```python
import numpy as np
import pandas as pd
import cv2
import os
from matplotlib import pyplot as plt
from pylab import imread
from skimage.color import rgb2gray
def imshows(ImageData, LabelData, rows, cols, gridType = False):
  # Convert ImageData and LabelData to List
  from matplotlib import pyplot as plt
  ImageArray = list(ImageData)
  LabelArray = list(LabelData)
  if(rows == 1 & cols == 1):
    fig = plt.figure(figsize=(20,20))
  else:
    fig = plt.figure(figsize=(cols*8,rows*5))

  for i in range(1, cols * rows + 1):
      fig.add_subplot(rows, cols, i)
      image = ImageArray[i - 1]
      # If the channel number is less than 3, we display as grayscale image
      # otherwise, we display as color image
      if (len(image.shape) < 3):
          plt.imshow(image, plt.cm.gray)
          plt.grid(gridType)
      else:
          plt.imshow(image)
          plt.grid(gridType)
      plt.title(LabelArray[i - 1])
  plt.show()
# Read Image
image_color = imread("Sample03/tom.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)

kernel_3_3 = np.ones((3, 3), np.float32) / 9
kernel_5_5 = np.ones((5, 5), np.float32) / 25

image_filter_3_3_01 = cv2.filter2D(image_color, -1, kernel_3_3)
image_filter_3_3_02 = cv2.filter2D(image_filter_3_3_01, -1, kernel_3_3)

image_filter_5_5_01 = cv2.filter2D(image_color, -1, kernel_5_5)
image_filter_5_5_02 = cv2.filter2D(image_filter_5_5_01, -1, kernel_5_5)

def variance_of_laplacian(image):
    # compute the Laplacian of the image and then return the focus
    # measure, which is simply the variance of the Laplacian
    return cv2.Laplacian(image, cv2.CV_64F).var()
```

```python
blur_mesurement = variance_of_laplacian(image_color)
blur_mesurement_3_3_01 = variance_of_laplacian(image_filter_3_3_01)
blur_mesurement_3_3_02 = variance_of_laplacian(image_filter_3_3_02)
print("Blur Measurement of image_color:", blur_mesurement)
print("Blur Measurement of image_filter_3_3_01:", blur_mesurement_3_3_01)
print("Blur Measurement of image_filter_3_3_02:", blur_mesurement_3_3_02)
print()

text = "Blurry measurement"
fm = blur_mesurement
image_color_text = image_color.copy()
cv2.putText(image_color_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

text = "Blurry measurement"
fm = blur_mesurement_3_3_01
image_filter_3_3_01_text = image_filter_3_3_01.copy()
cv2.putText(image_filter_3_3_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

text = "Blurry measurement"
fm = blur_mesurement_3_3_02
image_filter_3_3_02_text = image_filter_3_3_02.copy()
cv2.putText(image_filter_3_3_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

ShowThreeImages(image_color, image_filter_3_3_01, image_filter_3_3_02)
ShowThreeImages(image_color_text, image_filter_3_3_01_text, image_filter_3_3_02_text)
```

