ECE 558 FINAL PROJECT

NAME: Vikram Pande

UNITY_ID: vspande

Student ID: 200473855

LAPLACIAN BLOB DETECTOR

INTRODUCTION:

The blob detection is important task in Computer Vision which gives us the idea about regions, points in the images which differ in properties. The complementary information about the regions is obtained with the help of blobs which is not surely obtained by Edge or Corner detectors. Blob helps detect the presence of object in the given image. With blobs, centre and surrounding corners / edges can be obtained.

ALGORITHM OUTLINE:

1) **Generate a Laplacian of Gaussian Filter:**
The Laplacian of Gaussian Kernel is created with variance (σ) which is specified as 1.6 initially. Scaling factor k is specified as $\sqrt{2}$. The output of the function is multiplied with different number of variances created by scaling factor the normalize LoG. LoG is achieved with the help of following formula:

$$LoG(x,y) = [\frac{(x^2+y^2-2\sigma^2)}{2\pi\sigma^2}]e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Functions used:

```
def log(sigma:float) -> np.ndarray:
```

The function takes variance as input and generates LoG Kernel.

2) **Build Laplacian Scale Space:**
For Convolution, used the code from Project 2 with 4 types of padding and Kernel Types. To build a Laplacian Scale Space, used zero-padding. Initial scale is taken as 1.6 and going for n iterations with multiplication factor of $\sqrt{2}$. Doing so, the kernel size remains fix and image size changes at each iteration. The time for convolution could be reduced by doing so. Basically, the function takes image, changes its size and convolve the image with LoG kernel. Saving the square of response for current level of scale space in the final stage of function.

Functions used:

```
def laplacian_scalespace(input_img: np.array, sigmaCount: int) -> np.ndarray:
```

The function takes image (RGB/GREY) and sigmaCount (Number of Steps) as input and builds Laplacian Scale Space. Steps of Variance are taken randomly such as 3, 5, 7.

3) **Non-Max Suppression:**

To perform NMS, taking the output from Laplacian Scale Space and applying 2D NonMaxSuppression. Threshold is set as 0.005. NMS is applied in two steps, first to check if the centre value is greater than the neighbouring pixels around 3x3 kernel, if yes then set as 1 else 0. By so, getting ImageMaxima results. Second, to check if the center value is greater than values above and below for a particular instance of Laplacian Scale Space.

Functions used:

```
def nms2d(scalespace: np.ndarray ,threshold = threshold) -> np.ndarray:
def nms(img_maxima: np.ndarray, scalespaceLaplacian: np.ndarray) ->
np.ndarray:
```

The function nms2d takes scalespace and threshold as input and filter image based on threshold to obtain ImageMaxima. The function nms takes ImageMaxima and instances of Laplacian Scale Space to give the final Non-Max Suppressed output suppressing the values less than threshold to obtain final NMS points.

4) **Draw circles:**

The circles are drawn to display the maxima of the image by taking $radius$ and (x, y) co-ordinates. cv2.circle() is used.

Functions used:

```
def drawcircles(img: np.ndarray,nms_result:np.ndarray) -> np.ndarray:
```

The function takes image and result from NonMaxSuppression to display circles.

Description of Python files:

Convolution Helper Function Python File:

convolution.py

Contains Convolution and Padding functions.

Blob Detection Helper Function Python File:

