Module 4F12: Computer Vision

Solutions to Examples Paper 2

1. *Perspective projection and vanishing points*

   (a) Consider a line in 3D space, defined in camera-centered coordinates:

$$\begin{aligned} \mathbf{X}_c &= \mathbf{a} + \lambda \mathbf{b} \\ \Rightarrow \mathbf{x} &= f\left(\frac{a_x + \lambda b_x}{a_z + \lambda b_z}, \frac{a_y + \lambda b_y}{a_z + \lambda b_z}\right) \end{aligned}$$

   As $\lambda \to \infty$, we move further down the line, and $\mathbf{x}$ converges to the vanishing point:

$$\mathbf{x}_{vp} = f\left(\frac{b_x}{b_z}, \frac{b_y}{b_z}\right)$$

   The vanishing point depends only on the orientation of the line and not its position. When $b_z = 0$, the line is parallel to the image plane and the vanishing point is at infinity.

   (b) Renaissance artists generally restricted themselves to projections onto vertical planes (since the pictures tended to be hanging on walls), hence the vertical lines were parallel to the image plane and had no vanishing points. However, Renaissance artists had a good grasp of the theory of perspective, and would introduce vanishing points for vertical lines when painting scenes onto ceiling frescos, trying to give the illusion that the scene was actually "up there".

   (c) Parallel planes meet at infinity in the world. This projects to lines in the image, often referred to as horizon lines. To prove this, consider a plane in 3D space defined as follows:

$$\mathbf{X}_c.\mathbf{n} = d$$

   where $\mathbf{n} = (n_x, n_y, n_z)$ is the normal to the plane. We can analyse horizon lines by writing the perspective projection in the following form:

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{f\mathbf{X}_c}{Z_c}$$

   Taking the scalar product of both sides with $\mathbf{n}$ gives:

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} . \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{f\mathbf{X}_c.\mathbf{n}}{Z_c} = \frac{fd}{Z_c}$$

As $Z_c \to \infty$ we move away from the camera and we find

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} . \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = 0$$

Thus the equation of the horizon line is

$$n_x x + n_y y + f n_z = 0$$

which depends only on the orientation of the plane, and not its position. Thus parallel planes meet at a horizon line in the image.

2. *Rigid body transformations*

Inspection of the figure reveals that

$$X_c = -Y \quad , \quad Y_c = -Z + h \quad , \quad Z_c = X + 4h$$

The rigid body transformation between world and camera coordinates is therefore

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & h \\ 1 & 0 & 0 & 4h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3. *Calibration (3D)*

(a) The overall projection matrix must take account of rigid body motion between the camera and the scene, perspective projection onto the image plane and CCD imaging. This can be accomplished as follows.

Rigid body transformation

There is a rigid body transformation between the world coordinates $\tilde{\mathbf{X}}$ and the camera-centered coordinates $\tilde{\mathbf{X}}_{\mathrm{C}}$.

$$\tilde{\mathbf{X}}_{\mathrm{C}} = \mathrm{P}_r \ \tilde{\mathbf{X}} \ , \text{ where } \quad \mathrm{P}_r = \left[\begin{array}{ccc|c} & \mathrm{R} & & \mathbf{T} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$\tilde{\mathbf{X}}$ is the homogeneous representation of the world point $\mathbf{X}$, and likewise for $\tilde{\mathbf{X}}_{\mathrm{C}}$. P_r is the rigid body transformation matrix (rotation and translation).

Perspective projection

The next stage is perspective projection of $\tilde{\mathbf{X}}_{\mathrm{C}}$ onto $\tilde{\mathbf{x}}$ in the image plane:

$$\tilde{\mathbf{x}} = \mathrm{P}_p \ \tilde{\mathbf{X}}_{\mathrm{C}} \ , \text{ where } \quad \mathrm{P}_p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\tilde{\mathbf{x}} = (sx, sy, s)$ is the homogeneous representation of the image point $\mathbf{x} = (x, y)$. P_p is the perspective projection matrix.

CCD imaging

Finally, we have to convert to pixel coordinates $\mathbf{w} = (u, v)$:

$$\tilde{\mathbf{w}} = \mathrm{P}_c \, \tilde{\mathbf{x}} \text{ , where } \mathrm{P}_c = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\tilde{\mathbf{w}} = (su, sv, s)$ is the homogeneous representation of the pixel coordinates $\mathbf{w} = (u, v)$. P_c is the CCD calibration matrix.