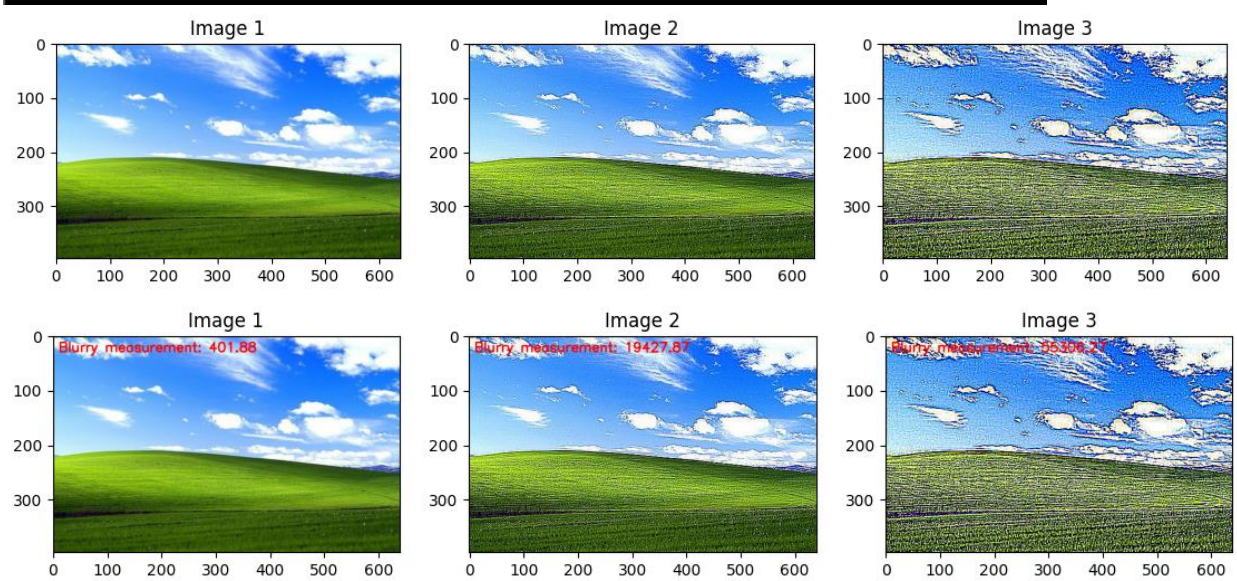
```
Blur Measurement of image_color: 1403.851635077047
Blur Measurement of image_filter_3_3_01: 104.61403226650074
Blur Measurement of image_filter_3_3_02: 39.79793081032584
```

```python
blur_mesurement = variance_of_laplacian(image_color)
blur_mesurement_5_5_01 = variance_of_laplacian(image_filter_5_5_01)
blur_mesurement_5_5_02 = variance_of_laplacian(image_filter_5_5_02)
print("Blur Measurement of image_color:", blur_mesurement)
print("Blur Measurement of image_filter_5_5_01:", blur_mesurement_5_5_01)
print("Blur Measurement of image_filter_5_5_02:", blur_mesurement_5_5_02)
print()

text = "Blurry measurement"
fm = blur_mesurement_5_5_01
image_filter_5_5_01_text = image_filter_5_5_01.copy()
cv2.putText(image_filter_5_5_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

text = "Blurry measurement"
fm = blur_mesurement_5_5_02
image_filter_5_5_02_text = image_filter_5_5_02.copy()
cv2.putText(image_filter_5_5_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

ShowThreeImages(image_color, image_filter_5_5_01, image_filter_5_5_02)
ShowThreeImages(image_color_text, image_filter_5_5_01_text, image_filter_5_5_02_text)
```

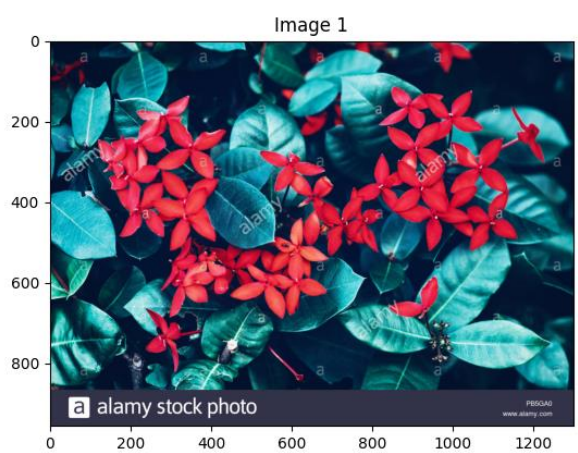
```
Blur Measurement of image_color: 1403.851635077047
Blur Measurement of image_filter_5_5_01: 31.140022647688728
Blur Measurement of image_filter_5_5_02: 8.32152548703258
```

