Stereogrammetry:
3D measurements from photographs

John Coffey, Cheshire, UK.

2024

Key words: 3D image reconstruction, parallax, photogrammetry, panoramic stitching,
epipolar, essential matrix, SIFT, RANSAC and SVD algorithms
constrained least-squares optimisation, pseudo-inverse matrix

1 Introduction

Like other mammals, we humans have stereoscopic vision – with our two eyes we can judge the distance of an object. Our brain combines the images seen by each eye and makes sense of the parallax. 'Parallax' is the apparent displacement of one object with respect to another when both are viewed from a slightly different position. You see it strongly when you hold your hand in front of your face and look at the scene beyond first with the left eye only then with only the right.

The phenomenon was used from the early days of photography to present an image which appeared three-dimensional. The photographer took two photos of the static scene, moving the camera rightwards between shots through the distance between our eyes. The two printed photos were mounted side by side and viewed through an optical device which presented only the left photograph to the left eye, and only the right one to the right eye. Using mathematics, parallax can be turned into a precise tool for measuring the relative positions of point objects in a scene by looking at them from two or more points of view. The maths essentially runs the image creation process backwards to find the positions which object points must have occupied for them to appear as they do over several photographs.

The technique was developed and used extensively for aerial reconnaissance during and after the second world war. Downwards pointing cameras were attached to the wing tips of an aeroplane which flew low over the terrain, taking simultaneous pairs of photographs. Alternatively, photos were recorded by a single camera at fast frame rates which ensured overlap of important objects between adjacent frames as the plane flew. The ability to see in 3D was a major intelligence advantage. One notable British pioneer was Prof Edgar 'Tommy' Thompson, member and later president of the Photogrammetric Society from 1952. He designed the Thompson-Watts mechanical plotting machine for precision mapping from aerial photographs. Parallel developments were carried out in Germany and the USA, so by 1960 the subject was well developed and widely used, and supported with dedicated peer-reviewed

technical journals. Today there is the International Society for Photogrammetry and Remote Sensing.

The technique has found wide application to surveying and architecture, and is widely used with aerial drones. It has become one aspect of computer vision. In recent years several web sites[1] have offered sophisticated programs by which the amateur can obtain from 10 or more overlapping images a 3D virtual reconstruction of an entire scene. Some also create a textured mesh for used within 3D modelling software.

Though this photogrammetry software can be used without in-depth technical knowledge, I have wished to understand how it works and to derive some of the mathematics for quantitative measurement, at least in the simplest cases. This article, therefore, is my own selective account of how parallax can be turned into a 3D measuring tool. The essential mathematics is projective geometry and linear algebra. One particular question is 'how many point objects must be observed and from how may camera positions and angles for an unambiguous determination of their positions to be made?'. We shall also see that the subject shares algorithms with other aspects of computer vision and image manipulation such as the stitching of overlapping photographs into one wide panorama.

There are two main parts to photogrammetry:

1. identifying common object points in two or more images taken from different positions,
2. using the positions of paired points in two or more images to determine the positions of the objects in the scene. This may involve the intermediate stage of determining the relative positions and orientations of the cameras.

In this simple and incomplete survey of this large subject I start in §2 by setting out the basics of parallax and explaining how object points can be readily calculated by triangulation provided the features are exactly paired in two overlapping images *and* the camera positions are known. In §3 I describe the more complicated geometry of determining the camera positions when they have not been set up in a controlled and known way, but need to be deduced from the overlapping photographs themselves. The clever algorithm for this uses 'epipolar geometry'. I then return to the first stage of photogrammetry – algorithms for automatically identifying features in overlapping images. §4 describes the SIFT algorithm which is one of the more powerful algorithms for feature extraction. §5 outlines the RANSAC algorithm which is applied to identify spurious matching of pairs of points. The section includes some comments on the challenges of numerical optimisation.

Much of the technical detail and example calculations are in the appendices.

- Appendix 1 explains the mapping of one quadrilateral onto another, the most basic operation in matching two overlapping images.
- Appendix 2 shows how this is extended to stitching two overlapping photographs together into a wide panorama. The operation uses constrained least-squares to obtain a good

[1] Meshroom by AliceVision, PixPro, Polycam, 3DF-Zephyr, Fano Zone 3D and PhotoModeler are six examples.

fit, and the mathematics of this is used in other algorithms where a best fit is needed. An example is given of warping one photograph to fit another.

- Appendix 3 is about SVD (Singular Value Decomposition), a widely used method for factoring matrices. Though SVD is not used specifically in my descriptions of photogrammetry, it is closely related to the recovery of camera positions by epipolar geometry, and is interesting in its own right. There is a description of the Moore-Penrose pseudo-inverse matrix used for least-squares data fitting.
- Appendix 4 gives details of Christopher Longuet-Higgin's 1980 algorithm for recovering camera positions using epipolar geometry. Some example calculations are included.
- Appendix 5 is about the SIFT algorithm.

2 Controlled translation of camera without rotation

2.1 Geometry

We start at the simplest situation and envisage an idealised scene which is completely empty apart from a few distinct points of light all of whose images the camera records. The camera is assumed to introduce no aberration, as would the pin-hole of a camera obscura[2]. To avoid the complication of image inversion which would occur with a real screen placed behind the pin-hole, I will represent the image as if it appeared on a plane placed in front of the focal point as in Figure 1. The camera is pointing towards an infinitely distant point V and the image plane is normal to OV at distance d. The essential feature of the geometry is that for each camera position a point object in the scene maps to a single point on the image plane. I used the same geometry in an article on perspective also on *www.mathstudio.co.uk*.

