MINI-LAB ASSIGNMENT 3 SPRING 2024
ELEC 3520: IoT & Cyber-physical Systems
Department of Electrical Engineering
College of Engineering, Design and Computing
University of Colorado Denver

Student's Name: Joseph Oler
Student ID: 110676165
Date: 4/6/2024

Question 1:

Using CV2 library to blur an image. You should show in the report your original and the blurred image including code and explanation.

Video Link - https://youtu.be/_aX2A8lERzg?si=tdKD5-DBUV4eb9gg

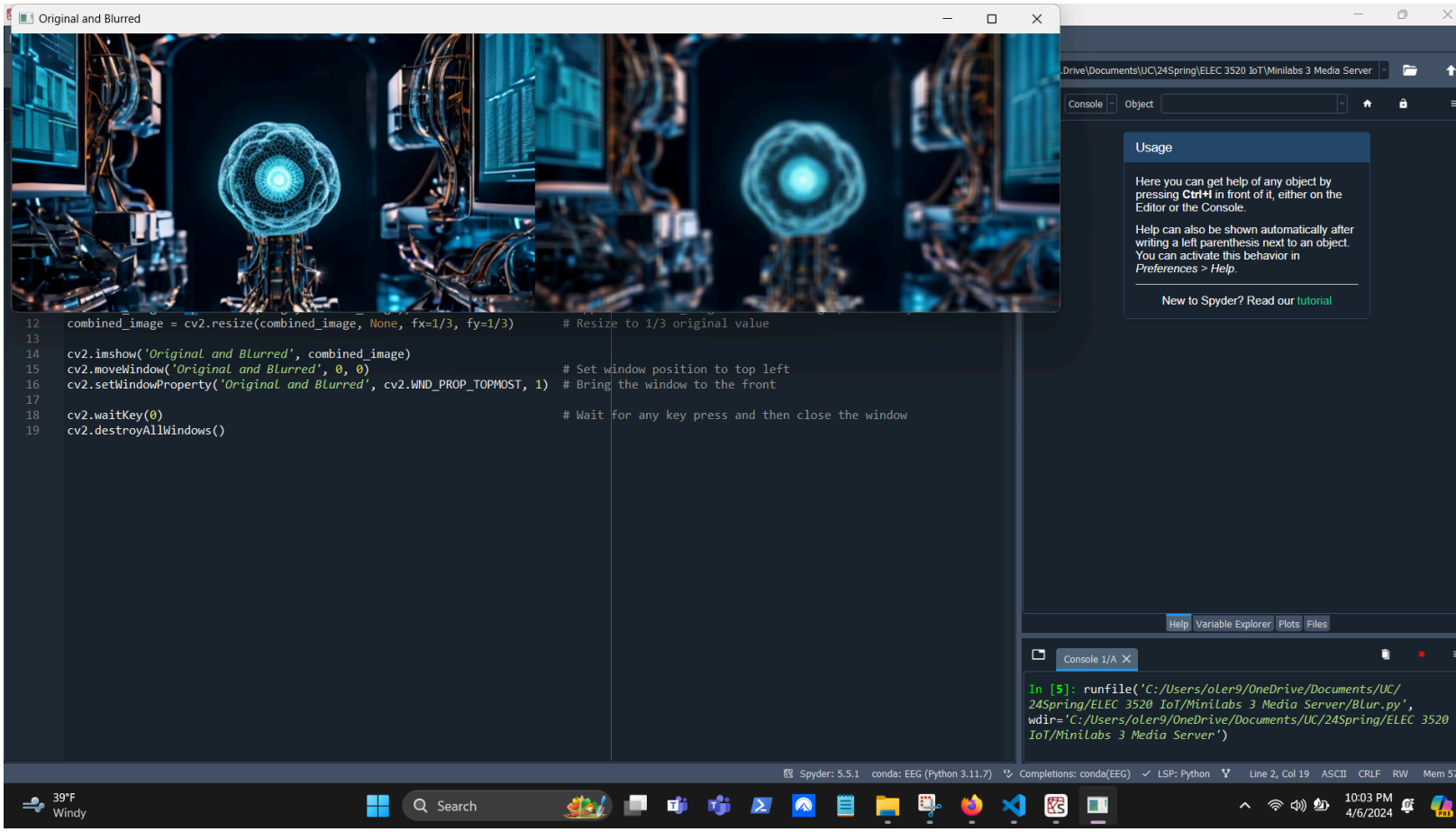
Code - Description in code, uploaded as Blur.py

