

# Exercise 2: Rigid and perspective transformations in homogeneous coordinates

02504 Computer vision

Morten R. Hannemose, [mohan@dtu.dk](mailto:mohan@dtu.dk), DTU Compute

February 9, 2024

## Mathematical exercises: Pinhole camera

In these exercises we will assume a *modern* camera with completely square pixels. What are the skew parameters then?

### Exercise 2.1

Reuse the `box3d` function from last week. Assume that  $f = 600$ ,  $\alpha = 1$ ,  $\beta = 0$ , and  $\delta x = \delta y = 400$ . Given a traditional camera, what is the resolution in pixels?

Also assume  $\mathbf{R} = \mathbf{I}$ , and  $\mathbf{t} = [0, .2, 1.5]^T$ . Use `projectpoints` from last week, to project the points.

Are all the points are captured by the image sensor?

Where does the corner  $\mathbf{P}_1 = [-0.5, -0.5, -0.5]$  project to?

### Exercise 2.2

Create a new or change your function `projectpoints` to a version that also takes `distCoeffs` as an input. The list `distCoeffs` should contain the distortion coefficients  $[k_3, k_5, k_7, \dots]$ . Make the function work for at least 3 coefficients.

Test your function with the same setup as in [Exercise 2.1](#) and but assume that the distortion is

$$\Delta r(r) = -0.2r^2. \quad (2)$$

What are the distortion coefficients in this case?

Where does the corner  $\mathbf{P}_1$  project to?

Are all the points captured by the image sensor?

Plot the results and try changing the distortion coefficients. Do they behave as they should?

## Exercise 2.3

Download the following image:

[https://people.compute.dtu.dk/mohan/02504/gopro\\_robot.jpg](https://people.compute.dtu.dk/mohan/02504/gopro_robot.jpg)

The image has been captured using a GoPro. Assume that the focal length is 0.455732 times the image width, and a reasonable guess of principal point,  $\alpha$ , and  $\beta$ . The distortion coefficients are

$$k_3 = -0.245031, \quad k_5 = 0.071524, \quad k_7 = -0.00994978$$

What is  $K$ ?

## Exercise 2.4

Implement a function `undistortImage` that takes an image, a camera matrix, and distortion coefficients and returns an undistorted version of the same image. Use the following as an outline of your function

```
x, y = np.meshgrid(np.arange(im.shape[1]), np.arange(im.shape[0]))
p = np.stack((x, y, np.ones(x.shape))).reshape(3, -1)

q = ...
q_d = ...
p_d = ...

x_d = p_d[0].reshape(x.shape).astype(np.float32)
y_d = p_d[1].reshape(y.shape).astype(np.float32)
assert (p_d[2]==1).all(), 'You did a mistake somewhere'
im_undistorted = cv2.remap(im, x_d, y_d, cv2.INTER_LINEAR)
```

Test the function by undistorting the image from the previous exercise. Are the lines straight now?

*Tip:* Reshape `r` back to an image and show it to check that it looks like expected.

# Homographies

## Exercise 2.5

Consider the following points on a plane

$$\mathbf{p}_{2a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_{2b} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \quad \mathbf{p}_{2c} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{p}_{2d} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

Using the homography

$$\mathbf{H} = \begin{bmatrix} -2 & 0 & 1 \\ 1 & -2 & 0 \\ 0 & 0 & 3 \end{bmatrix},$$

map the points using the homography ( $\mathbf{H}\mathbf{q}_2$ ).

## Exercise 2.6

Create a function `hest` that takes two sets of points in 2D, `q1` and `q2`, and returns the estimated homography matrix using the linear algorithm. You can extract the singular vector corresponding to the smallest singular value with SVD. However, Numpy returns  $\mathbf{V}^T$  instead of  $\mathbf{V}$ , so you can do

```
U, S, VT = np.linalg.svd(B)
VT[-1, :]
```

Test your function by using the points from the exercise above. Do you get the exact same numbers in your homography, or are they scaled? Explain why this is fine.

## Exercise 2.7

Create a helper function `normalize2d`. This function finds the transformation  $\mathbf{T}$  such that  $\mathbf{q}_{ih} = \mathbf{T}\mathbf{p}_{ih}$ , has mean  $[0, 0]$  and standard deviation  $[1, 1]$  for all  $\mathbf{q}_i$ .

