1. Feature detection

(a) Let \( I(x, y) \) denote the image, and \( G_\sigma(x, y) \) denote the 2D Gaussian kernel:

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

The 2D convolution can be decomposed into two 1D convolutions as follows:

\[
G_\sigma(x, y) \ast I(x, y) = \frac{1}{2\pi\sigma^2} \int \int I(x-u, y-v) \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \, du \, dv
\]

\[
= \frac{1}{\sqrt{2\pi}\sigma} \int \exp\left(-\frac{u^2}{2\sigma^2}\right) \left[ \frac{1}{\sqrt{2\pi}\sigma} \int I(x-u, y-v) \exp\left(-\frac{v^2}{2\sigma^2}\right) \, dv \right] \, du
\]

\[
= \frac{1}{\sqrt{2\pi}\sigma} \int \exp\left(-\frac{u^2}{2\sigma^2}\right) \left[ g_\sigma(y) \ast I(x-u, y) \right] \, du
\]

\[
= g_\sigma(x) \ast \left[ g_\sigma(y) \ast I(x, y) \right]
\]

where \( g_\sigma(x) \) is a 1D Gaussian kernel:

\[
g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)
\]

The kernel size is usually chosen so that the truncated values are less than 0.1% of the central value. If \( \sigma = 2 \) and the kernel size is \( 2n + 1 \) pixels, then

\[
\exp\left[\frac{-(n+1)^2}{2(2)^2}\right] < 0.001 \iff n > 6 , \text{ so use } n = 7
\]

So we would require a 15 pixel kernel for the 1D convolutions, or a \( 15 \times 15 \) pixel kernel for the 2D convolutions. Each 2D convolution requires \( 15 \times 15 \) multiply and accumulate operations, while each 1D convolution requires 15 multiply and accumulate operations. The speedup offered by the 1D option is therefore \( 15/2 = 7.5 \) times.

(b) Consider a row of three pixels with intensities \( I_{k-1}, I_k, I_{k+1} \).
The first-order derivative at A is approximately \((I_k - I_{k-1})\) pixel\(^{-1}\).
The first-order derivative at C is approximately \((I_{k+1} - I_k)\) pixel\(^{-1}\).
Hence the second-order derivative at B is approximately

\[
(I_{k+1} - I_k) - (I_k - I_{k-1}) = (I_{k+1} - 2I_k + I_{k-1})\ \text{pixel}^{-2}
\]

This can be computed by convolution with the kernel \([1 \ -2 \ 1]\).

Intensity discontinuities occur at zero-crossings of the second-order derivatives (the intensities have already been smoothed). Linear interpolation can be used to locate the discontinuity to sub-pixel accuracy.

<table>
<thead>
<tr>
<th>Pixel no.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity</td>
<td>48</td>
<td>50</td>
<td>53</td>
<td>56</td>
<td>64</td>
<td>79</td>
<td>98</td>
<td>115</td>
<td>126</td>
<td>132</td>
<td>133</td>
</tr>
<tr>
<td>2nd-order deriv.</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>-2</td>
<td>-6</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
</tr>
</tbody>
</table>

The zero at pixel 3 is not a zero-crossing. So there is only one intensity discontinuity located 2/3 of the way between pixels 6 and 7.

2. **Perspective projection and camera models**

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

\[ X_c \cdot n = d \]

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

\[
\begin{bmatrix}
  x \\
  y \\
  f
\end{bmatrix}
= \frac{fX_c}{Z_c}
\]

Taking the scalar product of both sides with \(n\) gives:

\[
\begin{bmatrix}
  x \\
  y \\
  f
\end{bmatrix}
\cdot
\begin{bmatrix}
  n_x \\
  n_y \\
  n_z
\end{bmatrix}
= \frac{fX_c \cdot n}{Z_c} = \frac{fd}{Z_c}
\]

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

\[
\begin{bmatrix}
  x \\
  y \\
  f
\end{bmatrix}
\cdot
\begin{bmatrix}
  n_x \\
  n_y \\
  n_z
\end{bmatrix}
= 0
\]

Thus the equation of the horizon line is

\[ n_x x + n_y y + fn_z = 0 \]

which depends only on the orientation of the plane, and not its position. Thus a set of parallel planes meet at a horizon line in the image.
(b) The rigid body transformation between world and camera-centered coordinates (in mm) is
\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix} = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & -100 \\
1 & 0 & 0 & 300 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_w \\
Y_w \\
Z_w \\
1
\end{bmatrix}
\]

Perspective projection onto the image plane (focal length 20mm) can be written as
\[
\begin{bmatrix}
sx \\
sy \\
s
\end{bmatrix} = \begin{bmatrix}
20 & 0 & 0 & 0 \\
0 & 20 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix}
\]

Pixels \((u, v)\) are found by sampling the image plane using the CCD array described in the question (50 pixels per mm). If the pixel origin is at the top left hand corner of the CCD array (viewed from the optical centre) and the \(u\)-axis runs left to right while the \(v\)-axis runs top to bottom (as usual) then:
\[
u = -50x + 200, \quad v = -50y + 200
\]

In matrix form we have
\[
\begin{bmatrix}
su \\
sv \\
s
\end{bmatrix} = \begin{bmatrix}
-50 & 0 & 200 \\
0 & -50 & 200 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
sx \\
sy \\
s
\end{bmatrix}
\]

Concatenating the matrices gives
\[
\begin{bmatrix}
su \\
sv \\
s
\end{bmatrix} = \begin{bmatrix}
200 & -1000 & 0 & 60000 \\
200 & 0 & -1000 & 160000 \\
1 & 0 & 0 & 300
\end{bmatrix}
\begin{bmatrix}
X_w \\
Y_w \\
Z_w \\
1
\end{bmatrix}
\]

To check the answer, substitute \((u, v) = (200, 200)\): we should obtain the ray corresponding to the optical axis, which has equation \(Z_w = 100, Y_w = 0\) (by inspection).
\[
\frac{su}{s} = 200 = \frac{200X_w - 1000Y_w + 60000}{X_w + 300} \quad \Leftrightarrow \quad Y_w = 0
\]
\[
\frac{sv}{s} = 200 = \frac{200X_w - 1000Z_w + 160000}{X_w + 300} \quad \Leftrightarrow \quad Z_w = 100
\]

This is as we expected.

3. Transformations and projection

(a) (i) **Euclidean.** 3 degrees of freedom: 2 translation and 1 rotation. Invariants include length and area plus all below.

(ii) **Similarity.** 4 degrees of freedom: as above plus 1 scaling. Invariants include length ratios, angles, bilateral symmetry plus all below.
(iii) **Affine.** 6 degrees of freedom: as above plus 2 shear. Invariants include parallelism, area ratios, skew symmetry, length ratios along parallel lines plus all below.

(iv) **Projective.** 8 degrees of freedom: as above plus 2 from the equation of the horizon line. Invariants include concurrency, collinearity, intersection, tangency, inflections and cross-ratios.

(b) The image of Q can be found by intersecting the diagonals in the image.

Then the sine rule tells us that

$$\frac{\sin \theta}{\sqrt{2}} = \frac{\sin 45^\circ}{\sqrt{5}} \iff \theta = 26.57^\circ$$

The y-coordinates of p and q are $1.5 \times \tan \theta = 0.75$, and the x-coordinate of p is the same as its y-coordinate. So p is (0.75, 0.75), q is (1.5, 0.75) and r is (1.5, 0).

4. **Stereo vision**

(a) Any four from: epipolar constraint, ordering, uniqueness, disparity gradient, figural continuity.

(b) (i)
(ii) To derive the epipolar constraint, let \( \mathbf{X} = [X \ Y \ Z]^T \) and \( \mathbf{X}' = [X' \ Y' \ Z']^T \). The two coordinate systems are related by an isometry:

\[
\mathbf{X}' = R \mathbf{X} + \mathbf{T}
\]

where \( R \) and \( \mathbf{T} \) are given in the question. Taking the vector product with \( \mathbf{T} \) and then the scalar product with \( \mathbf{X}' \), we obtain

\[
\mathbf{X}' \cdot (\mathbf{T} \times R \mathbf{X}) = 0 \ \Leftrightarrow \ \mathbf{X}'^T (\mathbf{T}_x \times R \mathbf{X}) = 0
\]

where

\[
\mathbf{T}_x \equiv \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}
\]

Finally, defining the essential matrix \( \mathbf{E} \equiv \mathbf{T}_x \times R \) and multiplying \( \mathbf{X} \) by \( f/Z \) and \( \mathbf{X}' \) by \( f/Z' \) gives

\[
\begin{bmatrix} x' & y' & f \end{bmatrix} \mathbf{E} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = 0
\]

In this example \( f = 15 \text{mm} \). The essential matrix can be enumerated and the epipolar constraint is (image coordinates in mm):

\[
\begin{bmatrix} x' & y' & 15 \end{bmatrix} \begin{bmatrix} 0 & -100 & 0 \\ -100 & 0 & -100\sqrt{3} \\ 0 & 100\sqrt{3} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 15 \end{bmatrix} = 0
\]

The epipole \( \mathbf{p}_e \) in the left image lies in the nullspace of \( \mathbf{E} \):

\[
\begin{bmatrix} 0 & -100 & 0 \\ -100 & 0 & -100\sqrt{3} \\ 0 & 100\sqrt{3} & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ 15 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow y_e = 0, \ x_e = -15\sqrt{3}
\]

The epipole and epipolar lines look like this:

Left image
(iii) **Disadvantages.** For parallel image planes (with \( T \) along the cameras’ x-axes) the epipolar lines will be raster lines, making the search for stereo correspondences much easier. Matching by cross-correlation will be more accurate since there will be less distortion between the images.

**Advantages.** A greater angle between the image planes allows better triangulation and hence structure estimates which are less sensitive to noise. The two cameras will also have a larger field of view in common.

5. **Visual motion**

(a) If \( \mathbf{X} = [X \, Y \, Z]^T \) is the position of a world point in the camera-centered coordinate system, then \( \mathbf{p} = f\mathbf{X}/Z \) (perspective projection). Differentiating with respect to time, we obtain

\[
\dot{\mathbf{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} = \frac{f\dot{X}}{Z} - \frac{fX\dot{Z}}{Z^2}
\]

For a camera moving with linear velocity \( \mathbf{U} \) and angular velocity \( \mathbf{\Omega} \), we know that \( \dot{\mathbf{X}} = -\mathbf{U} - \mathbf{\Omega} \times \mathbf{X} \) (rigid body motion). Substituting into the equation for \( \dot{\mathbf{p}} \), and also writing \( \mathbf{p} = f\mathbf{X}/Z \), we obtain

\[
\dot{\mathbf{p}} = \frac{f}{Z} (-\mathbf{U} - \mathbf{\Omega} \times \mathbf{X}) - \frac{\mathbf{p} \dot{Z}}{Z}
\]

Again using \( \mathbf{p} = f\mathbf{X}/Z \), and noting that \( \dot{Z} \) is the \( \mathbf{k} \) component of \( \dot{\mathbf{X}} \), we obtain

\[
\dot{\mathbf{p}} = -\frac{f\mathbf{U}}{Z} - \mathbf{\Omega} \times \mathbf{p} - \frac{\mathbf{p} \dot{Z}}{Z} (-\mathbf{U} - \mathbf{\Omega} \times \mathbf{X}) \cdot \mathbf{k}
\]

Once again using \( \mathbf{p} = f\mathbf{X}/Z \), we obtain

\[
\dot{\mathbf{p}} = -\frac{f\mathbf{U}}{Z} - \mathbf{\Omega} \times \mathbf{p} + \frac{\mathbf{(U,k)p}}{Z} + \frac{\mathbf{p} \Omega}{Z} \left( \mathbf{\Omega} \times \frac{Z\mathbf{p}}{f} \right) \cdot \mathbf{k}
\]

Cancelling the \( Z \)’s, we obtain the result

\[
\dot{\mathbf{p}} = -\frac{f\mathbf{U}}{Z} + \frac{\mathbf{(U,k)p}}{Z} - \mathbf{\Omega} \times \mathbf{p} + \frac{\mathbf{[\Omega, p, k]p}}{f}
\]

(b) (i) From the vector equation for \( \dot{\mathbf{p}} \), we can deduce expressions for \( \dot{x} \) and \( \dot{y} \):

\[
\dot{x} = -\frac{fU_1 + xU_3}{Z} - f\Omega_2 + y\Omega_3 + \frac{xy}{f}\Omega_1 - \frac{x^2}{f}\Omega_2
\]

\[
\dot{y} = -\frac{fU_2 + yU_3}{Z} + f\Omega_1 - x\Omega_3 - \frac{xy}{f}\Omega_2 + \frac{y^2}{f}\Omega_1
\]

The rotational components of image velocity are

\[
\dot{x}_r = -f\Omega_2 + y\Omega_3 + \frac{xy}{f}\Omega_1 - \frac{x^2}{f}\Omega_2
\]

\[
\dot{y}_r = +f\Omega_1 - x\Omega_3 - \frac{xy}{f}\Omega_2 + \frac{y^2}{f}\Omega_1
\]
Substituting $\Omega = (0, -1/f, 0)$ and the coordinates of A, B, C, and D gives:

\[
\begin{bmatrix}
\dot{x}_r \\
\dot{y}_r
\end{bmatrix}_A = \begin{bmatrix} 2 \\
0 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_r \\
\dot{y}_r
\end{bmatrix}_B = \begin{bmatrix} 1 \\
0 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_r \\
\dot{y}_r
\end{bmatrix}_C = \begin{bmatrix} 2 \\
0 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_r \\
\dot{y}_r
\end{bmatrix}_D = \begin{bmatrix} 1 \\
0 \end{bmatrix}
\]

(ii) Subtracting the rotational components from the full image velocities, we obtain the translational components:

\[
\begin{bmatrix}
\dot{x}_t \\
\dot{y}_t
\end{bmatrix}_A = \begin{bmatrix} -1 \\
1 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_t \\
\dot{y}_t
\end{bmatrix}_B = \begin{bmatrix} 0 \\
0 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_t \\
\dot{y}_t
\end{bmatrix}_C = \begin{bmatrix} 1 \\
1 \end{bmatrix}, \\
\begin{bmatrix}
\dot{x}_t \\
\dot{y}_t
\end{bmatrix}_D = \begin{bmatrix} 0 \\
0 \end{bmatrix}
\]

These intersect at the focus of expansion, which happens to be at the point B.

The image coordinates of the focus of expansion are $(fU_1/U_3, fU_2/U_3)$ (easily proved from the image velocity equations), so

\[
\frac{fU_1}{U_3} = 0, \quad \frac{fU_2}{U_3} = -f \Rightarrow U_1 = 0, \quad U_2 = -U_3
\]

So the aircraft’s heading is $(0, -1, 1)$ (not $(0, 1, -1)$, since the translational image flow is away from the focus of expansion so the aircraft is moving forwards along the optical axis).

(iii) Point D lies on the horizon, since it is not the focus of expansion but has zero translational image velocity.

6. Applications

(a) Book work: see the section on measuring image velocity in Handout 5 (optical flow and feature tracking).

Application: divide image into blocks and measure $(\dot{u}, \dot{v})$ for each block. If the motion is consistent with a a pan/tilt of the camera (constant image velocity throughout) then cancel this effect by resampling the CCD with a constant offset (the CCD will have to be bigger than the desired image size).
(b) Book work: see the section on time to contact in Handout 5. The time to contact can be used to guide the manipulator to approach the fruit carefully without damaging it.

(c) Book work: see the section on planar object recognition in Handout 3. The models can be acquired from sample images by extracting edges, identifying distinguished points (eg. bitangents to concavities and points of inflection) and mapping the part’s outline into a canonical frame: the canonical frame signature is the model for that part. Invariance to lighting conditions is achieved through edge extraction, viewpoint invariance comes from the use of canonical frames.

(d) A single view of a pointing hand (or arm) is ambiguous: the ‘piercing point’, where the line defined by the hand intersects the screen, cannot be uniquely determined but is constrained to a line, which is the projection of the hand’s line in the image (see (a) below).

With a second camera we obtain a similar constraint in the other image (c). There exists a planar projective transformation (8 degrees of freedom) that maps one image of the screen onto the other. This transformation can be calibrated by observing the four corners of the screen. We exploit this to transform the constraint lines into a common ‘canonical’ view of the screen, and hence find their intersection (b).

The piercing point can then be projected back into the two images (small circles in (a) and (b)); if the four reference points are known its world coordinates can also be calculated.

The user’s arm can be tracted using B-spline snakes (book work, see Handout 2).

Andrew Gee
March 1997
1. Feature detection

(a) [Book work] Both edges and corners are useful for describing image structure. Compared with raw images, they offer significant data reduction while preserving much of the image’s useful information content. Edges provide the more complete description: it is possible to recognise many structures in a line drawing of a scene. The disadvantage of edges becomes apparent when attempting to use them for motion analysis. The aperture problem renders the motion of an edge ambiguous: we cannot recover the component of image velocity along the edge. Corners do not suffer from this problem, and are often the feature of choice for stereo and motion applications. [2]

(b) [Book work] The rate of change of $I$ in the direction $\hat{n}$ is found by taking the scalar product of $\nabla I$ and $\hat{n}$:

$$I_n \equiv \nabla I(x, y) \cdot \hat{n} \Rightarrow I_n^2 = \frac{n^T \nabla I \nabla I^T n}{n^T n} = \frac{n^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} n}{n^T n}$$

where $I_x \equiv \partial I/\partial x$, etc. Next we smooth $I_n^2$ by convolution with a Gaussian kernel:

$$C_n(x, y) = G_\sigma(x, y) * I_n^2 = \frac{n^T \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} n}{n^T n}$$

where $\langle \rangle$ is the smoothed value. The smoothed change in intensity in direction $\hat{n}$ is therefore given by

$$C_n(x, y) = \frac{n^T A n}{n^T n}$$

where $A$ is the $2 \times 2$ matrix

$$A = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

Elementary eigenvector theory tells us that

$$\lambda_1 \leq C_n(x, y) \leq \lambda_2$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $A$. So, if we try every possible orientation $\hat{n}$, the maximum change in intensity we will find is $\lambda_2$, and the minimum value is $\lambda_1$.

