```
"""
CIVE 6374 – Optical Imaging Metrology
Professor: Dr. Craig Glennie
Author: Joshua Genova
Lab # 2
Description: Applying Image Corrections
Deadline: March 08, 2023 10:00 AM
"""
import numpy as np
import math
import matplotlib.pyplot as plt
from matplotlib import cm


def get_fiducial(xc, yc, A_mat, x_delta, y_delta):
    idx = 0
    xf = np.zeros(len(xc))
    yf = np.zeros(len(yc))
    fid_coords = np.zeros(shape=(len(xc),2))
    delta_xy = np.array([[x_delta], [y_delta]])
    for i in range(len(xc)):
        temp_c = np.array([[xc[i]], [yc[i]]])
        xf[idx], yf[idx] = np.dot(A_mat, temp_c) + delta_xy
        fid_coords[idx][0] = xf[idx]
        fid_coords[idx][1] = yf[idx]
        idx += 1
    print(f'Fiduciary Coordinates: \n{fid_coords}\n')
    return  fid_coords

def get_pp(fid_coords, pp):
    idx = 0
    fid_coords_pp = np.zeros(shape=(len(fid_coords),2))
    r_vals = np.zeros(shape=(len(fid_coords),1))
    for i in range(len(fid_coords)):
        x_temp = fid_coords[i][0] - pp[0]
        y_temp = fid_coords[i][1] - pp[1]
        fid_coords_pp[idx][0] = x_temp
        fid_coords_pp[idx][1] = y_temp
        r_vals[idx] = math.sqrt(x_temp**2 + y_temp**2)
        idx += 1
    print(f'Principal Point Offset Correction: \n {fid_coords_pp}\n')
    print(f'Radius (r): \n {r_vals}\n')
    return fid_coords_pp, r_vals

def get_radial(coords, r, K):
    idx = 0
    rad_correction = np.zeros(shape=(len(coords),2))
    corrected = np.zeros(shape=(len(coords),2))
    for i in range(len(coords)):
        rad_correction[idx][0] = -coords[i][0]*(K[0] + K[1]*r[i]**2 + K[2]*r[i]**4)
        rad_correction[idx][1] = -coords[i][1]*(K[0] + K[1]*r[i]**2 + K[2]*r[i]**4)
        corrected[idx][0] = coords[i][0] + rad_correction[idx][0]
        corrected[idx][1] = coords[i][1] + rad_correction[idx][1]
        idx += 1
    print(f'Radial Lens Distortion (delta): \n {rad_correction}\n')
    print(f'Radial Lens Distortion Correction: \n {corrected}\n')
```

```
    return rad_correction

def get_decentering(coords, r, P):
    idx = 0
    dec_correction = np.zeros(shape=(len(coords),2))
    corrected = np.zeros(shape=(len(coords),2))
    for i in range(len(coords)):
        dec_correction[idx][0] = -(P[0]*(r[i]**2 + 2*coords[i][0]**2) + 2*P[1]*coords[i]
[0]*coords[i][1])
        dec_correction[idx][1] = -(P[1]*(r[i]**2 + 2*coords[i][1]**2) + 2*P[0]*coords[i]
[0]*coords[i][1])
        corrected[idx][0] = coords[i][0] + dec_correction[idx][0]
        corrected[idx][1] = coords[i][1] + dec_correction[idx][1]
        idx += 1
    print(f'Decentering Lens Distortion(delta): \n {dec_correction}\n')
    print(f'Decentering Lens Distortion Correction: \n {corrected}\n')
    return dec_correction

def get_atmos(coords,r, c, K):
    idx = 0
    atmos_correction = np.zeros(shape=(len(coords),2))
    corrected = np.zeros(shape=(len(coords),2))
    for i in range(len(coords)):
        atmos_correction[idx][0] = -coords[i][0]*K*(1 + r[i]**2/c**2)
        atmos_correction[idx][1] = -coords[i][1]*K*(1 + r[i]**2/c**2)
        corrected[idx][0] = coords[i][0] + atmos_correction[idx][0]
        corrected[idx][1] = coords[i][1] + atmos_correction[idx][1]
        idx += 1
    print(f'Atmospheric Refration (delta): \n {atmos_correction}\n')
    print(f'Atmospheric Refration Correction: \n {corrected}\n')
    return atmos_correction

def new_coords(pp, rad, dec, atm):
    idx = 0
    new_coords = np.zeros(shape=(len(pp),2))
