Image Interpolation, Rotation and Straightening

Report for Lab Exercise 1 for the Computer Vision Course

M. Pfundstein (10452397)
T.M. Meijers (10647023)
University of Amsterdam
The Netherlands

November 10, 2014

1 Introduction

Written exercises

This report contains all the written answers to the theory questions in ***Geometric Transformations on Images Interpolation, Affine Transformations, Imaging, Re-Projection and Pseudo-3D****: Lab Exercise 1 for the first 2 weeks of Beeldverwerken*

Programming exercises

For the programming exercises the choice was made to explain them fully by comments. This means that besides this subsection, no other part of this report will refer to the programming exercises. The code is commented well enough so that every step is clear, and the structure of the code is well organized. The main file (*main.m*) is organized into sections which correspond to the structure in the assignment. Every section refers to which helper functions were written to solve the exercise.

2 Interpolation

2.1

The definition of the function nearest-neighbour is: `nearest-neighbour = F(floor(x))`

2.2

The two equations we have to solve for a and b are: $a = F(k+1) - F(k)$
$b = (1+k)F(k) - kF(k+1)$, so:
This can be combied to:
$f(x) \approx f_1(x) = (1+k-x)F(k) + (x-k)F(k+1)$

2.3

When we write that in matrix form we get:
$$\begin{pmatrix} -1 & 1 \\ 1+k & -k \end{pmatrix} \begin{pmatrix} F(k) \\ F(k+1) \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}$$

2.4

$x \in [k, k+1], \;\; y \in [l, l+1]$
$a = x - k, \;\; b = y - l$

We know that: $f(x,l) = (1-a)F(k.l) + aF(k+1,l)$
$f(x,l+1) = (1-a)F(k,l+1) + aF(k+1,l+1)$

Hence we get by substituting: $f(x,y) = (1-b)f(x,l) + bf(x,l+1) =$
$(1-a)(1-b)F(k,l) +$
$(1-a)bF(k,l+1) +$
$abF(k+1,l+1) +$
$a(1-b)F(k+1,l)$

3 Rotation

3.1

The rotation matrix is: $\begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix}$

3.2

To obtain the new points x' and y', we have to multiply the input vector with the rotation matrix above. Hence we get:
$\begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$

3.3

To rotate about an arbitrary point c we can use the following rotation. First we translate the input vector by -c, thus ancoring it at the origin. Then we rotate by applying the rotation matrix. Finally we translate the result back by c.
$\begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} \begin{pmatrix} x - c_1 \\ y - c_2 \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$

3.4

Of course, nothing happens to the center of a rotation:
$\vec{c} - \vec{c} = \vec{0}$,
$R\,\vec{0} = \vec{0}$,
$\vec{0} + \vec{c} = \vec{c}$

4 Affine Transformation

4.1

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ f \end{pmatrix}$$

The above has no solution because the dimensions conflict with each other. The matrix is a 2x3 matrix while the vector is a 2x1 vector.

4.2

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ f \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ f' \\ 1 \end{pmatrix}$$

As can be seen, in the matrix equation above the dimensions agree. The matrix $\begin{pmatrix} a & b \\ d & e \\ 0 & 0 \end{pmatrix}$ handles the rotation, while the column vector $\begin{pmatrix} c \\ f \\ 1 \end{pmatrix}$ translates the vector.

4.3

$$\begin{pmatrix} cos(\phi) & -sin(\phi) & c \\ sin(\phi) & cos(\phi) & f \\ 0 & 0 & 1 \end{pmatrix}$$

The matrix above rotates (counter clockwise) by ϕ, and translates with vector $\begin{pmatrix} c \\ f \end{pmatrix}$

4.4

The corners of the to be transformed image:

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} ax_1 + by_1 + c & ax_2 + by_2 + c & ax_3 + by_3 + c \\ dx_1 + ey_1 + f & dx_2 + ey_2 + f & dx_3 + ey_3 + f \end{pmatrix}$$

4.5

Because we only need a position vector and two direction vectors to span the parralelogram that will be transformed. The fourth point is a linear combination of the other three.

5 Re-projecting Images

5.1

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{23} & m_{33} \end{pmatrix} \begin{pmatrix} c \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11}x + m_{12}y + m_{13} \\ m_{21}x + m_{22}y + m_{23} \\ m_{31}x + m_{23}y + m_{33} \end{pmatrix}$$

5.2

2 equations, third row is only a scalar:

$$\begin{pmatrix} m_{11}x + m_{12}y + m_{13} \\ m_{21}x + m_{22}y + m_{23} \\ m_{31}x + m_{23}y + m_{33} \end{pmatrix} = \begin{pmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{pmatrix}$$

5.3

Each point has two equations, hence we have eight degrees of freedom meaning eight parameters.

5.4

m_{33} is not really doing anything. We can normalize the matrix by dividing every component by m_{33}. Hence m_{33} will then be 1.