We can classify image structure at each pixel by looking at the eigenvectors of $A$:

**No structure:** (smooth variation) $\lambda_1 \approx \lambda_2 \approx 0$
1D structure: (edge) $\lambda_1 \approx 0$ (direction of edge), $\lambda_2$ large (normal to edge)

2D structure: (corner) $\lambda_1$ and $\lambda_2$ both large and distinct

It is necessary to calculate $A$ at every pixel and mark corners where the quantity $\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$ exceeds some threshold ($\kappa \approx 0.04$ makes the detector a little “edge-phobic”). Note that $\det A = \lambda_1\lambda_2$ and $\text{trace } A = \lambda_1 + \lambda_2$.

An approximation to the first order spatial derivative $I_x$ mid-way between the pixel $(x, y)$ and the pixel $(x + 1, y)$ is $I(x + 1, y) - I(x, y)$. This can be computed by an horizontal 1D convolution with the kernel $[1 -1]$. An approximation to $I_y$ can be obtained by a similar vertical convolution. [5]

(c) If we looked at the eigenvalues of

\[
\begin{bmatrix}
I_x^2 & I_xI_y \\
I_xI_y & I_y^2
\end{bmatrix}
\]

we would find that one of them is always zero, and we would therefore never classify a point as a corner! This is because, if we assume $I(x, y)$ to be differentiable everywhere, there will always be a direction where $I_n^2 = 0$ (perpendicular to $\nabla I$).

This problem goes away if we smooth the derivatives: around a corner the direction of $\nabla I$ will not be stable, so we will not be able to find a direction in which the smoothed value of $I_n^2$ is zero.

If the image is known to be noisy, it might also be necessary to smooth the intensity array $I(x, y)$ before applying the corner detector. [3]

2. Perspective projection and the cross-ratio

(a) [Book work] Intrinsic camera parameters describe properties of the camera’s lens and CCD array, typically the focal length, the size and shape of the pixels and the point where the optical axis pierces the CCD array. Extrinsic camera parameters describe the position of the camera with respect to the scene: there are six degrees of freedom, three translational and three rotational. [2]

(b) The cross-ratio of the collinear points A, B, C and D will be the same measured in the image and on the roadside. In the image:

\[
\frac{AD \times BC}{BD \times AC} = \frac{6 \times 2}{3 \times 5} = 0.8
\]

On the roadside:

\[
\frac{AD \times BC}{BD \times AC} = \frac{BC}{AC} = \frac{BC}{400} = 0.8 \iff BC = 320m
\]

The roadside distance from sign A to the phone box is therefore $400 - 320 = 80m$. [4]

(c) Given the intrinsic camera parameters, we know the directions of the rays to the points A, B, C and D and also the direction of the optical axis (all in the camera-centered coordinate system). These directions are sketched overleaf.
The camera is looking straight ahead, so its optical axis is in the direction of motion. The ray OD is parallel to the side of the road, so the heading of the car can be found from the angle $\theta$ between the optical axis and OD. The line through A, B and C is the roadside: it is parallel to OD and can be positioned by moving it away from O until the distance AC is 400m. Then the distance from the car to the phone box is the length OB. [4]

3. Planar projective and affine transformations

(a) [Book work] (i) Since the transformation operates on homogeneous coordinates, the overall scale of the transformation matrix does not matter and we could, for instance, set $p_{33}$ to 1. The transformation therefore has 8 degrees of freedom.

The image of a square could take any of the following forms:

- Translation (2 DOF)
- Rotation
- Scaling
- Shear
- Stretch
- Fanning - equation of horizon line gives 2 DOF

(ii) The camera can be calibrated using four points. There are eight unknowns ($p_{11} \ldots p_{32}$) and each observed point provides two linear equations in the unknowns.
If more points were available, the calibration could be improved using linear least squares.

(iii) The invariants of the plane projective transformation include concurrency and collinearity, order of contact, tangent discontinuities and cusps, the cross-ratio of four collinear points, and measurements in a canonical frame. [4]

(b) [Book work] The planar affine model is appropriate when the distance from the camera to the scene is large compared with the depths of objects in the scene (say $Z_c > 10\Delta Z_c$). An affine model is easier to calibrate than a projective one, requiring only three points. The calibration tends to be less sensitive to noise (it is better conditioned) and the model, being linear, is easier to use. Finally, the affine model admits more geometric invariants than the projective one, which is often useful for object recognition and other applications. [2]

(c) (i) Parallelism is preserved under affine imaging, but not under general perspective imaging. The square is foreshortened and therefore not parallel to the image plane. Had perspective effects been significant, at least one pair of the square’s parallel sides would have converged towards a vanishing point in the image. However, the sides have remained parallel, so an affine model is appropriate.

(ii) Under an affine camera model, parallelism and length ratios along parallel lines are preserved. Since two of the sides of B are parallel to the sides of the square, it follows that B is a right-angled triangle. Furthermore, the ratios of the parallel sides of A and B as measured in the image are the same as the ratios in the world. The ratios (expressed as A:B) are 5:7.5 for the vertical sides and 5:10 for the oblique sides (ratios expressed as A:B). The perpendicular sides of B are therefore of relative lengths 7.5 and 10. Pythagoras’ theorem gives the relative length of the hypotenuse as 12.5. The relative lengths of the sides of B are therefore $7.5:10:12.5 = 3:4:5$ (clockwise from the vertical side). [4]

4. **Stereo vision**

(a) [Book work] Triangulation describes the process of recovering the depth of a point observed by two cameras. If the cameras are calibrated (that is, their intrinsic parameters and relative position are known) the position of the point in the left and right images can be associated with rays in 3D space. These should intersect (this is the epipolar constraint) and the point of intersection determines the 3D position of the point. [2]

(b) (i)

$$
R = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
-1 & 0 & 0
\end{bmatrix}, \quad T = \begin{bmatrix}
-d \\
0 \\
d
\end{bmatrix}
$$

(ii) $X'_c$ and $p'$ are parallel, so $X'_c \times p' = 0$. Substituting $(RX_c + T)$ for $X'_c$ and $(Z_c p / f)$ for $X_c$ gives

$$
\left(\frac{Z_c}{f}R p + T\right) \times p' = 0
$$
Substituting for \( R \) and \( T \) gives

\[
\left( \frac{Z_c}{f} \begin{bmatrix} f & -d \\ y & 0 \\ -x & d \end{bmatrix} + \begin{bmatrix} x' \\ y' \end{bmatrix} \right) \times \begin{bmatrix} x' \\ y' \end{bmatrix} = 0
\]

Evaluating the vector product gives

\[
Z_c y + \frac{Z_c x y'}{f} - y' d = 0 \quad (1)
\]

\[
-\frac{2}{f} Z_c x x' + dx' - f Z_c + df = 0 \quad (2)
\]

\[
Z_c y' - dy' - \frac{Z_c y x'}{f} = 0 \quad (3)
\]

(1) and (3) give

\[
y' d = Z_c y + \frac{Z_c x y'}{f} = Z_c y' - \frac{Z_c y x'}{f}
\]

\[
\Leftrightarrow y(f + x') = y'(f - x)
\]

This is the epipolar constraint. Rearranging (2) gives

\[
Z_c (f^2 + xx') = df(f + x')
\]

which allows the recovery of depth from the image \( x \)-coordinates. [3]

(iii) Since the aircraft appears stationary in the left image, the aircraft is travelling along a ray from the left camera’s optical centre, and its trajectory in the right image will be an epipolar line. Substituting \((x, y) = (0, 1)\) cms into the epipolar constraint gives

\[
f + x' = fy' \quad \text{(distances in cms)}
\]

By comparison, the aircraft’s trajectory in the right image is

\[
y' = 1 + 1.25x' \quad \text{(distances in cms)}
\]

Hence \(1/f = 1.25\), so the cameras’ focal length is 0.8 cm (or 8 mm). Substituting \(x = 0\) into the depth recovery equation gives

\[
Z_c f^2 = df(f + x') \Leftrightarrow Z_c = d + \frac{d}{f} x'
\]

But since \(x' = 0.01t\) (for \(x'\) in m), \(d = 100\) m and \(f = 0.008\) m we have

\[
Z_c = 100 + \frac{100}{0.008} \times 0.01t \quad \text{(Zc in metres, t in seconds)}
\]

Hence \(\dot{Z}_c = 1/0.008 = 125\) m/s. Also, from the basic perspective equations, we know that

\[
x = 0 = \frac{f X_c}{Z_c} \Leftrightarrow \dot{X}_c = 0
\]

and

\[
y = 0.01 \text{ m} = \frac{f Y_c}{Z_c} \Leftrightarrow Y_c = 1.25 Z_c \Rightarrow \dot{Y}_c = 1.25 \dot{Z}_c
\]

So the aircraft’s speed is \(\sqrt{1^2 + 1.25^2} \times \dot{Z}_c = 1.6 \times 125 = 200\) m/s (720 km/h). [4]
5. **Structure from motion**

(a) [Book work] The motion of the scene relative to the camera is

\[
\begin{bmatrix}
\dot{X}_c \\
\dot{Y}_c \\
\dot{Z}_c
\end{bmatrix}
= -
\begin{bmatrix}
U_1 \\
0 \\
0
\end{bmatrix}
\]

Perspective projection onto the image plane can be expressed as

\[
x = \frac{fX_c}{Z_c} \Rightarrow \dot{x} = \frac{f\dot{X}_c}{Z_c} - \frac{fX_c\dot{Z}_c}{Z_c^2}
\]

Substituting for $\dot{X}_c$ and $\dot{Z}_c$ gives

\[
\dot{x} = -\frac{fU_1}{Z_c}
\]

Thus the slopes of the trajectories in the $x$-$t$ slice are $-fU_1/Z_c$. The slope of A’s trajectories is $-10$ mm/s, so the depth of A is given by

\[
-\frac{50U_1}{Z_c} = -10 \Leftrightarrow Z_c = 5U_1
\]

Similarly, the slope of B’s trajectories is $-50/3$ mm/s, so the depth of B is $3U_1$. [4]

(b) Since C is a curved surface, as the camera moves it “looks around” the ball and sees a different contour of C. The depths of the imaged edges change ($\dot{Z}_c \neq 0$) and the $x$-$t$ trajectories are no longer straight lines. Since the trajectories appear to be straight, we can conclude that the depth of the ball is large compared to its radius, so that the second term in (4), and the change in $Z_c$ in the first term in (4), are negligible\(^1\). As a rough estimate, let’s say that the depth should be at least ten times the radius, so $Z_c > 1$ m for the ball C. The slope of C’s trajectories is about the same as A’s, so $Z_c \approx 5U_1$ for C. Hence $5U_1 > 1$ and an approximate lower bound for $U_1$ is $0.2$ m/s. [4]

(c) Figure 5(b) shows a trajectory with a shallow slope occluding a trajectory with a larger slope. Assuming the scene is stationary, this infers a distant object occluding a nearby object, which is clearly impossible. [2]

6. **Applications**

(a) This is a structure from motion problem where the motion is completely unknown. The basic approach is to process two images from the sequence at a time. The two images should be acquired close together so that they cover overlapping parts of the chapel. The visual motion of as many image points as possible should be determined: this could be achieved by tracking the features (edges and corners), measuring the optical flow across the whole image or solving the discrete correspondence problem (having first constrained the search for correspondences by estimating

\(^1\)Attempts to infer the depth of the ball by measuring its diameter in the image (and using $\Delta x = f\Delta X_c/Z_c$) are flawed for two reasons: (i) for a curved object $Z_c$ will not be constant, and (ii) we are not necessarily viewing a true diameter in the image, so $\Delta X_c \neq 20$ cm.
the fundamental matrix and the epipolar geometry). The motion trajectories can be used to solve for the camera’s motion, though care has to be taken as this is usually an ill-conditioned problem. One of the more robust techniques is to use motion parallax: the relative motion of two aligned features can be used to solve for the camera’s direction of translation, with the rotation and relative depths following readily (there will be an unavoidable speed-scale ambiguity). In the absence of aligned features, affine motion parallax can be used to hallucinate aligned points from groups of nearby points. Unless optical flow is used (and optical flow is very difficult to determine accurately), these geometrical techniques will only recover the structure of the chapel near strong geometrical features (edges and corners). For a more complete geometric model of the chapel, it will be necessary to interpolate between these features, perhaps drawing on shading and texture to refine the interpolation. More advanced structure from motion techniques integrate information from more than two views to construct the 3D model. The model could be made more realistic by attempting to capture surface texture as well as geometric structure. This is readily achieved using texture mapping, whereby portions of an image are warped and “stuck” onto the geometric model in the desired places. [5]

(b) There are two stages to this problem: (i) identify where the hoardings lie in an image, and (ii) replace that portion of the image with another advertisement. The first task is greatly simplified by knowing where the hoardings are around the pitch. If the TV cameras are mounted on tripods, and we measure the orientation of the camera using servos, then we know, roughly, when a hoarding is in the image and where it is. From this rough initial guess, the localisation of the hoarding can be refined using image-based techniques: edge detection or correlation against a (warped) template view of the hoarding. Having accurately localised the hoarding, the next stage is to replace it with another advertisement. A template of the new advertisement must be warped to fit the slot in the image. If the camera is some distance from the hoarding (as it is bound to be), then the warping can be achieved by a planar affine transformation, which maps the template hoarding onto the right shape in the image (which will, in general, be a parallelogram). The affected pixels in the image are then replaced with the new pixels before the frame is transmitted.

(c) The change in apparent area of the approaching car can be used to estimate the time to contact with the camera: if this is above some threshold, then it is safe to pull out into the fast lane. The approaching car must be detected and then tracked for a period of time, long enough to estimate the divergence of the image velocity field around the car. The tracking could be achieved using a B-spline snake: as the snake deforms the image divergence can be estimated from the change in the snake’s area: \( \nabla \cdot \mathbf{v} \approx \dot{a}(t)/a(t) \). The time to contact is then given by \( 2/\nabla \cdot \mathbf{v} \). One of the hardest parts of the process is likely to be detecting the approaching car and automatically initialising the snake. A simple approach would be to look at difference images (formed by subtracting the previous image in the sequence from the current one). The outline of approaching cars will be prominent in the difference
image. Also prominent will be the lane markings, though these can be identified (since we’ll know, roughly, where to expect them in the image) and ignored. Other parts of the scene (at least those parts around the road surface) should not respond significantly in the difference image. [5]

(d) The camera’s motion is a pure rotation, so if we observe the same feature in two overlapping images we know that the rays $\mathbf{p}$ and $\mathbf{p}'$ in the two camera-centered coordinate systems are related by a rotation matrix: $\mathbf{p} = \mathbf{R}\mathbf{p}'$. Rays and homogeneous pixel coordinates are related via the intrinsic camera calibration matrix $\mathbf{C}$ ($\tilde{\mathbf{w}} = \mathbf{C}\mathbf{p}$), so corresponding pixels in the two views are related by a projectivity: $\tilde{\mathbf{w}} = \mathbf{CRC}^{-1}\mathbf{w}'$. We can calibrate the projectivity using four points observed in the two views. The four correspondences could be found by correlation-based matching, using constraints (ordering, uniqueness, etc.) to limit the search. If more than four matches are found, then the calibration can be refined using least squares. If we find the projectivities between many pairs of overlapping images in the sequence, we can concatenate the projectivities to relate any image to the north pointing view. We can then use the image-to-north projectivities to warp the images as if they had been acquired by a north pointing camera. Finally, the warped images can be stitched together to construct the panoramic view to the north. [5]

Andrew Gee
April 1997
Module I12: Computer Vision and Robotics

Solutions to 1998 Tripos Paper

1. Feature detection

(a) (i) [Book work]

\[ G_\sigma(x, y) * I(x, y) = g_\sigma(x) * [g_\sigma(y) * I(x, y)] \]

where

\[ g_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{x^2}{2\sigma^2} \right) \]

(ii) The kernel is usually truncated so that the discarded values are less than 1/1000 of the peak value. If we discard the sample \((n + 1)\) pixels from the centre of the kernel, the size of the kernel will be \(2n + 1\) pixels. We can find \(n\) by solving:

\[
\exp \left[ -\frac{(n + 1)^2}{2\sigma^2} \right] < \frac{1}{1000} \\
\iff n > 3.7\sigma - 1
\]

For \(\sigma = 1\) we get \(n > 2.7\). Rounding up to the nearest integer we find that \(n = 3\) and the size of the kernel is 7 pixels.

We cannot convolve pixels which are less than 4 pixels away from the border of the image. This leaves a sub-image of size \(506 \times 506\) for smoothing. The 2D scheme requires \(506 \times 506 \times 7 \times 7 = 1.3 \times 10^7\) operations. The 1D scheme requires \(506 \times 506 \times 7 \times 2 = 3.6 \times 10^6\) operations.

(b) By examining first order Taylor expansions, we find that

\[
\frac{d^2 I}{dx^2}_{(x,y)} \approx I(x-1,y) - 2I(x,y) + I(x+1,y)
\]

and

\[
\frac{d^2 I}{dy^2}_{(x,y)} \approx I(x,y-1) - 2I(x,y) + I(x,y+1)
\]

It follows that the Laplacian can be estimated as follows:

\[
\nabla^2 I_{(x,y)} = \frac{d^2 I}{dx^2}_{(x,y)} + \frac{d^2 I}{dy^2}_{(x,y)} \approx I(x-1,y) + I(x+1,y) + I(x,y-1) + I(x,y+1) - 4I(x,y)
\]

This estimate can be computed by convolving with the kernel

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]
(c) [Book work] The principle advantage of the Marr-Hildreth operator is computational efficiency: edge detection requires only a single convolution and the detection of zero-crossings. Conversely, the Canny operator requires an additional, costly search for a local maximum normal to the gradient direction. The advantage of the Canny operator is enhanced robustness to noise. Any differential operator amplifies noise. The Canny operator computes only first derivatives and then searches for a local maximum (which is equivalent to a zero-crossing of the second derivative) normal to the gradient. The Marr-Hildreth operator computes second derivatives both along and normal to the edge. Computation of the second derivative along the edge emphasizes noise in that direction while serving no purpose in edge detection.

