Lab3\preLab3.py

```python
"""
CIVE 6374 – Optical Imaging Metrology
Professor: Dr. Craig Glennie
Author: Joshua Genova
Lab # 3
Description: Relative Orientation
Deadline: March 22, 2023 10:00 AM
"""

"""
Correct images before Lab 3 Tasks
1.) Affine Transformation into fiducial coordinates
2.) Principal Point Offset
3.) Radial Lens Distortion
4.) Decentering Lens Distortion
5.) Atmospheric Refraction
"""

import numpy as np
import math

 # Given from calibration certificate
focal_length = 153.358 # mm
principal_point_offset = [-0.006, 0.006] # [xp, yp] mm
radial_lens_distortion = [0.8878e-4, -0.1528e-7, 0.5256e-12] # [K0, K1, K2]
decentering_distortion = [0.1346e-06, 0.1224e-07] # [P1, P2]
c = focal_length # speed of light
# Given from handout
H = 1860/1000 # [km] elevation
h = 1100/1000 # [km] ground elevation
scale_number = 5000
image_size = 9 # in square
k_atmos = ((2410*H)/(H**2 -6*H + 250) - (2410*h)/(h**2 - 6*h + 250)*(h/H))*1e-6

# Image 1
delta_X1 = -122.01704301790505
delta_Y1 = 123.53429666924897
A1 = 0.011899426266928175
B1 = 0.00000299767744395384
C1 = -0.00000134050132901044
D1 = 0.011901264695956251
A_mat1 = np.array([[A1, B1], [C1, D1]])

# Image 2
delta_X2 = -122.19211044565897
delta_Y2 = 123.51804729053579
A2 = 0.011900088285313318
B2 = -8.456447779614914e-06
C2 = 7.403491422692827e-06
D2 = 0.011901033060072988
A_mat2 = np.array([[A2, B2], [C2, D2]])
```

```python
def get_fiducial(xc, yc, A_mat, x_delta, y_delta):
    delta_xy = np.array([[x_delta], [y_delta]])
    temp_c = np.array([[xc], [yc]])
    xf, yf = np.dot(A_mat, temp_c) + delta_xy
    return  xf[0], yf[0]

def get_pp(xf, yf, pp):
    x_pp = xf - pp[0]
    y_pp = yf - pp[1]
    r = math.sqrt(x_pp**2 + y_pp**2)
    return x_pp, y_pp, r

def get_radial(x_pp, y_pp, r, K):
    x_rad = -x_pp*(K[0] + K[1]*r**2 + K[2]*r**4)
    y_rad = -y_pp*(K[0] + K[1]*r**2 + K[2]*r**4)
    return x_rad, y_rad

def get_decentering(x_pp, y_pp , r, P):
    dec_x =  -(P[0]*(r**2 + 2*x_pp**2) + 2*P[1]*x_pp*y_pp)
    dec_y =  -(P[1]*(r**2 + 2*y_pp**2) + 2*P[0]*x_pp*y_pp)
    return dec_x, dec_y

def get_atmos(x_pp, y_pp ,r, c, K):
    atmos_x = -x_pp*K*(1 + r**2/c**2)
    atmos_y = -y_pp*K*(1 + r**2/c**2)
    return atmos_x, atmos_y

def get_total(xf, yf, pp, K, P, c, k_atmos):
    x_pp, y_pp, radius = get_pp(xf, yf, pp)
    x_rad, y_rad = get_radial(x_pp, y_pp, radius, K)
    x_dec, y_dec = get_decentering(x_pp, y_pp, radius, P)
    x_atmos, y_atmos = get_atmos(x_pp, y_pp, radius, c, k_atmos)

    x_total = x_pp + x_rad + x_dec + x_atmos
    y_total = y_pp + y_rad + y_dec + y_atmos
    return x_total, y_total

def get_left(xc, yc):
    xf, yf = get_fiducial(xc, yc, A_mat1, delta_X1, delta_Y1)
    xl, yl = get_total(xf, yf, principal_point_offset, radial_lens_distortion,
decentering_distortion, focal_length, k_atmos)
    return xl, yl

def get_right(xc, yc):
    xf, yf = get_fiducial(xc, yc, A_mat2, delta_X2, delta_Y2)
    xr, yr = get_total(xf, yf, principal_point_offset, radial_lens_distortion,
decentering_distortion, focal_length, k_atmos)
    return xr, yr

if __name__=="__main__":

    xc1 = [9460, 17400, 10059, 19158, 11844, 17842]
