

```

1  import numpy as np
2  import pandas as pd
3  import cv2
4  import os
5  from matplotlib import pyplot as plt
6  from pylab import imread
7  from skimage.color import rgb2gray
8  def imshow(ImageData, LabelData, rows, cols, gridType = False):
9      # Convert ImageData and LabelData to List
10     from matplotlib import pyplot as plt
11     ImageArray = list(ImageData)
12     LabelArray = list(LabelData)
13     if(rows == 1 & cols == 1):
14         fig = plt.figure(figsize=(20,20))
15     else:
16         fig = plt.figure(figsize=(cols*8,rows*5))
17
18     for i in range(1, cols * rows + 1):
19         fig.add_subplot(rows, cols, i)
20         image = ImageArray[i - 1]
21         # If the channel number is less than 3, we display as grayscale image
22         # otherwise, we display as color image
23         if (len(image.shape) < 3):
24             plt.imshow(image, plt.cm.gray)
25             plt.grid(gridType)
26         else:
27             plt.imshow(image)
28             plt.grid(gridType)
29             plt.title(LabelArray[i - 1])
30     plt.show()
44     # Read Image
45     image_color = imread("Sample03/tom.jpg")
46     # Convert Image into Gray
47     image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)
48
49     kernel_3_3 = np.ones((3, 3), np.float32) / 9
50     kernel_5_5 = np.ones((5, 5), np.float32) / 25
51
52     image_filter_3_3_01 = cv2.filter2D(image_color, -1, kernel_3_3)
53     image_filter_3_3_02 = cv2.filter2D(image_filter_3_3_01, -1, kernel_3_3)
54
55     image_filter_5_5_01 = cv2.filter2D(image_color, -1, kernel_5_5)
56     image_filter_5_5_02 = cv2.filter2D(image_filter_5_5_01, -1, kernel_5_5)
57
58     def variance_of_laplacian(image):
59         # compute the Laplacian of the image and then return the focus
60         # measure, which is simply the variance of the Laplacian
61         return cv2.Laplacian(image, cv2.CV_64F).var()

```

```

68 blur_mesurement = variance_of_laplacian(image_color)
69 blur_mesurement_3_3_01 = variance_of_laplacian(image_filter_3_3_01)
70 blur_mesurement_3_3_02 = variance_of_laplacian(image_filter_3_3_02)
71 print("Blur Measurement of image_color:", blur_mesurement)
72 print("Blur Measurement of image_filter_3_3_01:", blur_mesurement_3_3_01)
73 print("Blur Measurement of image_filter_3_3_02:", blur_mesurement_3_3_02)
74 print()
75
76 text = "Blurry measurement"
77 fm = blur_mesurement
78 image_color_text = image_color.copy()
79 cv2.putText(image_color_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
80
81 text = "Blurry measurement"
82 fm = blur_mesurement_3_3_01
83 image_filter_3_3_01_text = image_filter_3_3_01.copy()
84 cv2.putText(image_filter_3_3_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
85
86 text = "Blurry measurement"
87 fm = blur_mesurement_3_3_02
88 image_filter_3_3_02_text = image_filter_3_3_02.copy()
89 cv2.putText(image_filter_3_3_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
90
91 ShowThreeImages(image_color, image_filter_3_3_01, image_filter_3_3_02)
92 ShowThreeImages(image_color_text, image_filter_3_3_01_text, image_filter_3_3_02_text)

```

```

Blur Measurement of image_color: 1403.8516350777047
Blur Measurement of image_filter_3_3_01: 104.61403226650074
Blur Measurement of image_filter_3_3_02: 39.79793081032584

```



```

94 blur_measurement = variance_of_laplacian(image_color)
95 blur_measurement_5_5_01 = variance_of_laplacian(image_filter_5_5_01)
96 blur_measurement_5_5_02 = variance_of_laplacian(image_filter_5_5_02)
97 print("Blur Measurement of image_color:", blur_measurement)
98 print("Blur Measurement of image_filter_5_5_01:", blur_measurement_5_5_01)
99 print("Blur Measurement of image_filter_5_5_02:", blur_measurement_5_5_02)
100 print()
101
102 text = "Blurry measurement"
103 fm = blur_measurement_5_5_01
104 image_filter_5_5_01_text = image_filter_5_5_01.copy()
105 cv2.putText(image_filter_5_5_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
106
107 text = "Blurry measurement"
108 fm = blur_measurement_5_5_02
109 image_filter_5_5_02_text = image_filter_5_5_02.copy()
110 cv2.putText(image_filter_5_5_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
111
112 ShowThreeImages(image_color, image_filter_5_5_01, image_filter_5_5_02)
113 ShowThreeImages(image_color_text, image_filter_5_5_01_text, image_filter_5_5_02_text)