We can now express the overall imaging process, from $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{w}}$, as a single matrix multiplication in homogeneous coordinates:

$$\begin{aligned} \tilde{\mathbf{w}} &= \mathrm{P}_{ps} \, \tilde{\mathbf{X}} \\ \text{where } \mathrm{P}_{ps} &= \mathrm{P}_c \, \mathrm{P}_p \, \mathrm{P}_r \\ &= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{ccc|c} & \mathrm{R} & & \mathbf{T} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned}$$

P_{ps} is the camera projection matrix for a perspective camera. It is a 3×4 matrix with 10 degrees of freedom. The product $\mathrm{P}_c \, \mathrm{P}_p$ accounts for all the *intrinsic* (or internal) camera parameters. P_r accounts for the *extrinsic* parameters. P_{ps} is *not* a general 3×4 matrix, but has a special structure composed of P_r, P_p and P_c.

(b) For calibration purposes, it is easier to consider another camera model, the **projective camera**, which is described by the *general* 3×4 matrix P:

$$\tilde{\mathbf{w}} = \mathrm{P} \, \tilde{\mathbf{X}}, \text{ where } \mathrm{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

The projective camera has 11 degrees of freedom (since the overall scale of P does not matter). It is often far more convenient to deal with a projective camera than a perspective one, since we do not have to worry about any nonlinear constraints on the elements of P.

The projective camera can be calibrated by observing the corners of the squares on the table. We have to know the positions of the corners in the image (we could extract these by hand or use a corner detector) and in the world (we would measure these with a ruler). Each point we observe gives us a pair of equations:

$$u = \frac{su}{s} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

$$v \;=\; \frac{sv}{s} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

Since we are observing a calibrated scene, we know X, Y, and Z, and we observe the pixel coordinates u and v in the image. So we have two linear equations in the unknown camera parameters. Since there are 11 unknowns (the overall scale of P does not matter), we need to observe at least 6 points to calibrate the camera.

The equations can be solved using orthogonal least squares. First, we write the equations in matrix form:

$$\mathrm{A}\ \mathbf{p} = \mathbf{0}$$

where $\mathbf{p}$ is the 12×1 vector of unknowns (the twelve elements of P), A is the $2n \times 12$ matrix of coefficients and n is the number of observed calibration points. The aim of orthogonal least squares is to minimize the squared residuals of these equations, *subject to a unit length constraint on* $\mathbf{p}$. Without this constraint, the squared residuals would be zero for $\mathbf{p} = \mathbf{0}$, which is not a very interesting solution. Since the overall scale of P does not matter, the unit length constraint is valid. So, we aim to minimize

$$\frac{\mathbf{p}^T\ \mathrm{A}^T\ \mathrm{A}\ \mathbf{p}}{\mathbf{p}^T\ \mathbf{p}}$$

From Part IA mathematics we know that

$$\lambda_1 \leq \frac{\mathbf{p}^T\ \mathrm{A}^T\ \mathrm{A}\ \mathbf{p}}{\mathbf{p}^T\ \mathbf{p}} \leq \lambda_{12}$$

where λ_1 is the smallest eigenvalue of $\mathrm{A}^T\ \mathrm{A}$ and λ_{12} is the largest eigenvalue of $\mathrm{A}^T\ \mathrm{A}$. Furthermore, we know that

$$\lambda_1 = \frac{\mathbf{p}^T\ \mathrm{A}^T\ \mathrm{A}\ \mathbf{p}}{\mathbf{p}^T\ \mathbf{p}}$$

when $\mathbf{p}$ is equal to $\mathbf{u}_1$, the eigenvector corresponding to λ_1. Thus, the orthogonal least squares solution corresponds to the eigenvector of $\mathrm{A}^T\ \mathrm{A}$ with the smallest corresponding eigenvalue.

It is essential to adjust the height of the table, since if we do not we are not exercising all the degrees of freedom of the camera model and the set of linear equations will not be independent. Consequently, the least squares procedure will not find a unique solution (there will be a degenerate zero eigenvalue).

