要将上述的Python代码与之前提供的HTML代码结合在一起，你需要使用一个后端框架（如Flask或Django）来处理图像上传和图像处理的逻辑。下面是一个使用Flask框架的示例：

首先，确保你已安装 Flask 和 OpenCV，可以使用以下命令来安装它们：

```shell

pip install flask opencv-python

```

然后，创建一个名为 `app.py` 的 Python 文件，并将以下代码添加到文件中：

```python

from flask import Flask, render\_template, request

import cv2

import os

app = Flask(\_\_name\_\_)

# 加载人脸检测器

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/process', methods=['POST'])

def process():

# 获取上传的文件

file = request.files['imageInput']

# 保存上传的文件

file\_path = 'static/' + file.filename

file.save(file\_path)

# 加载图像

image = cv2.imread(file\_path)

# 将图像转换为灰度

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# 进行人脸检测

faces = face\_cascade.detectMultiScale(gray\_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# 对检测到的人脸进行模糊处理

for (x, y, w, h) in faces:

# 提取人脸区域

face\_roi = image[y:y+h, x:x+w]

# 对人脸区域进行模糊处理

blurred\_face = cv2.GaussianBlur(face\_roi, (99, 99), 30)

# 将模糊处理后的人脸区域放回原图像

image[y:y+h, x:x+w] = blurred\_face

# 保存处理后的图像

output\_path = 'static/blurred\_image.jpg'

cv2.imwrite(output\_path, image)

return render\_template('result.html', input\_image=file\_path, output\_image=output\_path)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

接下来，在同一目录下创建一个名为 `templates` 的文件夹，并在该文件夹中创建两个 HTML 文件：`index.html` 和 `result.html`。`index.html` 文件包含了上传表单，而 `result.html` 文件用于显示处理后的图像。

`index.html` 的内容如下：

```html

<!DOCTYPE html>

<html>

<head>

<title>人脸模糊网站</title>

<style>

/\* CSS样式 \*/

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

}

#preview {

max-width: 100%;

margin-bottom: 20px;

}

</style>

</head>

<body>

<div class="container">

<h1>人脸模糊网站</h1>

<form action="/process" method="post" enctype="multipart/form-data">

<input type="file" name="imageInput">

<br>

<button type="submit">模糊人脸</button>

</form>

</div>

</body>

</html>

```

`result.html` 的内容如下：

```html

<!DOCTYPE html>

<html>

<head>

<title>处理结果</title>

<style>

/\* CSS样式 \*/

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

}

#inputImage, #outputImage {

max-width: 100%;

margin-bottom: 20px;

}

</style>

</head>

<body>

<div class="container">

<h1>处理结果</h1>

<h2>原始图像：</h2>

<img id="inputImage" src="{{ input\_image }}" alt="Input Image">

<h2>模糊处理后的图像：</h2>

<img id="outputImage" src="{{ output\_image }}" alt="Output Image">

</div>

</body>

</html>

```

请确保在同一目录下创建一个名为 `static` 的文件夹，用于保存上传的图像和处理后的图像。

最后，在终端中运行以下命令启动 Flask 应用：

```shell

python app.py

```

现在，通过访问 `http://localhost:5000`，你将看到一个带有文件上传表单的页面。选择一张图片并点击“模糊人脸”按钮，然后你将被重定向到显示处理结果的页面。

这个示例将用户上传的图像保存在服务器上，并在服务器上进行人脸检测和模糊处理。处理后的图像将被保存在 `static/blurred\_image.jpg` 文件中，并在结果页面中显示。请注意，这只是一个基本的示例，你可以根据需要进行自定义和扩展。

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated