

Exercises in Basic Mathematical Tools for Imaging and Visualization

Exercise 4 (P) Application of the Pseudo-Inverse - Computation of the Projection Matrix

In a previous exercise we demonstrated that the following problem can be solved by the application of the *Pseudo-Inverse*:

$$\mathbf{Q} = \mathbf{P}\mathbf{M}$$

where \mathbf{Q} and \mathbf{M} are known and the matrix \mathbf{P} is unknown.

In this exercise we present an application for the usage of the Pseudo-Inverse in the following setting.

By \mathbf{Q} we denote a matrix containing N measured image points, with one point in each column. The image points are represented using homogeneous coordinates of a point $\mathbf{p} = [u, v, 1]^T$. The matrix \mathbf{M} contains one point of the 3D model in every column (also in homogeneous coordinates). \mathbf{M} has N columns. The matrix \mathbf{P} is the so-called *Projection Matrix* which takes the 3D points and projects them into the image, see also Figure 1.

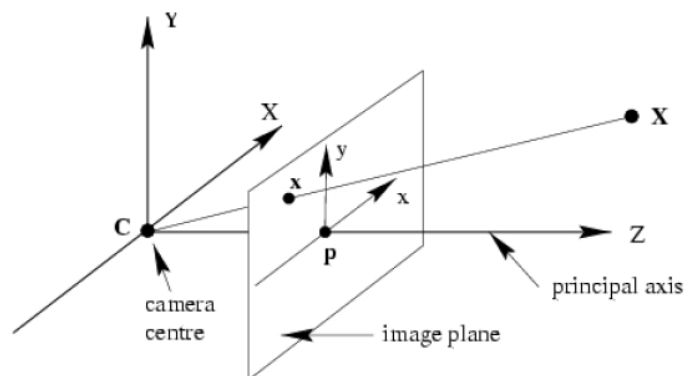


Figure 1: Projection of a 3D image \mathbf{X} to the 2D point \mathbf{x} in the image plane.

- Given the points in 3D and their projection in 2D. How can we compute the projection matrix \mathbf{P} ?
- Program the computation of the projection matrix in MATLAB by implementing the following steps
 - Generate a set of 3D points with homogeneous coordinates in a $(4 \times N)$ matrix.
 - Generate a simple (3×4) projection matrix \mathbf{P} , e.g. $\mathbf{P} = [\text{diag}([2 \ 2 \ 1]) \ \text{zeros}(3,1)]$.
 - Project the 3D points to 2D by using \mathbf{P} .
 - Reconstruct the projection matrix by using the MATLAB command `pinv()`.
 - Reconstruct the projection matrix by using the SVD for computing the Pseudo-Inverse as described in the last exercise.

- c) Repeat the above reconstruction of \mathbf{P} , but this time make the scenario more realistic by adding noise to the simulation.
- Add noise to the 3D points before projecting them (`randn()`). This simulates measurement errors of the 3D points.
 - Project the points.
 - Add noise to the projected 2D points. This simulates measurement errors of the 2D points.
 - Compute the projection matrix \mathbf{P} by using the noisy 2D-3D correspondences created above. Experiment with different levels of noise.

Exercise 5 (P) Image Warping

One of the most basic tasks in image processing is the so called *Image Warping*. Given an image I and a transformation \mathbf{H} , compute the transformed image I_H , see also Figures 3 and 2.