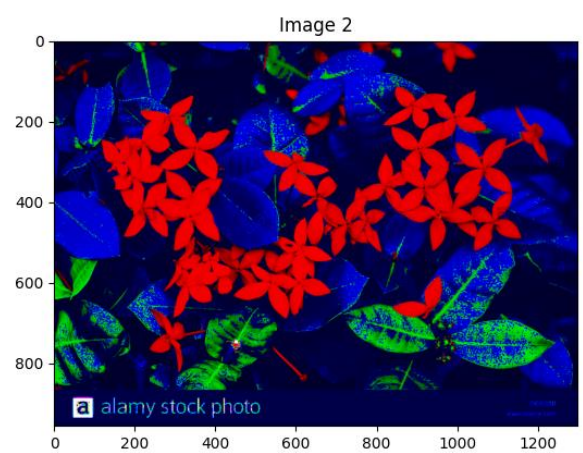
```python
# Read Image
image_color = imread("Sample03/windowxp.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)

kernel_sharpen_01 = np.array([[-1,-1,-1],
                              [-1, 9,-1],
                              [-1,-1,-1]])

kernel_sharpen_02 = np.array([[-1,-1,-1,-1,-1],
                              [-1,-1,-1,-1,-1],
                              [-1,-1,25,-1,-1],
                              [-1,-1,-1,-1,-1],
                              [-1,-1,-1,-1,-1]])

image_color_sharpen_01 = cv2.filter2D(image_color, -1, kernel_sharpen_01)
image_color_sharpen_02 = cv2.filter2D(image_color, -1, kernel_sharpen_02)
blur_mesurement = variance_of_laplacian(image_color)
blur_mesurement_sharpen_01 = variance_of_laplacian(image_color_sharpen_01)
blur_mesurement_sharpen_02 = variance_of_laplacian(image_color_sharpen_02)
print("Sharpen Measurement of image_color:", variance_of_laplacian(image_color))
print("Sharpen Measurement of image_color_sharpen_01:", variance_of_laplacian(image_color_sharpen_01))
print("Sharpen Measurement of image_color_sharpen_02:", variance_of_laplacian(image_color_sharpen_02))
print()

text = "Blurry measurement"
fm = blur_mesurement
image_color_text = image_color.copy()
cv2.putText(image_color_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

text = "Blurry measurement"
fm = blur_mesurement_sharpen_01
image_color_sharpen_01_text = image_color_sharpen_01.copy()
cv2.putText(image_color_sharpen_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

text = "Blurry measurement"
fm = blur_mesurement_sharpen_02
image_color_sharpen_02_text = image_color_sharpen_02.copy()
cv2.putText(image_color_sharpen_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)

ShowThreeImages(image_color, image_color_sharpen_01, image_color_sharpen_02)
ShowThreeImages(image_color_text, image_color_sharpen_01_text, image_color_sharpen_02_text)
```

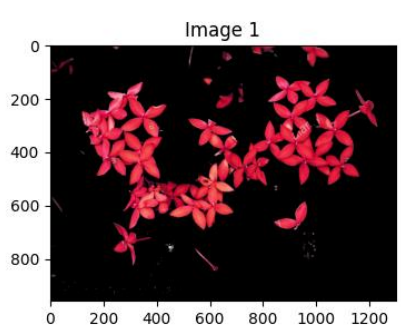
```
Sharpen Measurement of image_color: 401.88179073956996
Sharpen Measurement of image_color_sharpen_01: 19427.87461415046
Sharpen Measurement of image_color_sharpen_02: 55306.27306907043
```

