

```
In [40]: 1 import numpy as np
          2 import skimage as sk
          3 import skimage.io as skio
          4 from skimage import filters
          5 import cv2 as cv
```

```
In [41]: 1 def trim_borders(img, percentage=0.08):
          2     # trim borders from the image to reduce noise for alignment
          3     # get dimensions
          4     height, width = img.shape[:2]
          5     # crop by a percentage
          6     border_height = int(height * percentage)
          7     border_width = int(width * percentage)
          8     # crop
          9     new_img = img[border_height:height-border_height, border_width:width-border_width]
         10     return new_img
```

```
In [42]: 1 def edge_detect_trim(img):
2         height, width = img.shape[:2]
3
4         # convert to 8-bit
5         img8 = cv.convertScaleAbs(img)
6
7         # increase contrast
8         clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
9         img8 = clahe.apply(img8)
10
11        # apply gaussian blur to reduce noise
12        img_blur = cv.GaussianBlur(img8, (3, 3), 0)
13
14        # generate edge map
15        edges = filters.sobel(img)
16
17        # conver to 8-bit
18        edges = cv.convertScaleAbs(edges)
19
20        # find contours
21        contours, _ = cv.findContours(edges, cv.RETR_EXTERNAL, cv.CHAI
22
23        if not contours:
24            print("no contours")
25            return img
26
27        # get the bounding box of the largest contour
28        x, y, w, h = cv.boundingRect(np.concatenate(contours))
29
30        # ensure bounding box is not too small
31        if x > 0.08 * width:
32            x = 0
33
34        if y > 0.08 * height:
35            y = 0
36
37        if w < 0.92 * width:
38            w = width
39
40        if y < 0.92 * height:
41            h = height
42
43        # crop the image using the bounding box
44        cropped_img = img[y:y+h, x:x+w]
45
46        height, width = cropped_img.shape[:2]
47
48        return cropped_img
```

```
In [43]: 1 def L2(image1, image2):
2         l2 = np.sqrt(np.sum((image1-image2) ** 2))
3         return l2
```

```
In [44]: 1 def L2_linalg(image1, image2):
2         l2 = np.linalg.norm(image1-image2, ord=2)
3         return l2
```

```
In [45]: 1 def NCC(image1, image2):
2         # dot product between two normalized vectors:
3         # (image1./||image1|| and image2./||image2||)
4         A = image1 / np.linalg.norm(image1)
5         B = image2 / np.linalg.norm(image2)
6         d = np.sum(A * B)
7         # print('d', d)
8         return d
```

```
In [46]: 1 def SSD(image1, image2):
2         return np.sum((image1 - image2) ** 2)
```

```
In [47]: 1 def align_channel(channel1, channel2, drange=15):
2         """
3         align channel2 with channel1
4         exhaustively search over a window of possible displacements
5         score each using image matching metric (eg, L2 norm, NCC)
6         take displacement with best score
7         """
8
9         # channel1_copy = trim_borders(channel1)
10        # channel2_copy = trim_borders(channel2)
11
12        best_offset = (0, 0)
13        min_score = float('inf')
14        best_shifted = channel2
15
16        # search over window of possible displacements
17        for x in range(-drange, drange + 1):
18            for y in range(-drange, drange + 1):
19                shifted = np.roll(channel2, shift=(x, y), axis=(0, 1))
20                score = SSD(channel1, shifted)
21
22                if score < min_score:
23                    min_score = score
24                    best_offset = (x, y)
25                    best_shifted = shifted
26
27        # print('naive final offset', best_offset)
28        return best_offset
```

```
In [50]: 1 def pyramid_align(channel1, channel2, levels=10, drange=15):
2         channel1_copy = channel1[:]
3         channel2_copy = channel2[:]
4
5         #     channel1_copy = trim_borders(channel1)
6         #     channel2_copy = trim_borders(channel2)
7
8         pyr_channel1 = [channel1_copy]
9         pyr_channel2 = [channel2_copy]
10
11        # build image pyramid
12        real_levels = 0
13        for l in range(levels):
14            if pyr_channel1[-1].size < 32:
15                break
16            real_levels += 1
17            pyr_channel1.append(cv.resize(pyr_channel1[-1], (0, 0), fx
18            pyr_channel2.append(cv.resize(pyr_channel2[-1], (0, 0), fx
19
20        # default offset is 0
21        offset = (0, 0)
22
23        # iterate from coarsest to finest
24        for level in range(real_levels - 1, -1, -1):
25            pc1 = pyr_channel1[level]
26            pc2 = pyr_channel2[level]
27
28            offset = (2 * offset[0], 2 * offset[1])
29
30            # shift by previous offset
31            pc2 = np.roll(pc2, shift=offset, axis=(0,1))
32
33            # get new offset
34            new_offset = align_channel(pc1, pc2, drange)
35
36            offset = (offset[0] + new_offset[0], offset[1] + new_offse
37
38        print('pyramid final offset', offset)
39        final_shifted = np.roll(channel2_copy, shift=offset, axis=(0,1
40        return final_shifted # offset
```

```
In [51]: 1 def edge_align(channel1, channel2, levels=10, drange=15):
2         #     channel1_copy = trim_borders(channel1)
3         #     channel2_copy = trim_borders(channel2)
4
5         channel1_copy = channel1[:]
6         channel2_copy = channel2[:]
7
8         edges1 = filters.sobel(channel1_copy)
9         edges2 = filters.sobel(channel2_copy)
10
11         offset = pyramid_align(edges1, edges2, levels, drange)
12         print('edge final offset', offset)
13
14         final_shifted = np.roll(channel2_copy, shift=offset, axis=(0,1))
15
16         return final_shifted
```

```
In [52]: 1 def color_image(imname="data/cathedral.jpg"):
2         print('imname', imname)
3
4         # read in the image
5         im = skio.imread(imname)
6
7         # convert to double (might want to do this later on to save me
8         im = sk.img_as_float(im)
9
10        # compute the height of each part (just 1/3 of total)
11        height = np.floor(im.shape[0] / 3.0).astype(int)
12
13        # separate color channels
14        b = im[:height]
15        g = im[height: 2*height]
16        r = im[2*height: 3*height]
17
18        # trim before aligning
19        b = edge_detect_trim(b)
20        g = edge_detect_trim(g)
21        r = edge_detect_trim(r)
22
23        height = min(b.shape[0], g.shape[0], r.shape[0])
24        width = min(b.shape[1], g.shape[1], r.shape[1])
25
26        b = b[:height, :width]
27        g = g[:height, :width]
28        r = r[:height, :width]
29
30        # align
31        ag = pyramid_align(b, g)
32        ar = pyramid_align(b, r)
33        # b = trim_borders(b)
34
35        # create a color image
36        im_out = np.dstack([ar, ag, b])
37
38        # save the image
39        fname = './out_path/pyr_align_edge_crop/out_{}.jpg'.format(imn
40        skio.imwrite(fname, im_out)
41
42        # display the image
43        # skio.imshow(im_out)
44        # skio.show()
```

```
In [35]: 1 def color_all():
2         import os
3
4         # Get the list of all files and directories
5         path = "data/"
6         dir_list = os.listdir(path)
7         dir_list = [path + d for d in dir_list]
8
9         dir_list.remove('data/.DS_Store')
10
11        for f in dir_list:
12            color_image(f)
```

```
In [53]: 1 color_image()
```

```
imname data/cathedral.jpg
pyramid final offset (-338, -375)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
pyramid final offset (-334, -376)
```