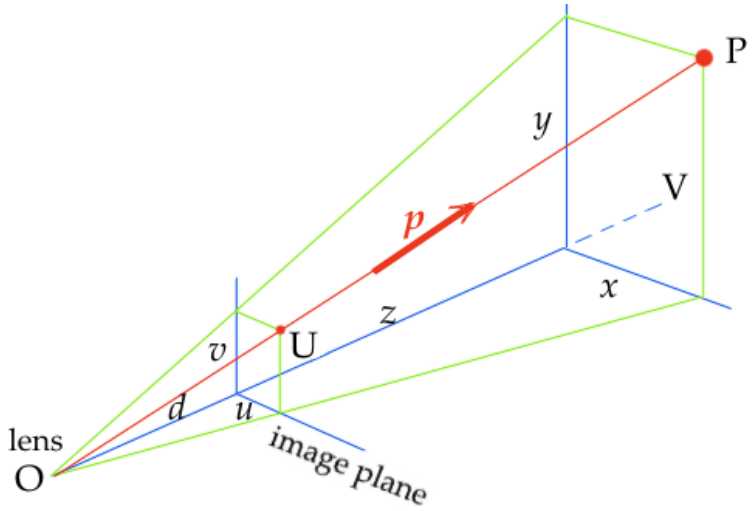

Figure 1: Geometry of image formation at one camera position. The focus is at origin O. Point P is projected to Q in the photograph along ray vector **a**. Note the left-handed xyz frame of axes.

[2] All lenses introduce geometrical and chromatic aberration. This can be allowed for in photogrammetry by calibrating the lens on a scene containing many precisely known point objects.

Referring to Figure 1, a ray of light leaves the object P and is recorded on the image plane at U, all such rays converging on the pin-hole 'lens' at O. P is at (x, y, z) and U at (u, v, d). u and x, v and y are in simple proportion:

$$\frac{u}{d} = \frac{x}{z}, \qquad \frac{v}{d} = \frac{y}{z} \qquad \text{so } u = x\,\frac{d}{z}, \quad v = y\,\frac{d}{z}, \qquad z > d. \tag{1}$$

These are the fundamental relations of linear perspective. A ray coming along the line of sight maps to the picture's origin, $(u, v) = (0, 0)$. Any points along the infinite line of **p** will be projected to the same point in the photograph. Given only a single image point, we have no idea of the range of the object.

There can be no parallax with just one camera position. This remains true even if the camera is rotated about its focal point since rotation does not change the angular separation of rays from the various object points. So what can be determined about one single object seen from just two camera positions? I will explain the geometry in the 2D case where all objects and the forward directions of the cameras lie in the one xz plane, as illustrated in the left and centre panels of Figure 2. In the left panel the camera has been displaced laterally (or a second parallel camera used) through a from O_1 to O_2 without rotation or change in z. In the central panel the translation is through b in z. Both x and z translations create parallax.

The shape of triangle PO_1O_2 is defined by the angles θ_1, θ_2, but its size depends on distance a or b. This is a feature of photogrammetry – only relative positions and ratio of distances can be recovered unless the system is calibrated by knowing at least one distance accurately. Put another way, since the photographic images may be scaled to any comfortable

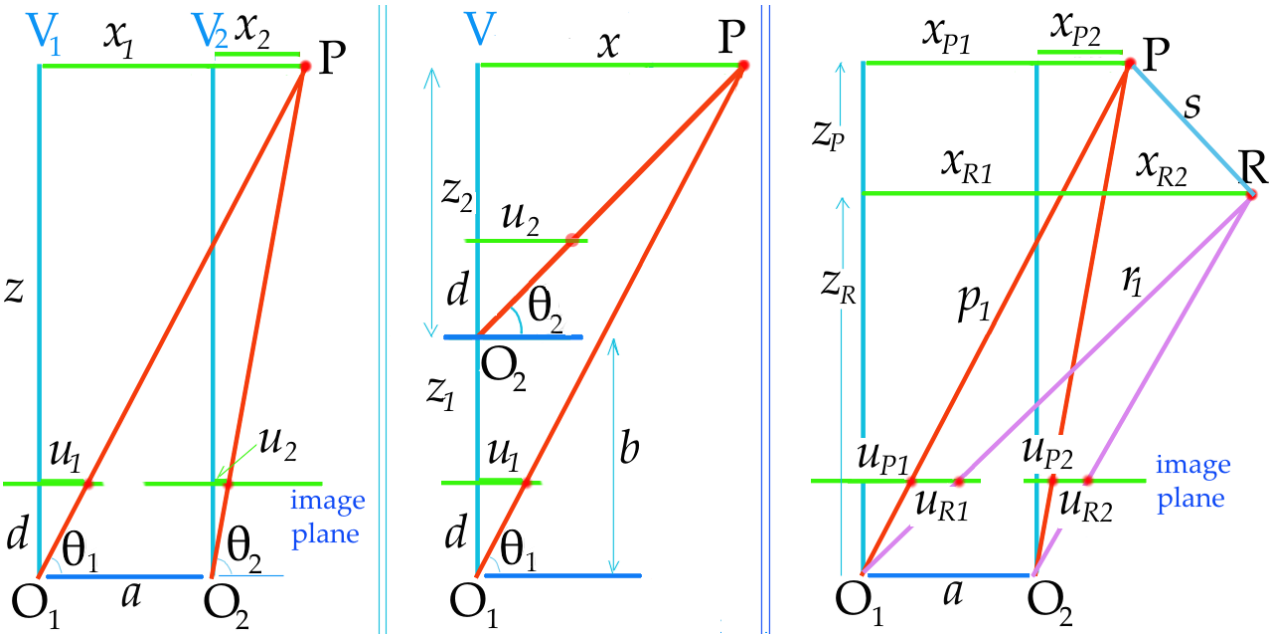

Figure 2: Left: image positions u_1, u_2 in x direction of object P when camera is moved laterally through a from O_1 to O_2. Centre: Translation in z by b. Right: image positions of two fixed objects P and R. x and u distances are measured from the forward direction of the respective camera.

size for observation, the scale must be determined from measurements on known distances. The focus-image plane distance d in Figures 1 and 2 is essentially a scale factor.