Test it out with some 2D points and make sure the mean and standard deviation are as expected.

## Exercise 2.8

Improve your `hest` function by adding an option (`normalize=true/false`) to normalize the points with `normalize2d` before estimating the homography. Apply the  $\mathbf{T}$  matrices to the estimated  $\mathbf{H}$  so the estimated homography still operates on non-normalized points.

## Exercise 2.9

Generate 100 random 2D points, and a random homography. Map the points using the homography, and use `hest` to estimate the homography from the points. You can use the following code as a starting point:

```

q2 = np.random.randn(2, 100)
q2h = np.vstack((q2, np.ones(1, 100)))
H_true = np.random.randn(3,3)
q1h = H_true@q2h
q1 = Pi(q1h)

```

## Exercise 2.10

Take a piece of paper and draw at least four  $\times$ -marks on it in random locations. Make sure to number them.

Put a small object on top of the paper, and use your phone to take pictures of your paper from two different viewpoints ( $A$  and  $B$ ) Get the  $x, y$  coordinates of your  $\times$ -marks in image coordinates. You can get the coordinates of clicked points with `plt.ginput`.

Use your annotated points to estimate the homography from image  $A$  to image  $B$ . Can you click on any point in one image and show where it is in the other image? What happens when you click on the object?

## Exercise 2.11

Re-create image  $A$ , using only pixel intensities from image  $B$ . Generate an overlay of the two images. Does the object on top align?

To accomplish this, use the following function:

```

def warpImage(im, H):
    imWarp = cv2.warpPerspective(im, H, (im.shape[1], im.shape[0]))
    return imWarp

```

It takes an image and a homography and returns the image warped with the homography.

# Solutions

## Answer of exercise 2.1

In a traditional camera, the principal point is exactly in the middle of the sensor. So, for this camera the sensor has  $2 \times 400 = 800$  pixels along each dimension i.e. a resolution of  $800 \times 800$  pixels.

The projection matrix reads

$$\mathcal{P} = \begin{bmatrix} 600 & 0 & 400 & 600 \\ 0 & 600 & 400 & 720 \\ 0 & 0 & 1 & 1.5 \end{bmatrix}, \quad (1)$$

Some points have an  $y$  value greater than 800, and are not visible in the image, as they are outside the image sensor.

$\mathbf{P}_1$  projects to  $[100, 220]^T$

## Answer of exercise 2.2

The first coefficient is negative, so this is barrel distortion.

The distortion coefficients are  $k_3 = -0.2$ ,  $k_5 = 0$ ,  $k_7 = 0, \dots$

$\mathbf{P}_1$  projects to  $[120.4, 232.24]^T$

The projection now reads:

$$\mathbf{q} = \mathbf{K} \Pi^{-1} \left[ \text{dist} \left[ \Pi \left( \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{Q} \right) \right] \right], \text{ where} \quad (3)$$

$$\text{dist}(\mathbf{p}) = \mathbf{p}(1 - 0.2 \|\mathbf{p}\|_2^2). \quad (4)$$

Notice in particular the transformations between inhomogeneous and homogeneous coordinates ( $\Pi$ ).

All the points are now projecting inside the image, and will thus be visible.

## Answer of exercise 2.3

The principal point is in the center of the image.  $\mathbf{K} = \begin{bmatrix} 875 & 0 & 960 \\ 0 & 875 & 540 \\ 0 & 0 & 1 \end{bmatrix}$ .

## Answer of exercise 2.5

We convert the  $\mathbf{p}_i$ s to homogeneous coordinates, and compute

$$\mathbf{q}_{ih} = \mathbf{H}\mathbf{p}_{ih}.$$

This gives us:

$$\mathbf{q}_1 = \begin{bmatrix} -\frac{1}{3} \\ \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}, \quad \mathbf{q}_2 = \begin{bmatrix} \frac{1}{3} \\ -2 \\ -2 \end{bmatrix}, \quad \mathbf{q}_3 = \begin{bmatrix} -1 \\ -\frac{4}{3} \\ \frac{4}{3} \end{bmatrix}, \quad \mathbf{q}_4 = \begin{bmatrix} -1 \\ -2 \\ -2 \end{bmatrix}$$

## Answer of exercise 2.6

You should obtain the same homography, but multiplied with a scalar such that  $\|\mathbf{H}\|_F = 1$ .