    for i in range(len(new_coords)):
        new_coords[idx][0] = pp[i][0] + rad[i][0] + dec[i][0] + atm[i][0]
        new_coords[idx][1] = pp[i][1] + rad[i][1] + dec[i][1] + atm[i][1]
        idx += 1
    print(f'Total Correction: \n {new_coords}\n')
    return new_coords


##########################################################################################
def new_radial(x, y, K, a):
    r2 = x**2 + y**2
    r4 = r2**2
    distorted = -a * (K[0] + K[1]*r2 + K[2]*r4)
    return distorted

def new_tangential(x, y, P, a):
    r2 = x**2 + y**2
    tangential_x = -(P[0]*(r2 + 2*x**2) + 2*P[1]*x*y)
    tangential_y = -(P[1]*(r2 + 2*y**2) + 2*P[0]*x*y)

    if a == 0:
        tangential = tangential_x
```

```
        else:
            tangential = tangential_y

        return tangential

    def new_atmos(x, y, c, K, a):
        r2 = x**2 + y**2
        atmos = -a*K*(1 + r2/c**2)
        return atmos

    def max_val(x, y, K, P, c, k_atmos, S):
        radial_x = abs(new_radial(x, y, K, x))
        radial_y = abs(new_radial(x, y, K, y))
        print(f'max radial x: {radial_x}, max radial y: {radial_y}')

        tangential_x = abs(new_tangential(x, y, P, 0))
        tangential_y = abs(new_tangential(x, y, P, 1))
        print(f'max tangential x: {tangential_x}, max tangential y: {tangential_y}')

        atmos_x = abs(new_atmos(x, y, c, k_atmos, x))
        atmos_y = abs(new_atmos(x, y, c, k_atmos, y))
        print(f'max atmospheric x: {atmos_x}, max atmospheric y: {atmos_y}')

        print(f'max radial x (ground): {radial_x*S}, max radial y (ground): {radial_y*S}')
        print(f'max tangential x (ground): {tangential_x*S}, max tangential y (ground):
    {tangential_y*S}')
        print(f'max atmospheric x (ground): {atmos_x*S}, max atmospheric y (ground): {atmos_y*S}')


    def create_grid(img_size, K, P, c, k_atmos):
        # convert img size to mm
        img_size_to_mm = img_size * 25.4 / 2

        x = np.linspace(-img_size_to_mm, img_size_to_mm, endpoint=True)
        y = np.linspace(-img_size_to_mm, img_size_to_mm, endpoint=True)
        X, Y = np.meshgrid(x, y)

        # radial
        x_rad = new_radial(X, Y, K, X)
        y_rad = new_radial(X, Y, K, Y)

        fig1 = plt.figure()
        ax1 = fig1.add_subplot(111, projection='3d')
        ax1.plot_surface(X, Y, x_rad, cmap='viridis')
        ax1.set_xlabel("X (mm)")
        ax1.set_ylabel("Y (mm)")
        ax1.set_zlabel("\nX Correction")
        ax1.set_title("X Radial Distortion Correction")
        fig2 = plt.figure()
        ax2 = fig2.add_subplot(111, projection='3d')
        ax2.plot_surface(X, Y, y_rad, cmap='viridis')
        ax2.set_xlabel("X (mm)")
        ax2.set_ylabel("Y (mm)")
        ax2.set_zlabel("\nY Correction")
        ax2.set_title("Y Radial Distortion Correction")
```

```
    # tangential
    x_tan = new_tangential(X, Y, P, 0)
    y_tan = new_tangential(X, Y, P, 1)

    fig3 = plt.figure()
    ax3 = fig3.add_subplot(111, projection='3d')
    ax3.plot_surface(X, Y, x_tan, cmap='viridis')
    ax3.set_xlabel("X (mm)")
    ax3.set_ylabel("Y (mm)")
    ax3.set_zlabel("\nX Correction")
    ax3.set_title("X Tangential Distortion Correction")
    fig4 = plt.figure()
    ax4 = fig4.add_subplot(111, projection='3d')
    ax4.plot_surface(X, Y, y_tan, cmap='viridis')
    ax4.set_xlabel("X (mm)")
    ax4.set_ylabel("Y (mm)")
    ax4.set_zlabel("\nY Correction")
    ax4.set_title("Y Tangential Distortion Correction")