2. Perspective and weak perspective projection

(a) [Book work] The mapping from camera-centered coordinates \((X_c, Y_c, Z_c)\) to pixel coordinates \((u, v)\) involves a perspective projection onto the image plane \((x, y)\) followed by an anisotropic scaling and translation in the image plane to account for the dimensions and positioning of the CCD array.

The perspective projection is a non-linear operation in Cartesian coordinates:

\[
x = \frac{fX_c}{Z_c}, \quad y = \frac{fY_c}{Z_c}
\]

where \(f\) is the focal length of the camera. This can be rewritten as a linear operation in homogeneous coordinates:

\[
\begin{bmatrix}
x \\
y \\
s
\end{bmatrix} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix}
\]

The mapping from image plane coordinates \((x, y)\) to pixel coordinates \((u, v)\) is given by:

\[
u = u_0 + k_u x , \quad v = v_0 + k_v y
\]

where the optical axis intersects the CCD array at the pixel with coordinates \((u_0, v_0)\) and there are \(k_u\) pixels per unit length in the \(u\) direction and \(k_v\) in the \(v\) direction. In homogeneous coordinates, this becomes

\[
\begin{bmatrix}
su \\
sv \\
s
\end{bmatrix} = \begin{bmatrix}
k_u & 0 & u_0 \\
0 & k_v & v_0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
s
\end{bmatrix}
\]

Concatenating the two transformations, we obtain

\[
\begin{bmatrix}
su \\
sv \\
s
\end{bmatrix} = \begin{bmatrix}
k_u f & 0 & u_0 \\
0 & k_v f & v_0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1
\end{bmatrix}
\]
(b) (i) 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:

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

(ii) Under full perspective we have

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

Under weak perspective we have

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

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

$$
u - \nu_A = \nu - \left(\frac{u Z_c + u_0 \Delta Z}{Z_c}\right) \left(\frac{Z_c}{Z_A}\right) = \left(\frac{u Z_A - Z_c}{Z_c}\right) - \left(\frac{Z_c}{Z_A}\right) = (u - u_0) \frac{\Delta Z}{Z_A}
$$

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.

(iii) [Book work] 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.

3. Camera calibration and planar affine projection

(a) [Book work] (i) The relationship is valid under the assumption that the image is formed by a “pinhole camera”, such that rays pass through a single point (the optical centre) before striking the image plane. The relationship does not account for nonlinear distortion, which affects all real cameras to some extent.
(ii) Geometrically, $s$ can be thought of as a scale factor, controlling the size of the image formed by an object in the world. It depends on the distance $Z_c$ of the object from the camera. The elements $p_{ij}$ describe an isometry (a rotation and translation), followed by perspective projection onto an image plane and sampling of the plane by a CCD array.

Algebraically, $s$ and the elements $p_{ij}$ allow the imaging process to be expressed as a linear relationship in homogeneous coordinates. In Cartesian coordinates the perspective image formation process cannot be expressed linearly, requiring a division by $Z_c$.

(iii) The process of estimating the elements $p_{ij}$ is known as camera calibration. It is necessary to observe a scene where the world positions of several distinguished features are known. For example, we might set up the camera to view a calibrated grid of some sort.

There are 11 parameters to estimate (since the overall scale of the projection matrix does not matter, we could, for example, set $p_{34}$ to 1). Each point we observe gives us a pair of equations. Setting $p_{34}$ to 1 we obtain:

$$u = \frac{su}{s} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + 1}$$

$$v = \frac{sv}{s} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + 1}$$

Since we are observing a familiar 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, we need to observe at least 6 points to calibrate the camera. The equations can be solved using linear least squares.

It is essential that the points used for calibration are in a general configuration. If they are not, for instance if they are coplanar or even co-linear, then 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 linear least squares procedure will
not find a unique solution (the pseudo-inverse will not exist). Nonlinear distortion is not included in the projective camera model. Any residuals remaining after the linear least squares procedure are partly due to nonlinear distortion, as well as errors in localising the corners in the image and in the world.

(b) (i) It is straightforward to calculate the perspective scale factor $s$ at the four corners of the visible part of the road junction:

\[-23, 3] : \quad s = -0.001 \times 23 + 0.001 \times 3 + 1.0 = 0.98 \\
[5, 30] : \quad s = 0.001 \times 5 + 0.001 \times 30 + 1.0 = 1.035 \\
[45, -24] : \quad s = 0.001 \times 45 - 0.001 \times 24 + 1.0 = 1.021 \\
[15, -48] : \quad s = 0.001 \times 15 - 0.001 \times 48 + 1.0 = 0.967 \\

So $0.967 \leq s \leq 1.035$.

The projection is virtually affine. The camera is positioned such that the depth variation across the scene is less than 3.5%. To simplify subsequent calculations, it is reasonable to adopt an affine camera model:

\[
\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 10.8 & 8.2 & 230.5 \\ 5.4 & -5.6 & 142.0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{5}
\]

(ii) Substituting $(u, v) = (100, 100)$ into (5) allows us to solve for the car’s world position $(X, Y)$. Solving the two linear simultaneous equations gives the car’s position as $(-10.26, -2.40)$ metres.

Differentiating (5) gives:

\[
\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 10.8 & 8.2 \\ 5.4 & -5.6 \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{6}
\]

Substituting $(\dot{u}, \dot{v}) = (180, 90)$ into (6) allows us to solve for the car’s physical $(\dot{X}, \dot{Y})$. Solving the two linear simultaneous equations gives the car’s velocity as $(16.67, 0.0)$ metres/second. So the car’s speed is 60 km/h, which seems reasonable.

4. Stereo vision

(a) [Book work] The two camera-centered coordinate systems are related by a rotation $R$ and translation $T$:

$$X'_c = RX_c + T$$

Taking the vector product with $T$, we obtain

$$T \times X'_c = T \times RX_c + T \times T$$

$$\Rightarrow T \times X'_c = T \times RX_c$$

Taking the scalar product with $X'_c$, we obtain

$$X'_c \cdot (T \times X'_c) = X'_c \cdot (T \times RX_c)$$

$$\Rightarrow X'_c \cdot (T \times RX_c) = 0 \tag{7}$$
Recall that a vector product can be expressed as a matrix multiplication:

\[
T \times X_c = T \times X_c
\]

where \( T = \begin{bmatrix}
0 & -T_z & T_y \\
T_z & 0 & -T_x \\
-T_y & T_x & 0
\end{bmatrix} \)

So equation (7) can be rewritten as

\[
X'_c (T \times R X_c) = 0 \\
\Leftrightarrow X'_c^T E X_c = 0,
\]

where \( E = T \times R \)

The constraint also holds for rays \( p = [x \ y \ f]^T \), which are parallel to the camera-centered position vectors \( X_c \):

\[
p'^T E p = 0
\]

(8)

\( E \) is a 3 \( \times \) 3 matrix known as the essential matrix. It has maximum rank 2.

(b) (i) The epipoles lie in the null spaces of \( E \) and \( E^T \).

\[
E : \begin{bmatrix}
0 & c & 0 \\
-c & 0 & a \\
0 & -a & 0
\end{bmatrix} \begin{bmatrix}
x_e \\
y_e \\
f
\end{bmatrix} = 0 \Leftrightarrow \begin{bmatrix}
x_e \\
y_e \\
f
\end{bmatrix} = \begin{bmatrix}
af/c \\
0 \\
f
\end{bmatrix}
\]

\[
E^T : \begin{bmatrix}
0 & -c & 0 \\
c & 0 & -a \\
0 & a & 0
\end{bmatrix} \begin{bmatrix}
x_e \\
y_e \\
f
\end{bmatrix} = 0 \Leftrightarrow \begin{bmatrix}
x_e \\
y_e \\
f
\end{bmatrix} = \begin{bmatrix}
af/c \\
0 \\
f
\end{bmatrix}
\]

So the epipoles are at \((af/c, 0)\) (image plane coordinates) in both views.

(ii) The translation is in the direction of the ray to the epipole, so it is in the direction \( \pm (af/c, 0, f) \). We cannot deduce whether the motion is forwards or backwards, since both result in the same epipolar geometry.
(iii) The epipolar line structure is the same in both views. All the epipolar lines pass through the epipole. See sketch under (iv) below.

(iv) The image trajectory of a point will lie on an epipolar line. We cannot deduce whether the point will move towards or away from the epipole (depends on the direction of translation). See trajectory sketched on right image below.

5. **Optical flow and time to contact**

(a) [Book work] To estimate the motion field \( \mathbf{v} \) from image intensities \( I(x, y, t) \) it is necessary to make a key assumption. We assume that any change in a pixel’s intensity over time is a result of translation of the local intensity distribution. This assumption is not generally true (in fact, this will only be the case for Lambertian surfaces moving under homogeneous, isotropic, time-invariant illumination), but nevertheless allows us to derive a useful result.

\[
\delta I = -\frac{\partial I}{\partial x} \times \delta x - \frac{\partial I}{\partial y} \times \delta y = -\frac{\partial I}{\partial x} \times \left( \frac{dx}{dt} \times \delta t \right) - \frac{\partial I}{\partial y} \times \left( \frac{dy}{dt} \times \delta t \right)
\]

\[
\Rightarrow \frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} \times \frac{dx}{dt} + \frac{\partial I}{\partial y} \times \frac{dy}{dt} = 0
\]

\[
\Leftrightarrow \frac{\partial I}{\partial t} + \left[ \frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y} \right]^T \left[ \frac{dx}{dt} \quad \frac{dy}{dt} \right]^T = 0 \Leftrightarrow \frac{\partial I}{\partial t} + \nabla I \cdot \mathbf{v} = 0 \quad (9)
\]

Since we can measure \( \frac{\partial I}{\partial t} \) and \( \nabla I \), equation (9) provides a single constraint on \( \dot{x} \) and \( \dot{y} \), the two elements of \( \mathbf{v} \).

\( \mathbf{v} \) estimated in this way is known as the **optical flow**. It is only an approximation to the underlying image velocity, since in reality changes in a pixel’s intensity are not totally determined by translation of the local intensity pattern.
Note that (9) provides an expression for the component of optical flow along the image intensity gradient. This is a manifestation of the aperture problem: we can only recover the component of image motion perpendicular to an intensity edge.

Given the single constraint on the optical flow (9), techniques exist to estimate the full optical flow $\mathbf{v}$ across the whole image. They usually involve a global optimization which tries to find a field that satisfies (9) and also a local smoothness constraint.

(b) (i) The rigid body motion equation describing the rate of change of camera-centered coordinates is

$$\dot{\mathbf{X}_c} = -\mathbf{U} - \Omega \times \mathbf{X}_c$$

For pure translation along the optical axis, this simplifies to

$$\dot{\mathbf{X}_c} = -\mathbf{U} \quad \text{where} \quad \mathbf{U} = [0 \ 0 \ U]^T$$

The relationship between rays and camera-centered coordinates is

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

Differentiating this expression, we obtain

$$\dot{\mathbf{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} = \frac{f}{Z_c} \dot{\mathbf{X}}_c - \frac{f}{Z_c^2} \mathbf{X}_c \dot{Z}_c$$

We now make some substitutions:

$$\mathbf{X}_c = \frac{Z_c}{f} \mathbf{p} \quad \dot{\mathbf{X}}_c = -\mathbf{U} \quad \dot{Z}_c = \dot{\mathbf{X}}_c \mathbf{k} = -U$$

The result of these substitutions is:

$$\dot{\mathbf{p}} = -\frac{f U}{Z_c} + \frac{U \mathbf{p}}{Z_c} = \begin{bmatrix} xU \\ yU \\ 0 \end{bmatrix}$$

This is a divergent motion field with focus of expansion at the point where the optical axis pierces the image plane.
(ii) Substituting the perspective projection equations \( x = fX_c/Z_c \) and \( y = fY_c/Z_c \) into the equation of the plane gives

\[
Z_c = Z_0 + \frac{pZ_c x}{f} + \frac{qZ_c y}{f} \Leftrightarrow \frac{1}{Z_c} = \frac{1}{Z_0} \left( 1 - \frac{px}{f} - \frac{qy}{f} \right)
\]

Substituting this expression into the image velocity equations gives

\[
\dot{x} = \frac{xU}{Z_c} = \frac{xU(1 - \frac{px}{f} - \frac{qy}{f})}{Z_0} = \frac{U}{Z_0} \left( x - \frac{px^2}{f} - \frac{qxy}{f} \right)
\]

\[
\dot{y} = \frac{yU}{Z_c} = \frac{yU(1 - \frac{px}{f} - \frac{qy}{f})}{Z_0} = \frac{U}{Z_0} \left( y - \frac{pxy}{f} - \frac{qy^2}{f} \right)
\]

(iii) The divergence of the image velocity field is

\[
\nabla \cdot \mathbf{v} = \frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y} = \frac{U}{Z_0} \left( 1 - \frac{2px}{f} - \frac{qy}{f} + 1 - \frac{px}{f} - \frac{2qy}{f} \right)
\]

At the principle point \((x = y = 0)\), this simplifies to

\[
\nabla \cdot \mathbf{v} = \frac{2U}{Z_0}
\]

The time to contact is given by

\[
t_c = \frac{Z_0}{U} = \frac{2}{\nabla \cdot \mathbf{v}}
\]

In practice, the divergence can be estimated by tracking a closed contour in the image (using a B-spline snake) and using Green’s theorem to relate \( \nabla \cdot \mathbf{v} \) to the area \( a \) enclosed by the contour and the rate of change of area:

\[
\frac{da}{dt} \approx a(t) \nabla \cdot \mathbf{v}
\]

6. Applications

(a) Both edge and corner detectors are fully data driven, bottom-up and very local. Corner detectors tend to produce outputs that are poorly localised and temporally unstable. Edge detectors tend to produce fragmented edgel chains. They are both sensitive to clutter. For these reasons, it is very difficult to establish correspondences between edge and corner features in successive frames.

B-splines provide a compact representation of complete contours. They can be fitted to image gradients in an efficient and robust way, rejecting fits that do not preserve shape (eg. affine B-spline snakes). In this way, they are not easily distracted by clutter. Their weaknesses are that they often need hand fitting in the first frame and, like all edge based techniques, they suffer from the aperture problem locally.
(b) The camera should look down on the ground plane (the floor). The image-to-ground mapping has 8 degrees of freedom:

\[
\begin{bmatrix}
    su \\
    sv \\
    s
\end{bmatrix} = \begin{bmatrix}
    p_{11} & p_{12} & p_{13} \\
    p_{21} & p_{22} & p_{23} \\
    p_{31} & p_{32} & 1
\end{bmatrix} \begin{bmatrix}
    X \\
    Y \\
    1
\end{bmatrix}
\]

This can be calibrated using four known points on the floor (and their corresponding image positions), though for greater accuracy more correspondences should be used (with least squares).

Consecutive images can be subtracted from each other (time differencing) to detect moving people. The bottom of the moving region should correspond to the point of contact between the person and the floor. The calibration can then be used to translate this into a world position on the floor.

People can be tracked using cross-correlation of the moving blobs, or using B-spline snakes.

(c) For a parallel camera geometry, the raster lines are epipolar lines. Edges and corners can be detected in the left and right images, and matched using ordering, epipolar and figural constraints. From a large set of correspondences, it is possible to calculate a dense disparity map. Then an intermediate view can be generated by taking one image, and shifting all the pixels a fraction of the way towards their positions in the other image (i.e. the shift is proportional to the disparity).

(d) Given that the cameras are some distance from the workspace, an affine model is appropriate:

\[
\begin{bmatrix}
    u \\
    v
\end{bmatrix} = \begin{bmatrix}
    p_{11} & p_{12} & p_{13} & p_{14} \\
    p_{21} & p_{22} & p_{23} & p_{24}
\end{bmatrix} \begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

The left and right cameras can be calibrated by moving the gripper to four predetermined points in 3D space, and tracking its image position using affine B-spline snakes. More points (and least squares) could be used for better accuracy.

With two calibrated affine cameras, it is straightforward to triangulate to recover structure. Each point observed in left and right images gives us 4 equations in the 3 unknowns \((X, Y, Z)\). These can be solved using least squares.

The user needs to specify the target in each view. The calibration can then be used to determine the world position of the target, and the gripper moved to the right location for a grasping manoeuvre.

Andrew Gee & Roberto Cipolla
January 1998
1. Feature detection

(a) $I(x, y)$ is a function of many variables, including the position of the camera; the properties of the lens and the CCD; the shape of the structures in the scene; the nature and distribution of light sources; and the reflectance properties of the visible surfaces.

Edge detection is commonly used in the first stage of many computer vision applications, since edges provide a compact representation of image structure and are invariant to illumination effects. Compared with raw images, edges offer significant data reduction while preserving much of the image’s useful information content (it is possible to recognise many structures in a line drawing of a scene). In contrast, most of the discarded information is not useful for discovering scene structure and motion.

(b) (i) To preserve the mean intensity of the image, $A$ should be set to $1 + 4 + 6 + 4 + 1 = 16$.

(ii) The smoothed pixels $s(x)$ are obtained by discrete convolution:

$$s(x) = \sum_{i=-2}^{2} g(i)I(x - i)$$

where $g(-2) = 1/16$, $g(-1) = 1/4$, $g(0) = 3/8$, $g(1) = 1/4$ and $g(2) = 1/16$.

Intensity discontinuities are localised by differentiating the smoothed pixels to obtain the gradient $d(x)$. This can be achieved by convolution with the kernel $[1 \ -1]$:

$$d(x) = s(x) - s(x - 1)$$

Intensity discontinuities are then localised at local maxima of $d(x)$.

(iii) 2D convolution with a Gaussian kernel $G_\sigma(x, y)$ can be decomposed into two 1D convolutions with a Gaussian kernel $g_\sigma(x)$ as follows:

$$G_\sigma(x, y) * I(x, y) = \frac{1}{2\pi\sigma^2} \int \int I(x - u, y - v) \exp \left( -\frac{u^2 + v^2}{2\sigma^2} \right) \, du \, dv$$

$$= \frac{1}{\sqrt{2\pi}} \int \exp \left( -\frac{u^2}{2\sigma^2} \right) \left[ \frac{1}{\sqrt{2\pi}\sigma} \int I(x - u, y - v) \exp \left( -\frac{v^2}{2\sigma^2} \right) \, dv \right] \, du$$

$$= \frac{1}{\sqrt{2\pi}} \int \exp \left( -\frac{u^2}{2\sigma^2} \right) [g_\sigma(y) * I(x - u, y)] \, du$$

$$= g_\sigma(x) * [g_\sigma(y) * I(x, y)]$$
So the discrete 1D kernel should be applied first along the image rows, then along the columns (or the other way around). Performing two 1D convolutions is much faster than performing a single 2D convolution.

(c) Averaging neighbouring pixels once is equivalent to convolution with the kernel \([\frac{1}{2} \frac{1}{2}]\). Averaging twice is equivalent to convolution with the kernel

\[
\begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix} \ast \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix} = \begin{bmatrix}
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
\end{bmatrix}
\]

Averaging three times is equivalent to convolution with the kernel

\[
\begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix} \ast \begin{bmatrix}
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
\end{bmatrix} = \begin{bmatrix}
\frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8}
\end{bmatrix}
\]

Averaging four times is equivalent to convolution with the kernel

\[
\begin{bmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{bmatrix} \ast \begin{bmatrix}
\frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16}
\end{bmatrix} = \begin{bmatrix}
\frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \frac{1}{16}
\end{bmatrix}
\]

which is the kernel in (b). So 1D smoothing with the kernel in (b) is equivalent to successively averaging neighbouring pixels. Four averaging operations are required.

Assessors’ remarks: The bookwork component produced very poor answers. Most candidates gave correct expressions for convolution and differentiation and were able to show how to apply the 1D filter in 2D smoothing. Many candidates got the correct filter kernel for the smoothing filter.

2. Camera calibration and vanishing points

(a) There is a rotation \(R\) and translation \(T\) between the world coordinates \(\tilde{\mathbf{X}}\) and the camera-centered coordinates \(\tilde{\mathbf{X}}_c\) (both expressed in homogeneous coordinates).

\[
\tilde{\mathbf{X}}_c = \begin{bmatrix} R & T \end{bmatrix} \tilde{\mathbf{X}}
\]

The next stage is perspective projection of \(\tilde{\mathbf{X}}_c\) onto \(\tilde{\mathbf{x}}\) in the image plane. Assume the focal length is \(f\):

\[
\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} \tilde{\mathbf{X}}_c
\]

\(\tilde{\mathbf{x}} = (sx, sy, s)\) is the homogeneous representation of the image point \(\mathbf{x} = (x, y)\). Finally, we have to convert to pixel coordinates \(\mathbf{w} = (u, v)\). Assume the optical axis intersects the image plane at the pixel with coordinates \((u_0, v_0)\) and there are \(k_u\) pixels per unit distance in the \(u\) direction and \(k_v\) in the \(v\) direction:

\[
\tilde{\mathbf{w}} = \begin{bmatrix} k_u & 0 & u_0 \\
0 & k_v & v_0 \\
0 & 0 & 1 \\
\end{bmatrix} \tilde{\mathbf{x}}
\]
where $\hat{w} = (su, sv, s)$ is the homogeneous representation of $w$. We can now express the overall imaging process, from $\hat{X}$ to $\hat{w}$, as a single matrix multiplication in homogeneous coordinates:

$$\hat{w} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{X} = P \hat{X},$$

$P$ is a $3 \times 4$ matrix, so the process can be expressed as

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(b) $P$ can be estimated by observing the images of known 3D points. 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 known 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:

$$Ap = 0$$

where $p$ is the $12 \times 1$ 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 orthogonal least squares solution corresponds to the eigenvector of $A^T A$ with the smallest corresponding eigenvalue.

It is essential that the calibration points are not coplanar, since otherwise 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

$$P = \begin{bmatrix} f k_u & 0 & u_0 & 0 \\ 0 & f k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f k_u & 0 & u_0 \\ 0 & f k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= C \begin{bmatrix} R & T \end{bmatrix} = \begin{bmatrix} CR & CT \end{bmatrix}$$
it is apparent that we need to decompose the left 3×3 sub-matrix of P into an upper triangular matrix C and an orthogonal (rotation) matrix R. This can be achieved using QR decomposition. T can then be recovered using

\[ T = C^{-1} \begin{bmatrix} p_{14} & p_{24} & p_{34} \end{bmatrix}^T \]

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 C has a zero in the middle of its top row, as expected. If we scale the matrix C 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 \(c_{13}\) and \(c_{23}\), and the products \(fk_u\) and \(fk_v\) by looking at \(c_{11}\) and \(c_{22}\). It is not possible to decouple the focal length from the pixel scaling factors.

(c) Distant points on lines parallel to the world X-axis can be represented in homogeneous coordinates as

\[ \tilde{X} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

Applying the projection matrix P, we find the image of these points is

\[ \begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \end{bmatrix} \]

The vanishing point of lines which are parallel to the world X-axis is therefore \((u, v) = (p_{11}/p_{31}, p_{21}/p_{31})\).

**Assessors’ remarks:** This question was mainly bookwork and was very well-answered. Many candidates clearly demonstrated they had understood camera models and calibration. In part (c), few candidates derived the vanishing point by considering the projection of a point at infinity using the projection matrix.

3. **Line to line transformations**

(a)
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:

\[
\begin{align*}
  l_c - l_a &= \frac{(X_c - X_a)(p - qr)}{(rX_c + 1)(rX_a + 1)} \\
  l_c - l_b &= \frac{(X_c - X_b)(p - qr)}{(rX_c + 1)(rX_b + 1)} \\
  \Rightarrow \frac{l_c - l_a}{l_c - l_b} &= \frac{(X_c - X_a)(rX_b + 1)}{(X_c - X_b)(rX_a + 1)} \tag{10}
\end{align*}
\]

Similarly

\[
\begin{align*}
  l_d - l_a &= \frac{(X_d - X_a)(rX_b + 1)}{(X_d - X_b)(rX_a + 1)} \\
  \frac{l_d - l_a}{l_d - l_b} &= \frac{(X_d - X_a)(X_c - X_b)}{(X_d - X_b)(X_c - X_a)} \tag{11}
\end{align*}
\]

Dividing (11) by (10) we obtain the cross-ratio:

\[
\frac{(l_d - l_a)(l_c - l_b)}{(l_d - l_b)(l_c - l_a)} = \frac{(X_d - X_a)(X_c - X_b)}{(X_d - X_b)(X_c - X_a)}
\]

Hence the cross-ratio is invariant under perspective projection.

(b) As the height of the pastry changes, the projected point moves along a line in 3D space (the line emanating from the light source in Fig. 2 on the question paper). The image of the projected point moves along the image of this line, which is also a line (assuming linear perspective projection with no radial distortion).

So the transformation between the projected point’s position in 3D space and the pixel coordinates \((u, v)\) of its image is a 1D projective one. Since the 3D position of the projected point is a linear function of the height of the pastry \(Z\), it follows that the transformation between the height of the pastry and the pixel coordinates of the point’s image is also a projective one:

\[
\begin{bmatrix}
  su \\
  sv \\
  s
\end{bmatrix}
= 
\begin{bmatrix}
  p_{11} & p_{12} \\
  p_{21} & p_{22} \\
  p_{31} & p_{32}
\end{bmatrix}
\begin{bmatrix}
  Z \\
  1
\end{bmatrix}
\]

Even though we need only look at one of the coordinates \(u\) or \(v\) to calibrate the system and recover the pastry height \(Z\), in practice there will be some measurement noise to deal with. 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 and subsequently recover $Z$ from
the image of the projected point.

To calibrate the system, we should observe the projected point for at least three
known pastry heights. Each observation provides us with two equations in the six
unknowns $p_{11} \ldots p_{32}$:

\[
\begin{align*}
    u &= su = \frac{p_{11}Z + p_{12}}{p_{31}Z + p_{32}} \\
    v &= sv = \frac{p_{21}Z + p_{22}}{p_{31}Z + p_{32}}
\end{align*}
\]

Three observations provide six equations, which are sufficient to solve for the cal-


ter {\frac{s u}{s}} = \frac{p_{11}Z + p_{12}}{p_{31}z + p_{32}}
\]
\]

After calibration, both the $u$ and $v$ image coordinates of the projected point should
be used to estimate the height of the pastry $Z$. This gives us two equations, in

\[
\begin{align*}
    u &= \frac{su}{s} = \frac{p_{11}Z + p_{12}}{p_{31}Z + p_{32}} \\
    v &= \frac{sv}{s} = \frac{p_{21}Z + p_{22}}{p_{31}Z + p_{32}}
\end{align*}
\]

The over-constrained set of equations can then be solved using orthogonal least squares to recover the calibration matrix up
to scale.

Assessors’ remarks: Part (a) was very poorly answered despite being bookwork. Many
candidates were unable to to simplify the ratio and cross-ratio expressions. Reasonable
ttempts were made at part (b) but many failed to give adequate details on the use of least-squares in calibration.

4. Stereo vision

(a) The fundamental matrix $F$ relates points in the left and right images of a stereo

\[
\tilde{w}'^T F \tilde{w} = 0
\]

where $\tilde{w} = (u, v, 1)$ are the point’s pixel coordinates in the left image, and $\tilde{w}'$ are
the coordinates of the corresponding point in the right image. The constraint arises
from the requirement that the rays from the two cameras’ optical centres through
$\tilde{w}$ and $\tilde{w}'$ must intersect at a point in space. $F$ has zero determinant and can be
determined only up to scale.

$F$ can be estimated from point correspondences. Each point correspondence $\tilde{w} \leftrightarrow \tilde{w}'$
generates one constraint on $F$:

\[
\begin{bmatrix}
    u' & v' & 1
\end{bmatrix}
\begin{bmatrix}
    f_{11} & f_{12} & f_{13} \\
    f_{21} & f_{22} & f_{23} \\
    f_{31} & f_{32} & f_{33}
\end{bmatrix}
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix} = 0
\]

This is a linear equation in the unknown elements of $F$. Given eight or more perfect
correspondences (image points in general position, no noise), $F$ can be determined
uniquely up to scale by solving the simultaneous linear equations. In practice, there may be more than eight correspondences and the image measurements will be noisy. The system of equations can then be solved by least squares, or using a robust regression scheme to reject outliers.

The linear technique does not enforce the constraint that \( \det F = 0 \). If the eight image points are noisy, then the linear estimate of \( F \) will not necessarily have zero determinant and the epipolar lines will not meet at a point. Nonlinear techniques exist to estimate \( F \) from 7 point correspondences, enforcing the rank 2 constraint.

(b) Consider first the location of the left epipole expressed as a ray \( p_e \) to the image plane.

Referring to the figure, the position of the left camera’s epipole is \( p_e \) in the left camera’s coordinate system and \( \lambda T \) in the right camera’s coordinate system. Relating the coordinate systems, we obtain

\[
\lambda T = Rp_e + T
\]

Taking the vector product with \( T \), we obtain

\[
0 = T \times Rp_e \quad \Leftrightarrow \quad Ep_e = 0
\]

So the location of the epipole in the left image lies in the null space of \( E \). We can relate rays to pixels using the camera calibration matrix \( C \), since \( \tilde{w} = Cp \). Hence:

\[
Ep_e = 0
\]

\[
\Leftrightarrow EC^{-1}\tilde{w}e = 0
\]

\[
\Leftrightarrow C'^{-T}EC^{-1}\tilde{w}e = 0
\]

\[
\Leftrightarrow F\tilde{w}e = 0
\]

The corresponding result for the right epipole is \( F^T\tilde{w}'_e = 0 \).
So, to find the left epipole, solve:

\[
\begin{bmatrix}
0 & 1 & -200 \\
1 & 0 & -2700 \\
-200 & 2300 & 80000
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
\]

The solution, by inspection, is \((u, v) = (2700, 200)\). To find the right epipole, solve:

\[
\begin{bmatrix}
0 & 1 & -200 \\
1 & 0 & 2300 \\
-200 & -2700 & 80000
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
\]

The solution, again by inspection, is \((u, v) = (-2300, 200)\).

(c) Given the intrinsic camera parameters, we can use the following equations to express the epipoles as rays to the image plane:

\[
\begin{align*}
u &= u_0 + k_u x, \quad v = v_0 + k_v y
\end{align*}
\]

In this case \(u_0 = v_0 = 200\) pixels and \(k_u = k_v = 50\) pixels/mm. For the left epipole, \((u, v) = (2700, 200)\) gives \((x, y) = (50, 0)\) mm. For the right epipole \((u, v) = (-2300, 200)\) gives \((x, y) = (-50, 0)\) mm. Since the focal length is 50mm, it follows that \(p_e = (50, 0, 50)\) mm and \(p'_e = (-50, 0, 50)\) mm. This tells us that the cameras are arranged symmetrically at right angles, more or less as shown in the diagram above. We cannot, however, deduce the distance between the two optical centres.

Assessors’ remarks: This question tested the candidates’ understanding of epipolar geometry and the fundamental matrix. Part (a) was answered fairly well, with most candidates able to explain the significance of the fundamental matrix and how it can be estimated from point correspondences. Many candidates did not seem to possess the ability to distinguish the relevant from the irrelevant, and wasted a lot of time deriving the epipolar constraint from first principles, when this was clearly not required. Part (b) was also fairly well answered, with most candidates able to find the two epipoles, and some also able to derive the result that the epipoles lie in the null space of F. Part (c) was extremely poorly answered, with only a couple of candidates possessing the insight to find the rays to the epipoles and deduce the relative position and orientation of the cameras by inspection.

5. Structure from motion and motion parallax

(a) The focus of expansion is the point in the image where the optical flow is instantaneously zero (apart from all points on the horizon). The translational motion field is radial with respect to the focus of expansion, which lies at the intersection of the image plane with a line parallel to the direction of translation through the optical centre. To find the image plane location of the focus of expansion in terms of \(U\), we simply solve the motion equations for \(\Omega = 0\) and \(\dot{x} = \dot{y} = 0\).

\[
\begin{align*}
-fU_1 x_0 U_3 = 0 & \iff x_0 = \frac{fU_1}{U_3} \\
-fU_2 y_0 U_3 = 0 & \iff y_0 = \frac{fU_2}{U_3}
\end{align*}
\]

34
So the focus of expansion lies at \((f \frac{U_1}{U_3}, f \frac{U_2}{U_3})\).

(b) (i) Motion parallax vectors are found by subtracting the image velocities of the coincident points. This gives \((-10, 0)\) mm/sec for the pair of points at A, and \((0, -5)\) mm/sec for the pair at B.

(ii) Since the rotational component of image velocity is the same for coincident points, the parallax vectors arise solely from the difference in the translational components. Since the translational component is radial with respect to the focus of expansion, the two parallax vectors can be extrapolated and the focus of expansion located at the point of intersection. In this case, the focus of expansion lies at \((x, y) = (0, 0)\), the centre of the image plane. It follows from (a) that \(U_1 = U_2 = 0\), so the direction of translation is along the camera’s optical axis.

By inspection, the image velocities can be decomposed into translational and rotational components as follows.

One of the points at A and one of the points at B has no translational component of image velocity, so these two points must lie on the horizon.

The rotational image velocities are consistent with a pure rotation about the optical axis. Writing the image motion equations for pure rotation and \(\Omega = (0, 0, \Omega_3)\) gives:

\[
x = y\Omega_3, \quad \dot{y} = -x\Omega_3
\]

Substituting the values from the left hand figure above gives \(\Omega_3 = 1\) rad/sec.

Turning now to the translational velocities, the image motion equations simplify to the following form for pure translation along the optical axis:

\[
\dot{x} = \frac{xU_3}{Z_c}, \quad \dot{y} = \frac{yU_3}{Z_c}
\]

The translational velocities in the right hand figure above reveal that the aircraft is flying forwards (phew) and that, disregarding the two points on the horizon, the points at B and C are twice as far away as the point at A. It is not possible to say how far away these points are unless we know \(U_3\): this is the speed-scale ambiguity.
The aircraft is performing an anticlockwise roll manoeuvre. The presence of a horizon line through AB suggests that the aircraft is currently banked at an angle of 45°.

(c) Approximate motion parallax vectors can be generated using a technique called affine motion parallax. A group of four tracked points can be thought of as three points A, B and C on a plane, and one point P off the plane. A point Q can be imagined, such that Q is coincident in the image with P but is coplanar with ABC. At the next time instant, the motion of points A, B, C and P can be measured, and the motion of point Q estimated from the motion of A, B and C by assuming that the plane ABC deforms in an affine manner (ie. assume that perspective effects are negligible). The relative image motion of P and Q gives one motion parallax vector. Other groups of four points can generate more motion parallax vectors, allowing the direction of translation to be estimated.

Assessors’ remarks: This question tested the candidates’ understanding of structure and ego-motion recovery from motion parallax vectors. The question was answered fairly well by the few candidates who attempted it. Most could derive the expression for the position of the focus of expansion in (a) and the two parallax vectors in (b). Many could deduce the position of the focus of expansion from the parallax vectors, but very few could deduce more about the aircraft’s motion or the scene structure in (b). The vast majority of attempts included comprehensive descriptions of the affine motion parallax technique in (c).

6. Applications

(a) Before the camera is rotated, assume, without loss of generality, that the camera is aligned with the world coordinate system and hence

\[
\tilde{w} = C \begin{bmatrix} I & O \end{bmatrix} \tilde{X} = C \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = CX
\]

It follows that

\[
X = C^{-1} \tilde{w}
\]

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

\[
\tilde{w}' = C \begin{bmatrix} R & O \end{bmatrix} \tilde{X} = CR \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = CRX = CRC^{-1} \tilde{w}
\]

Hence the relationship between points in the original image and corresponding points in the second image is a plane to plane projectivity. The projectivity can be estimated by observing at least four corresponding points in the two images. Each correspondence gives a constraint of the form

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

$$Ap = 0$$

where $p$ is the $9 \times 1$ 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.

The 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 $I(\mathbf{w})$ in the second image associated with each pixel $\mathbf{w}'$ 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.

(b) The weak perspective model is accurate when all points visible in the scene lie at approximately the same depth from the camera (compared with the viewing distance). This would be a suitable model when fixating on road signs, but not when viewing the road at a larger scale.

Using an affine camera (a linear generalisation of the weak perspective model), the transformation between world and image planes is

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

The transformation can be calibrated using three points. Given a calibrated affine camera, the equations above can be rearranged to recover scene structure:

$$X = \frac{p_{22}u - p_{12}v + p_{33}p_{12} - p_{13}p_{22}}{p_{11}p_{22} - p_{21}p_{12}}$$
$$Y = \frac{p_{11}u - p_{21}u + p_{33}p_{13} - p_{11}p_{23}}{p_{11}p_{22} - p_{21}p_{12}}$$

A scheme to recognise road signs could proceed as follows:
• Extract the borders of the signs by grouping straight line segments in the image. Localise the corners of the sign at the intersections of the line segments.

• Use three of the corners to calibrate the linear affine transformation between the world and image planes (this is possible since we know the dimensions of the sign’s border).

• Use the calibrated camera model to construct a true view of the entire sign (ie. recover the scene structure). In this way we remove viewpoint dependence.

• Compare the true view with each sign in the database. One way to do this would be to threshold the true view to form a black and white image (thus removing illumination effects). If the database contains thresholded true views of all the road signs, we can calculate a correlation coefficient between the thresholded true view and each sign in the database.

• Identify the sign as the one which gives the highest correlation coefficient.

(c) Consider the target viewed under weak perspective at time \( t \). The relationship between its image position \((u, v)\) and its position \((X, Y)\) on the world plane is

\[
\begin{bmatrix}
  u \\
v \\
1
\end{bmatrix} =
\begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & p_{22} & p_{23} \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} = P_t
\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix}, \text{ say}
\]

At time \( t + \Delta t \), the target has moved and the relationship is

\[
\begin{bmatrix}
  u' \\
v' \\
1
\end{bmatrix} =
\begin{bmatrix}
p'_{11} & p'_{12} & p'_{13} \\
p'_{21} & p'_{22} & p'_{23} \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} = P_{t+\Delta t}
\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix}, \text{ say}
\]

The relationship between the two images is therefore

\[
\begin{bmatrix}
  u' \\
v' \\
1
\end{bmatrix} = P_{t+\Delta t}
\begin{bmatrix}
X \\
Y \\
1
\end{bmatrix} = P_{t+\Delta t} P_t^{-1}
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} = P
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}, \text{ say}
\]

The bottom row of \( P \) must be \([0 \ 0 \ 1]\) for the above equation to hold. Hence, the target’s image undergoes a 2D affine transformation as the target moves. Assuming the target is outlined by high contrast features, it can be tracked using an affine snake as follows:

(a) Initialise a B-spline near the edge.

(b) Select a number of evenly spaced sample points along the B-spline.

(c) From each sample point, search normal to the spline for an edge in the image (using standard edge detection techniques).
(d) Apply an elastic “force” at each sample point, proportional to the distance to the edge.

(e) Calculate the elastic energy $E$ associated with the forces.

(f) Move the control points to minimize $E$ (least squares).

(g) Project the control points onto the subspace of allowable affine deformations. Use orthogonal projection to find the closest valid transformation.

(h) Repeat from (c).

The algorithm converges to produce a spline which closely follows the edge in the image and only ever deforms in an affine manner. By running the algorithm continuously, if the target moves, then the spline moves with it. The affine constraint improves resilience to noise and background clutter.

(d) The mapping from the robot table to the image can be modelled as a planar projective transformation:

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

The transformation can be calibrated by observing four points whose world coordinates are known. For high accuracy, these points should span the entire playing area: the corners of the robot table would be ideal. Each point gives us two linear equations in the eight unknowns. If more points are available, then the calibration can be improved using orthogonal least squares.

Once we have calibrated the camera, given the image position $(u, v)$ of a point on the robot we can uniquely determine the position of the point on the play area. For a plane to plane projectivity, we have

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{bmatrix} \lambda X \\ \lambda Y \\ \lambda \end{bmatrix}$$
\[
\begin{bmatrix}
\lambda X \\
\lambda Y \\
\lambda
\end{bmatrix} =
\begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & p_{22} & p_{23} \\
p_{31} & p_{32} & 1
\end{bmatrix}^{-1}
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}
\]

\[
\Leftrightarrow X = \frac{p_{11}^i u + p_{12}^i v + p_{13}^i}{p_{31}^i u + p_{32}^i v + p_{33}^i}, \quad Y = \frac{p_{21}^i u + p_{22}^i v + p_{23}^i}{p_{31}^i u + p_{32}^i v + p_{33}^i}
\]

We would have to observe two points on the robot to completely fix its position and orientation on the play table. If we require a robust and fully automatic system, these points must be easy to detect in the image. We could, perhaps, attach two bright LEDs to the robot and track them using a corner detection algorithm. Alternatively, we could attempt to track the edges of the robot (or the edges of a high-contrast pattern painted onto the robot) using a “snake”. The intersections of pairs of edges would then give us the points we require to localise the robot. Since the scene is likely to be reasonably static, with the exception of the moving robot, the robot could be detected in the first instance by time-differencing consecutive images.

**Assessors’ remarks:** This question was answered very well by many of the candidates, and very poorly by others who seemed to regard it as a last resort option. The good solutions were characterised by an ability to distinguish the important from the trivial, backed up by a sound understanding of the theory to be applied. The poor solutions often rambled on about completely irrelevant topics.
Module I12: Computer Vision and Robotics

Solutions to 2000 Tripos Paper

1. Feature detection

(a) An edge is an image feature with discontinuity in intensity in one direction. The directional derivative of the image intensity is small along an edge but large (and locally maximum) normal to the edge.

The first stage of most edge detection algorithms is to smooth the image $I(x,y)$ by convolution with a 2D Gaussian kernel $G_\sigma(x,y)$:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

In practice, the intensity of a smoothed pixel is computed by discrete convolution:

$$S(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} G_\sigma(i,j)I(x-i,y-j)$$

For computational efficiency, the 2D convolution can be decomposed into two 1D convolutions:

$$G_\sigma(x,y) * I(x,y) = g_\sigma(x) * [g_\sigma(y) * I(x,y)]$$

where $g_\sigma(x)$ is a 1D Gaussian kernel:

$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

The computation speedup offered by the 1D option is $n/2$. $n$ is usually chosen so that the truncated values are less than 0.1% of the central value. For $\sigma = 1$ pixel, a suitably truncated 1D Gaussian kernel is 7 pixels wide ($n = 3$).

The next step is to find the gradient of the smoothed image $S(x,y)$ at every pixel. This can be achieved by convolving $S(x,y)$ with the kernel $[1 - 1]$ in the $x$ and $y$ directions. The resulting gradient estimate applies half way between the pair of pixels being convolved.

(b) A corner, or feature of interest, is an image feature with distinguishable structure in two dimensions. It corresponds to a peak in the local autocorrelation function of the image. Corners can be detected efficiently using the eigenvalue approach outlined below.

The rate of change of intensity $I$ in the direction $\mathbf{n}$ is found by taking the scalar product of $\nabla I$ and $\hat{\mathbf{n}}$:

$$I_n \equiv \nabla I(x,y) \cdot \hat{\mathbf{n}} \Rightarrow I_n^2 = \frac{\mathbf{n}^T \nabla I \nabla I^T \mathbf{n}}{\mathbf{n}^T \mathbf{n}} = \frac{\mathbf{n}^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \mathbf{n}}{\mathbf{n}^T \mathbf{n}}$$
where $I_x \equiv \partial I/\partial x$, etc. Next we smooth $I_n^2$ by convolution with a Gaussian kernel:

$$C_n(x, y) = G_\sigma(x, y) * I_n^2 = \frac{n^T \begin{bmatrix} \langle I_n^2 \rangle & \langle I_n I_y \rangle \\ \langle I_n I_y \rangle & \langle I_n^2 \rangle \end{bmatrix} n}{n^T n}$$

where $\langle \rangle$ is the smoothed value. The smoothed change in intensity in direction $n$ is therefore given by

$$C_n(x, y) = \frac{n^T A n}{n^T n}$$

where $A$ is the $2 \times 2$ matrix

$$\begin{bmatrix} \langle I_n^2 \rangle & \langle I_n I_y \rangle \\ \langle I_n I_y \rangle & \langle I_n^2 \rangle \end{bmatrix}$$

Elementary eigenvector theory tells us that

$$\lambda_1 \leq C_n(x, y) \leq \lambda_2$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $A$. So, if we try every possible orientation $n$, the maximum change in intensity we will find is $\lambda_2$, and the minimum value is $\lambda_1$.

We can classify image structure at each pixel by looking at the eigenvectors of $A$:

**No structure:** (smooth variation) $\lambda_1 \approx \lambda_2 \approx 0$

**1D structure:** (edge) $\lambda_1 \approx 0$ (direction of edge), $\lambda_2$ large (normal to edge)

**2D structure:** (corner) $\lambda_1$ and $\lambda_2$ both large

It is necessary to calculate $A$ at every pixel and mark corners where the quantity $\lambda_1 \lambda_2 - \kappa (\lambda_1 + \lambda_2)^2$ exceeds some threshold ($\kappa \approx 0.04$ makes the detector a little “edge-phobic”). Note that $\det A = \lambda_1 \lambda_2$ and trace $A = \lambda_1 + \lambda_2$, so the required eigenvalue properties can be obtained directly from the elements of $A$.

Assessors’ remarks: The bookwork component produced reasonable answers. Most candidates gave correct expressions for convolution and differentiation in edge detection but made errors in the smoothing kernels required for corner detection.

2. Camera calibration and vanishing points

(a) The relationship is valid under the assumption that the image is formed by a “pinhole camera”, such that rays pass through a single point (the optical centre) before striking the image plane. The relationship does not account for nonlinear distortion, which affects all real cameras to some extent.

(b) Each reference point gives two linear equations in the unknown elements $p_{ij}$:

$$u_i = \frac{su_i}{s} = \frac{p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}}$$

$$\Leftrightarrow 0 = p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14} - u_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34})$$

$$v_i = \frac{sv_i}{s} = \frac{p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}}{p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}}$$

$$\Leftrightarrow 0 = p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24} - v_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34})$$

42
Since the projection matrix operates on homogeneous coordinates, it can be estimated only up to scale (eleven degrees of freedom). We therefore require a minimum of six reference points to perform the calibration. More reference points could be used in a least squares framework to improve the estimate. [30%]

(c) The relationship can be rewritten as follows

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3
\end{bmatrix} =
\begin{bmatrix}
  p_{11} & p_{12} & p_{13} & p_{14} \\
  p_{21} & p_{22} & p_{23} & p_{24} \\
  p_{31} & p_{32} & p_{33} & p_{34}
\end{bmatrix}
\begin{bmatrix}
  X_1 \\
  X_2 \\
  X_3 \\
  X_4
\end{bmatrix}
\]

where \( u = x_1 / x_3, \ v = x_2 / x_3, \ X = X_1 / X_4, \ Y = X_2 / X_4 \) and \( Z = X_3 / X_4 \). Points at infinity in the world can be expressed by setting \( X_4 \) to zero, and points at infinity in the image plane can be expressed by setting \( x_3 \) to zero. [20%]

(d) Distant points on lines parallel to the world \( X \)-axis can be represented in homogeneous coordinates as

\[
\tilde{X} =
\begin{bmatrix}
  1 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

Applying the projection matrix \( P \), we find the image of these points is

\[
\begin{bmatrix}
  s u \\
  s v \\
  s 
\end{bmatrix} =
\begin{bmatrix}
  p_{11} & p_{12} & p_{13} & p_{14} \\
  p_{21} & p_{22} & p_{23} & p_{24} \\
  p_{31} & p_{32} & p_{33} & p_{34}
\end{bmatrix}
\begin{bmatrix}
  1 \\
  0 \\
  0 \\
  0
\end{bmatrix} =
\begin{bmatrix}
  p_{11} \\
  p_{21} \\
  p_{31}
\end{bmatrix}
\]

The vanishing point of lines which are parallel to the world \( X \)-axis is therefore \((u, v) = (p_{11} / p_{31}, p_{21} / p_{31})\). Similarly, the vanishing point of lines which are parallel to the world \( Y \)-axis is \((u, v) = (p_{12} / p_{32}, p_{22} / p_{32})\), and the vanishing point of lines which are parallel to the world \( Z \)-axis is \((u, v) = (p_{13} / p_{33}, p_{23} / p_{33})\).

Note that the projection matrix can be written in the form

\[
P = [KR|KT]
\]

where \( R \) is the rotation matrix between camera and world coordinates, \( T \) is the translation vector between camera and world coordinates, and \( K \) is the \( 3 \times 3 \) matrix of the camera’s intrinsic parameters:

\[
K =
\begin{bmatrix}
  f & 0 & u_0 \\
  0 & f & v_0 \\
  0 & 0 & 1
\end{bmatrix}
\]

Since the vanishing points do not depend on \( KT \) (the fourth column of \( P \)), it follows that they do not depend on the position of the camera. [40%]

Assessors’ remarks: This question was mainly bookwork and was very well answered. Many candidates clearly demonstrated they had understood camera models and calibration. In part (c) most candidates struggled to represent points at infinity in image and world although they were able to use alternative techniques to derive the vanishing point.
3. Planar projective transformations

(a) (i) When the camera is viewing a plane, the relationship between pixels and world positions is given by

\[
\begin{pmatrix}
  su \\
  sv \\
  s
\end{pmatrix}
= 
\begin{pmatrix}
  p_{11} & p_{12} & p_{13} \\
  p_{21} & p_{22} & p_{23} \\
  p_{31} & p_{32} & p_{33}
\end{pmatrix}
\begin{pmatrix}
  X \\
  Y \\
  1
\end{pmatrix}
\]

or \( \hat{w} = P\tilde{X}^p \) for short. For a second image of the same point, we have \( \hat{w}' = P'\tilde{X}^p \).

It follows that \( \hat{w}' = P'P^{-1}\hat{w} = T\hat{w} \), where \( T = P'P^{-1} \) is a 3 × 3 matrix. Hence the relationship between points in the original image and corresponding points in the second image is a 2D projective transformation. [20%]

(ii) Assume, without loss of generality, that before the camera is rotated, the camera is aligned with the world coordinate system and hence

\[
\hat{w} = K \begin{bmatrix} 1 & | & 0 \end{bmatrix} \begin{pmatrix} X \\
  Y \\
  Z \\
  1
\end{pmatrix} = K \begin{pmatrix} X \\
  Y \\
  Z
\end{pmatrix} = KX
\]

where \( K \) is the 3 × 3 matrix of intrinsic camera parameters:

\[
K = \begin{bmatrix}
  f k_u & 0 & u_0 \\
  0 & f k_v & v_0 \\
  0 & 0 & 1
\end{bmatrix}
\]

It follows that \( X = K^{-1}\hat{w} \)

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

\[
\hat{w}' = K \begin{bmatrix} R & | & 0 \end{bmatrix} \begin{pmatrix} X \\
  Y \\
  Z \\
  1
\end{pmatrix} = KR \begin{pmatrix} X \\
  Y \\
  Z
\end{pmatrix} = KRX = KRK^{-1}\hat{w} = T\hat{w}
\]

where \( T = KRK^{-1} \). Hence the relationship between points in the original image and corresponding points in the second image is a 2D projective transformation. [20%]

(b) Since the transformation operates on homogeneous coordinates, the overall scale of the transformation matrix does not matter and we could, for instance, set \( t_{33} \) to 1. The transformation therefore has 8 degrees of freedom.

The image of a square could take any of the forms shown on the next page.
(c) The equation of the line in the first image is $l^T \tilde{w} = 0$, where $l = [l_1 \ l_2 \ l_3]^T$. Since $\tilde{w} = T^{-1} \tilde{w}'$, it follows that the equation of the line in the second image is $l^T T^{-1} \tilde{w}' = 0$, or simply $l' = T^{-T} l$. \[20\%\]

(d) The equation of the conic in the first image is $\tilde{w}^T C \tilde{w} = 0$, where

$$C \equiv \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

Using again the relationship $\tilde{w} = T^{-1} \tilde{w}'$, we find the equation of the corresponding conic in the second image as follows:

$$ (T^{-1} \tilde{w}')^T C T^{-1} \tilde{w}' = 0 \iff \tilde{w}'^T T^{-T} C T^{-1} \tilde{w}' = 0 $$

Alternatively, the conic in the second image can be expressed simply as $C' = T^{-T} C T^{-1}$. \[20\%\]

**Assessors’ remarks:** Part (a)(i) was very poorly answered despite being a minor variation on a problem seen in lectures. Only two candidates were able to derive the equation of the line and conic after the projective transformation in (c) and (d).

4. **Stereo vision**

(a) Matching constraints which can be used to find point correspondences in stereo vision include the epipolar constraint, the ordering constraint, the uniqueness constraint, the disparity gradient limit and figural continuity.

A typical algorithm for finding a large number of matches between left and right images proceeds as follows:

**Unguided matching.** Local search and normalized cross-correlation of strong features results in a small number of seed matches.

**Compute the epipolar geometry.** The seed matches are used to estimate the fundamental matrix $F$. Since many of the seed matches are likely to be erroneous, some sort of robust regression is preferred to least squares. One approach
is to find an $F$ which is consistent with a significant subset of the seed matches, and reject the rest as outliers.

**Guided matching.** Now that $F$ is known, the search for further matches can be restricted to a narrow band around the epipolar lines. Other constraints (ordering, figural continuity, etc.) can be used to choose between mutually exclusive matches which satisfy the epipolar constraint.

(b) (i) The determinant of the estimated fundamental matrix is not zero, so $F$ has full rank and not the expected rank 2. It will be impossible to find an epipole lying in the null space of $F$ (since $F$ has no null space), and the epipolar lines will not meet at a point.

The error could have arisen as a consequence of a number of different factors:

- Inaccurate localisation of point features in the left and right images.
- Incorrect matching of point features.
- The cameras are not well described by the pinhole model (for example, they exhibit significant radial distortion).
- Any combination of the above.

