Beeldverwerken Lab 1

Toon Meijer (11016914), Marcel van de Lagemaat (10886699)

April 2019

1 Introduction

2 Interpolation

2.1

$F(x) = floor(x + 0.5)$

2.2

The two equations that we should solve are:

$$\begin{cases} F(k) = ak + b \\ F(k+1) = a(k+1) + b \end{cases}$$

2.3

From the system of equations of question 2.2 follows:

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

Filling in a in the other equation then gives us:

$(F(k+1) - F(k))k + b = F(k) \Leftrightarrow$
$b = F(k) + kF(k) - kF(k+1) \Leftrightarrow$
$b = (1+k)F(k) - kF(k+1)$

2.4

To find $f(k, y)$ the first to equations to solve are:

$$\begin{cases} F(k,l) = al + b \\ F(k,l+1) = a(l+1) + b = al + a + b \end{cases}$$

In matrix notation:

$$\left[\begin{array}{cc|c} a & b & F(k,l) \\ a & a+b & F(k,l+1) \end{array}\right] \rightarrow \left[\begin{array}{cc|c} a & b & F(k,l) \\ 0 & a & F(k,l+1) - F(k,l) \end{array}\right]$$

$a = F(k,l+1) - F(k,l) \rightarrow$
$F(k,l) = (F(k,l+1) - F(k,l))l + b \rightarrow$
$b = F(k,l)(l+1) - lF(k,l+1)$

From these a and b can be concluded:

$$\begin{aligned} f(k,y) &= (F(k,l+1) - F(k,l))y + F(k,l)(l+1) - lF(k,l+1) \\ &= yF(l,l+1) - yF(k,l) + F(k,l) - lF(k,l+1) \\ &= F(k,l)(1+l-y) + F(k,l+1)(y-l) \end{aligned}$$

substituting $y - l = \alpha$ gives:
$f(k,y) = (1-\alpha)F(k,l) + \alpha F(k,l+1)$

Using the same steps for $f(k+1,y)$ gives:

$f(k+1,y) = (1-\alpha)F(k+1,l) + \alpha F(k+1,l+1)$
Then, again using the same steps, for $f(x,y)$ solving the system of equations:

$$\begin{cases} F(k,y) = ak + b \\ F(k+1,y) = a(l+1) + b \end{cases}$$

Gives:
$f(x,y) = (1+k-x)F(k,y) + (x-k)F(k+1,y)$

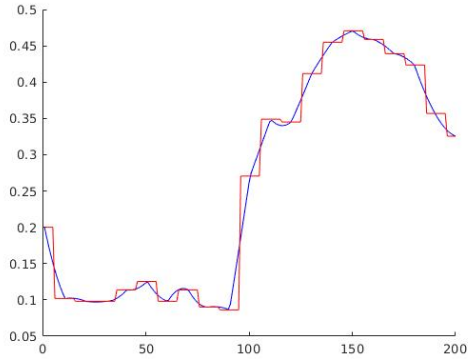
Substituting $k - x = \beta$:
$f(x,y) = (1+\beta)F(k,y) + \beta F(k+1,y)$

Then substituting the previously calculated $F(k,y)$ and $F(k+1,y)$:
$f(x,y) = (1-\beta)((1-\alpha)F(k,l) + \alpha F(k,l+1)) + \beta((1-\alpha)F(k+1,l) + \alpha F(k+1,l+1)) \rightarrow$

$$\begin{aligned} f(x,y) = &(1-\beta)(1-\alpha)F(k,l) + \\ &(1-\beta)\alpha F(k,l+1) + \\ &\beta(1-\alpha)F(k+1,l) + \\ &\alpha\beta F(k+1,l+1) \end{aligned}$$

2.1 Programming

Below is the resulting graph from plotting the profile of both 'linear' and 'nearest'.



3 Rotation

3.1

$R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$

3.2

Use the following matrix multiplication, and fill in the x and y values of the point.

$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

Replace the θ with the degree with which to rotate the point. For a counterclockwise rotation, keep the degree a positive value. For a clockwise rotation, make the degree negative.

3.3

Substract point c from the point to be rotated, thus making point c the temporary origin. Now use the standard rotation matrix to rotate the point. Then add c again to the rotated point.