For lateral translation, measuring x from the forwards directions at the two lens positions, we have $x_1 = x_2 + a$. From Eq 1

$$x_1 = \frac{u_1 z}{d} = \frac{u_2 z}{d} + a \quad \text{so} \quad \frac{z}{d} = \frac{a}{u_1 - u_2} \quad \text{or} \quad u_1 - u_2 = \frac{ad}{z}. \tag{2}$$

This is the parallax equation. Numerically the parallax is $u_1 - u_2$. It is determined by the sideways shift in viewing position and decreases with distance of the object, and is the same for all objects at the same distance z. From this

$$x_1 = \frac{u_1 a}{u_1 - u_2}, \qquad x_2 = \frac{u_2 a}{u_1 - u_2}, \qquad z = \frac{ad}{u_1 - u_2}. \tag{3}$$

x/a and z/a are therefore found, though recovering x_1 and z depends on knowing a accurately and being certain that there has been no rotation of the camera. Alternatively, if a is not initially known but one length s in the scene is known precisely – *e.g.* between points marked on a rod – a can be found, and from a all other lengths. In practice, because of sensitivity to camera position and orientation, it is necessary to record images from several camera positions and seek convergence of co-ordinate values by some form of averaging.

The maths is straightforward. Suppose s, the distance PR, is known precisely.

$$s^2 = (x_P - x_R)^2 + (z_P - z_R)^2.$$

From Eq 3

$$\begin{aligned} \frac{s^2}{a^2} &= \left(\frac{u_{P1}}{u_{P1} - u_{P2}} - \frac{u_{R1}}{u_{R1} - u_{R2}}\right)^2 + \left(\frac{d}{u_{P1} - u_{P2}} - \frac{d}{u_{R1} - u_{R2}}\right)^2, \\ &= \left(\frac{u_{P2}}{u_{P1} - u_{P2}} - \frac{u_{R2}}{u_{R1} - u_{R2}}\right)^2 + \left(\frac{d}{u_{P1} - u_{P2}} - \frac{d}{u_{R1} - u_{R2}}\right)^2, \\ &= \frac{d^2(u_{P1} - u_{P2} - u_{R1} + u_{R2})^2 + (u_{P2}u_{R1} - u_{P1}u_{R2})^2}{(u_{P1} - u_{P2})^2(u_{R1} - u_{R2})^2}. \end{aligned} \tag{4}$$

This value does not change if the suffices 1 and 2 are interchanged. It is important that the u values be measured from the centre of the screen, since if there is an offset h, $(u_{P2} + h)(u_{R1} + h) - (u_{P1} + h)(u_{R2} + h) \neq u_{P2}u_{R1} - u_{P1}u_{R2}$.

For translation in both the x and z directions

$$x_1 = \frac{u_1(u_2 - ad)}{d(u_2 - u_1)}, \qquad x_2 = \frac{u_2(bu_1 - ad)}{d(u_2 - u_1)}, \qquad z_1 = \frac{bu_2 - ad}{(u_2 - u_1)}, \qquad z_2 = \frac{bu_1 - ad}{(u_2 - u_1)}. \tag{5}$$

In general lateral translation is to be preferred since it gives larger parallax for objects in front of the camera, where most objects of interest will be. Translation in z is better at detecting parallax between objects well to the side of the field of view, so may be appropriate with a wide angle lens.

2.2 Practical demonstration

To give a concrete example of the above theory for lateral translation I set up two simple cases in which the point objects and the centre of the camera's lens were all in the same xz plane. (To anticipate §3, this is called the 'epipolar plane' of the object points.) A digital single lens reflex camera was mounted on a wooden board so that it could be slid along a table, nominally without rotation. The point objects were black dots drawn on a wooden bar, the bar being placed on the table as in the photograph of Figure 3. Point 1 was 20 cm from Point 2 and the others were at 10 cm intervals.

The first stage was to calibrate the images and determine the image-focus distance d. I placed the bar parallel to the x axis with Point 3 in line with the forwards direction of the camera. With the focal length of the zoom lens fixed at 24 mm, photos were taken with the bar moved to eight distances z. From the photographic image at each distance the position of each object point was measured in terms of its u position in pixels from the left edge of the digital screen using the MPos software from Sourceforge.net. The actual distance recorded was z', measured with a rule with respect to the camera's mounting plate. This differed from the focal point by a distance g. g can be determined according to Eq 1 such that $u/d = x/(z' + g))$ for all points. Clearly u and d must have the same units, as must x and z. Best fit values calculated in a spreadsheet were $g = -18$ mm and $d = 1736$ pixels. Allowing for the front face of the lens being 50 mm in front of the reference position on the mounting board at $z' = 0$, the focal point was 32 mm behind the front of the lens.