Given the projective camera matrix, we can attempt to recover the intrinsic and extrinsic parameters using QR decomposition. Writing

$$\begin{aligned}
\mathrm{P}_{ps} &= \begin{bmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{cccc|c} & & & & \\ & & \mathrm{R} & & \mathbf{T} \\ & & & & \\ \hline 0 & 0 & 0 & & 1 \end{array}\right] = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \\ \mathrm{R} & \mathbf{T} \\ & \end{array}\right] \\
&= \mathrm{K} \left[\begin{array}{c|c} \mathrm{R} & \mathbf{T} \end{array}\right] = \left[\begin{array}{c|c} \mathrm{K\,R} & \mathrm{K}\ \mathbf{T} \end{array}\right]
\end{aligned}$$

it is apparent that we need to decompose the left 3×3 sub-matrix of P into an upper triangular matrix K and an orthogonal (rotation) matrix R. This can be achieved using QR decomposition: here's the appropriate sequence of Matlab commands:

```
[X Y] = qr( flipud(PL)' ); K = flipud( fliplr(Y') ); R = flipud(X');
```

where `PL` is the left 3×3 sub-matrix of P. Since there are multiple solutions, it is necessary to check that the appropriate elements of K are positive, change the signs of some columns of K if necessary, and compensate by changing the signs of the corresponding rows of R. $\mathbf{T}$ can then be recovered using

$$\mathbf{T} = \mathrm{K}^{-1} [\, p_{14} \;\; p_{24} \;\; p_{34} \,]^T$$

If the camera we're calibrating is high quality (so it does something approaching a perspective projection onto a well mounted CCD array) and the calibration has been properly performed, we should find that the recovered intrinsic matrix K has a zero in the middle of its top row, as expected. If we scale the matrix K so that it has a 1 in its lower right hand corner (this is acceptable, since the overall scale of P does not matter), then we can recover the principle point (u_0, v_0) by looking at k_{13} and k_{23}, and the products fk_u and fk_v by looking at k_{11} and k_{22}. It is not possible to decouple the focal length from the pixel scaling factors.

(c) Nonlinear distortion is not included in the projective camera model. Any residuals remaining after the orthogonal least squares procedure are partly due to nonlinear distortion, as well as errors in localising the corners in the image and in the world.

4. *Planar projective transformations*

The mapping from world to image plane is described by a planar projective transformation P^p:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

The transformation can be calibrated by observing four points whose world coordinates are known. Each point gives us two linear equations in the eight unknowns.

X=0, Y=0, x=0, y=0

$$0 = \frac{sx}{s} = \frac{p_{13}}{1} \;, \quad 0 = \frac{sv}{s} = \frac{p_{23}}{1} \;\Rightarrow\; p_{13} = p_{23} = 0$$

X=1, Y=0, x=1, y=–0.5

$$1 = \frac{sx}{s} = \frac{p_{11} + p_{13}}{p_{31} + 1} \;, \quad -0.5 = \frac{sy}{s} = \frac{p_{21} + p_{23}}{p_{31} + 1}$$

$$\Rightarrow p_{11} + p_{13} - p_{31} = 1 \;, \quad 2p_{21} + 2p_{23} + p_{31} = -1$$

X=0, Y=1, x=–0.5, y=1

$$-0.5 = \frac{sx}{s} = \frac{p_{12} + p_{13}}{p_{32} + 1} \ , \quad 1 = \frac{sy}{s} = \frac{p_{22} + p_{23}}{p_{32} + 1}$$
$$\Rightarrow 2p_{12} + 2p_{13} + p_{32} = -1 \ , \quad p_{22} + p_{23} - p_{32} = 1$$

X=1, Y=1, x= 1/3, y=1/3

$$1/3 = \frac{sx}{s} = \frac{p_{11} + p_{12} + p_{13}}{p_{31} + p_{32} + 1} \ , \quad 1/3 = \frac{sy}{s} = \frac{p_{21} + p_{22} + p_{23}}{p_{31} + p_{32} + 1}$$
$$\Rightarrow 3p_{11} + 3p_{12} + 3p_{13} - p_{31} - p_{32} = 1 \ , \quad 3p_{21} + 3p_{22} + 3p_{23} - p_{31} - p_{32} = 1$$

We now have a total of 8 equations in 8 unknowns, which we can write in matrix form:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 2 & 1 & 0 \\ 0 & 2 & 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 3 & 3 & 3 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 3 & 3 & 3 & -1 & -1 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

We solve this by matrix inversion (Matlab):