    yc1 = [-2292, -1661, -10883, -10412, -17253, -18028]

    xc2 = [1411, 9416, 2275, 11129, 4160, 10137]
    yc2 = [-2081, -1167, -10787, -10048, -17085, -17690]
    idx = 0
    fid_coords = np.zeros(shape=(len(xc1),2))
    pp_coords = np.zeros(shape=(len(xc1),2))
    rad_correction = np.zeros(shape=(len(xc1),2))
    rad_corrected = np.zeros(shape=(len(xc1),2))
    dec_correction = np.zeros(shape=(len(xc1),2))
    dec_corrected = np.zeros(shape=(len(xc1),2))
    atmos_correction = np.zeros(shape=(len(xc1),2))
    atmos_corrected = np.zeros(shape=(len(xc1),2))
    total_correction = np.zeros(shape=(len(xc1),2))
    correction = np.zeros(shape=(len(xc1),2))

    for i in range(len(xc1)):
        # get fiducial coordinates
        xf, yf = get_fiducial(xc1[i], yc1[i], A_mat1, delta_X1, delta_Y1)
        fid_coords[idx][0] = xf
        fid_coords[idx][1] = yf

        # apply principal point offset correction
        x_pp, y_pp, radius = get_pp(xf, yf, principal_point_offset)
        pp_coords[idx][0] = x_pp
        pp_coords[idx][1] = y_pp

        # apply radial lens distortion correction
        x_rad, y_rad = get_radial(x_pp, y_pp, radius, radial_lens_distortion)
        rad_correction[idx][0] = x_rad
        rad_correction[idx][1] = y_rad
        rad_corrected[idx][0] = x_pp + x_rad
        rad_corrected[idx][1] = y_pp + y_rad

        # apply decentering lens distortion correction
        x_dec, y_dec = get_decentering(x_pp, y_pp, radius, decentering_distortion)
        dec_correction[idx][0] = x_dec
        dec_correction[idx][1] = y_dec
        dec_corrected[idx][0] = x_pp + x_dec
        dec_corrected[idx][1] = y_pp + y_dec

        # apply atmospheric refraction correction
        x_atmos, y_atmos = get_atmos(x_pp, y_pp, radius, focal_length, k_atmos)
        atmos_correction[idx][0] = x_atmos
        atmos_correction[idx][1] = y_atmos
        atmos_corrected[idx][0] = x_pp + x_atmos
        atmos_corrected[idx][1] = y_pp + y_atmos

        # total correction
        total_correction[idx][0] = x_pp + x_rad + x_dec + x_atmos
        total_correction[idx][1] = y_pp + y_rad + y_dec + y_atmos

        x_total, y_total = get_total(xf, yf, principal_point_offset, radial_lens_distortion,
decentering_distortion, focal_length, k_atmos)
        correction[idx][0] = x_total
        correction[idx][1] = y_total


        xl, yl = get_left(xc1[i], yc1[i])
        xr, yr = get_right(xc2[i], yc2[i])
        idx += 1

    # print(f'Fiduciary Coordinates: \n{fid_coords}\n')
    # print(f'Principal Point Offset Correction: \n {pp_coords}\n')
    # print(f'Radial Lens Distortion (delta): \n {rad_correction}\n')
    # print(f'Radial Lens Distortion Correction: \n {rad_corrected}\n')
    # print(f'Decentering Lens Distortion(delta): \n {dec_correction}\n')
    # print(f'Decentering Lens Distortion Correction: \n {dec_corrected}\n')
    # print(f'Atmospheric Refration (delta): \n {atmos_correction}\n')
    # print(f'Atmospheric Refration Correction: \n {atmos_corrected}\n')
    # print(f'Total Correction: \n {total_correction}\n')
    # print(f'Total Correction: \n {correction}\n')
```