C:\Users\oler9\OneDrive\Documents\UC\24Spring\ELEC 3520 IoT\Minilabs 3 Media Server\Blur.py

Blur.py*

```python
import cv2
import numpy as np

# Load the image
path = r'C:\Users\oler9\OneDrive\Pictures\3.png'
image = cv2.imread(path)

ksize = (21, 21)              # Kernel takes average of values in surrounding pixels (10 L-R) & (10 U-D) respectively
blurred_image = cv2.blur(image, ksize)                                  # Apply blur

combined_image = np.hstack((image, blurred_image))                      # Appends blurred_image to end of image pixel array
combined_image = cv2.resize(combined_image, None, fx=1/3, fy=1/3)       # Resize to 1/3 original value

cv2.imshow('Original and Blurred', combined_image)
cv2.moveWindow('Original and Blurred', 0, 0)                            # Set window position to top left
cv2.setWindowProperty('Original and Blurred', cv2.WND_PROP_TOPMOST, 1)  # Bring the window to the front

cv2.waitKey(0)                                                          # Wait for any key press and then close the window
cv2.destroyAllWindows()
```

Output - Appears in top left corner



Question 2:

Use the example of Python Webserver, add the blurring function in question (1) so that it shows both the original and blurred image after you press on the Upload button as shown in the picture below.

Video Link - https://youtu.be/dDIt6Xpv7RE?si=31bRmiB2pavKJgNy

Server.py -

Server.py | template.html

C: > Users > oler9 > OneDrive > Documents > UC > 24Spring > ELEC 3520 IoT > Minilabs 3 Media Server > Server.py > index

```python
#@author: Joseph Oler
from flask import Flask, request, redirect, url_for, render_template
from flask import send_from_directory
import os
import cv2
import numpy as np
from werkzeug.utils import secure_filename

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'

def allowed_file(filename):
    return ('.' in filename) and (filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS)       #(peri

@app.route('/')                             # If first stop in server
def index():
    return render_template('template.html') # Generate template

@app.route('/upload', methods=['POST'])
def upload():
    if 'file' not in request.files:         # If not file selected when upload button pressed
        return redirect(request.url)        # reload webpage
    file = request.files['file']            # read file
    if file.filename == '':                 # if file was nothing
        return redirect(request.url)        # reload webpage
    if file and allowed_file(file.filename):                            # If file exists and is allowed
        filename = secure_filename(file.filename)                       # Secure file name creation
        up_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)   # Path to upload
        file.save(up_path)            # Saves the uploaded file to the location of the file path
        image = cv2.imread(up_path)   # AFTER upload to location, can then read from location
```

```python
        ksize = (21, 21)                                                # Kernel takes average of values in surrounding pixels (10 L-R) &
        blurred_image = cv2.blur(image, ksize)                          # Apply blur
        blur_path = os.path.join(app.config['UPLOAD_FOLDER'], filename+'_blur.png') # Blur path
        cv2.imwrite(blur_path, blurred_image)                           # Save blurred image

        combined_image = np.hstack((image, blurred_image))              # Appends blurred_image to end of image pixel array
        combined_image = cv2.resize(combined_image, None, fx=1/3, fy=1/3)       # Resize to 1/3 original value
        combined_file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename+'_combined.png')    # Path to combined image
        cv2.imwrite(combined_file_path, combined_image)                 # Writes the new image at the path location / names the new file
        return render_template('template.html', filename=filename+'_combined.png')     # Reloads webpage,
    else:
        return redirect(request.url)        # else reload webpage

@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)


if __name__ == '__main__':      # Begin
    app.run(debug=True)
```

Template.html - to be honest I use chat gpt to make it look nicer / format the html better

Server.py | template.html

C: > Users > oler9 > OneDrive > Documents > UC > 24Spring > ELEC 3520 IoT > Minilabs 3 Media Server > tem

```html
@author: Joseph Oler

<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Uploader</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }

        .container {
            max-width: 600px;
            margin: 50px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        h1 {
            text-align: center;
            margin-bottom: 30px;
        }

        form {
            text-align: center;
```