main.py

Contains LoG, LaplacianScaleSpace, NMS, DrawCircles functions

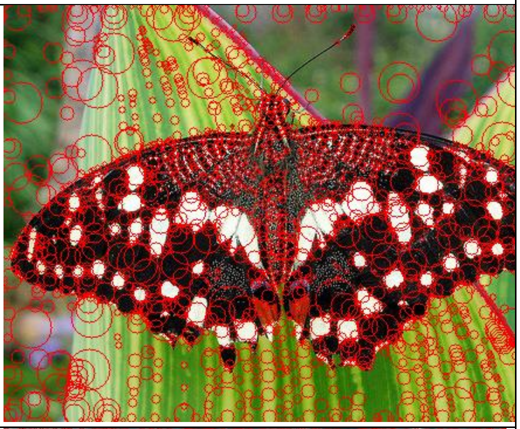
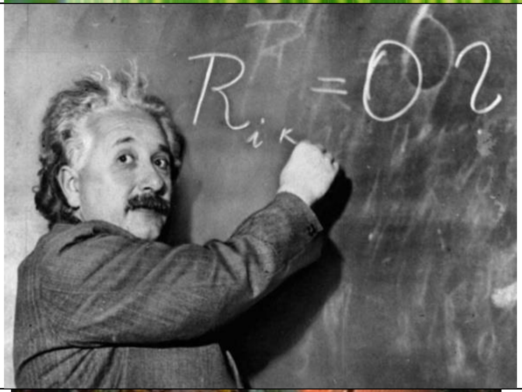
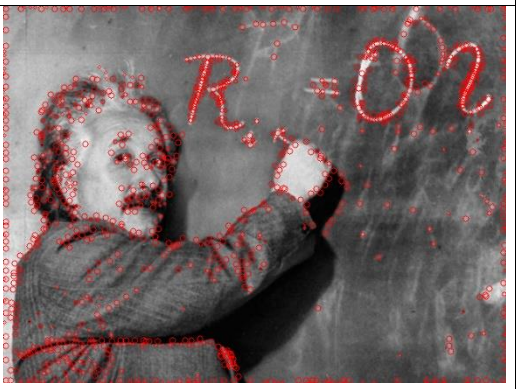
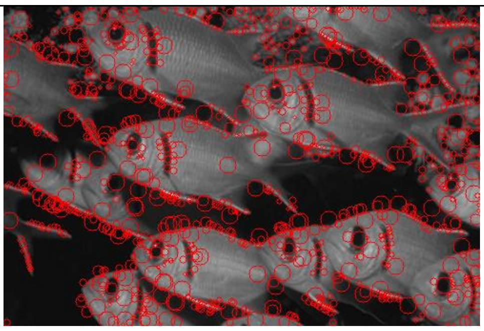
One-Click Run Python File:

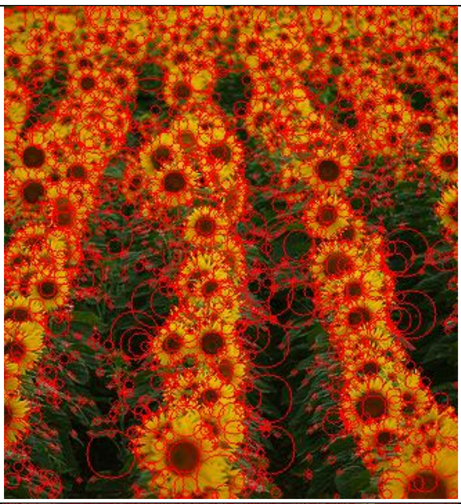
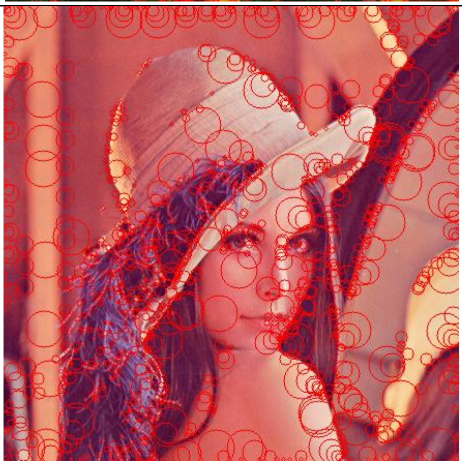
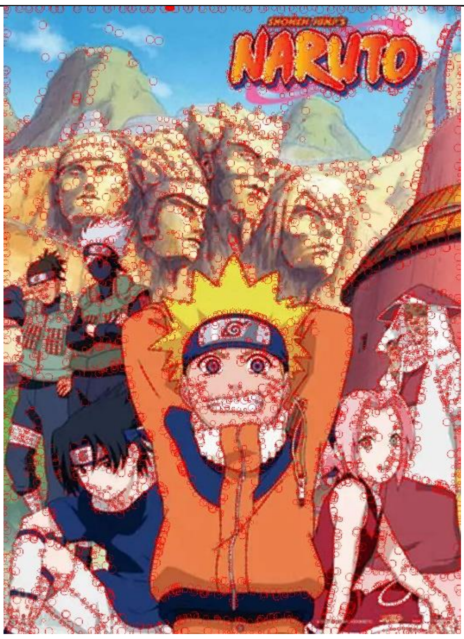
run_tests.py