```

```

Blur Measurement of image_color: 1403.8516350777047
Blur Measurement of image_filter_5_5_01: 31.140022647688728
Blur Measurement of image_filter_5_5_02: 8.32152548703258

```



```

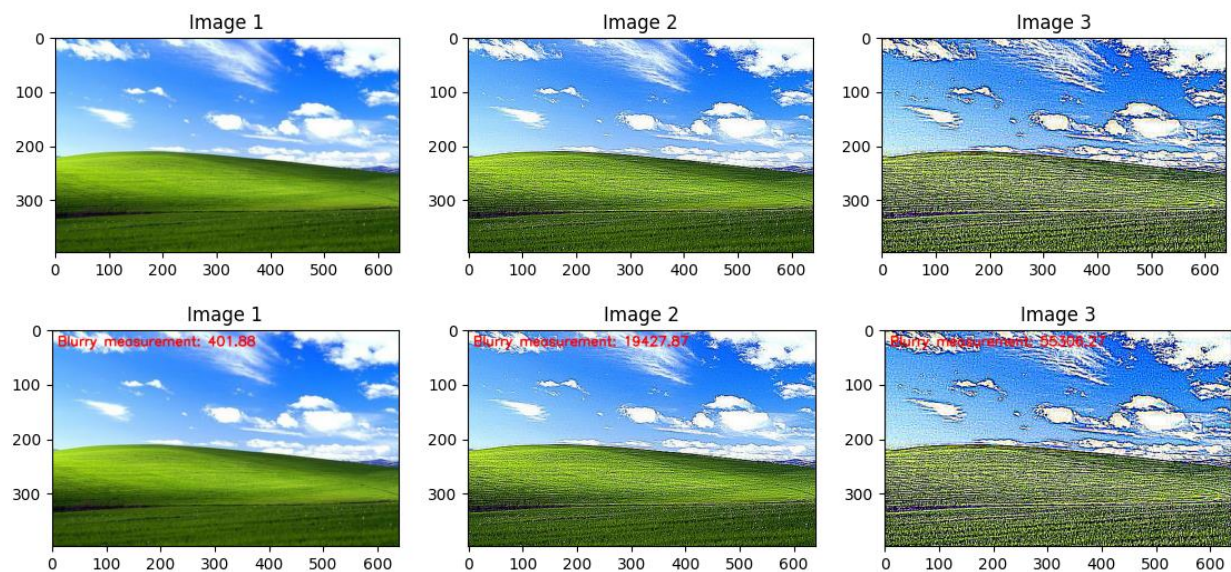
223 # Read Image
224 image_color = imread("Sample03/windowxp.jpg")
225 # Convert Image into Gray
226 image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)
227
228 kernel_sharpen_01 = np.array([[ -1, -1, -1],
229                               [-1, 9, -1],
230                               [-1, -1, -1]])
231
232 kernel_sharpen_02 = np.array([[ -1, -1, -1, -1, -1],
233                               [-1, -1, -1, -1, -1],
234                               [-1, -1, 25, -1, -1],
235                               [-1, -1, -1, -1, -1],
236                               [-1, -1, -1, -1, -1]])
237
238 image_color_sharpen_01 = cv2.filter2D(image_color, -1, kernel_sharpen_01)
239 image_color_sharpen_02 = cv2.filter2D(image_color, -1, kernel_sharpen_02)
240
241 blur_mesurement = variance_of_laplacian(image_color)
242 blur_mesurement_sharpen_01 = variance_of_laplacian(image_color_sharpen_01)
243 blur_mesurement_sharpen_02 = variance_of_laplacian(image_color_sharpen_02)
244 print("Sharpen Measurement of image_color:", variance_of_laplacian(image_color))
245 print("Sharpen Measurement of image_color_sharpen_01:", variance_of_laplacian(image_color_sharpen_01))
246 print("Sharpen Measurement of image_color_sharpen_02:", variance_of_laplacian(image_color_sharpen_02))
247 print()
248
249 text = "Blurry measurement"
250 fm = blur_mesurement
251 image_color_text = image_color.copy()
252 cv2.putText(image_color_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
253
254 text = "Blurry measurement"
255 fm = blur_mesurement_sharpen_01
256 image_color_sharpen_01_text = image_color_sharpen_01.copy()
257 cv2.putText(image_color_sharpen_01_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
258
259 text = "Blurry measurement"
260 fm = blur_mesurement_sharpen_02
261 image_color_sharpen_02_text = image_color_sharpen_02.copy()
262 cv2.putText(image_color_sharpen_02_text, "{}: {:.2f}".format(text, fm), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
263
264 ShowThreeImages(image_color, image_color_sharpen_01, image_color_sharpen_02)
265 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