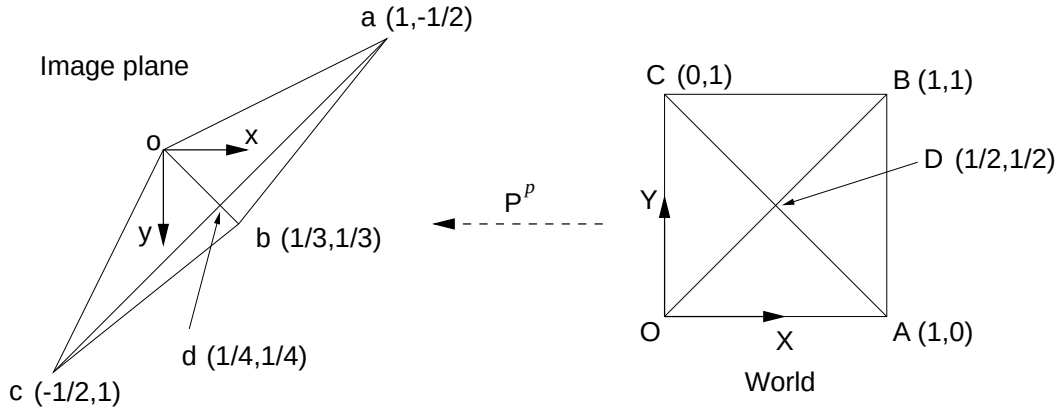
$$\mathrm{P}^p = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

The image of the point (0.5, 0.5) is

$$\mathrm{P}^p = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 2 \end{bmatrix}$$

in homogeneous coordinates, which is (0.25, 0.25) in Cartesian coordinates.

(b) We can check the answer by finding the intersection of the diagonals of the imaged square (since incidence is preserved under perspective projection).



5. *Projective transformation due to camera rotation*

(a) Before the camera is rotated, the camera is aligned with the world coordinate system and hence

$$\tilde{\mathbf{w}} = \mathrm{K}\left[\begin{array}{c|c} \mathrm{I} & \mathbf{O} \end{array}\right]\tilde{\mathbf{X}} = \mathrm{K}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathrm{K}\,\mathbf{X}$$

It follows that

$$\mathbf{X} = \mathrm{K}^{-1}\,\tilde{\mathbf{w}}$$

After rotating by R about the optical centre, the same world point $\mathbf{X}$ projects to a different image point $\tilde{\mathbf{w}}'$ as follows:

$$\tilde{\mathbf{w}}' = \mathrm{K}\left[\begin{array}{c|c} \mathrm{R} & \mathbf{O} \end{array}\right]\tilde{\mathbf{X}} = \mathrm{K}\,\mathrm{R}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathrm{K}\,\mathrm{R}\,\mathbf{X} = \mathrm{K}\,\mathrm{R}\,\mathrm{K}^{-1}\,\tilde{\mathbf{w}}$$

Hence the relationship between points in the original image and corresponding points in the second image is a plane to plane projectivity.

(b) The homography (2D projective transformation) can be estimated by observing at least four corresponding points in the two images. Each correspondence gives a constraint of the form

$$\begin{bmatrix} su' \\ sv' \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

By rearranging this matrix equation, it becomes clear how each correspondence provides two linear equations in the unknown elements of P:

$$u' = \frac{su'}{s} = \frac{p_{11}u + p_{12}v + p_{13}}{p_{31}u + p_{32}v + p_{33}}$$

$$v' = \frac{sv'}{s} = \frac{p_{21}u + p_{22}v + p_{23}}{p_{31}u + p_{32}v + p_{33}}$$

The set of constraints can be written in matrix form:

$$\mathrm{A}\,\mathbf{p} = \mathbf{0}$$

where $\mathbf{p}$ is the 9×1 vector of unknowns (the 9 elements of P), A is the $2n \times 9$ matrix of coefficients and n is the number of corresponding points observed in the two images. This can be solved using orthogonal least squares, as in question 3.

A *mosaic* can be constructed as follows. The camera is rotated around the optical centre and a sequence of images is acquired, with each image overlapping its

predecessor to some extent (say 50%). The plane to plane projectivity P relating consecutive pairs of images is estimated using correspondences in the overlap region. The correspondences can be located manually, or perhaps even automatically using some sort of correlation scheme. P is then used to *warp* one image into the coordinate frame of its predecessor, by finding the grey level $I(\tilde{\mathbf{w}})$ in the second image associated with each pixel $\tilde{\mathbf{w}}'$ in the frame of the first image. The two images can then be displayed in the same frame. Some sort of blending is required in the overlap region. This process is repeated for all pairs of images, allowing the entire sequence to be displayed in a single frame. If all has gone well (and the camera has not been translated as well as rotated), the seams should be invisible in the final composite mosaic.

6. *Line to line transformations*

   As discussed in the lecture notes, for the purposes of recovering structure along a line we really only need to calibrate for one component, say y:

$$\begin{bmatrix} sy \\ s \end{bmatrix} = \begin{bmatrix} p_{21} & p_{22} \\ p_{31} & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

   where X is the position of the train on the railway line. We can calibrate this using the y coordinates of the imaged markers.