Case 1

In the first test I placed the bar at arbitrary range and turned it at 45° to both x and z axes, as in Figure 3. I then took photographs with the camera moved sideways in x in 5 cm steps without turning or change in z. Image positions and corresponding angles θ are listed in Table 1. Here $\tan\theta = u/d = x/z$. Any two camera positions can be used to determine the

Figure 3: Five point objects on a wooden bar at mid-lens height. They are 10 or 20 cm apart.

	$u + 956$ pixels					θ rads				
a mm	Pt 1	Pt 2	Pt 3	Pt 4	Pt 5	Pt 1	Pt 2	Pt 3	Pt 4	Pt 5
−15	775	1242	1598			−0.104	0.165	0.370		
−10	623	1043	1369			−0.192	0.050	0.238		
−5	487	864	1158	1583		−0.270	−0.053	0.116	0.361	
0	347	678	939	1318	1850	−0.351	−0.160	-0.010	0.209	0.515
5	213	498	722	1054	1562	−0.428	−0.264	−0.135	0.056	0.349
10		330	516	797	1235		−0.361	-0.253	−0.092	0.161
15		145	299	530	895		−0.467	-0.378	−0.245	−0.035

Table 1: Image positions u of five point objects in Test 1, and their corresponding angles θ from the forwards direction at each lateral displacement a of the camera. The image centre is at 956 pixels.

relative positions of the points whose images they share, and if at least one dimension is known exactly, the absolute positions can be determined. Table 2 details calculation of the x and z co-ordinates of Point 2 only, based on all pairs of u values for the seven camera positions from which Point 2 could be seen. The values of a are all nominal values for positions 5 cm apart. There are ${}^7C_2 = 21$ pairs of $u_1,\ u_2$. Though these 21 measurements are be independent, coming as they do from only 7 positions of the camera, the errors in the x values are uncorrelated because of the reciprocal relation in Eq 2, and I see no ground for rejecting one pair in favour of another. The foot of Table 2 shows the point at $(-7 \cdot 5 \pm 0 \cdot 2, 47 \cdot 0 \pm 1 \cdot 5)$ cm from the central position of the camera. Table 3 gives the coordinates of all five object points and the distances between them – all very close to 10 cm or multiples thereof. Point 3 had been placed at $z' = 41 \cdot 4$ cm, $z = 39 \cdot 6$ cm, to compare with the mean measured value of $39 \cdot 9$. These co-ordinates are plotted in Figure 4. They fall convincingly on a straight line inclined at $\arctan 1 \cdot 03 = 46°$. Moreover, Point 3 is measured as only $0 \cdot 4$ cm away from where it was placed. Overall, therefore, the algorithm has worked quite well in this contrived example.

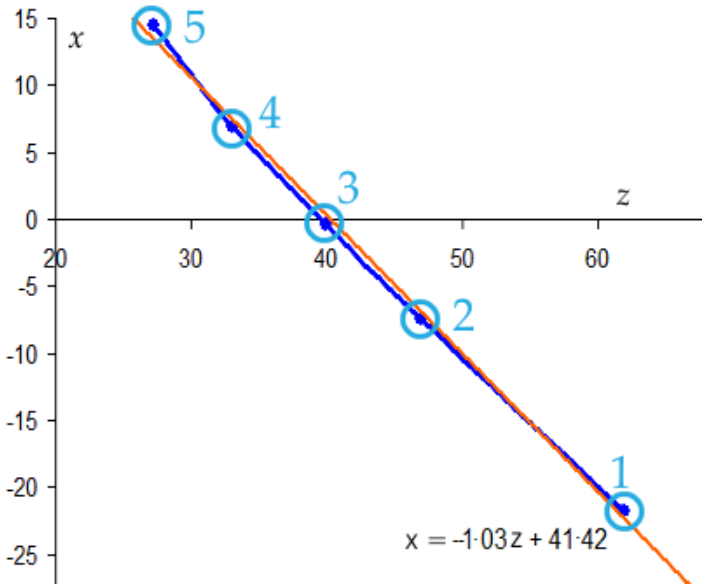

Figure 4: Points 5 to 1 along the wooden bar as determined by stereogrammetry.

camera posn 1 (cm)	u_1	u_2	$u_1 - 956$	$u_2 - 956$	a (cm)	x_1	x_1 (cm) wrt central posn	z (cm)
-15	1242	1043	286	87	5	7.2	-7.8	43.6
-15	1242	864	286	-92	10	7.6	-7.4	45.9
-15	1242	678	286	-278	15	7.6	-7.4	46.2
-15	1242	498	286	-458	20	7.7	-7.3	46.7
-15	1242	330	286	-626	25	7.8	-7.2	47.6
-15	1242	145	286	-811	30	7.8	-7.2	47.5
-10	1043	864	87	-92	5	2.4	-7.6	48.5
-10	1043	678	87	-278	10	2.4	-7.6	47.6
-10	1043	498	87	-458	15	2.4	-7.6	47.8
-10	1043	330	87	-626	20	2.4	-7.6	48.7
-10	1043	145	87	-811	25	2.4	-7.6	48.3
-5	864	678	-92	-278	5	-2.5	-7.5	46.7
-5	864	498	-92	-458	10	-2.5	-7.5	47.4
-5	864	330	-92	-626	15	-2.6	-7.6	48.8
-5	864	145	-92	-811	20	-2.6	-7.6	48.3
0	678	498	-278	-458	5	-7.7	-7.7	48.2
0	678	330	-278	-626	10	-8.0	-8.0	49.9
0	678	145	-278	-811	15	-7.8	-7.8	48.9
5	498	330	-458	-626	5	-13.6	-8.6	51.7
5	498	145	-458	-811	10	-13.0	-8.0	49.2
10	330	145	-626	-811	5	-16.9	-6.9	46.9
Mean							**-7.5**	**47.0**
St. Devn.							**0.2**	**1.5**

Table 2: Calculation of co-ordinates of Point 2 with respect to the central position of the camera.