In this particular example, the determinant of the matrix is near zero, so it is likely that no gross matching errors have occurred. The estimate may be improved by finding a rank 2 matrix which is very similar to the original estimate. This can be achieved by orthogonal projection of the original estimate onto the subspace of rank 2 matrices.

The original estimate might also be improved by obtaining more point correspondences and using them in a linear least squares technique to re-estimate $F$.

(ii) The fundamental matrix has an upper left $2 \times 2$ sub-matrix of zeros, which is indicative of an affine stereo scenario where the epipoles are at infinity. Such a situation can arise in two distinct ways:

- The cameras have large focal lengths and are viewing the scene from some distance. The vector joining the optical centres of the two cameras is not nearly parallel to either optical axis. In this case, the epipoles are finite but a long way from the optical centres, and the epipolar lines are nearly parallel. It is quite possible that, within the bounds of experimental error, the estimated fundamental matrix has an upper left $2 \times 2$ sub-matrix of zeros.
- The vector joining the optical centres of the two cameras is parallel to both image planes. In this case, both epipoles will be at infinity, regardless of the cameras’ focal lengths.

(iii) Recovery of metric structure requires knowledge of the cameras’ intrinsic parameters. Given these parameters, we can calculate the essential matrix $E$ from the fundamental matrix $F$, and then deduce from $E$ the relative translation and rotation of the two cameras. Metric structure can then be recovered by triangulation. Without the intrinsic parameters, we can only recover structure up to a 3D projective transformation.
Assessors’ remarks: This question tested the candidates’ understanding of stereo vision and the fundamental matrix. Part (a) was answered fairly well, with most candidates able to list four matching constraints and outline a plausible matching algorithm. Most candidates spotted that the matrix in (b)(i) was not rank 2, though most were far too vague about why this might be, assuming that the word “noise” was all that was required. Answers to (b)(ii) were generally poor, with few candidates spotting the affine fundamental matrix. Similarly few candidates were able to pick out the key stages of the metric reconstruction procedure in (b)(iii).

5. Structure from motion and ego-motion recovery

(a) In the image motion equation, the camera’s linear velocity $U$ always appears scaled by the distance $Z_c$ to the image point. Thus, it is only possible to recover the ratio of these two quantities, and not their absolute values. A camera moving with velocity $U$ through a scene $Z_c(x, y)$ would give rise to exactly the same motion field as a camera moving with velocity $\lambda U$ through a scene $\lambda Z_c(x, y)$. This is the speed-scale ambiguity. [10%]

(b) There are two distinct ways to estimate the image motion field $\dot{\mathbf{p}}$. The first is to track distinguished features, leading to a sparse estimate of $\dot{\mathbf{p}}$ at image locations where there are strong features. Feature tracking algorithms exploit spatiotemporal coherence: features should trace out smooth trajectories in the image over time. Spatiotemporal coherence will be lost if the sampling rate is not sufficient to disambiguate the motion (aliasing). The PAL frame rate of 25Hz is usually sufficient, as long as the density of features in the image is not too high.

Both edges and corners can be tracked in a Kalman filter framework. The filter predicts the feature’s position in each frame and also the uncertainty in the prediction. The uncertainty can be used to choose an appropriate search window. Corner tracking gives a full estimate of the image motion field $\dot{\mathbf{p}}$, while edge tracking can only estimate the component of motion perpendicular to the edge: this is the aperture problem.

The second way to estimate the image motion field is to use an intensity-based optical flow approach. If we assume that any change in a pixel’s intensity $I$ over time is a result of translation of the local intensity distribution, then we can derive the following constraint on the image motion field $\mathbf{v} \equiv (\dot{x}, \dot{y})$:

$$\frac{\partial I}{\partial t} + \nabla I \cdot \mathbf{v} = 0$$

Since we can measure $\partial I/\partial t$ and $\nabla I$, we have a single constraint on the two elements of $\mathbf{v}$. To be more precise, we have an estimate of the image flow in the direction of the image intensity gradient. This is another manifestation of the aperture problem. $\mathbf{v}$ estimated in this way is known as the optical flow. It is only an approximation to the underlying image velocity, since in reality changes in a pixel’s intensity are not totally determined by translation of the local intensity pattern.

Given the single constraint, several techniques exist to estimate the full flow $\mathbf{v}$ across the whole image. The techniques usually involve an expensive optimisation to find
a flow field $\mathbf{v}$ which is consistent with the constraint and also smooth (except at occluding edges, where jumps in $\mathbf{v}$ are allowed). The result is a dense estimate of $\mathbf{v}$ at every image pixel. However, the expense of the optical flow approach usually precludes its use for real-time vision systems, where the feature tracking alternative is preferred. [40%]

(c) For the case of pure translation, the image motion equation simplifies to

$$\dot{x} = -\frac{fU_1}{Z_c} + \frac{xU_3}{Z_c}, \quad \dot{y} = -\frac{fU_2}{Z_c} + \frac{yU_3}{Z_c}$$

where $\mathbf{U} \equiv [U_1 \ U_2 \ U_3]^T$. If $(\dot{x}, \dot{y}) = (5, 0)$ mm/sec at all points in the image, it follows that $U_3 = U_2 = 0$ and $fU_1/Z_c = -5$ mm/sec at all points in the image. The camera is therefore translating in the $x$ direction in front of a wall which is parallel to the image plane (since $Z_c$ is constant). It is not possible to deduce anything further about the camera’s focal length, the camera’s speed or the distance to the wall. [20%]

(d) For the case of pure rotation, the image motion equation simplifies to

$$\dot{x} = -f\Omega_2 + y\Omega_3 + \frac{xy}{f}\Omega_1 - \frac{x^2}{f}\Omega_2, \quad \dot{y} = f\Omega_1 - x\Omega_3 - \frac{xy}{f}\Omega_2 + \frac{y^2}{f}\Omega_1$$

where $\mathbf{\Omega} \equiv [\Omega_1 \ \Omega_2 \ \Omega_3]^T$. The only way to get an approximately constant image motion field is to assume a panning rotation ($\Omega_1 = \Omega_3 = 0$) with a long lens (large focal length $f$). In this case we get

$$\dot{x} = -f\Omega_2 - \frac{x^2}{f}\Omega_2 \approx -f\Omega_2 \quad \text{if } f \text{ is large}$$

and

$$\dot{y} = -\frac{xy}{f}\Omega_2 \approx 0 \quad \text{if } f \text{ is large}$$

We can therefore deduce that $f\Omega_2 = -5$ mm/sec and $f$ is large. It is not possible to deduce anything about the structure of the scene. [30%]

Assessors’ remarks: This question tested the candidates’ understanding of feature tracking, optical flow and the image motion equations. Almost all candidates were able to explain the speed-scale ambiguity in (a). Most gave acceptable accounts of feature tracking and optical flow in (b), though a few launched into irrelevant descriptions of structure-from-motion techniques. While most candidates were able to deduce the camera motion in (c), very few could repeat this feat in (d), even though panning rotation was analysed in detail in the lecture handouts.

6. Applications

(a) Before the camera is rotated, assume, without loss of generality, that the camera is aligned with the world coordinate system and hence

$$\mathbf{\bar{w}} = K \begin{bmatrix} 1 \ | \ O \end{bmatrix} \mathbf{\bar{X}} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KX$$
It follows that $X = K^{-1}\tilde{w}$. After rotating by $R$ about the optical centre, the same world point $X$ projects to a different image point $\tilde{w}'$ as follows:

$$\tilde{w}' = K\begin{bmatrix} R & O \end{bmatrix} \tilde{X} = KR \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = KRX = KRK^{-1}\tilde{w}$$

Hence the relationship between points in the original image and corresponding points in the second image is a plane to plane projectivity. The projectivity 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 $Ap = 0$, where $p$ is the $9 \times 1$ 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.

It is possible to recover the camera’s intrinsic parameters and the angle of rotation. Consider the projectivity between views $i$ and $j$: $P_{ij} = KR_{ij}K^{-1}$. Rearranging gives $R_{ij} = K^{-1}P_{ij}K$, and also we know that $R_{ij}R_{ij}^T = I$, since rotation matrices are orthogonal. This observation allows us to eliminate the unknown rotations:

$$(K^{-1}P_{ij}K)(K^TP_{ij}K^{-T}) = I \quad \Leftrightarrow \quad P_{ij}(KK^T)P_{ij} = KK^T$$

These linear equations in the unknown matrix $KK^T$ can be solved across a number of views (in fact, at least five images are required). $K$ can then be deduced from $KK^T$, and the rotations can subsequently be recovered. \[50\%\]

(b) The change in apparent area of the approaching target can be used to estimate the time to contact with the camera: this in turn can be used by the controller to adjust the robot’s approach speed. The approaching target must be detected and then tracked for a period of time, long enough to estimate the divergence of the image velocity field around the target. The tracking could be achieved using a B-spline snake: as the snake deforms, the image divergence can be estimated from the change in the snake’s area: $\nabla v \approx \dot{a}(t)/a(t)$. The time to contact is then given by $2/\nabla v$. We are assuming a pinhole camera model and constant velocity of the robot. The time to contact estimate also assumes that the looming target appears near the centre of the image. \[50\%\]
(c) The models can be acquired from sample images by extracting edges, identifying
distinguished points (eg. bitangents to concavities and points of inflection) and
mapping the part’s outline into a canonical frame: the canonical frame signature
is the model for that part. Invariance to lighting conditions is achieved through
dge extraction, viewpoint invariance comes from the use of canonical frames. For
cluttered scenes, perceptual grouping can be used to reduce the number of curve
segments to process.

(d) From the projection matrix and considering the points at infinity corresponding
to the three orthogonal directions, we can derive simple constraints on the elements
of the projection matrix:

\[
\begin{bmatrix}
\lambda_1 u_1 & \lambda_2 u_2 & \lambda_3 u_3 \\
\lambda_1 v_1 & \lambda_2 v_2 & \lambda_3 v_3 \\
\lambda_1 & \lambda_2 & \lambda_3
\end{bmatrix} = P
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]  \hspace{1cm} (12)

where \( \lambda_i \) are initially unknown scaling factors. This equation can be rearranged
and expressed in terms of the camera calibration matrix \( K \) and the camera orientation
(rotation matrix), \( R \):

\[
\begin{bmatrix}
u_1 & u_2 & u_3 \\
v_1 & v_2 & v_3 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
\lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3
\end{bmatrix} = KR
\]  \hspace{1cm} (13)

By exploiting the properties of the rotation matrix \( R \), we can rearrange (13) to
recover constraints on the intrinsic parameters of the camera and the unknown
scaling parameters \( \lambda_i \). In particular:

\[
\begin{bmatrix}
u_1 & u_2 & u_3 \\
v_1 & v_2 & v_3 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
\lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3
\end{bmatrix} = KK^T
\]  \hspace{1cm} (14)

Under the assumption of known aspect ratio and zero skew, (14) can be rewritten as
six linear equations (from six elements of the symmetric matrix) and can be solved
to recover three intrinsic parameters and the unknown scale factors \( \lambda^2 \).

The solution of (14) and substitution into (13) leads to the recovery of the 3 \( \times \) 3
sub-matrix of the projection matrix, \( KR \), which can then be easily decomposed to
obtain the rotation matrix \( R \).

The fourth column of the projection matrix depends on the position of the world
coordinate system relative to the camera coordinate system. An arbitrary reference
point can be chosen as the origin. Its image coordinates fix the translation \( T \) up to
an arbitrary scale factor \( \lambda_4 \):

\[
\lambda_4
\begin{bmatrix}
u_4 \\
v_4
\end{bmatrix} = P
\begin{bmatrix}
p_{11} & p_{12} & p_{13} & p_{14} \\
p_{21} & p_{22} & p_{23} & p_{24} \\
p_{31} & p_{32} & p_{33} & p_{34}
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix} = KT
\]  \hspace{1cm} (15)
In a single view and with no metric information, this scale is indeterminate and can be arbitrarily set, e.g. \( \lambda_4 = 1 \). For additional views, the image correspondences of a fifth point are required to fix this scale factor. This is equivalent to fixing the epipoles from the translational component of image motion under known rotation — two point correspondences are required to recover the direction of translation.

Reconstruction proceeds from the two projection matrices and point correspondences. First we choose three features in the left image and record image coordinates \((u, v)\). Next we find correspondences \((u', v')\) in the right view (we can use the epipolar lines to help). Finally, we recover 3D positions by triangulation.

The texture of each surface polygon is represented as bitmaps. The bitmap is obtained from one of the images (the least foreshortened one) and projectively warped to be fronto-parallel for loading into the VRML browser. The browser then warps the texture appropriately when viewing the model.

[50%]

Assessors’ remarks: This question was answered very well by many of the candidates, and very poorly by others who seemed to regard it as a last resort option. The good solutions were characterised by an ability to distinguish the important from the trivial, backed up by a sound understanding of the theory to be applied. The poor solutions often rambled on about completely irrelevant topics.

Andrew Gee & Roberto Cipolla
January 2000
1. Feature detection

(a) Let \( I(x, y) \) denote the image, and \( G_\sigma(x, y) \) denote the 2D Gaussian kernel:

\[
G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

(i) In practice, the intensity of a smoothed pixel is computed by discrete convolution:

\[
S(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} G_\sigma(i, j)I(x - i, y - j)
\]

[10%]

(ii) The 2D convolution can be decomposed into two 1D convolutions as follows:

\[
G_\sigma(x, y) * I(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} g_\sigma(i)I(x - i, y - j) g_\sigma(j) = g_\sigma(x) * [g_\sigma(y) * I(x, y)]
\]

where \( g_\sigma(x) \) is a 1D discrete approximation to the Gaussian kernel:

\[
g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)
\]

We would require a \((2n+1)\) pixel kernel for the 1D convolutions, or a \((2n+1) \times (2n+1)\) pixel kernel for the 2D convolutions. Each 2D convolution requires \((2n+1) \times (2n+1)\) multiply and accumulate operations, while each 1D convolution requires \((2n+1)\) multiply and accumulate operations. The speedup offered by the 1D option is therefore \((2n+1)/2\) times. [20%]

(iii) The kernel is usually truncated so that the discarded values are less than \(1/1000\) of the peak value. If we discard the sample \((n+1)\) pixels from the centre of the kernel, the size of the kernel will be \(2n+1\) pixels. We can find \(n\) by solving:

\[
\exp\left[-\frac{(n+1)^2}{2\sigma^2}\right] < \frac{1}{1000} \iff n > 3.7\sigma - 1
\]

For \(\sigma = 1\) we get \(n > 2.7\). Rounding up to the nearest integer we find that \(n = 3\) and the size of the kernel is 7 pixels. The coefficients are: [30%]

| 0.004 | 0.054 | 0.242 | 0.399 | 0.242 | 0.054 | 0.004 |

(b) The Marr–Hildreth operator convolves the image with a discrete version of the Laplacian of a Gaussian and then localises edges at the resulting zero-crossings.
Canny operator is a directional edge finder. It first localises the orientation of the edge by computing
\[ \hat{n} = \frac{\nabla (G_\sigma(x, y) * I(x, y))}{|\nabla (G_\sigma(x, y) * I(x, y))|} \]
and then searches for a local maximum of \( |\nabla (G_\sigma * I)| \) in the direction \( \hat{n} \). This is equivalent to finding zero-crossings in the directional second derivative of \( (G_\sigma * I) \) in the direction \( \hat{n} \).

The principle advantage of the Marr-Hildreth operator is computational efficiency: edge detection requires only a single convolution and the detection of zero-crossings. Conversely, the Canny operator requires an additional, costly search for a local maximum normal to the gradient direction. The advantage of the Canny operator is enhanced robustness to noise. Any differential operator amplifies noise. The Canny operator computes only first derivatives and then searches for a local maximum (which is equivalent to a zero-crossing of the second derivative) normal to the gradient. The Marr-Hildreth operator computes second derivatives both along and normal to the edge. Computation of the second derivative along the edge emphasizes noise in that direction while serving no purpose in edge detection.

Assessors’ remarks: A reasonably simple question with some very good answers. Many candidates did not give correct expressions for discrete convolution.

2. Geometric transformations and vanishing points
(a) (i) Parallel planes meet at lines in the image, often referred to as horizon lines. To prove this, consider a plane in 3D space defined as follows:
\[ X_c.n = d \]
where \( n = (n_x, n_y, n_z) \) is the normal to the plane. We can analyse horizon lines by writing the perspective projection in the following form:
\[ \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{fX_c}{Z_c} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \]
Taking the scalar product of both sides with \( n \) gives:
\[ \begin{bmatrix} x \\ y \\ f \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{fX_c.n}{Z_c} = \frac{fd}{Z_c} \]
As \( Z_c \to \infty \) we move away from the camera and we find
\[ \begin{bmatrix} x \\ y \\ f \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = 0 \]
Thus the equation of the horizon line is
\[ n_xx + n_yy + fn_z = 0 \]
which depends only on the orientation of the plane, and not its position. Thus a set of parallel planes meet at a horizon line in the image.

(ii) Consider a line parallel to the planes, with direction vector \( \mathbf{b} \), such that \( \mathbf{b} \cdot \mathbf{n} = 0 \):

\[
\mathbf{b} \cdot \mathbf{n} = n_x b_x + n_y b_y + n_z b_z = 0 \\
\Leftrightarrow n_x (f b_x / b_z) + n_y (f b_y / b_z) + f n_z = 0
\]

The vanishing point of the line in the image is \((f b_x / b_z, f b_y / b_z)\). The equation above shows that the vanishing point satisfies the equation of the horizon line.

(b) (i) The image of the circle is an ellipse. Consider the left side of the square. We know the world coordinates of A, the two corners and the vanishing point V1 (infinity). In the image, we know the coordinates of the two corners and V1. We can therefore use a cross-ratio to determine the image coordinates of A. Similar constructions on the other sides give the image coordinates of B, C and D.

However, four points are not sufficient to determine the equation of the ellipse: we require five. A further four points can be obtained by considering the intersections of the square’s diagonals with the circle. Consider the diagonal containing points E and G. In the world, we know the coordinates of the endpoints, the midpoint and E. In the image, we know the coordinates of the endpoints and the midpoint (by intersecting the two diagonals), so we can use a cross-ratio to find E. Again, similar constructions give the image coordinates of F, G and H.