```

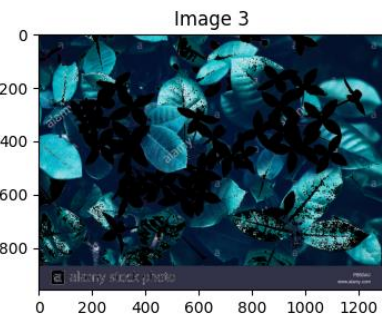
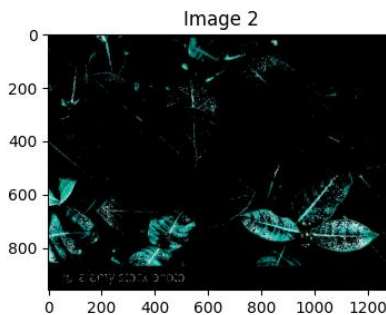
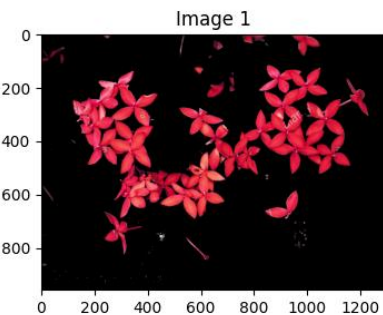
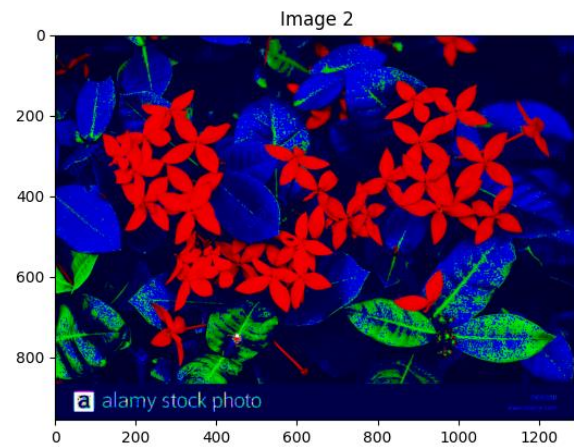




```

125 # Read Image
126 image_color = imread("Sample03/flower.jpg")
127 # Convert Image into Gray
128 image_gray = cv2.cvtColor(image_color, cv2.COLOR_RGB2GRAY)
129
130 def max_rgb_filter(image):
131     # split the image into its BGR components
132     (B, G, R) = cv2.split(image)
133     # find the maximum pixel intensity values for each
134     # (x, y)-coordinate,, then set all pixel values less
135     # than M to zero
136     M = np.maximum(np.maximum(R, G), B)
137     R[R < M] = 0
138     G[G < M] = 0
139     B[B < M] = 0
140     # merge the channels back together and return the image
141     return cv2.merge([B, G, R])
142
143 image_color_rgbmax = max_rgb_filter(image_color)
144 ShowTwoImages(image_color, image_color_rgbmax)
145
146 def SegmentColorImageByMask(IM, Mask):
147     Mask = Mask.astype(np.uint8)
148     result = cv2.bitwise_and(IM, IM, mask = Mask)
149     return result
150
151 image_maxR_mask = image_gray < 0
152 image_maxG_mask = image_gray < 0
153 image_maxB_mask = image_gray < 0
154
155 R = image_color_rgbmax[:, :, 0]
156 G = image_color_rgbmax[:, :, 1]
157 B = image_color_rgbmax[:, :, 2]
158
159 image_maxR_mask[(G == 0) & (B == 0)] = 1
160 image_maxG_mask[(R == 0) & (B == 0)] = 1
161 image_maxB_mask[(G == 0) & (R == 0)] = 1
162
163 image_maxR = SegmentColorImageByMask(image_color, image_maxR_mask)
164 image_maxG = SegmentColorImageByMask(image_color, image_maxG_mask)
165 image_maxB = SegmentColorImageByMask(image_color, image_maxB_mask)
166
167 ShowThreeImages(image_maxR, image_maxG, image_maxB)

