Beeldverwerken homework 1

Harm Manders(10677186) & Jonas van Oenen(10670947)
Group C

April 2016

These are our answers to the theory questions of the assignment. To test our matlab code simply run our main function e1_jonas_vanOenen_harm_manders. This should create a window with 6 different plots/images with above a title corresponding to a question. In this main function each question has a different section going from question 2 to 8.

2 Interpolation

2.1

The definition for nearest-neighbor interpolation is given below.

F(x) = floor(x+0.5)

2.2

The two equations needing to be solved are a and b.
$a = F(k+1) - F(k)$
$b = (k+1)F(k) - k(F(k+1))$

These two equations can be combined to create the equation below.
$f(x) = (k+1-x)F(k) + (x-k)F(k+1)$

2.3

This is the matrix equation form for equations a and b.

$$\begin{bmatrix} -1 & 1 \\ k+1 & -k \end{bmatrix} \begin{bmatrix} F(k) \\ F(k+1) \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

2.4

$f_1(x, y)$
$= (F(k, l+1) - F(k, l))y + (1+l)F(k, l) - (F(k, l+1))$

$= y(F(k,l+1)) - y(F(k,l)) + (1+l)(F(k,l) - (F(k,l+1))$
$= (y-l)F(k,l+1) + (1+l-y)F(k,l)$

$f_2(x,y)$
$= (F(k+1,l+1) - F(k+1,l))y + (1+l)F(k+1,l) - (F(k+1,l+1))$
$= (y-l)F(k+1,l+1) + (1+l-y)F(k+1,l)$

Neem y-l = β

$f_1(x,y) = \beta F(k,l+1) + (1-\beta)F(k,l)$
$f_2(x,y) = \beta F(k+1,l+1) + (1-\beta)F(k+1,l)$

$f_3(x,y)$
$= ((\beta F(k+1,l+1) + (1-\beta)F(k+1,l) - (\beta F(k,l+1) + (l-\beta)F(k,l))))x + (1+k)(\beta F(k,l+1) + (1-\beta)F(k,l) - k(\beta F(k+1,l+1) + (1-\beta)F(k+1,l)))$

$= x(\beta F(k+1,l+1) + (1-\beta)F(k+1,l) - x(\beta F(k,l+1) + (1-\beta)F(k,l) + (1+k)\beta F(k,l+1) + (1+k)(1-\beta)F(k,l) - k\beta F(k+1,l+1)k(1-\beta)F(k+1,l)))$

$= x\beta F(k+1,l+1) + x(1-\beta)F(k+1,l) - x\beta F(k,l+1) - x(1-\beta)F(k,l) + (1+k)\beta F(k,l+1) + (1+k)(l-\beta)F(k,l) - k\beta F(k+1,l+1) - (l-\beta)F(k+1,l)$

$= (x-k)\beta F(k+1,l+1) + (x-k)(1-\beta)F(k+1,l) + (1+k-x)\beta F(k,l+1) + (1+k-x)(1-\beta)F(k,l)$

Neem x-k = α

$f_3(x,y) = \alpha\beta F(k+1,l+1) + \alpha(1-\beta)F(k+1,l) + (1-\alpha)\beta F(k,l+1) + (1-\alpha)(1-\beta)F(k,l)$

3 Rotation

3.1

The matrix R given below is a rotation matrix in the counter clock wise direction.

R = $\begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3.2

To transform a point $(x,y)^t$ we will rotate it using the rotation matrix R.
$\begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

3.3

To perform a simple rotation around c, the vector c is first translated to the origin, then rotated, then translated back to c. The matrix equation is shown below.

$$\begin{bmatrix} cos\phi & -sin\phi & -cosC_1 + sinC_2 + C_1 \\ sin\phi & cos\phi & -sinC_1 - cosC_2 + C_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

3.4

When center c is rotate around c nothing happens, as demonstrated below.

$$\begin{bmatrix} cos\phi & -sin\phi & -cosC_1 + sinC_2 + C_1 \\ sin\phi & cos\phi & -sinC_1 - cosC_2 + C_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ 1 \end{bmatrix} = \begin{bmatrix} cosC_1 - sinC_2 - cosC_1 + sinC_2 + C_2 \\ sinC_1 + cosC_2 - sinC_1 - cosC_2 + C_2 \\ 1 \end{bmatrix}$$

This last matrix can be simplified to form the matrix below, showing that nothing happens.

$$\begin{bmatrix} C_1 \\ C_2 \\ 1 \end{bmatrix}$$

4 Affine Transformations

4.1

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

These matrices cannot be multiplied with each other, since the first is a 2x3 matrix and the second a 2x1.

4.2

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

This is how you can represent the matrices with help from homogeneous coordinates. The first $(a, b, d; e, 0, 0)^t$ part is the rotation, the $(c, f, 1)^t$ is the translation.

4.3

$$\begin{bmatrix} cos\phi & -sin\phi & c \\ sin\phi & cos\phi & f \\ 0 & 0 & 1 \end{bmatrix}$$

This is a rotation in the counter clockwise direction rotating ϕ degrees, and translation to coordinates $(c, f, 1)^t$

4.4

The corners of the transformed image can be used.

4.5

In the matrix the vectors a,d and b,e are for the rotation, the vector c,f is the position vector. The 4th point is simply a combination of the 3 other points.

5 Re-projecting images

5.1

$$\begin{bmatrix} m_{11} & m_{12} & m_{12} \\ m_{21} & m_{22} & m_{31} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}u & m_{12}v & m_{12} \\ m_{21}u & m_{22}v & m_{31} \\ m_{31}u & m_{32}v & m_{33} \end{bmatrix} = \begin{bmatrix} \lambda u' \\ \lambda v' \\ \lambda \end{bmatrix}$$

5.2

Every point correspondence has 2 equations.
$m_{31}u_1x_1 + m_{32}v_1x_1 + m_{33}x_1 = m_{11}u_1 + m_{12}v_1 + m_{13}$
$m_{31}u_1x_1 + m_{32}v_1x_1 + m_{33}x_1 = m_{21}u_1 + m_{22}v_1 + m_{23}$

5.3

Every point correspondence has 2 equations with 8 parameters which need to be filled in.

5.4

We can normalize the matrix by dividing $m_{11}...m_{32}$ by m_{33} which means m_{33} will be 1.

5.5

In 2 dimensions there are 4 point correspondences needed.

5.6

The unknown vector k is described below.

$$\vec{k} = \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix}$$

5.7

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

5.8

Take the kernel of the A, and thereby project $\vec{x}$ onto the null space.

5.9

As shown in previous answers only 8 parameters are needed for the equation and the 9th will be 1.

5.10

The command null(A) is very practical, since as described in 5.8, by taking the kernel of A $\vec{x}$ can be projected onto the null space.

7 Estimating a Camera's Projection Matrix

7.1

$$A = \begin{bmatrix} u_1 & v_1 & w_1 & 1 & 0 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 - x_1w_1 - x_1 \\ 0 & 0 & 0 & 0 & u_1 & v_1 & w_1 & 1 & -y_1u_1 & -y_1v_1 - y_1w_1 - y_1 \\ . & . & . & . & . & . & . & . & . & . \\ u_n & v_n & w_n & 1 & 0 & 0 & 0 & 0 & -x_nu_n & -x_nv_n - x_nw_n - x_n \\ 0 & 0 & 0 & 0 & u_n & v_n & w_n & 1 & -y_nu_n & -y_nv_n - y_nw_n - y_n \end{bmatrix}$$

7.2

Yes you can still represent all the equations. Matrix A will grow accordingly

7.3

No, there is no exact solution anymore, you now have to minimise the squared error