Point	position x (cm)	z (cm)	distance apart	actual distance
1	-21.8	62.0		
			20.8	19.9
2	-7.5	47.0		
			10.0	10.0
3	-0.4	39.9		
			10.0	10.1
4	6.9	33.0		
			9.5	10.0
5	14.4	27.3		
			50.2	50.1
1	-21.8	62.0		

Table 3: Average measured co-coordinates of Points 1 to 5 and the distances between pairs. The errors are largest for extreme Points 1 and 5, but the distance between them is correct.

Case 2

In this case I did not use pre-determined camera positions a, but rather took the separation of Points 1 and 3 as known to be $29 \cdot 9$ cm. A second wooden bar was added to the scene as seen in Figure 5. Separations were AB = BC = 10 cm, CD = 15 cm, to within $0 \cdot 1$ cm. The angles of the two bars from the x axis were arbitrary and not measured directly, but from the photograph can be seen to be about 55° anticlockwise for the thicker bar and about 22° clockwise for the bar used in Case 1.

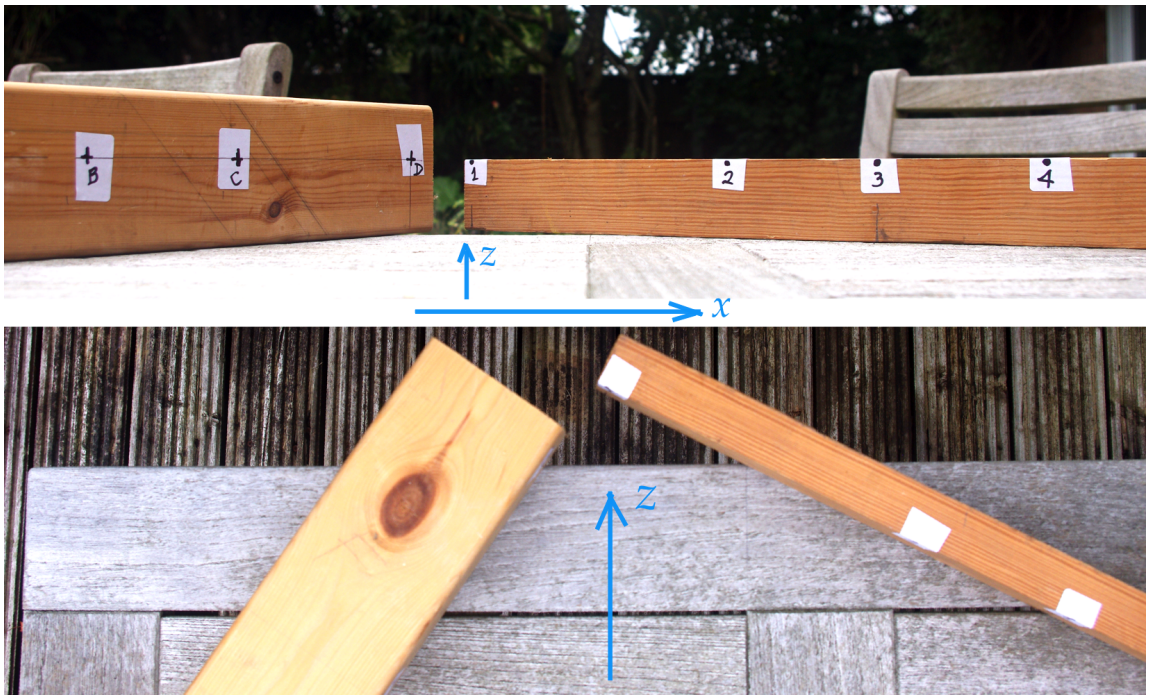

Figure 5: Case 2: Two wooden bars seen along the $+z$ direction (upper photo) and from above (lower). Points A and 5 are outside the image. (Ignore the garden furniture!)

My first step was to determine the relative positions of the nine camera stations. Using Eq 4 I calculated the separation a_{in} of pairs of camera positions, indexed from $i = 1$ to $n = 9$, assuming $s = 29 \cdot 9$ cm. There are 36 such pairs and I did carry out the calculation for all 36 though only a quarter of them are independent. The separation a_{jk} of adjacent pairs can be found by taking data only from images at camera positions j and k, but can also be found as the difference from a third position, as $a_{jm} - a_{mk}$. I examined the gain in accuracy in proportion to the computational effort from taking many combinations of three pairs and found only a marginal benefit. The ratio a/s differed typically by 0·001, less than 1%, between simply taking adjacent pairs and taking the difference of averages over many $a_{jm} - a_{mk}$ types. Separations are listed in Table 4. They average $4 \cdot 94 \pm 0 \cdot 28$ cm, close to the actual $5 \cdot 0 \pm 0 \cdot 1$ cm, with adjacent separations accurate to about 5%.

