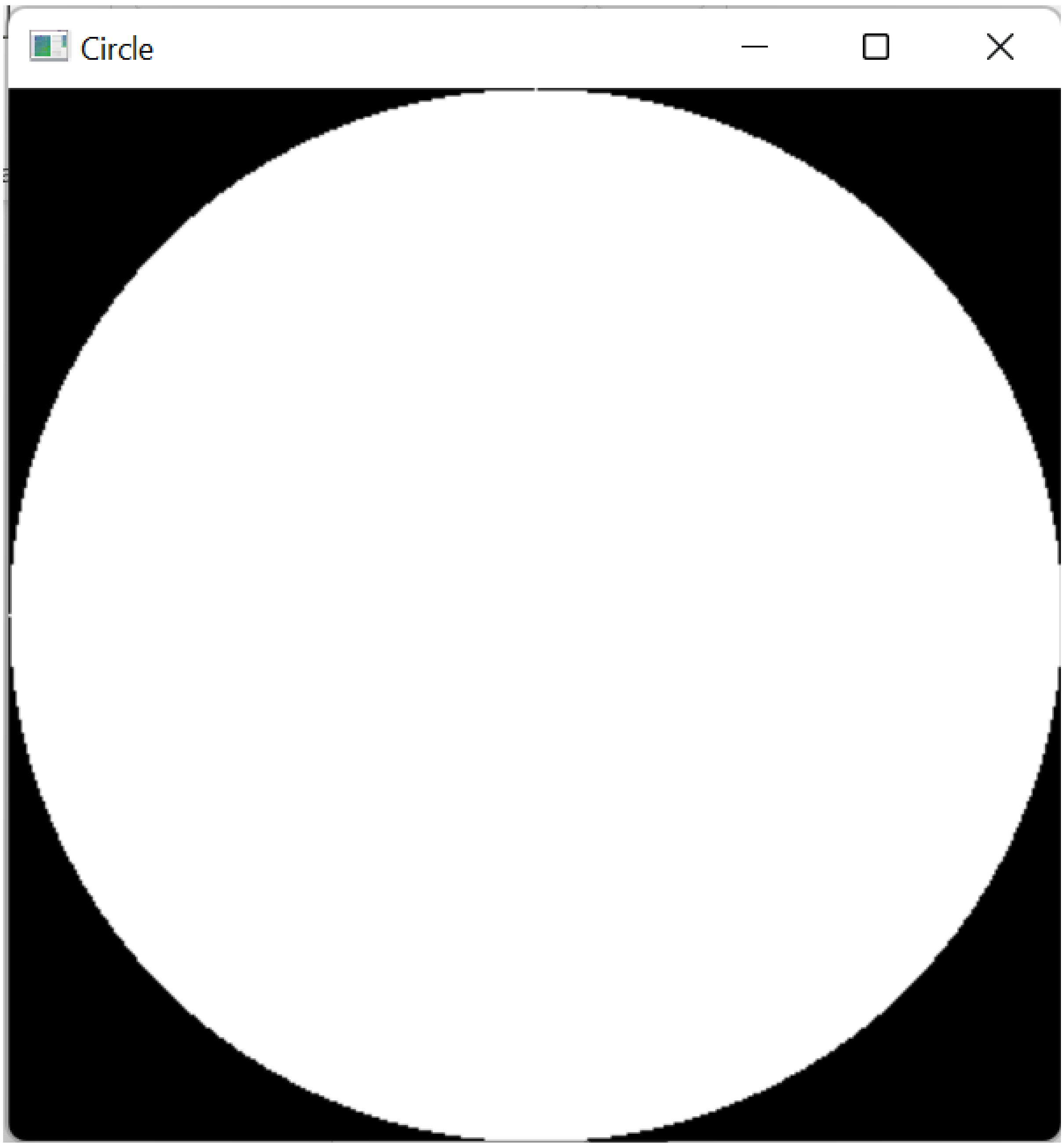


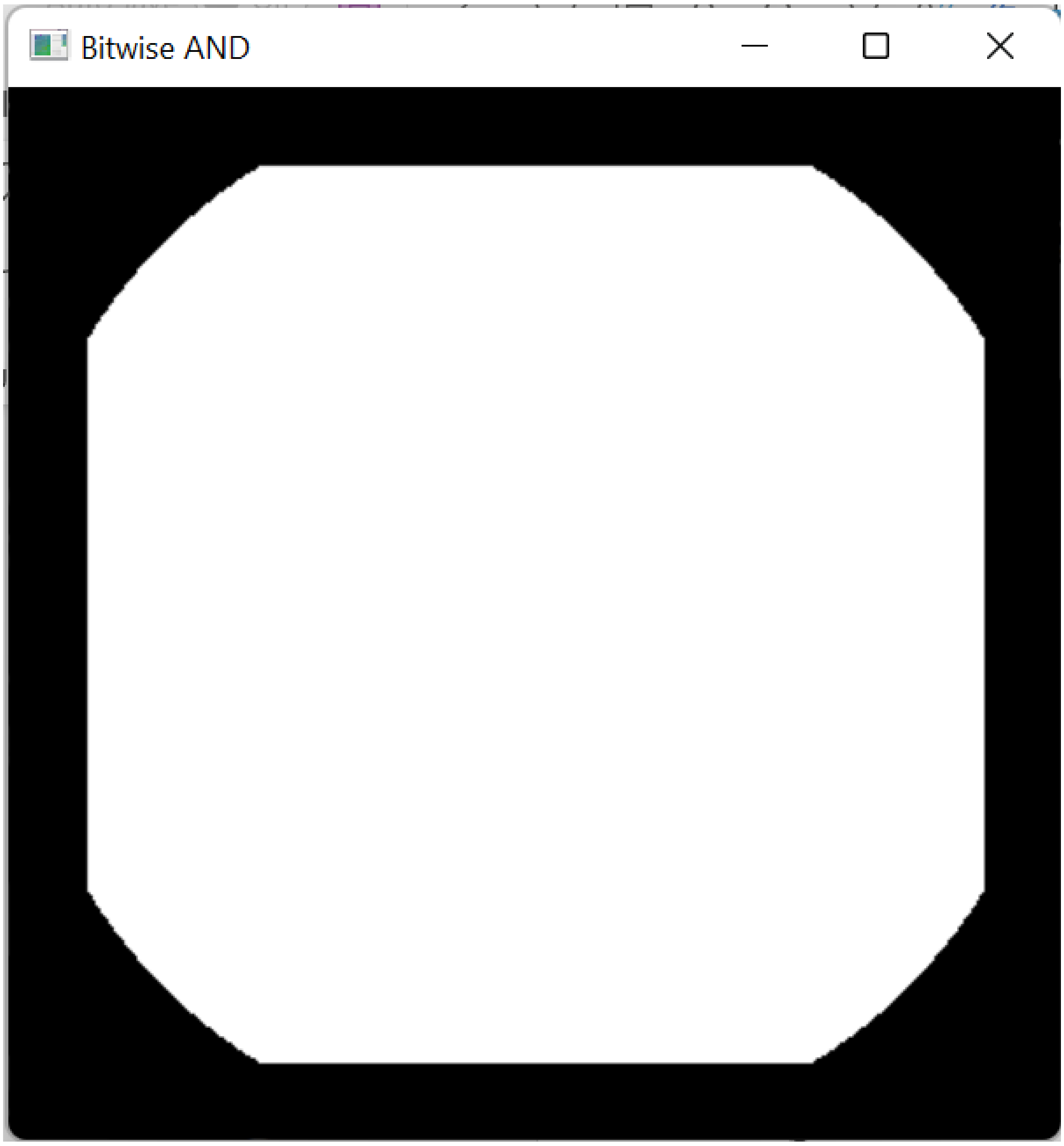


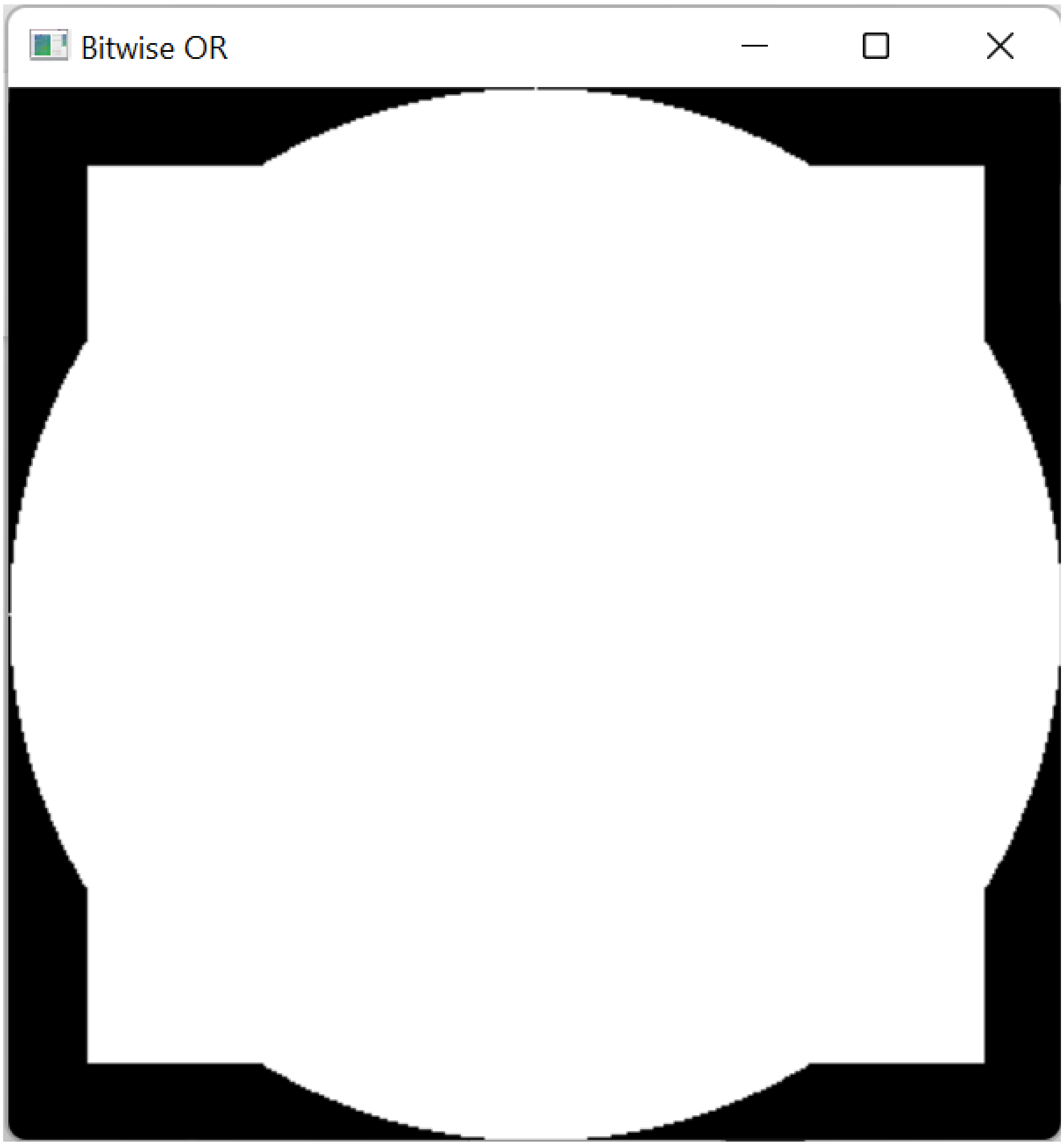
Bitwise Operator

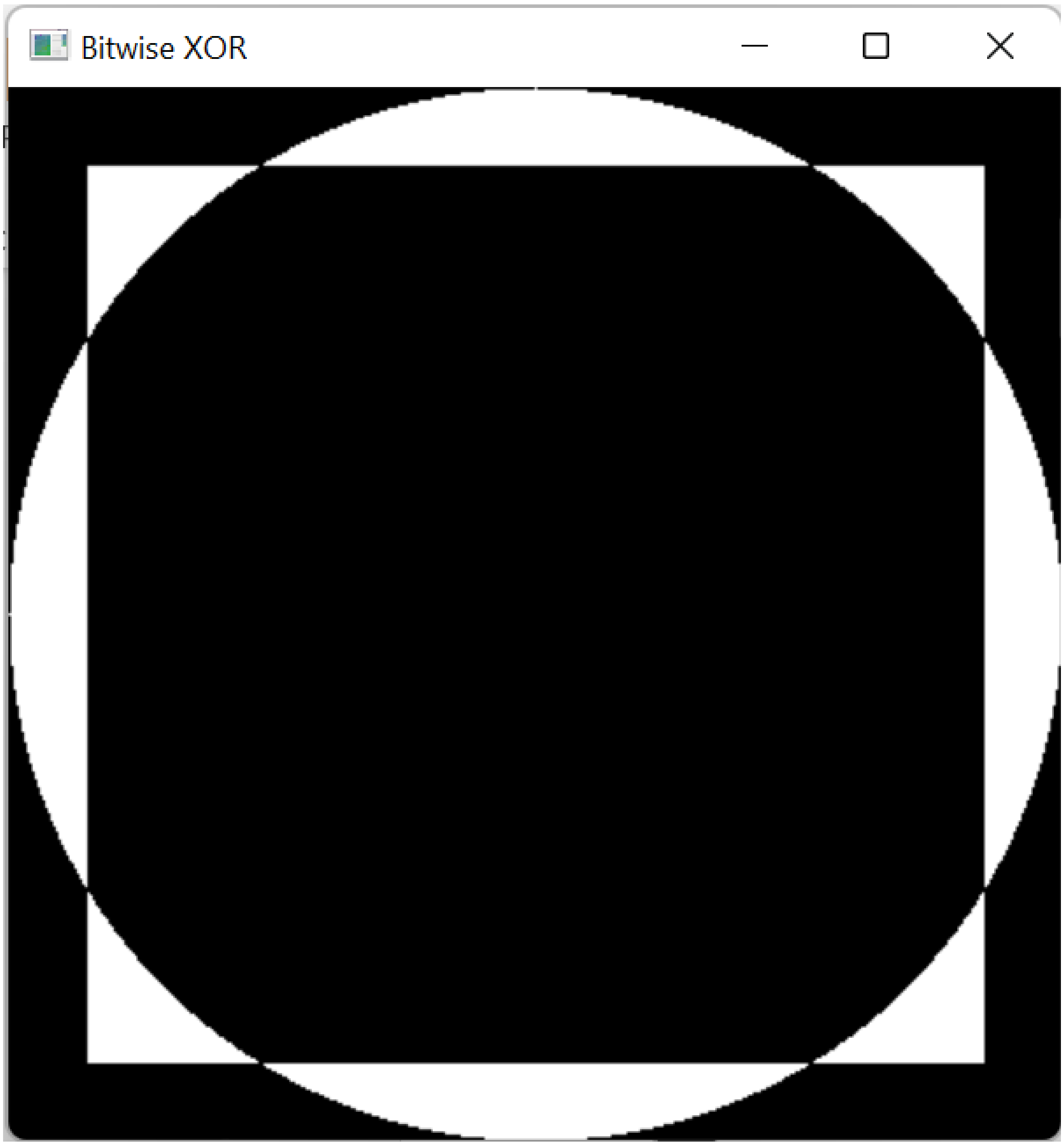
#pylint:disable=no-memberimport cv2 as cvimport numpy as npblank = np.zeros((400,400), dtype='uint8')rectangle = cv.rectangle(blank.copy(), (30,30), (370,370), 255, -1)circle = cv.circle(blank.copy(), (200,200), 200, 255, -1)cv.imshow('Rectangle', rectangle)cv.imshow('Circle', circle)# bitwise AND --> intersecting regionsbitwise_and = cv.bitwise_and(rectangle, circle)cv.imshow('Bitwise AND', bitwise_and)# bitwise OR --> non-intersecting and intersecting regionsbitwise_or = cv.bitwise_or(rectangle, circle)cv.imshow('Bitwise OR', bitwise_or)# bitwise XOR --> non-intersecting regionsbitwise_xor = cv.bitwise_xor(rectangle, circle)cv.imshow('Bitwise XOR', bitwise_xor)# bitwise NOTbitwise_not = cv.bitwise_not(circle)cv.imshow('Circle NOT', bitwise_not)cv.waitKey(0)