```



```

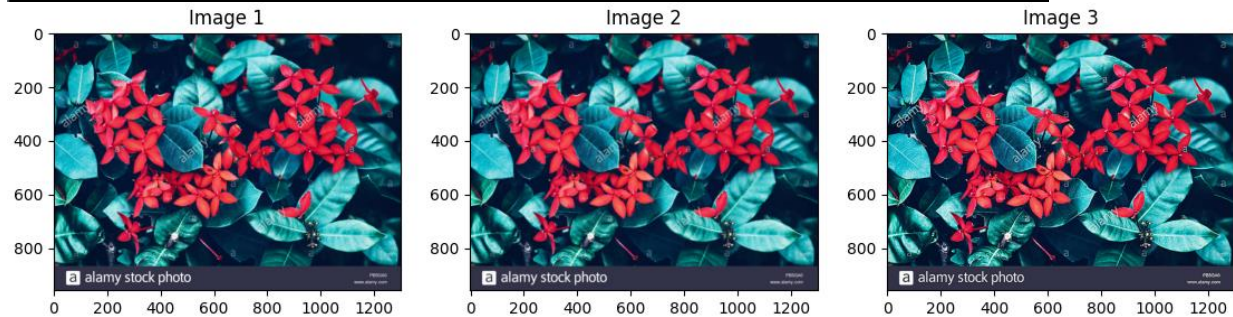
172 def variance_of_laplacian(image):
173     # compute the Laplacian of the image and then return the focus
174     # measure, which is simply the variance of the Laplacian
175     return cv2.Laplacian(image, cv2.CV_64F).var()
176
177 kernel_sharpen_01 = np.array([[ -1,-1,-1],
178                               [-1,9,-1],
179                               [-1,-1,-1]])
180
181 kernel_3_3 = np.ones((3, 3), np.float32) / 9
182
183 image_color_blur_sharpen_01 = cv2.filter2D(image_color, -1, kernel_3_3)
184 image_color_blur_sharpen_02 = cv2.filter2D(image_color_blur_sharpen_01, -1, kernel_sharpen_01)
185 blur_mesurement = variance_of_laplacian(image_color)
186 blur_mesurement_blur_sharpen = variance_of_laplacian(image_color_blur_sharpen_02)
187 print("Blur Measurement of image_color:", blur_mesurement)
188 print("Blur Measurement of image_color_blur_sharpen:", blur_mesurement_blur_sharpen)
189 print()
190 ShowThreeImages(image_color, image_color_blur_sharpen_01, image_color_blur_sharpen_02)
191
192
193 image_color_sharpen_blur_01 = cv2.filter2D(image_color, -1, kernel_sharpen_01)
194 image_color_sharpen_blur_02 = cv2.filter2D(image_color_sharpen_blur_01, -1, kernel_3_3)
195 blur_mesurement = variance_of_laplacian(image_color)
196 blur_mesurement_sharpen_blur = variance_of_laplacian(image_color_sharpen_blur_02)
197 print("Blur Measurement of image_color:", blur_mesurement)
198 print("Blur Measurement of image_color_sharpen_blur:", blur_mesurement_sharpen_blur)
199 print()
200 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

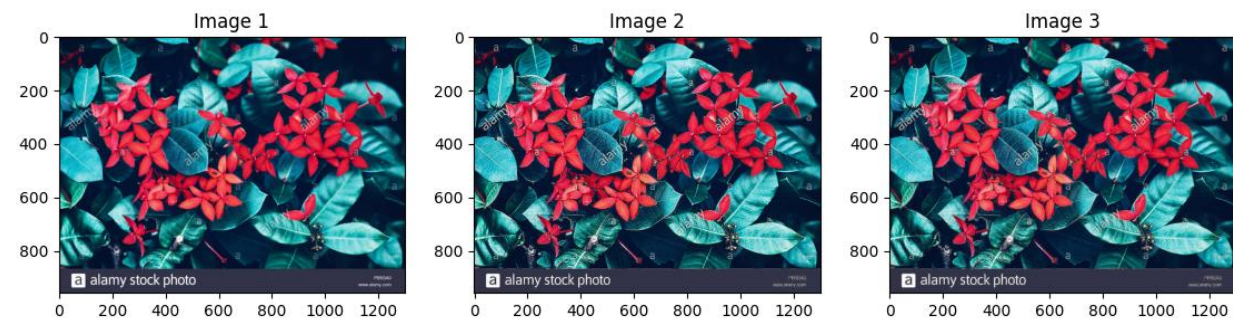
```



```

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