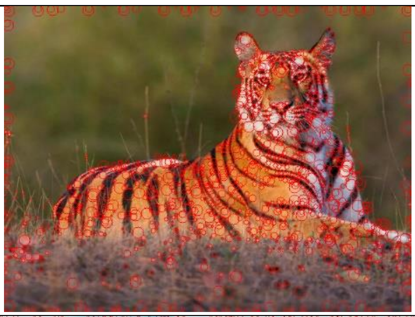
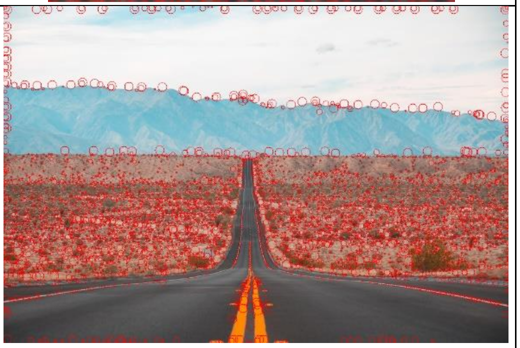
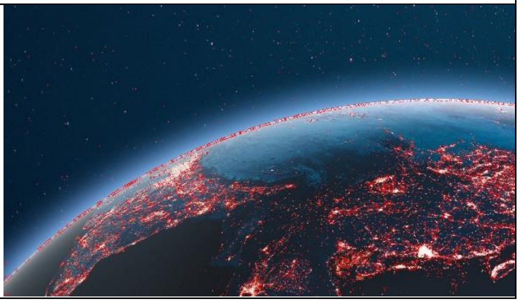
Contains 8 Images and Run functions.

RESULTS:

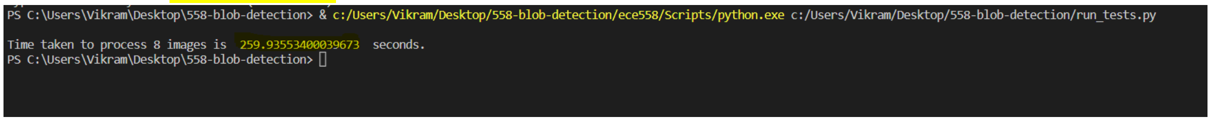
- 8 Images are processed with threshold and different scaling.
- Images were both RGB and GREY.
- The time taken to process all the **8 images** in first attempt was approximately **259 seconds.**
- Below are given the 9 original images and images with blobs detected.

ORIGNAL INPUT IMAGES	OUTPUT IMAGES WITH BLOBS
	
	
	

	
	
SHONEN JUMP'S NARUTO	 SHONEN JUMP'S NARUTO

TIME TAKEN: 259 SECONDS



```
PS C:\Users\Vikram\Desktop\558-blob-detection> & c:/Users/Vikram/Desktop/558-blob-detection/ece558/Scripts/python.exe c:/Users/Vikram/Desktop/558-blob-detection/run_tests.py

Time taken to process 8 images is  259.93553400039673  seconds.
PS C:\Users\Vikram\Desktop\558-blob-detection> []
```

It is observed that the time is dependent on the steps of sigma, if sigma steps are set as 7, time require to process image is more as compared to 2 or 3. The conclusion is that if we increase the steps in which we multiply scale factor with variance, the more it is, time requirement is more, circles with small radius form. If number of steps are small, circles are big, time requirement to process the image is less.

FOLDER STRUCTURE:

__pycache__ - cache

ece558 – python environment

ResultImages – Result Images

TestImages4Project – Test Images

Codes – main.py, convolution.py, run_tests.py

TOOLS and LIBRARIES:

IDE used: Visual Studio Code

Libraries used throughout the code:

```
from convolution import conv2
import time
import numpy as np
import time
import os
import cv2
import matplotlib.pyplot as plt
from convolution import conv2
from main import log, laplacian_scalespace, nms2d, nms, _drawcircles,
drawcircles
import math
```