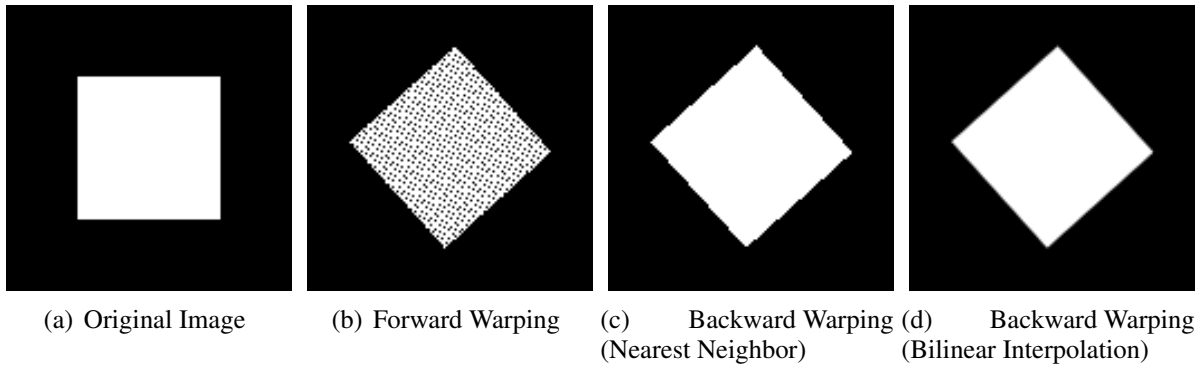


Figure 2: Image Warping. Examples for different warping methods for a simple transformation (rotation and translation).

We will assume that the transformation is given as (3×3) -matrix \mathbf{H} which operates on homogeneous points. So a point $\mathbf{p} = [u, v, 1]^T$ in the first image is transformed by $\mathbf{p}' = \mathbf{H}\mathbf{p}$. In order to transform the image I , we have to take all points \mathbf{p} of I , transfer them to the points $\mathbf{p}' = \mathbf{H}\mathbf{p}$ in the output image I_H , and then write the intensities at the new points, that is

$$I_H(\mathbf{p}') \leftarrow I(\mathbf{p}) .$$

The problem with image warping is that in general, the transformation will not map integer coordinates to integers but to real numbers ($\mathbf{p}' = [u', v', 1]^T$ where (u', v') are not always in $\mathbb{Z} \times \mathbb{Z}$). Since we are dealing with discrete images, this presents a problem, which has to be solved.

There are basically two approaches to this problem.

- *Forward Image Warping* computes for every point \mathbf{p} of the input image I the corresponding transformed point $\mathbf{p}' = \mathbf{H}\mathbf{p}$. The value of the point \mathbf{p}' is rounded and the respective intensity is stored at that position, that is $I_H(\text{round}(\mathbf{p}')) \leftarrow I(\mathbf{p})$. The drawback of this method is that in general not all points of the domain of the warped image are reached which leads to black pixels in the warped image.¹
- *Backward Image Warping* addresses the drawback of the forward warping. Instead of computing the destination for every point of the input image and this way missing some points of the resulting image, we tackle the problem the other way around. For every point \mathbf{p}' of the resulting image I_H the origin \mathbf{p} of this point is computed. This way, since we compute a value for every point in the warped image, no 'holes' in the resulting image can occur. In order to do this, we have to compute the inverse transformation \mathbf{H}^{-1} . The origin point to the point \mathbf{p}' of the resulting image is computed by $\mathbf{p} = \mathbf{H}^{-1}\mathbf{p}'$. The intensity value of the computed point \mathbf{x} is obtained an interpolation method of choice (nearest neighbor, bilinear, bicubic, ...).²

¹The Forward Image Warping is sometimes also referred to as using the *Lagrange Coordinate Frame*, where the motion of a point is described by giving the position of this point at the next frame. Can be thought of as: Where am I going?

²The Forward Image Warping is also referred to as using the *Euler Coordinate Frame*. Here the motion of a point is described by giving the position of this point at the previous frame. Can be thought of as: Where am I coming from?

Homework:

- a) Implement *Forward-Warping*.
- b) Implement *Backward-Warping* using the simple *Nearest-Neighbor* interpolation.
- c) Implement *Bilinear Interpolation*.
- d) Implement *Backward-Warping* using the simple *Bilinear Interpolation*.

Note: You will find material and the stubs for your implementation at the Lecture web page.



(a) Original Image



(b) Forward Warping



(c) Backward Warping (Nearest Neighbor)



(d) Backward Warping (Bilinear Interpolation)

Figure 3: Image Warping. Examples for different warping methods for a simple transformation (rotation and translation).