```
from skimage.io import imread
from skimage.data import camera
from scipy.ndimage import map_coordinates
import numpy as np
import matplotlib.pyplot as plt
import time
def linear_polar(img, o=None, r=None, output=None, order=1, cont=0, verbose =
0):
    if o is None: o = np.array(img.shape[:2])/2 - 0.5
    if r is None: r = (np.array(img.shape[:2])**2).sum()**0.5/2
    if output is None:
        shp = int(round(r)), int(round(r*2*np.pi))
        output = np.zeros(shp, dtype=img.dtype)
    elif isinstance(output, tuple):
        output = np.zeros(output, dtype=img.dtype)
    out_h, out_w = output.shape
    out_img = np.zeros((out_h, out_w), dtype=img.dtype)
    rs = np.linspace(0, r, out_h)
    ts = np.linspace(0, np.pi*2, out_w)
    xs = rs[:,None] * np.cos(ts) + o[1]
    ys = rs[:,None] * np.sin(ts) + o[0]
    # breakpoint()
    map_coordinates(img, (ys, xs), order=order, output=output)
    if verbose == 0:
        return output
    elif verbose > 0:
        return output, rs, ts, o, r, out_h, out_w

def polar_linear(img, o=None, r=None, output=None, order=1, cont=0):
    if r is None: r = img.shape[0]
    if output is None:
        output = np.zeros((r*2, r*2), dtype=img.dtype)
    elif isinstance(output, tuple):
        output = np.zeros(output, dtype=img.dtype)
    if o is None: o = np.array(output.shape)/2 - 0.5
    out_h, out_w = output.shape
    ys, xs = np.mgrid[:out_h, :out_w] - o[:,None,None]
    rs = (ys**2+xs**2)**0.5
    # breakpoint()
    ts = np.arccos(xs/rs)
    ts[ys<0] = np.pi*2 - ts[ys<0]
    ts *= (img.shape[1]-1)/(np.pi*2)
    map_coordinates(img, (rs, ts), order=order, output=output)
    return output
def map_pixel(i,j, img, o=None, r=None, output=None, order=1, cont=0, debug =
False, out_h = None, out_w = None):

    if o is None: o = np.array(img.shape[:2])/2 - 0.5
    if r is None: r = (np.array(img.shape[:2])**2).sum()**0.5/2
    if out_h is None or out_w is None or debug:
        if output is None:
            shp = int(round(r)), int(round(r*2*np.pi))
            output = np.zeros(shp, dtype=img.dtype)
        elif isinstance(output, tuple):
```

```
            output = np.zeros(output, dtype=img.dtype)
        out_h, out_w = output.shape
    y = i
    x = j
    y_orig = o[0]
    x_orig = o[1]

    rad = np.sqrt((x- x_orig)**2 + (y - y_orig)**2)
    theta =  (2*np.pi+(np.arctan2((y-y_orig), (x-x_orig))))%(2*np.pi)#(y-
y_orig)/(x-x_orig)) #(2*np.pi + np.arctan((y-y_orig)/(x-x_orig)))%(2*np.pi)
    # print(theta, y- y_orig, x-x_orig)

    rs = np.linspace(0, r, out_h)
    ts = np.linspace(0, np.pi*2, out_w)
    # breakpoint()
    r_index = np.digitize(rad, rs)
    theta_index = np.digitize(theta, ts)

    if debug:
        xs = rs[:,None] * np.cos(ts) + o[1]
        ys = rs[:,None] * np.sin(ts) + o[0]
        map_coordinates(img, (ys, xs), order=order, output=output)
        print(r_index, theta_index)
        print(ys[r_index, theta_index], xs[r_index, theta_index], i, j)
        # print(, i, j)
        print(output[r_index, theta_index])
        print(img[y,x])
        print(img[i,j])
    return r_index, theta_index, theta
    # breakpoint()
def unmap_pixel(radius,theta_idx, img,theta = None, o=None, r=None,
output=None, order=1, cont=0, debug = False, out_h = None, out_w = None):
    if r is None: r = img.shape[0]
    if output is None:
        output = np.zeros((r*2, r*2), dtype=img.dtype)
    elif isinstance(output, tuple):
        output = np.zeros(output, dtype=img.dtype)
    if o is None: o = np.array(output.shape)/2 - 0.5
    out_h, out_w = output.shape
    if theta == None:
        ts = np.linspace(0, 2*np.pi, img.shape[1])
        theta = ts[theta_idx]
    x = radius*np.cos(theta) + o[1]
    y = radius*np.sin(theta) + o[0]

    return int(np.round_(y)), int(np.round_(x))#r_index, theta_index, theta
if __name__ == '__main__':
    img = camera()
    ax = plt.subplot(311)
    ax.imshow(img)

    out, rs, ts,o, r, out_h, out_w = linear_polar(img, verbose = 1)
    print(time.time())

    oldtime = time.time()
```

```
106     coords = (120,1)
107     test =map_pixel(coords[0], coords[1],img[255:,:], o = o, r =r, out_h =
    out_h, out_w = out_w, debug= False )
108     original_args = unmap_pixel(test[0], test[1], out, output = img.shape,
    theta = test[2])
109     print(coords, original_args)
110     newtime = time.time()
111     print(newtime- oldtime)
112
113     ax = plt.subplot(312)
114     ax.imshow(out)
115     img = polar_linear(out, output=img.shape)
116     ax = plt.subplot(313)
117     ax.imshow(img)
118     plt.show()
```