From relative camera positions the positions of the object points A to D, 1 to 5 were calculated following the algorithm of Table 2. The results are in Table 4 and plotted in Figure 6. Camera position 1 is taken to be at $x = 0$. The distances between object points are close to the actual ones, and the angles of the wooden bars from the x axis are +54° and −27°.

pair j-k	a/s	a_{jk} (cm)
1-2	$0 \cdot 176$	$5 \cdot 26$
2-3	$0 \cdot 172$	$5 \cdot 14$
3-4	$0 \cdot 155$	$4 \cdot 63$
4-5	$0 \cdot 178$	$5 \cdot 32$
5-6	$0 \cdot 167$	$4 \cdot 99$
6-7	$0 \cdot 156$	$4 \cdot 68$
7-8	$0 \cdot 164$	$4 \cdot 89$
8-9	$0 \cdot 155$	$4 \cdot 62$

Table 4: Spacing a_{jk} of adjacent camera positions determined from Points 1 and 3 being separated by 29.9 cm.

Point	x (cm)	z (cm)	b (cm)
A	-13.0	57.0	9.6
B	-7.4	64.7	10.0
C	-1.7	72.9	14.2
D	7.1	84.2	5.7
1	10.9	88.4	20.1
2	28.7	79.1	9.9
3	37.6	75.0	10.0
4	46.5	70.4	9.9
5	55.0	65.4	

Table 5: Co-ordinates of the object points and the distance b between adjacent ones, as determined by stereogrammetry.

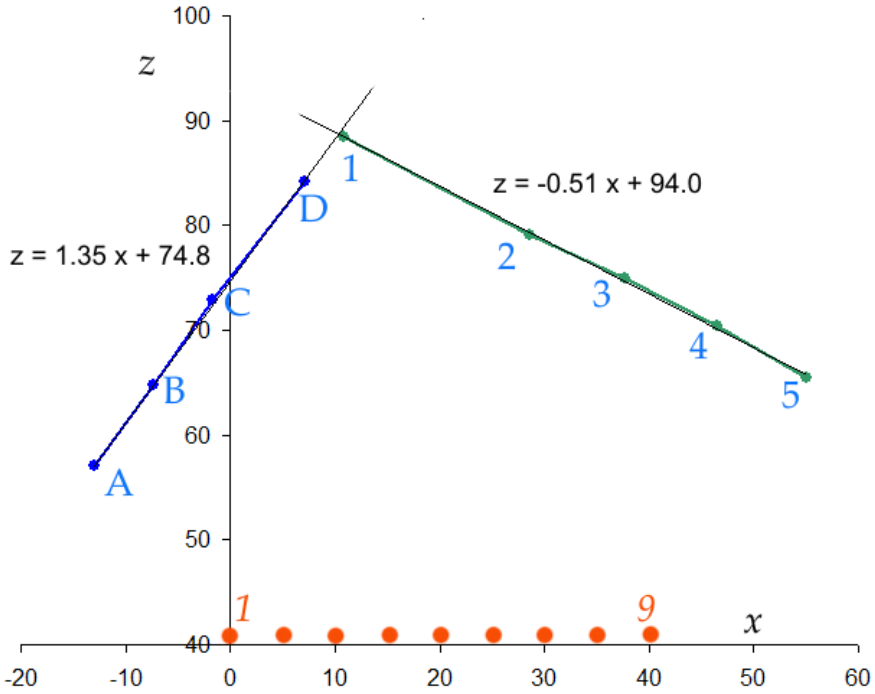

Figure 6: Case 2: Calculated positions 1 to 9 of the camera, and of the object points A to D, 1 to 5.

Cases 1 and 2 illustrate the practicalities of stereogrammetry when object points lie in the xz plane of the camera, when the translation is strictly in the x direction[3] and there is no rotation between images. The mathematics is simple – just triangulation. I am persuaded that for accurate measurements it is best to constrain the camera as in these cases, and control the spacing between camera stations, perhaps by having several identical cameras fixed onto a bar or by having the one camera advance along a bar in known steps.

3 Unconstrained camera positions and rotations

In this section we examine the general situation in which the camera moves to arbitrary and unknown positions and is turned at arbitrary angles, as when a person with a hand-held camera takes overlapping photographs while walking around the scene.

3.1 General features of photogrammetry software

The reconstruction algorithm in the practical examples in the previous section worked because there was only one unknown quantity – the separation a in x or b in z of adjacent camera positions. Once translation is random and the camera rotates about the y axis, two more unknown variables have been introduced into the simple 2D cases of the previous section. There are even more unknowns with a hand-held camera photographing a building. Inverse ray tracing can only be performed if there is a corresponding increase in information. Where can this come from? Since adding more camera stations just increases the number of unknowns, it must come from observing more object points at each camera station. Intuitively we might hope that if sufficient data points were recorded, the ambiguity in their reconstructed positions could be narrowed to a tolerable degree. Recovering information from images is akin to the common mathematical problem of solving n equations in k independent unknown variables. If $n = k$, there is either no solution or a unique one. If $n < k$, there is deficiency of information and hence non-removable ambiguity. If $n > k$, there is surplus information and each k-subset should give the same result. In practice there are random errors in measuring the position in the image so a best fit has to be found which minimises the discrepancies amongst k-subsets. This is the usual case in practice; many feature pairs in overlapping images are taken and a least-squares or equivalent fit approximation made.

A comment about camera rotation: rotation alone of the camera does not create parallax. If the camera is rotated about the focal point of the lens, the image records a different section of the scene, but the angles between rays from adjacent object points have not changed, so triangles as in Figure 2 cannot be constructed. For this reason taking photographs for 3D stereo reconstruction requires a different type of image overlap from taking ones for panoramic stitching. The former requires translation, especially laterally, and the latter requires rotation without translation. Panoramic stitching is possible because of a well known theorem of projective geometry, that any two quadrilaterals whose corresponding vertices can be joined by lines which pass through the same point (the focus) can be transformed one to

[3] It would also work for *known* translations involving the z direction.

the other by a projective transformation. Some maths is given in Appendix 1, and Appendix 2 gives an example of panoramic stitching with image warping.