3.4

Nothing happens, point c moves to the origin and then moves back to where it was before.
(1) c - c = $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

(2) $\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

(3) $\begin{bmatrix} 0 \\ 0 \end{bmatrix} + c = c$

4 Affine Transformations

4.1

If you try to write (1) as matrix equation you get:

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

The dimensions of the matrices don't match so this isn't a solvable equation. This means that affine transformation of 2D point is not possible in 2D.

4.2

Affine transformations in 2D can be done by using homogeneous coordinates. This means that the point in 2D becomes three-dimensional. This looks like this:

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

To interpret the solution of this equation you can divide the x and y values by the z value of the vector that comes out and then ignore the third dimension.

4.3

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) & c \\ sin(\phi) & cos(\phi) & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is now the general matrix for a rigid body motion. The result of multiplying a vector with this matrix is rotation of angle ϕ and translation of $(c, f)^T$

4.4

The most convenient points to choose for an affine translation are three corner points of the image. It doesn't really matter which three, the only important thing to keep in mind is to keep the order of the points the same when picking the points you want to map to. An example for these points then are (maatlab coordinates, starting at 1): (1, 1), (1, max(y)), (max(x), 1).

4.5

For affine transformations you have six unknown parameters and you need to solve six equations. This means you need three point correspondences for these parameters and each point will give you two equations to solve, so six in total. You don't need more than three points, because an affine transformation always transforms a parallelogram. To create a parallelogram you only need three points and the fourth will follow from those points.

5 Re-projecting images

5.1

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}x + m_{12}y + m_{13} \\ m_{21}x + m_{22}y + m_{23} \\ m_{31}x + m_{32}y + m_{33} \end{bmatrix} = \begin{bmatrix} \lambda x' \\ \lambda y' \\ \lambda \end{bmatrix}$$

5.2

Point correspondence for point $[u_1, v_1]^T$ can be written as these three equations:

$$m_{11}u_1 + m_{12}v_1 + m_{13} = \lambda_1 x_1$$
$$m_{21}u_1 + m_{22}v_1 + m_{23} = \lambda_1 y_1$$
$$m_{31}u_1 + m_{32}v_1 + m_{33} = \lambda_1$$

Then we eliminate the lambda, so we are left with two equations per point correspondence.

$$m_{11}u_1 + m_{12}v_1 + m_{13} = m_{31}u_1x_1 + m_{32}v_1x_1 + m_{33}x_1$$
$$m_{21}u_1 + m_{22}v_1 + m_{23} = m_{31}u_1y_1 + m_{32}v_1y_1 + m_{33}y_1$$

5.3

It should at least have eight parameters from four points.

5.4

This is because the general vector $[x_1, x_2, x_3]^T$ should be interpreted as $[x_1/x_3, x_2/x_3]^T$. This implies that we can normalize the matrix M for example by setting $m_{33} = 1$. That means there are really only 8 parameters in the matrix M that are relevant to our transformation.

5.5

Because M is of size 3x3 and we have one arbitrary element (m_{33}), we have 8 unknowns. We can solve this with 8 equations, so we need 4 point correspondences. In 3D, the matrix M is of size 3x4, so now we have 11 unknowns (again one arbitrary element m_{34}). Therefore we need 11 equations so we need at least 6 point correspondences to solve these equations.

5.6

The unknowns are the elements of the matrix M. If we collect them in a vector $\vec{m}$ we get:

$$\vec{m} = \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} \tag{1}$$

5.7

$$\begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 & -x_1 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 & -y_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 & -x_2 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 & -y_2 \\ . & . & . & . & . & . & . & . & . \\ u_n & v_n & 1 & 0 & 0 & 0 & -u_nx_n & -v_nx_n & -x_n \\ 0 & 0 & 0 & u_n & v_n & 1 & -u_ny_n & -v_ny_n & -y_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ . \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

5.8

We can avoid the trivial solution $\mathbf{x} = 0$ by normalizing the vector $\vec{m}$. If $\vec{m}$ is in the nullspace of the lefthand matrix above, we create a situation where the problem is exactly solvable.

5.9

Because a projective transformation uses a 3D matrix to do a 2D transformation. An actual 3D projective transformation uses 12-1=11 parameters.

5.10

The vector has to be in the null space of the matrix we describe in question 5.7. In question 5.8 we see that we can use it to solve the problem.

