

```
1 from skimage.io import imread
2 from skimage.data import camera
3 from scipy.ndimage import map_coordinates
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 def linear_polar(img, o=None, r=None, output=None, order=1, cont=0):
8     if o is None: o = np.array(img.shape[:2])/2 - 0.5
9     if r is None: r = (np.array(img.shape[:2])**2).sum()**0.5/2
10    if output is None:
11        shp = int(round(r)), int(round(r*2*np.pi))
12        output = np.zeros(shp, dtype=img.dtype)
13    elif isinstance(output, tuple):
14        output = np.zeros(output, dtype=img.dtype)
15    out_h, out_w = output.shape
16    out_img = np.zeros((out_h, out_w), dtype=img.dtype)
17    rs = np.linspace(0, r, out_h)
18    ts = np.linspace(0, np.pi*2, out_w)
19    xs = rs[:,None] * np.cos(ts) + o[1]
20    ys = rs[:,None] * np.sin(ts) + o[0]
21    map_coordinates(img, (ys, xs), order=order, output=output)
22    return output
23
24 def polar_linear(img, o=None, r=None, output=None, order=1, cont=0):
25     if r is None: r = img.shape[0]
26     if output is None:
27         output = np.zeros((r*2, r*2), dtype=img.dtype)
28     elif isinstance(output, tuple):
29         output = np.zeros(output, dtype=img.dtype)
30     if o is None: o = np.array(output.shape)/2 - 0.5
31     out_h, out_w = output.shape
32     ys, xs = np.mgrid[:out_h, :out_w] - o[:,None, None]
33     rs = (ys**2+xs**2)**0.5
34     # breakpoint()
35     ts = np.arccos(xs/rs)
36     ts[ys<0] = np.pi*2 - ts[ys<0]
37     ts *= (img.shape[1]-1)/(np.pi*2)
38     map_coordinates(img, (rs, ts), order=order, output=output)
39     return output
40
41 if __name__ == '__main__':
42     img = camera()
43     ax = plt.subplot(311)
44     ax.imshow(img)
45     out = linear_polar(img)
46     ax = plt.subplot(312)
47     ax.imshow(out)
48     img = polar_linear(out, output=img.shape)
49     ax = plt.subplot(313)
50     ax.imshow(img)
51     plt.show()
```