   X=0, y=0

$$0 = \frac{sy}{s} = p_{22}$$

   X=20, y=0.5

$$0.5 = \frac{sy}{s} = \frac{20p_{21} + p_{22}}{20p_{31} + 1}$$

   X=30, y=3/5

$$\frac{3}{5} = \frac{sy}{s} = \frac{30p_{21} + p_{22}}{30p_{31} + 1}$$

   Solving these gives $p_{22} = 0,\ p_{21} = 1/20,\ p_{31} = 1/20$, so

$$\begin{bmatrix} sy \\ s \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

   With this calibration we can recover structure X given the y component of the image position:

$$y = \frac{X}{X + 20} \quad \Leftrightarrow \quad X = \frac{20y}{1 - y}$$

We can differentiate this expression to find the train's velocity $\dot{X}$ in terms of the y component of image velocity $\dot{y}$:

$$\dot{X} = \frac{(1-y) \times 20\dot{y} + 20y \times \dot{y}}{(1-y)^2} = \frac{20\dot{y}}{(1-y)^2}$$

The alarm should sound when $\dot{X} > 40$:

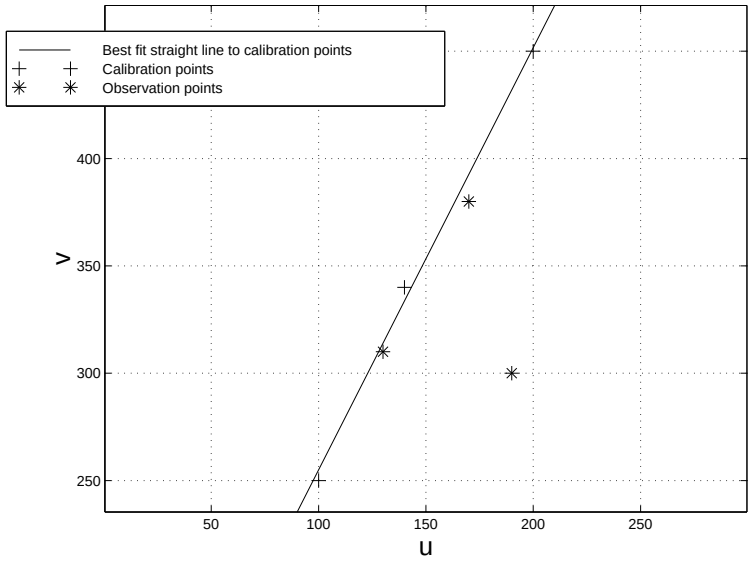
$$\frac{20\dot{y}}{(1-y)^2} > 40 \quad \Leftrightarrow \quad \dot{y} > 2(1-y)^2$$

So the limiting value of $\dot{y}$ should be $2(1-y)^2$.

7. *Calibration (1D)*

The transformation between the projected cross' position in 3D space and the pixel coordinates of its image is a 1D projective one. Since the 3D position of the projected cross is a linear function of the height of the table, it follows that the transformation between the height of the table and the pixel coordinates of the cross' image is also a projective one.

Having established this, we could proceed as in question 6 and calibrate the transformation between one of the pixel coordinates, u say, and the height of the table using the three calibration points given in the question. However, if we plot the three calibration points (marked '+' below) we see that they are not exactly collinear.



We clearly have some measurement noise to deal with. Also, one of the three points observed later (marked '$*$' above) is well off the line and is clearly an outlying measurement: it would be pointless to try to deduce the height of the table from this point. Since measurement errors are equally likely in the u and v directions, we

should use both u and v observations together with a least squares technique to find the optimal calibration matrix.

The transformation between table height X and cross position (u, v) can be written in the following form:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

The u and v coordinates of the calibration observations provide six equations:

X=50, u=100, v = 250

$$100 = \frac{su}{s} = \frac{50p_{11} + p_{12}}{50p_{31} + p_{32}}, \quad 250 = \frac{sv}{s} = \frac{50p_{21} + p_{22}}{50p_{31} + p_{32}}$$

X=100, u=140, v = 340

$$140 = \frac{su}{s} = \frac{100p_{11} + p_{12}}{100p_{31} + p_{32}}, \quad 340 = \frac{sv}{s} = \frac{100p_{21} + p_{22}}{100p_{31} + p_{32}}$$

X=200, u=200, v = 450

$$200 = \frac{su}{s} = \frac{200p_{11} + p_{12}}{200p_{31} + p_{32}}, \quad 450 = \frac{sv}{s} = \frac{200p_{21} + p_{22}}{200p_{31} + p_{32}}$$