In [54]: 1 color_all()

```
imname data/emir.tif
pyramid final offset (-9355, -10590)
pyramid final offset (-9299, -7030)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/monastery.jpg
pyramid final offset (-339, -379)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
pyramid final offset (-338, -378)
imname data/church.tif
pyramid final offset (-9497, -10563)
pyramid final offset (-9441, -10599)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/three_generations.tif
pyramid final offset (-8602, -10474)
pyramid final offset (-8745, -10479)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/melons.tif
pyramid final offset (-9321, -10626)
pyramid final offset (-9291, -10626)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/onion_church.tif
no contours
no contours
no contours
pyramid final offset (-9593, -7540)
pyramid final offset (-9537, -7527)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/train.tif
pyramid final offset (-9519, -10852)
pyramid final offset (-9413, -10778)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/tobolsk.jpg
pyramid final offset (-338, -383)
```


Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
pyramid final offset (-335, -384)
imname data/icon.tif
pyramid final offset (-9069, -10612)
pyramid final offset (-8815, -10447)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/cathedral.jpg
pyramid final offset (-338, -375)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
pyramid final offset (-334, -376)
imname data/self_portrait.tif
pyramid final offset (-9237, -10589)
pyramid final offset (-9081, -10580)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/harvesters.tif
no contours
no contours
no contours
pyramid final offset (-9536, -11052)
pyramid final offset (-9397, -11051)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/sculpture.tif
pyramid final offset (-9436, -10959)
pyramid final offset (-9390, -10977)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/lady.tif
pyramid final offset (-9306, -10716)
pyramid final offset (-9259, -10719)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

In [39]: 1 color_all()

```
imname data/emir.tif
edge final offset (-9356, -10591)
edge final offset (-9298, -10574)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/monastery.jpg
edge final offset (-344, -379)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
edge final offset (-338, -380)
imname data/church.tif
edge final offset (-9473, -10561)
edge final offset (-9440, -10603)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/three_generations.tif
edge final offset (-8603, -10479)
edge final offset (-8745, -10480)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/melons.tif
edge final offset (-9325, -10626)
edge final offset (-9293, -7084)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/onion_church.tif
no contours
no contours
no contours
edge final offset (-9593, -7538)
edge final offset (-9538, -7527)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/train.tif
edge final offset (-9521, -10854)
edge final offset (-9477, -10757)
```

Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to uint8 prior to saving to suppress this warning.

```
imname data/tobolsk.jpg
edge final offset (-338, -383)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
edge final offset (-335, -384)
imname data/icon.tif
edge final offset (-9069, -10612)
edge final offset (-8814, -10446)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
imname data/cathedral.jpg
edge final offset (-336, -373)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
edge final offset (-329, -373)
imname data/self_portrait.tif
edge final offset (-9238, -10588)
edge final offset (-9079, -10578)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
imname data/harvesters.tif
no contours
no contours
no contours
edge final offset (-9583, -11055)
edge final offset (-9468, -11046)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
imname data/sculpture.tif
edge final offset (-9436, -10959)
edge final offset (-9390, -10977)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
imname data/lady.tif
edge final offset (-9303, -10710)
edge final offset (-9237, -10734)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

In [38]: 1 color_image()

```
imname data/cathedral.jpg
edge final offset (-336, -373)
```

```
Lossy conversion from float64 to uint8. Range [0, 1]. Convert image to
uint8 prior to saving to suppress this warning.
```

```
edge final offset (-329, -373)
```

In []: 1