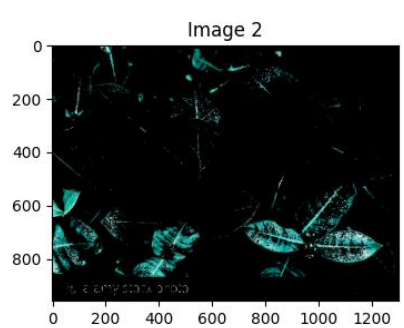


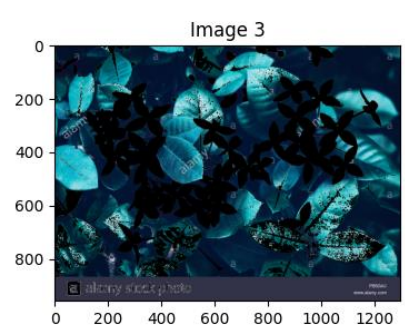
```python
# Read Image
image_color = imread("Sample03/flower.jpg")
# Convert Image into Gray
image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)

def max_rgb_filter(image):
    # split the image into its BGR components
    (B, G, R) = cv2.split(image)
    # find the maximum pixel intensity values for each
    # (x, y)-coordinate,, then set all pixel values less
    # than M to zero
    M = np.maximum(np.maximum(R, G), B)
    R[R < M] = 0
    G[G < M] = 0
    B[B < M] = 0
    # merge the channels back together and return the image
    return cv2.merge([B, G, R])

image_color_rgbmax = max_rgb_filter(image_color)
ShowTwoImages(image_color, image_color_rgbmax)
def SegmentColorImageByMask(IM, Mask):
    Mask = Mask.astype(np.uint8)
    result = cv2.bitwise_and(IM, IM, mask = Mask)
    return result

image_maxR_mask = image_gray < 0
image_maxG_mask = image_gray < 0
image_maxB_mask = image_gray < 0

R = image_color_rgbmax[:,:,0]
G = image_color_rgbmax[:,:,1]
B = image_color_rgbmax[:,:,2]

image_maxR_mask[(G == 0) & (B == 0)] = 1
image_maxG_mask[(R == 0) & (B == 0)] = 1
image_maxB_mask[(G == 0) & (R == 0)] = 1

image_maxR = SegmentColorImageByMask(image_color, image_maxR_mask)
image_maxG = SegmentColorImageByMask(image_color, image_maxG_mask)
image_maxB = SegmentColorImageByMask(image_color, image_maxB_mask)

ShowThreeImages(image_maxR, image_maxG, image_maxB)
```





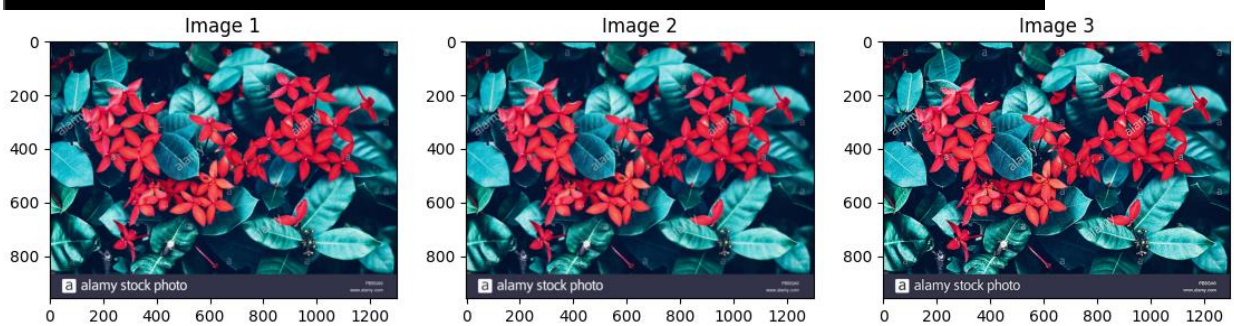




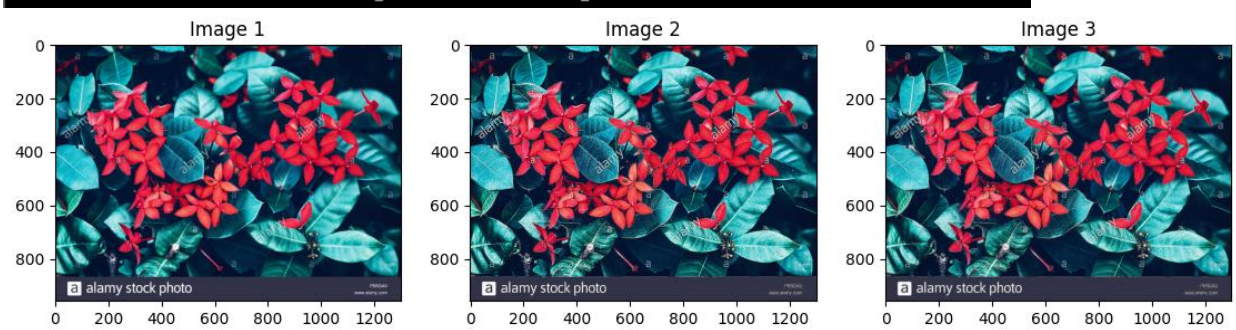


```python
def variance_of_laplacian(image):
    # compute the Laplacian of the image and then return the focus
    # measure, which is simply the variance of the Laplacian
    return cv2.Laplacian(image, cv2.CV_64F).var()

kernel_sharpen_01 = np.array([[-1,-1,-1],
                              [-1,9,-1],
                              [-1,-1,-1]])

kernel_3_3 = np.ones((3, 3), np.float32) / 9

image_color_blur_sharpen_01 = cv2.filter2D(image_color, -1, kernel_3_3)
image_color_blur_sharpen_02 = cv2.filter2D(image_color_blur_sharpen_01, -1, kernel_sharpen_01)
blur_mesurement = variance_of_laplacian(image_color)
blur_mesurement_blur_sharpen = variance_of_laplacian(image_color_blur_sharpen_02)
print("Blur Measurement of image_color:", blur_mesurement)
print("Blur Measurement of image_color_blur_sharpen:", blur_mesurement_blur_sharpen)
print()
ShowThreeImages(image_color, image_color_blur_sharpen_01, image_color_blur_sharpen_02)


image_color_sharpen_blur_01 = cv2.filter2D(image_color, -1, kernel_sharpen_01)
image_color_sharpen_blur_02 = cv2.filter2D(image_color_sharpen_blur_01, -1, kernel_3_3)
blur_mesurement = variance_of_laplacian(image_color)
blur_mesurement_sharpen_blur = variance_of_laplacian(image_color_sharpen_blur_02)
print("Blur Measurement of image_color:", blur_mesurement)
print("Blur Measurement of image_color_sharpen_blur:", blur_mesurement_sharpen_blur)
print()
ShowThreeImages(image_color, image_color_sharpen_blur_01, image_color_sharpen_blur_02)
```

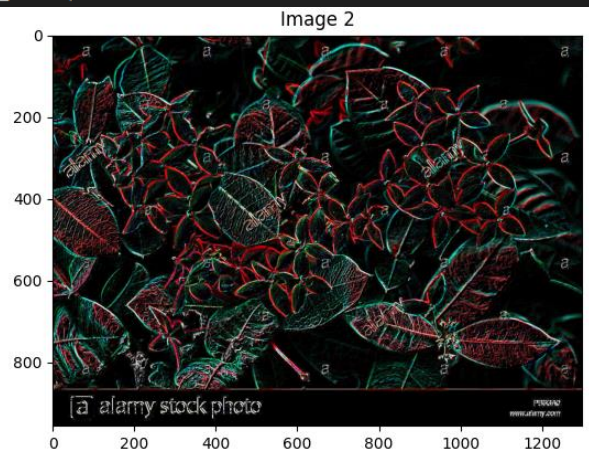
```
Blur Measurement of image_color: 536.5564347527203
Blur Measurement of image_color_blur_sharpen: 1520.3284946453266
```