```

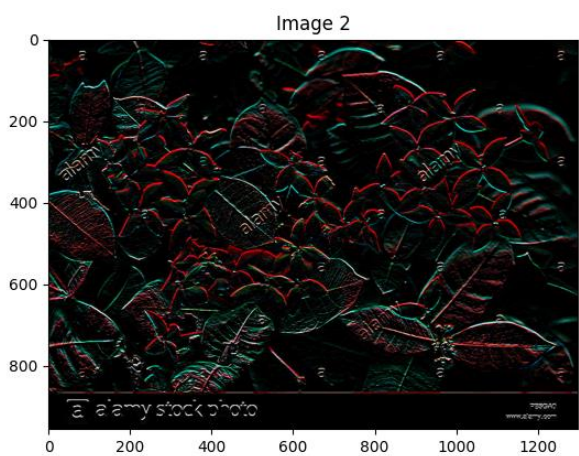
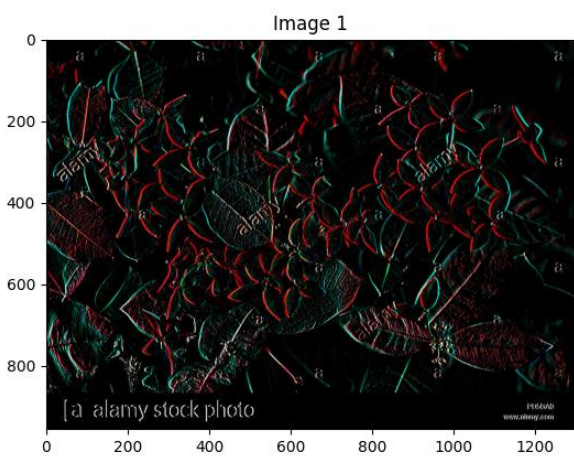
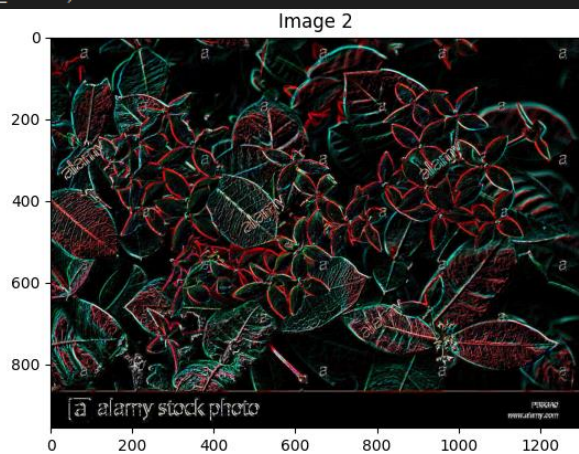




```

204 kernel_sobelX = np.array((
205     [-1, 0, 1],
206     [-2, 0, 2],
207     [-1, 0, 1]), dtype="int")
208
209 # construct the Sobel y-axis kernel
210 kernel_sobelY = np.array((
211     [-1, -2, -1],
212     [0, 0, 0],
213     [1, 2, 1]), dtype="int")
214
215 image_color_edge_sobelX = cv2.filter2D(image_color, -1, kernel_sobelX)
216 image_color_edge_sobelY = cv2.filter2D(image_color, -1, kernel_sobelY)
217 image_color_edge = image_color_edge_sobelX + image_color_edge_sobelY
218
219 ShowTwoImages(image_color, image_color_edge)
220 ShowTwoImages(image_color_edge_sobelX, image_color_edge_sobelY)





```



```

285 def read_and_check_images_from_folder(folder):
286     images = []
287     names = []
288     blur_check = 500
289     for filename in os.listdir(folder):
290         img = imread(os.path.join(folder, filename))
291         if img is not None:
292             blur_measurement = variance_of_laplacian(img)
293             if (blur_measurement < blur_check):
294                 text = "Blurry Image"
295             else:
296                 text = "Good Image"
297             cv2.putText(img, "{}".format(text), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
298             images.append(img)
299             names.append(filename)
300     path = "Image_BlurDetection_Output"
301     os.mkdir(path)
302     os.chdir(path)
303     for i in range(len(images)):
304         plt.imshow(names[i], images[i])
305
306 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

native americans: \*exist\*  
american government:



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

Image 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