Stereophotogrammetry packages such as Meshroom and 3D-Zephyr use algorithms which search for common features in many photographs taken from fairly arbitrary positions and reconstruct them in 3D. I do not know their particular methods, but offer some views on typical approaches and relevant established algorithms. Such packages progress through several major stages:

1. sorting the images to accept only those of adequate quality and which form an overlapping sequence,
2. identifying many fine features in each image, and logging their positions,
3. parametrising each such feature with a numerical 'signature' to allow unambiguous matching between two or more images,
4. searching for and eliminating spurious matches,
5. from the good quality matches, determining the relative positions and orientations of the cameras from which the respective photographs were taken, and where the object points are located with respect to a chosen reference frame of axes, up to a scale factor,
6. decorating each reconstructed object point with an element of the photograph at that point, to present to the user a fragmentary but recognisable 3D picture,
7. connecting the object points correctly into a 3D mesh for export.

Though these packages are remarkably effective, the results are better when the photographer has provided quality input. The photographs should be sharply in focus, with consistent lighting, good depth of field, no camera shake or grain, and taken in sequence with about 50% overlap, and with the main change between camera stations being a sideways translation. Reflections from polished surfaces confuse feature recognition, and blank, flat areas are of no use. Highly patterned scenes in which the same motifs recur, such as many parallel lines (e.g. railings, stairs, wallpaper), are likely to produce too many false matches. The best scenes, therefore, have plenty of contrast and variety in colour and shape of features. Below I explain the epipolar geometry algorithm devised in 1980 by Christopher Longuet-Higgins for determining the relative position and orientation of two camera station from one pair of overlapping photographs. This is supported by mathematical details in Appendix 4. Of the algorithms for extracting tiny features in photographs and matching them with ones in an overlapping image, SIFT is probably the most generally effective. It is outlined in §4 and illustrated in Appendix 5. The RANSAC algorithm for recognising genuine matched pairs of point features in two images is outlined in §5. These and similar algorithms have been around for years, undergoing significant development and spawning variants. Emphasis in their development has been on reliability and computational speed.

3.2 Epipolar algorithm for determining camera position and rotation

We assume that SIFT and RANSAC, or by-eye examination, have provided a sufficient numbers of correctly matching pairs of object points. How do we find the camera positions and thence the object positions?

The search for an affective algorithm started during World War II. A major part of the early theoretical developments was made by E.H. Thompson in the 1950s. He formulated the matter in terms of cubic equations in five unknowns, showing that the realtive positions of two cameras can be calculated from just five points common to both photographs. In 1981 a major development was devised by Christopher Longuet-Higgins through his interest in binocular vision in humans and animals. He showed[4] that there is an excellent algorithm for determining relative camera positions from eight points in two overlapping images without the cubic simultaneous equations. The proof is clearly explained in his article, but it is such an elegant algorithm that I outline it here and give some illustrative examples.

The vector geometry of two non-parallel cameras is shown in Figure 7. The calculation is made in the plane containing the two camera stations O_1, O_2 and one object point P. This is called the 'epipolar plane' or 'basal plane' of P and is different for (almost) all object points. The line O_1:O_2 is the base line and also the axis of intersection of epipolar planes for points Q, R, S, ... as seen from O_1, O_2. Textbooks also define two 'epipolar points' which are where the line O_1O_2 cuts each image, marked E_1, E_2 in the figure. However, the following argument makes no use of them. In §2 the xz plane was the epipolar plane and the epipolar points were at infinity because the two cameras were pointing in parallel directions. The mathematics is further explained in Appendix 4.

$\mathbf{p_1}$ is the vector from O_1 to P and similarly $\mathbf{p_2}$, whilst $\mathbf{b}$ is the unknown displacement vector between the camera stations. The normals to the epipolar plane, $\mathbf{n_1}$, $\mathbf{n_2}$, are also shown. They can be expressed as the cross products $\mathbf{n_1} = \mathbf{b} \times \mathbf{p_1}$, $\mathbf{n_2} = \mathbf{b} \times \mathbf{p_2}$. They are parallel to

[4] 'A computer algorithm for reconstructing a scene from two projections', Nature, Vol. 293, Sept 1981, p 133.

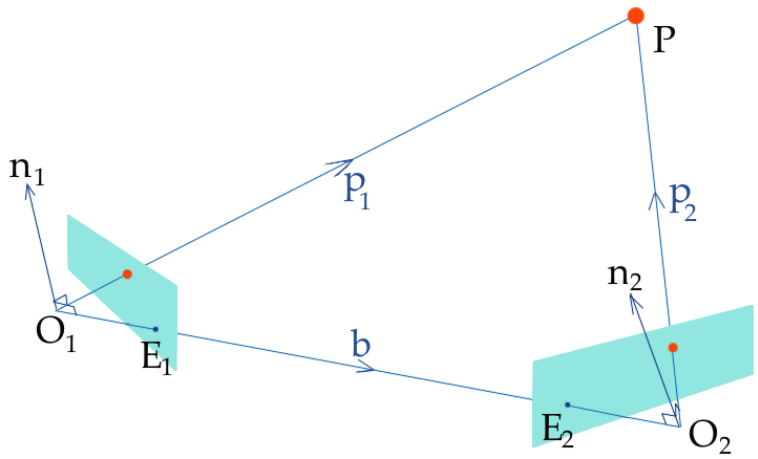

Figure 7: The base line O_1O_2 and object point P form the basal (epipolar) plane .

each other and also satisfy $\mathbf{n_1} \cdot \mathbf{b} = \mathbf{n_2} \cdot \mathbf{b} = \mathbf{n_1} \cdot \mathbf{p_1} = \mathbf{n_2} \cdot \mathbf{p_2} = 0$. Since we aim to find $\mathbf{b}$, these relations are a valuable constraint on its possible values. This is an example of a common strategy in mathematics – find a quantity which can be set to zero.