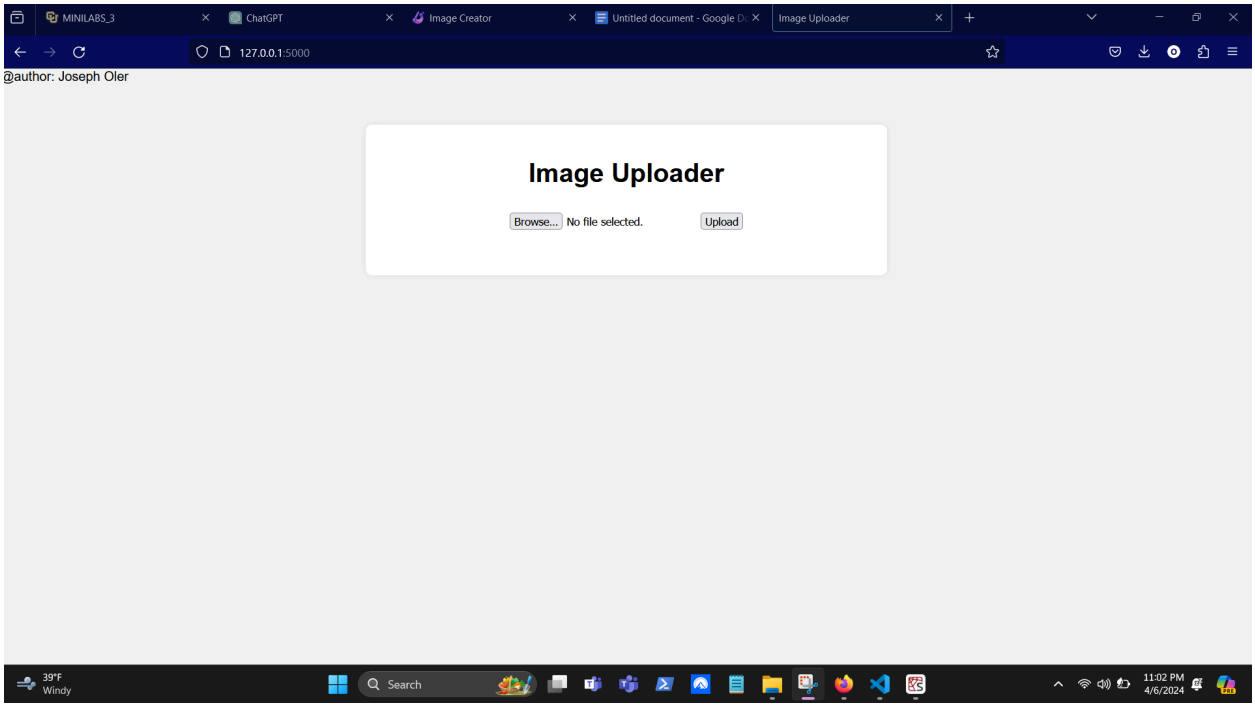
Server.py | template.html

C: > Users > oler9 > OneDrive > Documents > UC > 24Spring > ELEC 3520 IoT > Minilabs 3 Media Server > templates >

```
 4  <html>
 6  <head>
10      <style>
30          }
31
32          form {
33              text-align: center;
34          }
35
36          input[type="file"] {
37              margin-bottom: 20px;
38          }
39
40          img {
41              display: block;
42              margin: 20px auto;
43              max-width: 100%;
44              height: auto;
45          }
46      </style>
47  </head>
48
```

```
48
49  <body>
50      <div class="container">
51          <h1>Image Uploader</h1>
52          <form action="/upload" method="post" enctype="multipart/form-data">
53              <input type="file" name="file">
54              <input type="submit" value="Upload">
55          </form>
56          {% if filename %}
57          <h2>Uploaded Image:</h2>
58          <img src="{{ url_for('uploaded_file', filename=filename) }}" alt="Uploaded Image">
59          {% endif %}
60      </div>
61  </body>
62
63  </html>
```

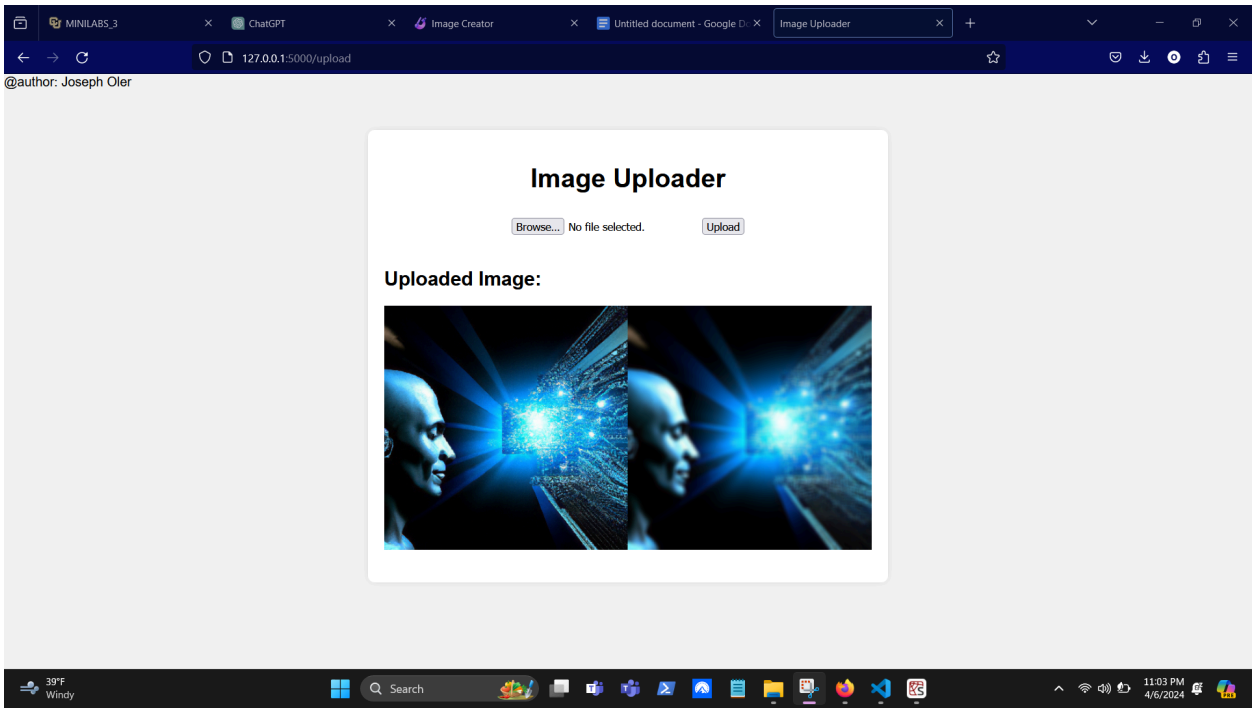
Output -



- Initial GET received

```
PS C:\Users\oler9\OneDrive\Documents\UC\24Spring\ELEC 3520 IoT\Minilabs 3 Media Server>  c:; cd 'c:\Users\oler9\OneDrive\Documents\UC\24Spring\ELEC 3520 IoT\Minilabs 3 Media Server'; &
 'c:\Users\oler9\miniconda3\envs\EEG\python.exe' 'c:\Users\oler9\.vscode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '60342' '
-' 'c:\Users\oler9\OneDrive\Documents\UC\24Spring\ELEC 3520 IoT\Minilabs 3 Media Server\Server.py'
 * Serving Flask app 'Server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 282-894-734
127.0.0.1 - - [06/Apr/2024 23:11:09] "GET / HTTP/1.1" 200 -
```

After upload -



- Pressing upload button creates POST, which uploads the file
- Page is then re-rendered using the template.html file and the recently uploaded photo

```
127.0.0.1 - - [06/Apr/2024 23:11:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/Apr/2024 23:11:53] "POST /upload HTTP/1.1" 200 -
127.0.0.1 - - [06/Apr/2024 23:11:54] "GET /uploads/5.png_combined.png HTTP/1.1" 200 -
```

Uploads Folder - files are uploaded correctly