6 Pinhole Camera Model

7 Estimating a Camera's Projection Matrix

7.1

The point correspondence for a point $[u_1, v_1, w_1]^T$:

$m_{11}u_1 + m_{12}v_1 + m_{13}w_1 + m_{14} = \lambda_1 x_1$
$m_{21}u_1 + m_{22}v_1 + m_{23}w_1 + m_{24} = \lambda_1 y_1$
$m_{31}u_1 + m_{32}v_1 + m_{33}w_1 + m_{34} = \lambda_1$

Eliminating the lambda:

$$m_{11}u_1 + m_{12}v_1 + m_{13}w_1 + m_{14} = m_{31}u_1x_1 + m_{32}v_1x_1 + m_{33}w_1x_1 + m_{34}x_1$$
$$m_{21}u_1 + m_{22}v_1 + m_{23}w_1 + m_{24} = m_{31}u_1y_1 + m_{32}v_1y_1 + m_{33}w_1y_1 + m_{34}y_1$$

Putting these equations in a matrix:

$$A = \begin{bmatrix} u_1 & v_1 & w_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 & -w_1x_1 & -x_1 \\ 0 & 0 & 0 & 0 & u_1 & v_1 & w_1 & 1 & -u_1y_1 & -v_1y_1 & -w_1y_1 & -y_1 \\ u_2 & v_2 & w_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 & -w_2x_2 & -x_2 \\ 0 & 0 & 0 & 0 & u_2 & v_2 & w_2 & 1 & -u_2y_2 & -v_2y_2 & -w_2y_2 & -y_2 \\ . & . & . & . & . & . & . & . & . & . & . & . \\ u_n & v_n & w_n & 1 & 0 & 0 & 0 & 0 & -u_nx_n & -v_nx_n & -w_nx_n & -x_n \\ 0 & 0 & 0 & 0 & u_n & v_n & w_n & 1 & -u_ny_n & -v_ny_n & -w_ny_n & -y_n \end{bmatrix}$$

7.2

Yes, but you will have to make a least squares approximation because of the higher number of points. This would be beneficial for accuracy.

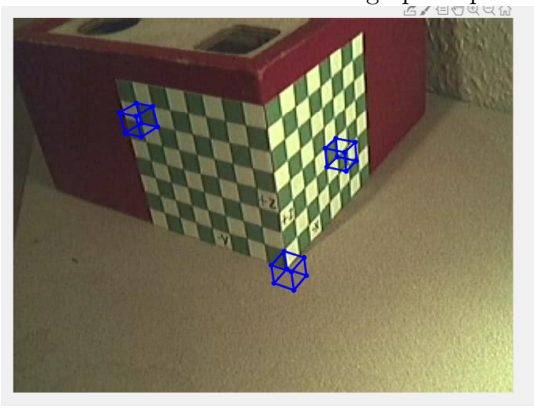
7.3

No there will not be an exact solution.

7.4

8 Projecting Cubes into a Picture

Below is the result from running "plotSquares1.m":

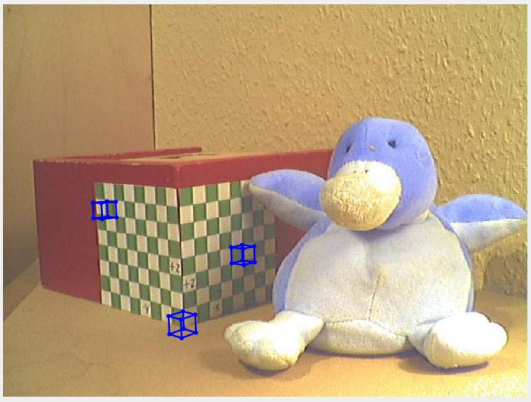


For part two of the assignment, we created function "enterCalibrationPoints.m", which takes as input an image and a number. The number tells the function how many points you want to mark. Then we input the resulting matrix in our "plotSquares1.m" function. The mininum amount of calibration points needed is 8. You can see from the pictures below that accuracy goes down with a lower number of points.

This is file "view1" with a calibration of 9 points.

This is file "view2" with a calibration of 8 points.

This is file "view3" with a calibration of 12 points.



This is file "view4" with a calibration of 8 points.