5.5

In 2 dimension we need 4 points .
(Extra:) In 3 dimensions we need 8 points (assuming that we transform a cube).

5.6

If we put all unknown parameters into a vector we get:

$$\vec{v} = \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{23} \\ m_{33} \end{pmatrix}$$

5.7

The full matrix equation is: (See 7.1. for a similiar derivation of the matrix)

$$\begin{pmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 & -x_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -y_1u_1 & -y_1v_1 & -y_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ u_8 & v_8 & 1 & 0 & 0 & 0 & -x_8u_8 & -x_8v_8 & -x_8 \\ 0 & 0 & 0 & u_8 & v_8 & 1 & -y_8u_8 & -y_8v_8 & -y_8 \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \\ m_{33} \end{pmatrix} = \vec{0}$$

5.8

By taking the kernel of A. The kernel projects $\vec{x}$ onto the null space.

5.9

See 5.3 and 5.4

5.10

Because if we have 4 points to transform, A will be a eight by nine matrix. Hence we can solve $A\,\vec{x} = \vec{0}$ by calculating the kernel of A.

6 Pinhole Camera Model

Due to time constraints the assignment corresponding to this section was not required nor done.

7 Estimating a Camera's Projection Matrix

7.1

The matrix equation for project an arbirary point (X,Y,Z) to the screen coordinates (x, y) is as follows: $\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

In order to get the values of the 12 unknown parameters m we can use the following steps:
First we expand the matrix equation to to:
$m_{11}X + m_{12}Y + m_{13}Z + m_{14} = \lambda x$
$m_{21}X + m_{22}Y + m_{23}Z + m_{24} = \lambda y$
$m_{31} + m_{32} + m_{33} + m_{34} = \lambda$

Substitute λ:
$m_{11}X + m_{12}Y + m_{13}Z + m_{14} = x(m_{31} + m_{23} + m_{33} + m_{34})$
$m_{21}X + m_{22}Y + m_{23}Z + m_{24} = y(m_{31} + m_{32} + m_{33} + m_{34})$

Set equations to zero:

$m_{11}X + m_{12}Y + m_{13}Z + m_{14} - m_{31} - m_{23} - m_{33} - m_{34} = 0$
$m_{21}X + m_{22}Y + m_{23}Z + m_{24} - m_{31} - m_{32} - m_{33} - m_{34} = 0$

If we do this for all points we get the following equation: $A\vec{x} = \vec{0}$ which expands to:

$$\begin{pmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -Y_1y_1 & -Z_1y_1 & -y_1 \\ . & . & . & . & . & . & . & . & . & . & . & \\ . & . & . & . & . & . & . & . & . & . & . & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -Z_nx_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_n & -Y_ny_n & -Z_ny_n & -y_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix} = \vec{0}$$

7.2

Yes it can, you always have as many rows as two times the number of points. The matrix dimensions will be $2p \times 12$ with p denoting the number of points. The more points the better the projection matrix will become.

7.3

No. The solution will be a 'best possible' solution. This is why we can't use the kernel method.

7.4

We need to minimize $A\vec{x} \approx \vec{0}$. Assume $|\vec{x}| = 1$. Hence $\vec{x}$ is a unit vector.
We can therefore minimize $|A\vec{x}|^2$ with $||\vec{x}||^2 = 1$.
This can be written as (1) $A\vec{x} \cdot A\vec{x}$ with $\vec{x} \cdot \vec{x} = 1$.
Using the singular value decomposition we get: $A = U\Sigma V^T$. We can thus write (1) as (2) $(U\Sigma V^T\vec{x}) \cdot (U\Sigma V^T\vec{x})$ with $\vec{x} \cdot \vec{x} = 1$.

Because U and V are both orthogornal matrices (2) simplifies to (3) $(\Sigma V^T \vec{x}) \cdot (\Sigma V^T \vec{x})$ with $\vec{x} \cdot \vec{x} = 1$.
To get rid of the V^T matrix, we assign a new variable $y = V^T \vec{x}$. Hence $\vec{x} = Vy$.
(3) becomes (4) $\Sigma y \cdot \Sigma y$ with $Vy \cdot Vy = 1$. Because V is orthogornal, this simplifies to
(5) $\Sigma y \cdot \Sigma y$ with $y \cdot y = 1$.
We know that Σ is a $2p \times 9$ matrix with p denoting the number of reference points. We can now minimize
$\Sigma y = \vec{0}$. Because Σ is the singular matrix that stores σ_1 to σ_9 in descending order ($\sigma_1 \geq \sigma_2 \geq .. \geq \sigma_9$), we need to chose y so that we only pick σ_9. Hence $y = \left[0,0,0,0,0,0,0,0,1\right]^T$. It follows that $\vec{x} = Vy = V\left[0,0,0,0,0,0,0,0,1\right]^T$.
This means that $\vec{x}$ is the last column of the matrix V.

8 Projecting Cubes into a Picture (Pseudo 3D)

This section does not have any theory questions