Writing these equations in matrix form gives:

$$\begin{bmatrix} 50 & 1 & 0 & 0 & -5000 & -100 \\ 100 & 1 & 0 & 0 & -14000 & -140 \\ 200 & 1 & 0 & 0 & -40000 & -200 \\ 0 & 0 & 50 & 1 & -12500 & -250 \\ 0 & 0 & 100 & 1 & -34000 & -340 \\ 0 & 0 & 200 & 1 & -90000 & -450 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{21} \\ p_{22} \\ p_{31} \\ p_{32} \end{bmatrix} = \mathbf{0}$$

We solve this by orthogonal least squares (see question 3 for the method, and use Matlab to calculate the eigenvector via SVD):

$$\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix} = \begin{bmatrix} 1.6934 & 35.1093 \\ 3.5839 & 127.5444 \\ 0.0044 & 1.0 \end{bmatrix}$$

where the solution has been arbitrarily scaled to give a 1 in the lower right hand corner.

By way of illustration, let's look at an alternative way to estimate P. We could set p_{32} to 1 from the outset. Our system of linear equations becomes:

$$\begin{bmatrix} 50 & 1 & 0 & 0 & -5000 \\ 100 & 1 & 0 & 0 & -14000 \\ 200 & 1 & 0 & 0 & -40000 \\ 0 & 0 & 50 & 1 & -12500 \\ 0 & 0 & 100 & 1 & -34000 \\ 0 & 0 & 200 & 1 & -90000 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{21} \\ p_{22} \\ p_{31} \end{bmatrix} = \begin{bmatrix} 100 \\ 140 \\ 200 \\ 250 \\ 340 \\ 450 \end{bmatrix}$$

We solve this via the pseudo-inverse (use Matlab):

$$\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & 1 \end{bmatrix} = \begin{bmatrix} 1.7462 & 33.2618 \\ 3.7011 & 123.8197 \\ 0.0046 & 1.0 \end{bmatrix}$$

The solution is very similar to the one obtained with the eigenvector technique. In general, however, the eigenvector technique is to be preferred, since it produces a less biased solution[1]. For low-dimension problems there is not much to choose between the two approaches, and we'll stick with the pseudo-inverse for the rest of this question.

We can now use this calibration to find the height of the table X for new observations of the cross. Each of the u and v observations provides a single equation in X:

$$u = \frac{su}{s} = \frac{Xp_{11} + p_{12}}{Xp_{31} + 1}\ , \quad v = \frac{su}{s} = \frac{Xp_{21} + p_{22}}{Xp_{31} + 1}$$

We could solve using either the u or v coordinates, but since measurement errors are equally likely in u and v we should really use both coordinates together with a least squares procedure. Rearranging the above equations gives:

$$\begin{bmatrix} p_{11} - up_{31} \\ p_{21} - vp_{31} \end{bmatrix} X = \begin{bmatrix} u - p_{12} \\ v - p_{22} \end{bmatrix}$$

These equations can be solved for X using a pseudo-inverse (use Matlab) and the calibration parameters found earlier.

(a) $(u, v) = (130, 310) \Rightarrow X = 82.2$ mm.
(b) $(u, v) = (170, 380) \Rightarrow X = 133.1$ mm.
(c) $(u, v) = (190, 300)$. As mentioned earlier, this is clearly an outlying observation.

Graphical approach

Alternatively, we could try a graphical approach to avoid the lengthy least squares calculations (though this is not really an issue if we're using Matlab). We can plot

[1] In the context of fitting a straight line to a set of points in a plane, the pseudo-inverse technique minimizes the distances between the line and the points in only one of the principal directions, while the eigenvector technique minimizes the orthogonal distances between the line and the points.

the three calibration points on graph paper, and then fit a straight line (using orthogonal least squares) by eye — see the graph on page 9. Next we project each of the calibration points onto the line (using orthogonal projection, again by eye) to obtain what are effectively noise-free calibration points. We can now proceed as in question 6, calibrating for just one of the image coordinates, for example:

$$\begin{bmatrix} sv \\ s \end{bmatrix} = \begin{bmatrix} p_{21} & p_{22} \\ p_{31} & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Finally, we project each of the noisy observation points onto the straight line and use the calibration to solve for the table height X.

8. *Weak perspective*

(a) Under weak perspective projection, we assume that all points lie at approximately the same depth Z_A from the camera. This allows the projection to be re-written as follows:

$$\begin{bmatrix} su_A \\ sv_A \\ s \end{bmatrix} = \begin{bmatrix} k_u f & 0 & 0 & u_0 Z_A \\ 0 & k_v f & 0 & v_0 Z_A \\ 0 & 0 & 0 & Z_A \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

(b) Under full perspective we have

$$u = \frac{k_u f X_c + u_0 Z_c}{Z_c}$$

Under weak perspective we have

$$\begin{aligned} u_A &= \frac{k_u f X_c + u_0 Z_A}{Z_A} = \left(\frac{k_u f X_c + u_0 Z_A}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) \\ &= \left(\frac{k_u f X_c + u_0 Z_c + u_0(Z_A - Z_c)}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) = \left(u + \frac{u_0 \Delta Z}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) \end{aligned}$$

where $\Delta Z \equiv Z_A - Z_c$. So

$$\begin{aligned} u - u_A &= u - \left(\frac{uZ_c + u_0 \Delta Z}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) = \left(\frac{uZ_A}{Z_c} - \frac{uZ_c + u_0 \Delta Z}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) \\ &= \left(\frac{u(Z_A - Z_c) - u_0 \Delta Z}{Z_c}\right)\left(\frac{Z_c}{Z_A}\right) = (u - u_0)\frac{\Delta Z}{Z_A} \end{aligned}$$

Similarly for $(v - v_A)$, we find that

$$v - v_A = (v - v_0)\frac{\Delta Z}{Z_A}$$

So the weak perspective approximation is perfect at the centre of the image, but gets progressively worse away from the centre.

(c) Weak perspective is a good approximation when the depth range of objects in the scene is small compared with the viewing distance. A good rule of thumb is that the viewing distance should be at least ten times the depth range.

The main advantage of the weak perspective model is that it is easier to calibrate than the full perspective model. The calibration requires fewer points with known world position, and, since the model is linear, the calibration process is also better conditioned (less sensitive to noise) than the nonlinear full perspective calibration.

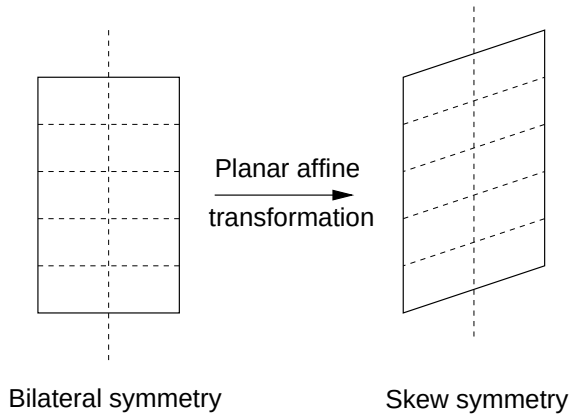
9. *Planar affine transformations*

(a) If the field of view is such that all points visible on the world plane lie at approximately the same depth from the camera (compared with the viewing distance), then the mapping from world plane (X, Y) to image plane (u, v) is approximately affine:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

This transformation has 6 degrees of freedom. Geometrically they signify rotation (1 DOF), scaling (1 DOF), shear (axis and magnitude, 2 DOF) and translation (2 DOF).

(b) The planar affine transformation preserves parallelism. Bilateral planar symmetry is transformed into a property called *skew symmetry*: the axis of symmetry is no longer orthogonal to the lines joining corresponding points.



The planar affine transformation preserves centroids. The centroid $(\bar{u}, \bar{v})$ of the transformed shape is at the same position as the transformed centroid $(\bar{X}, \bar{Y})$ of the original shape:

$$\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i = \frac{1}{n}\sum_{i=1}^{n} (p_{11}X_i + p_{12}Y_i + p_{13}) = p_{11}\frac{1}{n}\sum_{i=1}^{n} X_i + p_{12}\frac{1}{n}\sum_{i=1}^{n} Y_i + p_{13}$$

$$= \quad p_{11}\bar{X} + p_{12}\bar{Y} + p_{13}$$

Similarly

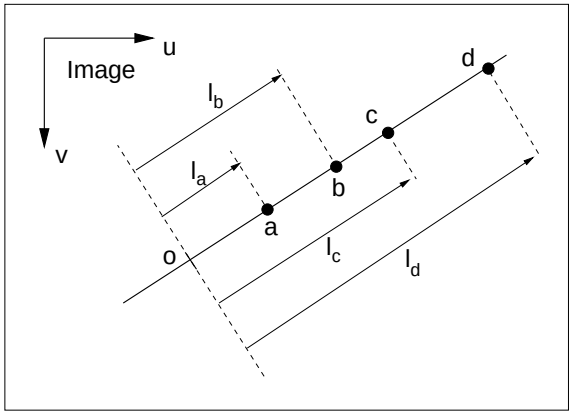
$$\bar{v} = p_{21}\bar{X} + p_{22}\bar{Y} + p_{23}$$

Hence

$$\begin{bmatrix} \bar{u} \\ \bar{v} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} \bar{X} \\ \bar{Y} \\ 1 \end{bmatrix}$$

(c) The planar projective transformation has two extra degrees of freedom. These specify "fanning": a square can transform to a quadrilateral with two vanishing points in the image. The equation of the horizon line contributes the two extra degrees of freedom.