(ii) When the camera is viewing a plane, the relationship between pixels and world positions is given by

\[
\begin{bmatrix}
s u \\ sv \\ s
\end{bmatrix}
= 
\begin{bmatrix}
p_{11} & p_{12} & p_{13} \\
p_{21} & p_{22} & p_{23} \\
p_{31} & p_{32} & p_{33}
\end{bmatrix}
\begin{bmatrix}
X \\ Y \\ 1
\end{bmatrix}
\]
This can be calibrated using the world and image positions of the square’s corners. For the circle, we first find its equation in the world plane in the form $\tilde{X}^T C \tilde{X} = 0$. Using the relationship $\tilde{X} = P^{-1} \tilde{w}$, we find the equation of the corresponding conic in the image as follows:

$$(P^{-1} \tilde{w})^T C P^{-1} \tilde{w}' = 0 \iff \tilde{w}'^T P^{-T} C P^{-1} \tilde{w}' = 0$$

Alternatively, the conic in the image can be expressed simply as $C' = P^{-T} C P^{-1}$. [20%]

**Assessors’ remarks:** A poorly attempted question. The derivation of the horizon was well-answered but candidates struggled with the second part on the use of invariants and an algebraic method to compute the transformed conics.

3. Calibration

(a) There is a rotation $R$ and translation $T$ between the world coordinates $\tilde{X}$ and the camera-centered coordinates $\tilde{X}_c$ (both expressed in homogeneous coordinates).

$$\tilde{X}_c = \begin{bmatrix} R & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tilde{X}$$

Assuming a pinhole camera model, with no nonlinear distortion, the next stage is perspective projection of $\tilde{X}_c$ onto $\tilde{x}$ in the image plane. Assume the focal length is $f$:

$$\tilde{x} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{X}_c$$

$\tilde{x} = (sx, sy, s)$ is the homogeneous representation of the image point $x = (x, y)$. Finally, we have to convert to pixel coordinates $\tilde{w} = (u, v)$. Assume the optical axis intersects the image plane at the pixel with coordinates $(u_0, v_0)$ and there are $k_u$ pixels per unit distance in the $u$ direction and $k_v$ in the $v$ direction:

$$\tilde{w} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{x}$$

where $\tilde{w} = (su, sv, s)$ is the homogeneous representation of $\tilde{w}$. We have assumed that the CCD array is planar and mounted perpendicular to the optical axis.

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

$$\tilde{w} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} \tilde{X} = P \tilde{X}, \ \text{say}$$
P is a $3 \times 4$ matrix, so the process can be expressed as

$$
\begin{bmatrix}
  su \\
  sv \\
  s
\end{bmatrix} = \begin{bmatrix}
  p_{11} & p_{12} & p_{13} & p_{14} \\
  p_{21} & p_{22} & p_{23} & p_{24} \\
  p_{31} & p_{32} & p_{33} & p_{34}
\end{bmatrix} \begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
$$

P can be estimated by observing the images of known 3D points. 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 known 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, in a non-degenerate configuration, to calibrate the camera. In practice, we would use more than 6 points to mitigate the effects of measurement noise.

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

$$Ap = 0$$

where $p$ is the $12 \times 1$ 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 orthogonal least squares solution corresponds to the eigenvector of $A^TA$ with the smallest corresponding eigenvalue.

The linear solution is, however, only approximate, since we have not taken into account the special structure of P. Ideally, the linear solution should be used as the starting point for nonlinear optimization, finding the parameters of the rigid body transformation, perspective projection and CCD mapping that minimize the errors between measured image points $(u_i, v_i)$ and projected (or modeled) image positions $(\hat{u}_i, \hat{v}_i)$:

$$\min_p \sum_i ((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2)$$

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

$$P = \begin{bmatrix}
  fk_u & 0 & u_0 & 0 \\
  0 & fk_v & v_0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
  R \\
  T
\end{bmatrix} = \begin{bmatrix}
  fk_u & 0 & u_0 \\
  0 & fk_v & v_0 \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  R \\
  T
\end{bmatrix}$$

$$= C \begin{bmatrix}
  R \\
  T
\end{bmatrix} = \begin{bmatrix}
  CR \\
  CT
\end{bmatrix}$$
it is apparent that we need to decompose the left $3 \times 3$ sub-matrix of $P$ into an upper triangular matrix $C$ and an orthogonal (rotation) matrix $R$. This can be achieved using QR decomposition. $T$ can then be recovered using

$$T = C^{-1} \begin{bmatrix} p_{14} & p_{24} & p_{34} \end{bmatrix}^T$$

It is not possible to decouple the focal length $f$ from the pixel scale factors $k_u$ and $k_v$. [60%]

(b) Assuming that the road is straight in the vicinity of the tunnel entrance, the mapping from the road to the image can be modeled as a 1D planar projective transformation. For the purpose of recovering structure along a line, we really only need to calibrate for one image coordinate. Given that the road probably runs from the bottom of the image to the top, we would use the image $v$ coordinate and not the $u$ coordinate (which would not vary much for different points on the road):

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

where $X$ is the position of the car on the road. We can calibrate this using the $v$ coordinates of the lane markings and the corresponding known distances on the road. Each calibration point gives us one linear equation in the three unknowns (the overall scale of $P$ does not matter): we therefore require three points. For high accuracy, we should use more calibration points (with least squares estimation of the projection matrix) spanning the entire visible length of the road.

Once we have calibrated the camera, given the image position $(u, v)$ of a car we can use the $v$ coordinate to determine the position of the car on the road. Moreover, we can differentiate the expression relating image to world coordinates to determine the car’s physical velocity from its apparent velocity in the image.

If the road was not straight in the vicinity of the tunnel entrance, we would need to calibrate the plane-to-plane projective transformation between the image $(u, v)$ and the world plane $(X, Y)$. For this, we would need a minimum of four calibration points. Once calibrated, a car’s image position $(u, v)$ could be mapped onto a world position $(X, Y)$, and the image velocity $(\dot{u}, \dot{v})$ could be used to infer the car’s speed. [40%]

Assessors’ remarks: Part (a) was predominantly book work and answered very well by most candidates: only the refinement of the calibration by nonlinear techniques posed problems. Many candidates also came up with good proposals for the practical application in part (b), the most common mistake being to decide on a particular camera model (eg, planar or linear) without suitable justification.

4. Stereo vision

(a) The essential matrix $E$ describes the epipolar geometry of a stereo rig in terms of rays $p = [x \ y \ f]^T$, where $(x, y)$ are the metric image plane coordinates of an observed point and $f$ is the camera’s focal length. It is therefore mostly used when the intrinsic parameters of the cameras are known. The left and right epipoles lie in the null spaces of $E$ and $E^T$ respectively, while the epipolar constraint is given by $p^T E p = 0$. To derive the essential matrix in terms of $R$ and $T$, we start with the
equation relating the two coordinate systems:

\[ X'_c = RX_c + T \Rightarrow T \times X'_c = T \times RX_c \]

\[ \Rightarrow X'_c (T \times RX_c) = 0 \Leftrightarrow \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \]

\[ \Leftrightarrow p'^T[T \times R]p = 0, \] since rays and camera-centered positions are parallel.

The essential matrix is therefore given by \( E = T \times R \). [30%]

(b) (i) To find the essential matrix from the fundamental matrix \( F \), we also need the intrinsic calibration matrices \( K \) and \( K' \) of the left and right cameras, where

\[ K = \begin{bmatrix} f k_u & 0 & u_0 \\ 0 & f k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \]

\( f \) is the focal length, \( k_u \) and \( k_v \) the pixel scale factors and \((u_0, v_0)\) the point where the optical axis intersects the image plane. Then \( E = K'^T FK \). [15%]

(ii) The essential matrix is clearly of the form \( T \times R \) for \( T = [1 \ 0 \ 0]^T \), so one solution is \( R = R_1 = I \) and \( \hat{T} = \hat{T}_1 = [1 \ 0 \ 0]^T \). By inspection, another solution is \( \hat{T} = -\hat{T}_1 \) and \( R = R_2 \), where

\[ R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \]

Since the essential matrix is only defined up to scale, we can also consider \(-E\) to get the remaining two solutions \( \hat{T} = -\hat{T}_1 \) and \( R = R_1 \), and \( \hat{T} = \hat{T}_1 \) and \( R = R_2 \).

In practice, the ambiguity can often be resolved by ensuring that all visible points are in front of the two cameras. In this case, however, the cameras’ image planes are aligned, so the “in front” condition will disambiguate the rotation but not the sign of the translation. However, by observing the image positions of a nearby point, it should be clear which camera is on the left and which on the right. [30%]

(iii) The unknown baseline length means we can recover structure only up to scale. To see this, consider the triangulation equations with an unknown baseline length \( \lambda \):

\[ X'_c = RX_c + \lambda \hat{T} \Rightarrow X'_c \times p' = (RX_c + \lambda \hat{T}) \times p' = 0 \]

\[ \Leftrightarrow 0 = \left( \frac{Z_c}{f} R p + \lambda \hat{T} \right) \times p' \Leftrightarrow \frac{Z_c}{\lambda f} (R p \times p') = -\hat{T} \times p' \]

The unknown depths \( Z_c \) appear scaled by the baseline length \( \lambda \), so if we double \( \lambda \) we also double all the recovered depths \( Z_c \). Since 3D structure is given by \( X_c = (Z_c/f) p \), doubling the depths \( Z_c \) doubles the size of the reconstructed scene. [25%]

Assessors’ remarks: Most were able to derive the essential matrix and relate it to the fundamental matrix. Only about half the candidates could demonstrate
multiple solutions for the decomposition of the essential matrix, and most of these
found only two solutions: the arbitrary sign of the essential matrix was largely
ignored. In the last part of the question, there was a good understanding of the
relationship between the baseline length and the scale of the 3D reconstruction.

5. Visual navigation

(a) For a translating camera, the focus of expansion is the point where a line through
the optical centre parallel to the direction of motion intersects the image plane. At
this point (as well as at all points on the horizon) the translational component of
the image velocity field is zero. The coordinates of the focus of expansion can be
found by solving the image motion equations for \( \dot{x} = 0 \) and \( \dot{y} = 0 \):

\[
-\frac{fu_1 + xu_3}{Z_c} = 0 \Leftrightarrow x_0 = \frac{fu_1}{u_3}
\]

\[
-\frac{fu_2 + yu_3}{Z_c} = 0 \Leftrightarrow y_0 = \frac{fu_2}{u_3}
\]

So the focus of expansion lies at \((f^2u_1u_3, f^2u_2u_3)\). Eliminating
\(u_1\) and \(u_2\) from the motion

\[
\dot{x} = \frac{u_3(x - x_0)}{Z_c}, \quad \dot{y} = \frac{u_3(y - y_0)}{Z_c} \Rightarrow \frac{\dot{x}}{\dot{y}} = \frac{(x - x_0)}{(y - y_0)}
\]

So the motion field is radial with respect to the focus of expansion. [30%]

(b) (i) We can convert pixel coordinates \((u, v)\) to metric image plane coordinates
\((x, y)\) using the CCD scaling equations:

\[
u = u_0 + ku, \quad v = v_0 + kv
\]

In this example, \(k_u = k_v = 50\) pixels/mm and \(u_0 = v_0 = 200\). After conversion,
we find the original focus of expansion was at \((-3.8, -0.4)\) mm, while the new one
is at \((5.8, -0.3)\) mm. Substituting into the formulae for the focus of expansion,
we find that before the change of direction \(U_1/U_3 = -0.475\) and \(U_2/U_3 = -0.05\),
while after the change of direction \(U_1/U_3 = 0.725\) and \(U_2/U_3 = -0.0375\). The
original direction of motion was therefore \([-0.475, -0.05, 1]^T\), while the new one
is \([0.725, -0.0375, 1]^T\). The angle between these two vectors is 61.3°. [20%]

(ii) The two heading vectors must both lie in the ground plane, so we can find the
ground plane normal in camera-centered coordinates by taking the vector product
of the heading vectors:

\[
[0.475, -0.05, 1]^T \times [0.725, -0.0375, 1]^T = [-0.0125, 1.2, 0.054]^T
\]

The optical axis is in the direction \([0, 0, 1]^T\). The angle between the optical axis
and the ground plane normal is 87.4°. The angle between the optical axis and the
horizontal ground plane is therefore 2.58°. [20%]

(iii) There are several problems with tracking the focus of expansion to measure
the angle of a turn. First, finding the focus of expansion involves measuring the
image velocities of several prominent corner features and locating their point of
intersection: this might be ill-conditioned if the focus of expansion is a long way from the visible portion of the image plane. Second, using the focus of expansion in this way requires that the robot does not rotate, which will be difficult to ensure in practice. Finally, the robot needs to move forward along its new path before the angle of turn can be measured and any correction applied.

A better way to control the robot would be to keep the focus of expansion near the centre of the image and allow the robot to rotate to change direction. To measure the angle of a turn, we could track a prominent, stationary corner feature as the robot rotates. Since the camera is calibrated, the image location of the feature can be converted to a ray, whose angle can be tracked through the turn. If the feature should slip out of view, another prominent feature can be located and tracked. Once the robot is pointing in the right direction, it can be kept on a steady course by adjusting the motor speeds should the focus of expansion drift across the image.  

Assessors’ remarks: The few candidates who attempted the question answered it fairly well (with one exception, hence the low average mark), with many gaining full marks for part (a). In part (b), most were able to deduce the change in the robot’s direction, though none found the angle between the optical axis and the horizontal (the most common mistake being to incorrectly label the angle between a line and a plane). In the last part of the question, several candidates came up with sensible, alternative approaches to visual navigation.
1. Feature detection

(a) (i) The first stage of most edge detection algorithms is to smooth the image $I(x, y)$ by convolution with a 2D Gaussian kernel $G_\sigma(x, y)$:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

This is a low pass filter to reduce the effect of additive image noise before differentiation. The latter amplifies high frequencies. The size of the filter determines the scale of finding edges. The larger the kernel the lower the low pass filter cut-off spatial frequency. \[20\%\]

(ii) The intensity of a smoothed pixel is computed by discrete convolution:

$$S(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} G_\sigma(i, j) I(x - i, y - j)$$

The 2D convolution can be decomposed into two 1D convolutions as follows:

$$G_\sigma(x, y) * I(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} g_\sigma(i) g_\sigma(j) I(x - i, y - j) = g_\sigma(x) * [g_\sigma(y) * I(x, y)]$$

where $g_\sigma(x)$ is a 1D discrete approximation to the Gaussian kernel:

$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$ \[20\%\]

(iii) The next step is to find the gradient of the smoothed image $S(x, y)$ at every pixel. This can be achieved by convolving $S(x, y)$ with the kernel $[1 \ -1]$ in the $x$ and $y$ directions. The resulting gradient estimate applies half way between the pair of pixels being convolved. \[10\%\]

(b)(i) The rate of change of intensity $I$ in the direction $\hat{n}$ is found by taking the scalar product of $\nabla I$ and $\hat{n}$:

$$I_n \equiv \nabla I(x, y) \cdot \hat{n} \Rightarrow I_n^2 = \frac{n^T \nabla I \nabla I^T \hat{n}}{n^T \hat{n}} = \frac{n^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \hat{n}}{n^T \hat{n}}$$

where $I_x \equiv \partial I / \partial x$, etc. \[20\%\]
(ii) We smooth $I^2_n$ by convolution with a Gaussian kernel:

$$C_n(x, y) = G_\sigma(x, y) * I^2_n = \frac{n^T \begin{bmatrix} \langle I^2_x \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I^2_y \rangle \end{bmatrix} n}{n^T n}$$

where $\langle \rangle$ is the smoothed value. The smoothed change in intensity in direction $n$ is therefore given by

$$C_n(x, y) = \frac{n^T A n}{n^T n}$$

where $A$ is the $2 \times 2$ matrix

$$\begin{bmatrix} \langle I^2_x \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I^2_y \rangle \end{bmatrix}$$

Elementary eigenvector theory tells us that

$$\lambda_1 \leq C_n(x, y) \leq \lambda_2$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $A$. So, if we try every possible orientation $n$, the maximum change in intensity we will find is $\lambda_2$, and the minimum value is $\lambda_1$.

We can detect a corner by looking at the eigenvectors of $A$. For a (corner) $\lambda_1$ and $\lambda_2$ both large. It is necessary to calculate $A$ at every pixel and mark corners where the quantity $\lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$ exceeds some threshold ($\kappa \approx 0.04$ makes the detector a little “edge-phobic”). Note that $\text{det} A = \lambda_1 \lambda_2$ and $\text{trace} A = \lambda_1 + \lambda_2$, so the required eigenvalue properties can be obtained directly from the elements of $A$. [30%]

Assessors’ remarks: A straightforward question which was extremely popular and answered well by most candidates.

2. Camera models and calibration

(a) [Book work] The relationship is valid under the assumption that the image is formed by a “pinhole camera”, such that rays pass through a single point (the optical centre) before striking the image plane. The relationship does not account for nonlinear distortion, which affects all real cameras to some extent.

Geometrically, $s$ can be thought of as a scale factor, controlling the size of the image formed by an object in the world. It depends on the distance $Z_c$ of the object from the camera.