    # radial + tangential
    fig5 = plt.figure()
    ax5 = fig5.add_subplot(111, projection='3d')
    ax5.plot_surface(X, Y, x_rad+x_tan, cmap='viridis')
    ax5.set_xlabel("X (mm)")
    ax5.set_ylabel("Y (mm)")
    ax5.set_zlabel("\nX Correction")
    ax5.set_title("X Combination Correction")
    fig6 = plt.figure()
    ax6 = fig6.add_subplot(111, projection='3d')
    ax6.plot_surface(X, Y, y_rad+y_tan, cmap='viridis')
    ax6.set_xlabel("X (mm)")
    ax6.set_ylabel("Y (mm)")
    ax6.set_zlabel("\nY Correction")
    ax6.set_title("Y Combination Correction")

    # atmospheric
    x_atmos = new_atmos(X, Y, c, k_atmos, X)
    y_atmos = new_atmos(X, Y, c, k_atmos, Y)

    fig7 = plt.figure()
    ax7 = fig7.add_subplot(111, projection='3d')
    ax7.plot_surface(X, Y, x_atmos, cmap='viridis')
    ax7.set_xlabel("X (mm)")
    ax7.set_ylabel("Y (mm)")
    ax7.set_zlabel("\nX Correction")
    ax7.set_title("X Atmospheric Refraction Correction")
    fig8 = plt.figure()
    ax8 = fig8.add_subplot(111, projection='3d')
    ax8.plot_surface(X, Y, y_atmos, cmap='viridis')
    ax8.set_xlabel("X (mm)")
    ax8.set_ylabel("Y (mm)")
    ax8.set_zlabel("\nY Correction")
    ax8.set_title("Y Atmospheric Refration Correction")

    plt.show()
    return
```

```
if __name__=="__main__":

    # 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
    # print(k_atmos)
    # Given from affine transformation:
    delta_X = -122.01704301790505
    delta_Y = 123.53429666924897
    A = 0.011899426266928175
    B = 0.000002997677447395384
    C = -0.00001340501329010 44
    D = 0.011901264695956251
    A_mat = np.array([[A, B], [C, D]])

    xc1 = [9460, 17400, 10059, 19158, 11844, 17842, 11781, 14009, 9617, 14686]
    yc1 = [-2292, -1661, -10883, -10412, -17253, -18028, -1174, -9749, -14502, -18204]

    xc2 = [1411, 9416, 2275, 11129, 4160, 10137, 3726, 6161, 1954, 6984]
    yc2 = [-2081, -1167, -10787, -10048, -17085, -17690, -854, -9528, -14416, -17948]

    # Image 1
    print("Image 1:")
    img1_fid = get_fiducial(xc1, yc1, A_mat, delta_X, delta_Y)
    img1_pp, img1_r = get_pp(img1_fid, principal_point_offset)
    img1_rad_correction = get_radial(img1_pp, img1_r, radial_lens_distortion)
    img1_dec_correction = get_decentering(img1_pp, img1_r, decentering_distortion)
    img1_atmos_correction = get_atmos(img1_pp, img1_r, c, k_atmos)
    img1_new_coords = new_coords(img1_pp, img1_rad_correction, img1_dec_correction,
img1_atmos_correction)


    delta_X = -122.19211044565897
    delta_Y = 123.51804729053579
    A = 0.011900088285313318
    B = -8.456447779614914e-06
    C = 7.403491422692827e-06
    D = 0.011901033060072988
    A_mat = np.array([[A, B], [C, D]])

    # Image 2
    print('-'*75)
    print("Image 2:")
    img2_fid = get_fiducial(xc2, yc2, A_mat, delta_X, delta_Y)
    img2_pp, img2_r = get_pp(img2_fid, principal_point_offset)
    img2_rad_correction = get_radial(img2_pp, img2_r, radial_lens_distortion)
```

```
    img2_dec_correction = get_decentering(img2_pp, img2_r, decentering_distortion)
    img2_atmos_correction = get_atmos(img2_pp, img2_r, c, k_atmos)
    img2_new_coords = new_coords(img2_pp, img2_rad_correction, img2_dec_correction,
img2_atmos_correction)


    # max_val(114.3, 114.3, radial_lens_distortion, decentering_distortion, c, k_atmos,
scale_number)


    grid_coords = create_grid(image_size, radial_lens_distortion, decentering_distortion, c,
k_atmos)
```