10. *Projective invariants*



The figure shows the image of four world points A, B, C and D, and the world origin O. Distances l measured along the image line from o are related to distances along the world line by a 1D projective transformation:

$$\begin{bmatrix} sl \\ s \end{bmatrix} = \begin{bmatrix} p & q \\ r & 1 \end{bmatrix} \begin{bmatrix} X \\ 1 \end{bmatrix}$$

Hence we obtain

$$l_i = \frac{pX_i + q}{rX_i + 1}$$

Ratios of lengths in the image and the world can be expressed as follows:

$$ac = l_c - l_a \quad = \quad \frac{(X_c - X_a)(p - qr)}{(rX_c + 1)(rX_a + 1)}$$

$$bc = l_c - l_b = \frac{(X_c - X_b)(p - qr)}{(rX_c + 1)(rX_b + 1)}$$

$$\Rightarrow \frac{ac}{bc} = \frac{AC(rX_b + 1)}{BC(rX_a + 1)} \tag{1}$$

So the ratios of lengths are *not* invariant (compare with the affine case, where they are).

Similarly,

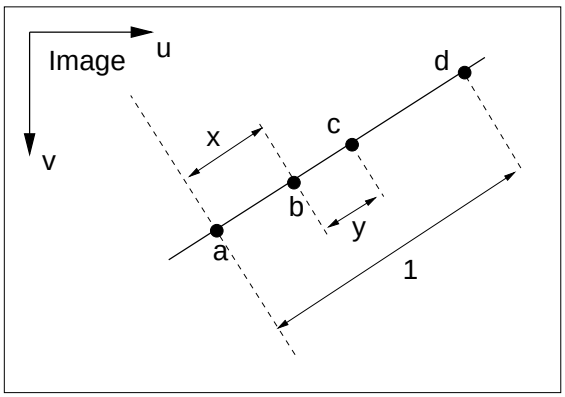
$$\frac{ad}{bd} = \frac{AD(rX_b + 1)}{BD(rX_a + 1)} \tag{2}$$

Dividing (1) by (2) we obtain

$$\frac{ac \times bd}{bc \times ad} = \frac{AC \times BD}{BC \times AD}$$

This is the **cross-ratio**, which is invariant to camera position, orientation and intrinsic parameters.

The four points can be permuted $4! = 24$ different ways. However, if we explore these permutations we find only six distinct values of the cross-ratio.



Using the lengths x and y defined in the figure, we can write the cross-ratio as follows:

$$\frac{ac \times bd}{bc \times ad} = \frac{(x + y)(1 - x)}{y} = \frac{x - x^2 - xy + y}{y}$$

We shall see common terms cropping up when we permute the points, so for convenience define $\alpha = x - x^2 - xy$ and $\beta = y$, so

$$\frac{ac \times bd}{bc \times ad} = \frac{\alpha + \beta}{\beta} = \tau \ , \quad \text{say}$$

The second distinct cross-ratio is found by permuting the points as follows;

$$\frac{ab \times cd}{ac \times bd} = \frac{x(1-x-y)}{(x+y)(1-x)} = \frac{x-x^2-xy}{x-x^2-xy+y} = \frac{\alpha}{\alpha+\beta} = \frac{\tau-1}{\tau}$$

The third distinct cross-ratio is found by permuting the points as follows;

$$\frac{ad \times bc}{ab \times cd} = \frac{y}{x(1-x-y)} = \frac{y}{x-x^2-xy} = \frac{\beta}{\alpha} = \frac{1}{\tau-1}$$

A further three distinct cross-ratios can be derived by taking the inverses of these expressions:

$$\frac{bc \times ad}{ac \times bd} = \frac{1}{\tau} \quad , \quad \frac{ac \times bd}{ab \times cd} = \frac{\tau}{\tau-1} \quad , \quad \frac{ab \times cd}{ad \times bc} = \tau - 1$$

All other permutations of the points yield one of these cross-ratios. In summary, then, there are six distinct cross-ratios, though they are not independent.

Roberto Cipolla
October 2020