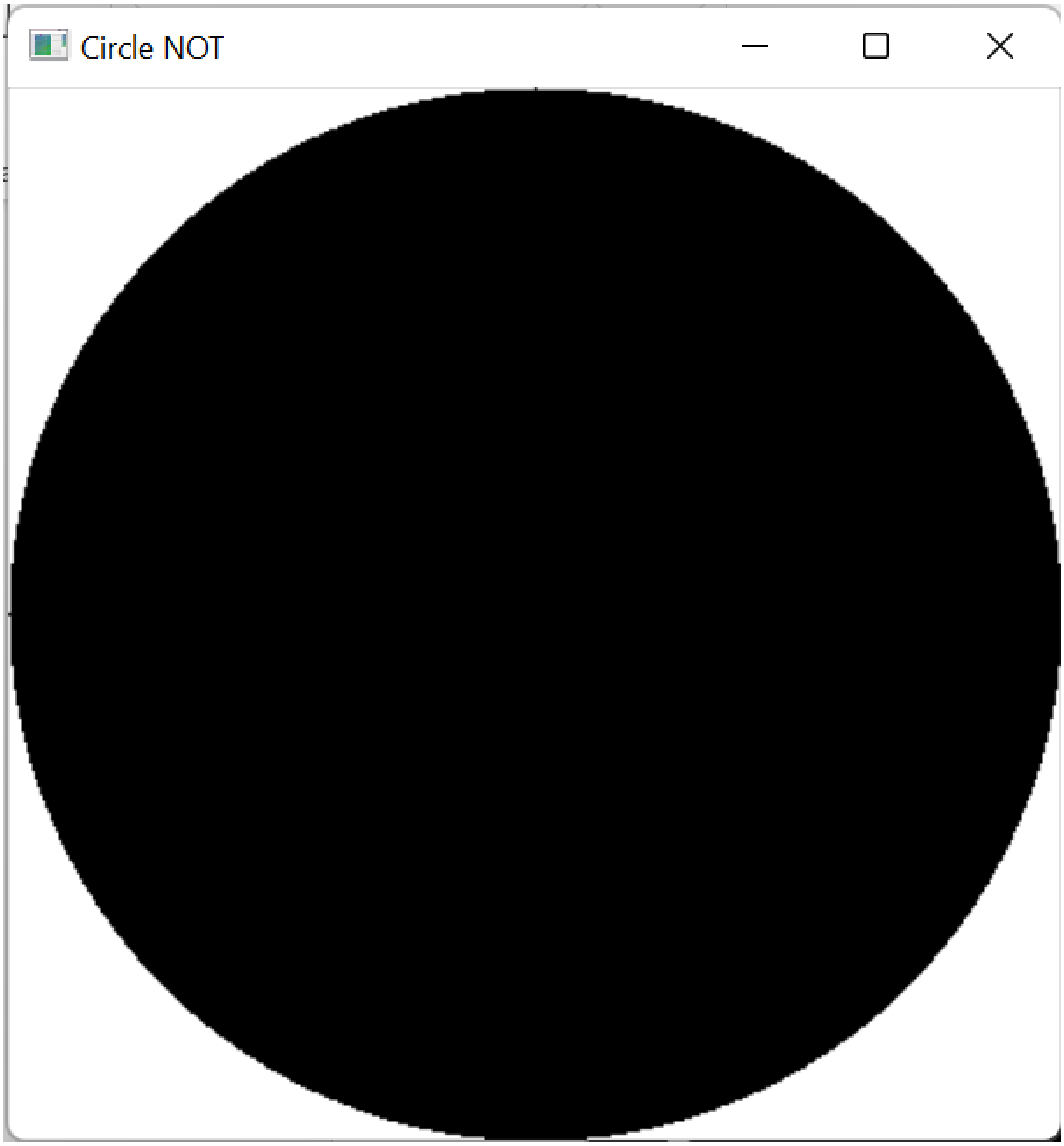



Circle


Bitwise AND

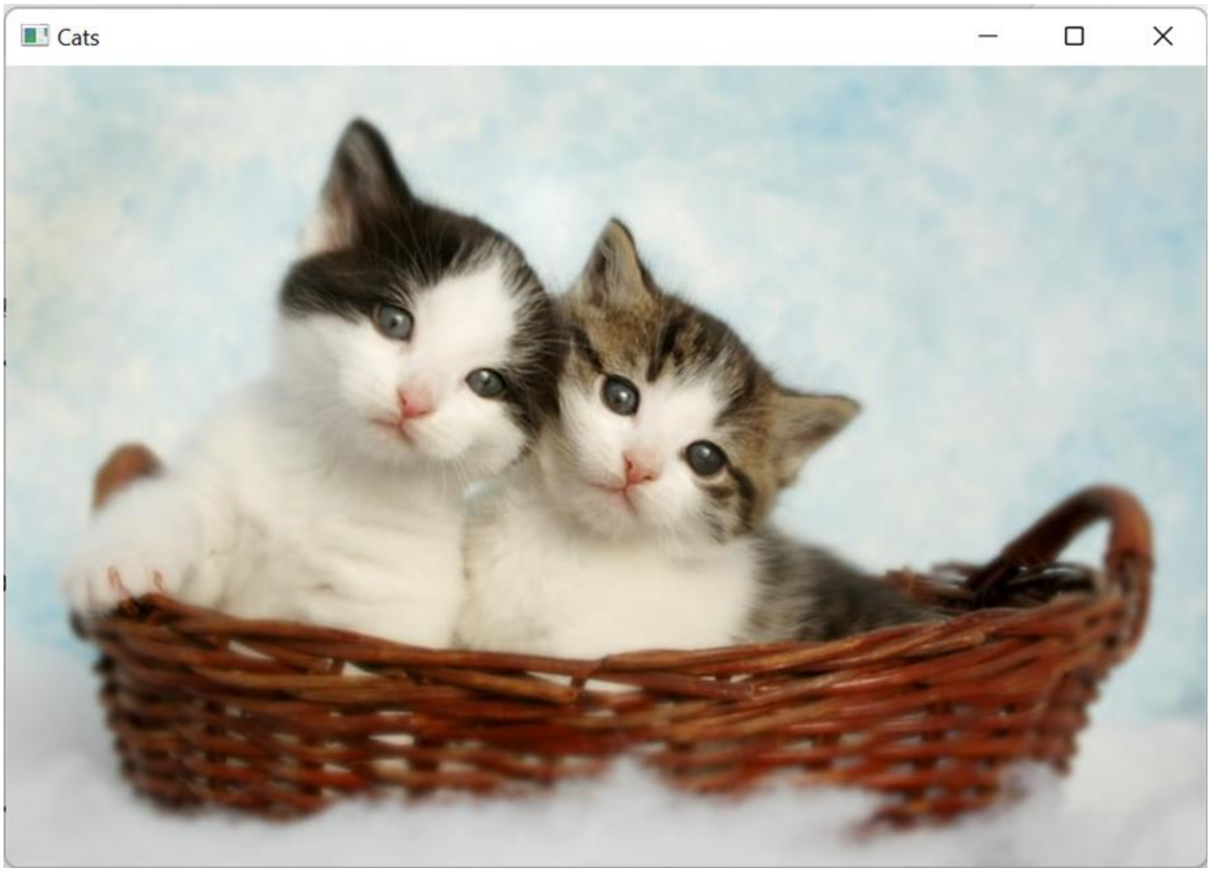

Bitwise OR

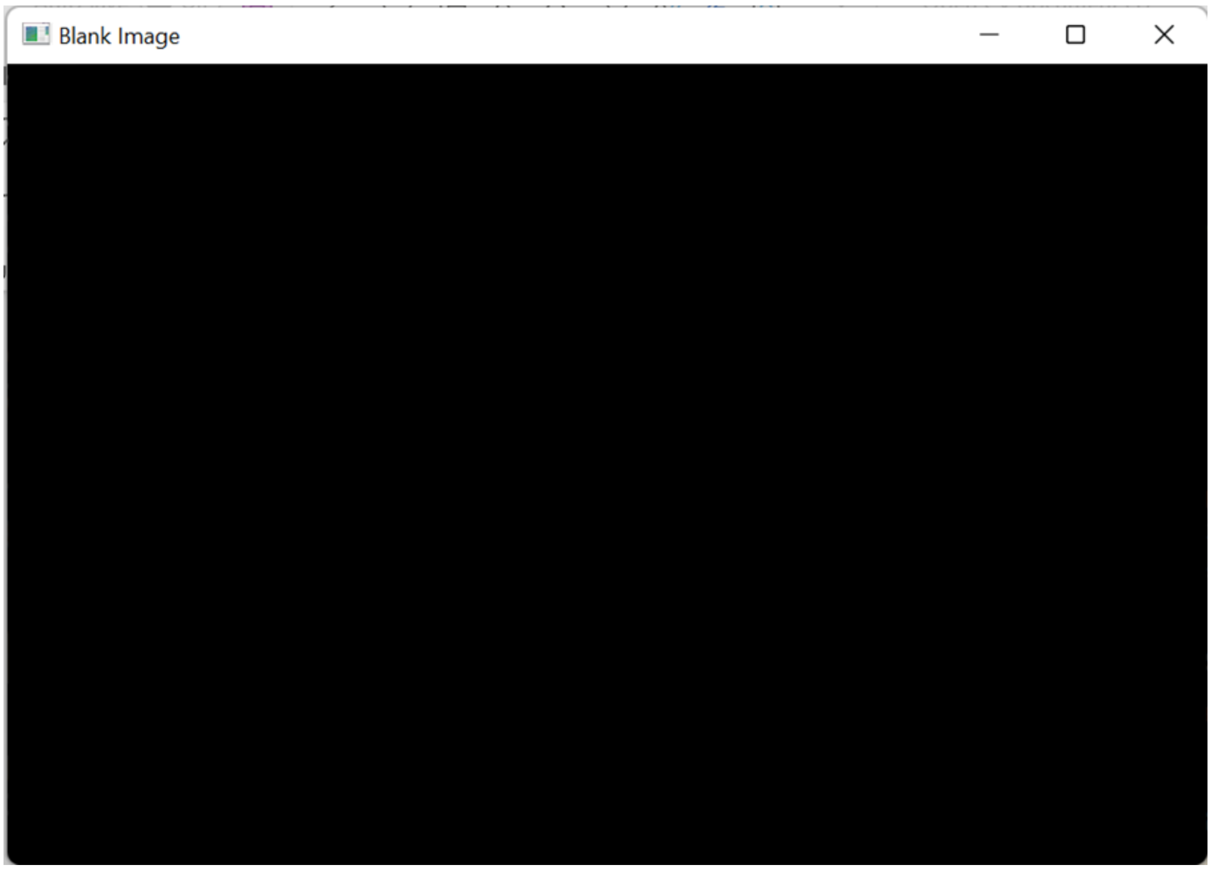

Bitwise XOR

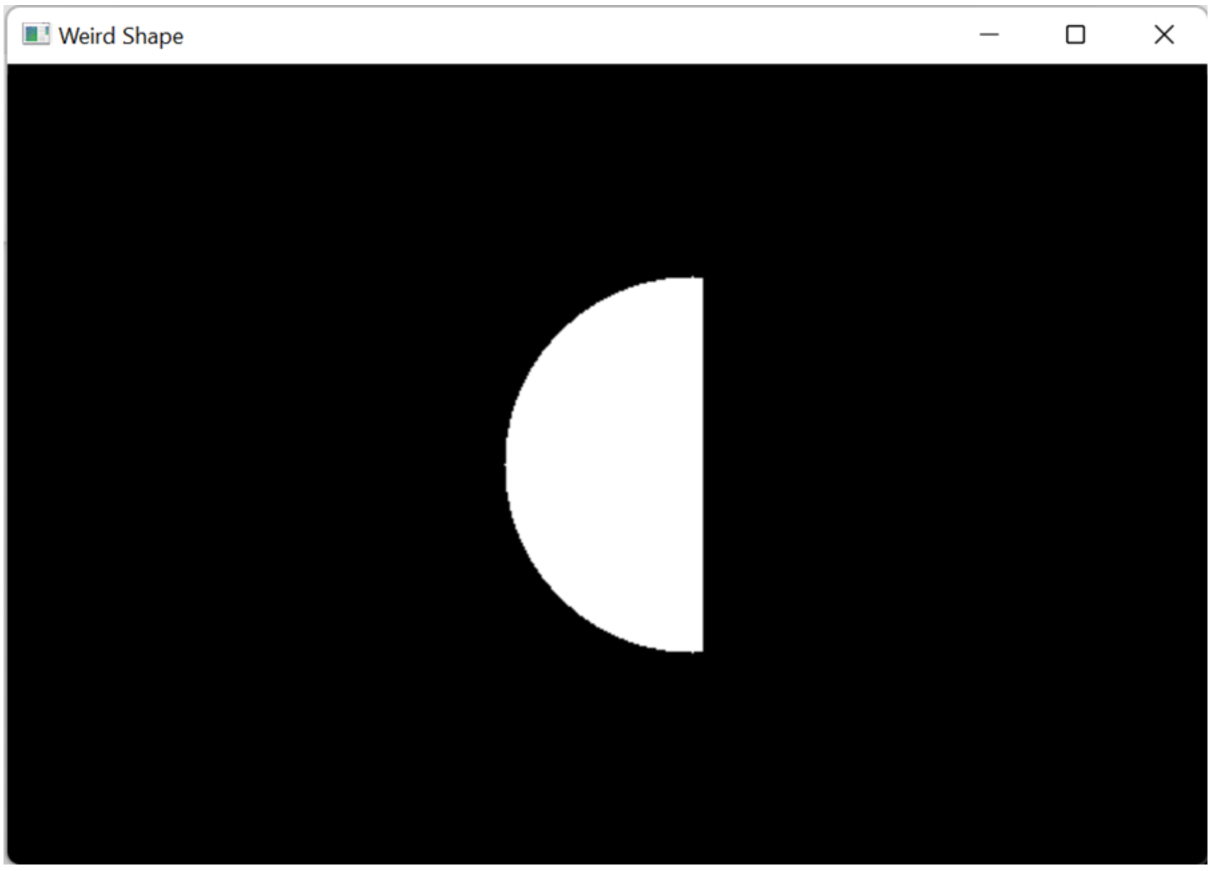


Masking

#pylint:disable=no-memberimport cv2 as cvimport numpy as npimg = cv.imread('../Resources/Photos/cats 2.jpg')cv.imshow('Cats', img)blank = np.zeros(img.shape[:2], dtype='uint8')cv.imshow('Blank Image', blank)circle = cv.circle(blank.copy(), (img.shape[1]//2 + 45,img.shape[0]//2), 100, 255, -1)rectangle = cv.rectangle(blank.copy(), (30,30), (370,370), 255, -1)weird_shape = cv.bitwise_and(circle,rectangle)cv.imshow('Weird Shape', weird_shape)masked = cv.bitwise_and(img,img,mask=weird_shape)cv.imshow('Weird Shaped Masked Image', masked)cv.waitKey(0)


Cats


Blank Image



Reference:

This library was explored from various websites and youtube channels which include : totorialspoint.com and freeCodeCamp.org