Algebraically, $s$ and the elements $p_{ij}$ allow the imaging process to be expressed as a linear relationship in homogeneous coordinates. In Cartesian coordinates the perspective image formation process cannot be expressed linearly, requiring a division by $Z_c$. [20%]

(b) The projection matrix can be written in the form

$$P = [K][R][T]$$
where $R$ is the rotation matrix between camera and world coordinates, $T$ is the translation vector between camera and world coordinates, and $K$ is the $3 \times 3$ matrix of the camera’s intrinsic parameters:

$$K = \begin{bmatrix}
fk_u & 0 & u_0 \\
0 & fk_v & v_0 \\
0 & 0 & 1
\end{bmatrix}$$

(A longer answer might extend to the following). There is a rotation $R$ and translation $T$ between the world coordinates $\tilde{X}$ and the camera-centered coordinates $\tilde{X}_c$ (both expressed in homogeneous coordinates).

$$\tilde{X}_c = \begin{bmatrix}
R & T \\
0 & 0 & 0 & 1
\end{bmatrix} \tilde{X}$$

Assuming a pinhole camera model, with no nonlinear distortion, the next stage is perspective projection of $\tilde{X}_c$ onto $\tilde{x}$ in the image plane. Assume the focal length is $f$:

$$\tilde{x} = \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \tilde{X}_c$$

$\tilde{x} = (sx, sy, s)$ is the homogeneous representation of the image point $x = (x, y)$. Finally, we have to convert to pixel coordinates $\tilde{w} = (u, v)$. Assume the optical axis intersects the image plane at the pixel with coordinates $(u_0, v_0)$ and there are $k_u$ pixels per unit distance in the $u$ direction and $k_v$ in the $v$ direction:

$$\tilde{w} = \begin{bmatrix}
k_u & 0 & u_0 \\
0 & k_v & v_0 \\
0 & 0 & 1
\end{bmatrix} \tilde{x}$$

where $\tilde{w} = (su, sv, s)$ is the homogeneous representation of $\tilde{w}$. We have assumed that the CCD array is planar and mounted perpendicular to the optical axis.

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

$$\tilde{w} = \begin{bmatrix}
fk_u & 0 & u_0 \\
0 & fk_v & v_0 \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
R & T \\
0 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix} \tilde{X} = P \tilde{X}, \text{ say}$$

$P$ is a $3 \times 4$ matrix, so the process can be expressed as

$$\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}$$
(c) P can be estimated by observing the images of known 3D points. Each point we observe gives us a pair of equations:

\[
\begin{align*}
u &= \frac{su}{s} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \\
v &= \frac{sv}{s} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}
\end{align*}
\]

Since we are observing a known 3D object, 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, in a non-degenerate configuration, to calibrate the camera. In practice, we would use more than 6 points to mitigate the effects of measurement noise.

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

\[ A \mathbf{p} = \mathbf{0} \]

where \( \mathbf{p} \) is the 12 x 1 vector of unknowns (the twelve elements of P), A is the 2n x 12 matrix of coefficients and \( n \) is the number of observed calibration points. The orthogonal least squares solution corresponds to the eigenvector of \( A^T A \) with the smallest corresponding eigenvalue.

The linear solution is, however, only approximate, since we have not taken into account the special structure of P. Ideally, the linear solution should be used as the starting point for nonlinear optimization, finding the parameters of the rigid body transformation, perspective projection and CCD mapping that minimize the errors between measured image points \((u_i, v_i)\) and projected (or modeled) image positions \((\hat{u}_i, \hat{v}_i)\):

\[
\min_{\mathbf{P}} \sum_i ((u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2)
\]

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

\[
\mathbf{P} = \begin{bmatrix} f k_u & 0 & u_0 & 0 \\ 0 & f k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} f k_u & 0 & u_0 \\ 0 & f k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix}
\]

it is apparent that we need to decompose the left 3 x 3 sub-matrix of P into an upper triangular matrix K and an orthogonal (rotation) matrix R. This can be achieved using QR decomposition. \( \mathbf{T} \) can then be recovered using

\[
\mathbf{T} = K^{-1} \begin{bmatrix} p_{14} & p_{24} & p_{34} \end{bmatrix}^T
\]

It is not possible to decouple the focal length \( f \) from the pixel scale factors \( k_u \) and \( k_v \). [40%]

(d) [Book work] 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. Under these conditions, the relationship between world coordinates and image pixel coordinates can be approximated as follows:

\[
\begin{bmatrix}
    u \\
    v
\end{bmatrix} =
\begin{bmatrix}
    p_{11} & p_{12} & p_{13} & p_{14} \\
    p_{21} & p_{22} & p_{23} & p_{24} \\
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    1
\end{bmatrix}
\]

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. [20%]

Assessors’ remarks: This essay-style, book work question was attempted by all but one of the candidates and was generally well answered. The better solutions exhibited sound editorial judgement: the ability to distinguish the relevant from the irrelevant, and the key points from the trivial details.

3. Planar projective transformations

(a) When the camera is viewing a plane, \( Z = 0 \) (without loss of generality) and the general projective relationship between pixels and world positions is given by

\[
\begin{bmatrix}
    su \\
    sv \\
    s
\end{bmatrix} =
\begin{bmatrix}
    t_{11} & t_{12} & t_{13} \\
    t_{21} & t_{22} & t_{23} \\
    t_{31} & t_{32} & t_{33} \\
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    1
\end{bmatrix}
\]

[20%]

(b) Since the transformation operates on homogeneous coordinates, the overall scale of the transformation matrix does not matter and we could, for instance, set \( t_{33} \) to 1. The transformation therefore has 8 degrees of freedom.

The image of a square could take any of the following forms:

Translation (2 DOF)  Rotation  Scaling
Shear  Stretch  Fanning - equation of horizon line gives 2 DOF

[20%]

(c) (i) Vanishing points corresponding to lines parallel to the \( x \)-axis and \( y \)-axis must lie on the horizon. Distant points on lines parallel to the world \( x \)-axis can
be represented in homogeneous coordinates as \([1 \ 0 \ 0]^T\). The vanishing point of lines which are parallel to the world \(x\)-axis is therefore \((u, v) = (t_{11}/t_{31}, t_{21}/t_{31})\). By a similar argument, the vanishing point of lines which are parallel to the world \(y\)-axis is therefore \((u, v) = (t_{12}/t_{32}, t_{22}/t_{32})\). If the equation of the horizon line is \(l \begin{bmatrix} su & sv & s \end{bmatrix}^T = 0\), then these two points on the line give us two constraints on \(l\):

\[
\begin{align*}
1. [t_{11} & \ t_{21} & t_{31}]^T = 0 \\
1. [t_{12} & \ t_{22} & t_{32}]^T = 0
\end{align*}
\]

Another way of looking at these constraints is that \(l\) is perpendicular to both \([t_{11} \ t_{21} \ t_{31}]^T\) and \([t_{12} \ t_{22} \ t_{32}]^T\). We can therefore find the horizon line by a vector product:

\[
\begin{vmatrix}
u & v & 1 \\
t_{11} & t_{21} & t_{31} \\
t_{12} & t_{22} & t_{32}
\end{vmatrix} = 0
\]

(ii) If the internal parameters, \(K\), are known, we can decompose the planar projective transformation as follows:

\[
\begin{bmatrix} 
    u \\
    v \\
    1 
\end{bmatrix} = K \begin{bmatrix} 
    r_{11} & r_{12} & T_x \\
    r_{21} & r_{22} & T_y \\
    r_{31} & r_{32} & T_z 
\end{bmatrix} \begin{bmatrix} 
    X \\
    Y \\
    1 
\end{bmatrix}
\]

where \(R\) is the rotation between the camera-centered coordinate system and the world plane, and \(T\) is the translation. Substituting the world and image coordinates of the first vanishing point, we have

\[
\begin{bmatrix} 
    t_{11} \\
    t_{21} \\
    t_{31} 
\end{bmatrix} = K \begin{bmatrix} 
    r_{11} & r_{12} & T_x \\
    r_{21} & r_{22} & T_y \\
    r_{31} & r_{32} & T_z 
\end{bmatrix} \begin{bmatrix} 
    1 \\
    0 \\
    0 
\end{bmatrix} = K \begin{bmatrix} 
    r_{11} \\
    r_{21} \\
    r_{31} 
\end{bmatrix}
\]

So we can recover the first column of the rotation matrix \(R\) as follows:

\[
\begin{bmatrix} 
    r_{11} \\
    r_{21} \\
    r_{31} 
\end{bmatrix} = K^{-1} \begin{bmatrix} 
    t_{11} \\
    t_{21} \\
    t_{31} 
\end{bmatrix}
\]

Similarly, the second column of the rotation matrix is given by

\[
\begin{bmatrix} 
    r_{12} \\
    r_{22} \\
    r_{32} 
\end{bmatrix} = K^{-1} \begin{bmatrix} 
    t_{12} \\
    t_{22} \\
    t_{32} 
\end{bmatrix}
\]

The third column of the rotation matrix follows from the fact that the matrix is orthogonal and has a determinant of plus one.

Alternatively, the orientation of the plane with respect to the camera (and hence the orientation of the camera with respect to the plane) can be recovered directly from the equation of the horizon line. First, we use the internal parameters to express the horizon line in terms of metric image plane coordinates \((x, y)\). Then, it is straightforward to show that the equation of the horizon line is

\[
n_xx + n_yy + fn_z = 0
\]
where \( \mathbf{n} = (n_x, n_y, n_z) \) is the normal to the plane in camera-centered coordinates, and \( f \) is the camera’s focal length (which is known). We can therefore recover \( \mathbf{n} \) from the equation of the horizon line. \( \mathbf{n} \) is all we need to know to describe the camera’s orientation with respect to the plane.

(d) We can derive a relationship for the transformation of lines by considering the relationship between points. Points on a line in the world satisfy the homogeneous equation \( \mathbf{X}^T \mathbf{L} = 0 \), while in the image we have \( \tilde{\mathbf{w}}^T \mathbf{l} = 0 \). Since \( \tilde{\mathbf{w}} = T \mathbf{X} \), we get

\[
\tilde{\mathbf{w}}^T \mathbf{l} = \mathbf{X}^T T^T \mathbf{l} = 0 \Rightarrow \mathbf{L} = T^T \mathbf{l}
\]

Each line correspondence therefore gives two equations in the unknown elements of \( T \). A minimum of four lines can be used to solve for \( T \) (since the scale does not matter).

Assessors’ remarks: An unpopular question. Very few candidates recovered the simple expression of the horizon and its dependence on camera orientation. Only one candidate derived the line to line relationship needed for calibration with lines (part (d)).

4. Stereo vision

(a) The fundamental matrix \( F \) relates points in the left and right images of a stereo pair:

\[
\tilde{\mathbf{w}}'^T F \tilde{\mathbf{w}} = 0
\]

where \( \tilde{\mathbf{w}} = (u, v, 1) \) are the point’s pixel coordinates in the left image, and \( \tilde{\mathbf{w}}' \) are the coordinates of the corresponding point in the right image. The constraint arises from the requirement that the rays from the two cameras’ optical centres through \( \tilde{\mathbf{w}} \) and \( \tilde{\mathbf{w}}' \) must intersect at a point in space. \( F \) has zero determinant and can be determined only up to scale.

\( F \) can be estimated from point correspondences. Each point correspondence \( \tilde{\mathbf{w}} \leftrightarrow \tilde{\mathbf{w}}' \) generates one constraint on \( F \):

\[
\begin{bmatrix}
  u' \\
  v' \\
  1
\end{bmatrix}
\begin{bmatrix}
  f_{11} & f_{12} & f_{13} \\
  f_{21} & f_{22} & f_{23} \\
  f_{31} & f_{32} & f_{33}
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}
= 0
\]

This is a linear equation in the unknown elements of \( F \). Given eight or more perfect correspondences (image points in general position, no noise), \( F \) can be determined uniquely up to scale by solving the simultaneous linear equations. In practice, there may be more than eight correspondences and the image measurements will be noisy. The system of equations can then be solved by least squares, or using a robust regression scheme to reject outliers.

The linear technique does not enforce the constraint that \( \det F = 0 \). If the eight image points are noisy, then the linear estimate of \( F \) will not necessarily have zero determinant and the epipolar lines will not meet at a point. Nonlinear techniques exist to estimate \( F \) from 7 point correspondences, enforcing the rank 2 constraint.
(b) The epipoles lie in the null spaces of $F$ and $F^T$. So, for the left epipole we have:

$$F \tilde{w}_e = 0$$

If $F$ were invertible, we would be able to write

$$\tilde{w}_e = F^{-1}0 = 0$$

which is a contradiction. It follows that $F$ is non-invertible and therefore has maximum rank 2.

To check the validity of the given fundamental matrix, we need simply evaluate its determinant:

$$\det F = -1 \times (18 \times 10^4 - 300 \times 541.4214) - 300 \times (-58.5786 + 0) = 0$$

$F$ is therefore a valid fundamental matrix.

(c) (i) The coordinates of the point in A’s image can be used to derive an epipolar line in B’s image. Likewise, the coordinates of the point in C’s image can be used to derive another epipolar line in B’s image. Where these two lines intersect is the point’s location in B’s image. Algebraically, if the point’s coordinates in B’s image are $(u, v)$, we have

$$\begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -300 \\ 1 & 0 & -541.4214 \\ -300 & -58.5786 & 18 \times 10^4 \end{bmatrix} \begin{bmatrix} 300 \\ 500 \\ 1 \end{bmatrix} = 0$$

$$\Leftrightarrow \begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -300 \\ 1 & 0 & -541.4214 \\ -300 & -58.5786 & 18 \times 10^4 \end{bmatrix} \begin{bmatrix} 200 \\ 60710.7 \end{bmatrix} = 0$$

for the A,B pair, and

$$\begin{bmatrix} 376 & 700 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & -300 \\ 1 & 0 & -541.4214 \\ -300 & -58.5786 & 18 \times 10^4 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

$$\Leftrightarrow \begin{bmatrix} 400 & 317.4214 & -311794.98 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

for the B,C pair. Eliminating $u$, we obtain

$$800.2642v - 433216.38 = 0 \Leftrightarrow v = 541$$

Substituting into the equation for either epipolar line, we obtain $u = 350$. So the point appears at $(350, 541)$ in B’s image.

(ii) The transfer technique will fail when the two epipolar lines in B’s image are coincident. For non-degenerate camera configurations, this will occur when the observed point lies in the trifocal plane defined by the three optical centres. Then both epipolar lines in B’s image will pass through the two epipoles (one epipole
for A’s optical centre, the other for B’s), and there will be no unique point of intersection. [20%]

Assessors’ remarks: A question which produced similar responses from many candidates, who were typically very knowledgeable about the fundamental matrix and its properties (parts (a) and (b)), but unable to solve two simultaneous linear equations accurately (part (c)(i)). Part (c)(ii), which required a little insight beyond book work, was not well answered.

5. Visual motion

(a) The motion of the scene relative to the camera is

\[
\begin{bmatrix}
\dot{X}_c \\
\dot{Y}_c \\
\dot{Z}_c
\end{bmatrix} = -
\begin{bmatrix}
U_1 \\
0 \\
0
\end{bmatrix}
\]

Perspective projection onto the image plane can be expressed as

\[
x = \frac{fX_c}{Z_c} \Rightarrow \dot{x} = \frac{f\dot{X}_c}{Z_c} - \frac{fX_c\dot{Z}_c}{Z_c^2}
\]

Substituting for \(\dot{X}_c\) and \(\dot{Z}_c\) gives

\[
\dot{x} = -\frac{fU_1}{Z_c}
\] [20%]

(b) (i) Since the camera motion is along the x axis, x-t slices through the spatiotemporal image reveal true trajectories of distinguished image features (ie. true values of \(\dot{x}\) which can be used for structure-from-motion analysis). Slices at other orientations do not track individual features and any apparent trajectories are therefore misleading. [10%]

(ii) From the analysis in (a), the slopes of trajectories in x-t slice are \(-fU_1/Z_c\). The slope of the left trajectory is \(-10\) mm/s, so the depth of the left side of the object is given by

\[
-\frac{fU_1}{Z_c} = -10 \Leftrightarrow Z_c = \frac{fU_1}{10}
\]

Similarly, the slope of the right trajectory is \(-50/3\) mm/s, so the depth of the right side of the object is \(3fU_1/50\). [20%]

(iii) We can easily combine the depth information from (ii) with information from the square image at time \(t = t_0\). From the spatiotemporal image, we know that the dimension of the square is 35 mm, so the \((x, y)\) image plane coordinates of the four vertices at \(t = t_0\) are (clockwise starting from the top left)

\[(0, 0) , \ (35, 0) , \ (35, 35) , \ (0, 35)\]

Substituting into the standard perspective projection equations

\[
x = \frac{fX_c}{Z_c}, \ y = \frac{fY_c}{Z_c}
\]
we obtain the vertices’ camera-centered coordinates

\[(0, 0, Z_c), (35Z_c/f, 0, Z_c), (35Z_c/f, 35Z_c/f, Z_c), (0, 35Z_c/f, Z_c)\]

Finally, substituting the depths found in (ii), we obtain the camera-centered coordinates

\[(0, 0, fU_1/10), (2.1U_1, 0, 3fU_1/50), (2.1U_1, 2.1U_1, 3fU_1/50), (0, 3.5U_1, fU_1/10)\]

This is as far as we can go without knowing \(U_1\) (the speed-scale ambiguity). The four corners of the object are clearly coplanar, since the left and right sides are both parallel to the image plane. This does not mean that the object is planar, though: it could, for example, bulge out towards the camera near its centre. This hypothesis is supported by occlusion of the left edge after one second: the object is not planar. [25%]

(c) The key difference between spatiotemporal image analysis and real-time feature tracking is the way in which trajectories are measured. With feature tracking, the trajectory is measured in real time (using a snake, for example), using past measurements to initiate the search in each frame. With spatiotemporal image analysis, the trajectory is measured by detecting edges in appropriate slices through the 3D image: both past and future measurements can be used to locate the feature at any particular time. Even if a trajectory contains significant gaps (due to temporary occlusion of the feature, for example), techniques like the Hough transform can recover the edge in its entirety.

So, with spatiotemporal image analysis we can recover trajectories more reliably, leading to more accurate and complete 3D reconstructions. Against this, we need a huge amount of memory to store the spatiotemporal image, and the analysis cannot happen in real time, but only after the entire set of images has been acquired. [25%]

Assessors’ remarks: The few candidates who attempted this question answered parts (a) and (b)(i)–(ii) very well, indicating a good understanding of basic spatiotemporal image analysis. In (b)(iii), not one candidate managed to combine the depth information with the \(t = 0\) image to deduce the vertex coordinates, though several correctly inferred a convex object from occlusion of the left edge at time \(t = 1\). Answers to (c) were generally poor, since the key advantage of spatiotemporal image analysis (the ability to use future as well as past measurements) went unnoticed by all the candidates.