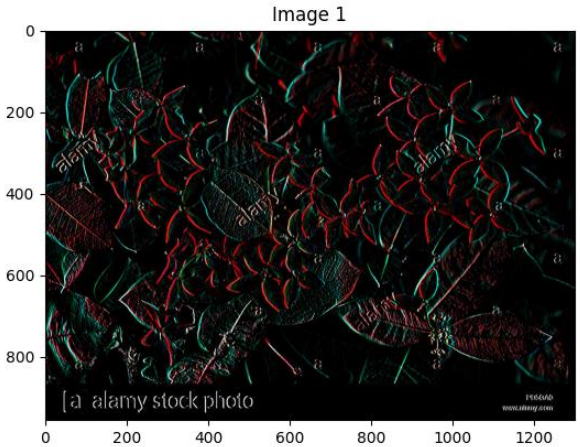


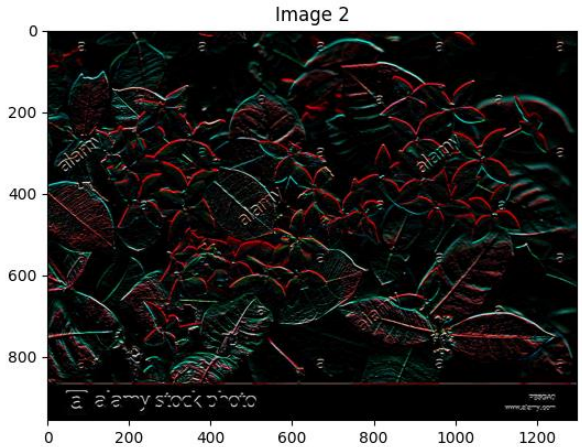

```
Blur Measurement of image_color: 536.5564347527203
Blur Measurement of image_color_sharpen_blur: 608.4553634123788
```



```python
kernel_sobelX = np.array((
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]), dtype="int")

# construct the Sobel y-axis kernel
kernel_sobelY = np.array((
    [-1, -2, -1],
    [0, 0, 0],
    [1, 2, 1]), dtype="int")

image_color_edge_sobelX = cv2.filter2D(image_color, -1, kernel_sobelX)
image_color_edge_sobelY = cv2.filter2D(image_color, -1, kernel_sobelY)
image_color_edge = image_color_edge_sobelX + image_color_edge_sobelY

ShowTwoImages(image_color, image_color_edge)
ShowTwoImages(image_color_edge_sobelX, image_color_edge_sobelY)
```







```python
def read_and_check_images_from_folder(folder):
    images = []
    names = []
    blur_check = 500
    for filename in os.listdir(folder):
        img = imread(os.path.join(folder, filename))
        if img is not None:
            blur_measurement = variance_of_laplacian(img)
            if (blur_measurement < blur_check):
                text = "Blurry Image"
            else:
                text = "Good Image"
            cv2.putText(img, "{}".format(text), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
            images.append(img)
            names.append(filename)
    path = "Image_BlurDetection_Output"
    os.mkdir(path)
    os.chdir(path)
    for i in range(len(images)):
        plt.imsave(names[i], images[i])

read_and_check_images_from_folder("Image_Input")
```

Image_Input	28/11/2021 9:46 PM	File folder	
Sample03	28/11/2021 8:06 PM	File folder	
Lab03.py	28/11/2021 9:46 PM	Python File	12 KB
Lab03.pyproj	26/11/2021 8:00 AM	Python Project	2 KB

Digital Image Processing ▸ Lab03 ▸ Lab03 ▸ Image_Input

Search Image_Input

Help

library ▾ Share with ▾ Slide show Burn New folder

boys.jpg

freereal.jpg

oblivion.jpg

patrick.jpg

yellowsmile.jpg

Image_BlurDetection_Output	28/11/2021 9:46 PM	File folder	
Image_Input	28/11/2021 9:46 PM	File folder	
Sample03	28/11/2021 8:06 PM	File folder	
Lab03.py	28/11/2021 9:46 PM	Python File	12 KB
Lab03.pyproj	26/11/2021 8:00 AM	Python Project	2 KB

age Processing ▸ Lab03 ▸ Lab03 ▸ Image_BlurDetection_Output Search Image_BlurDetection_Output

Help

library ▾ Share with ▾ Slide show Burn New folder

boys.jpg

freereal.jpg

oblivion.jpg

patrick.jpg

yellowsmile.jpg