The next stage invokes the subtle distinction between a vector and its representation as a matrix of components in a particular co-ordinate frame. The vector relations

$$\mathbf{p_2} = \mathbf{p_1} - \mathbf{b}, \qquad \mathbf{p_2} \cdot \mathbf{n_2} = \mathbf{0}, \qquad \mathbf{b} \times \mathbf{b} = \mathbf{0} \tag{6}$$

are independent of co-ordinate frame. However, the co-ordinates of P seen from O_2 are related to those from O_1 by rotation as well as translation $\mathbf{b}$. Similarly $\mathbf{b}$ is (b_{x1}, b_{y1}, b_{z1}) in the frame of camera 1 and (b_{x2}, b_{y2}, b_{z2}) in the frame of camera 2. If $\mathbf{R}$ is the matrix representing rotation about O_2, these co-ordinates are related by

$$\mathbf{p_2} = \mathbf{R}(\mathbf{p_1} - \mathbf{b}) \tag{7}$$

which is to be understood as a matrix equation with components in the respective co-ordinate frames. To be clear $\mathbf{Rb}$ as a matrix is still simply $\mathbf{b}$ as a vector; $\mathbf{b}$ has not been rotated, only its components transformed. Apply the constraint $\mathbf{p_2} \cdot \mathbf{n_2} = \mathbf{0}$ to Eq 7:

$$\mathbf{p_2} \cdot (\mathbf{b} \times \mathbf{p_2}) = \mathbf{p_2} \cdot [\mathbf{b} \times \mathbf{R}(\mathbf{p_1} - \mathbf{b})] = \mathbf{p_2} \cdot [\mathbf{b} \times \mathbf{R}\mathbf{p_1} - \mathbf{0}] = \mathbf{p_2} \cdot (\mathbf{b} \times \mathbf{R})\mathbf{p_1} = \mathbf{0}. \tag{8}$$

The product $\mathbf{b} \times \mathbf{R}$ is to be understood at $\mathbf{b} \times \mathbf{R_1}$, $\mathbf{b} \times \mathbf{R_2}$, $\mathbf{b} \times \mathbf{R_3}$ where $\mathbf{R_j}$ are the three columns of $\mathbf{R}$. Matrix $\mathbf{E} = \mathbf{b} \times \mathbf{R}$ converts the co-ordinates of $\mathbf{p_1}$ to those of $\mathbf{p_2}$. It is a 3×3 invertible matrix called the 'essential matrix', and has the highly fortuitous property that it can be decomposed into its two factors. For this $\mathbf{b}\times$ must be understood as a matrix operation. The cross product of two vectors is not a true vector (a 'polar vector'), but rather an 'axial vector' which behaves like an anti-symmetric second rank tensor. Therefore the operation $\mathbf{b_2}\times$ can be written as multiplication by the skew-symmetric matrix $\mathbf{B}$

$$\mathbf{E} = \mathbf{b} \times \mathbf{R} \equiv \mathbf{B}.\mathbf{R} = \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}. \tag{9}$$

The components of $\mathbf{B}$ are with respect to camera 2. If the elements of $\mathbf{E}$ can be found from the image points, it can be split as in Appendix 4 and the examples below to recover $\mathbf{B}$ and $\mathbf{R}$ separately, at which stage we have deduced the relative position of the cameras.

We have obtained $\mathbf{p_2 E p_1} = 0$. We do not know $\mathbf{p_1}$, $\mathbf{p_2}$, but do have their image co-ordinates (u, v) which from Eq 1 satisfy

$$\frac{p_{1x}}{p_{1z}} = \frac{u_1}{d}, \quad \frac{p_{1y}}{p_{1z}} = \frac{v_1}{d}, \qquad \frac{p_{2x}}{p_{2z}} = \frac{u_2}{d}, \quad \frac{p_{2y}}{p_{2z}} = \frac{v_2}{d}.$$

Using homogeneous co-ordinates

$$\frac{p_{2z}}{d} \begin{pmatrix} u_2 & v_2 & 1 \end{pmatrix} \mathbf{E} \frac{p_{1z}}{d} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = 0.$$

We do not know p_{1z} or p_{2z}, but do know that neither is zero since object points are at ranges $> d$. So the other factors must themselves be zero. We arrive at the final equation for the essential matrix,

$$\begin{pmatrix} u_2 & v_2 & 1 \end{pmatrix} \mathbf{E} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = 0. \tag{10}$$

E is the same for all points common to the two overlapping photographs, depending only on the relative positions of the cameras. Therefore we can examine 9 points and solve for the unknown elements of **E**. In fact only 8 are strictly necessary because the scale of the whole scene is undefined unless we know at least one physical length accurately. Only ratios of lengths can be found. In practice many maore point pairs are used.

The key to factoring $\mathbf{E} = \mathbf{BR}$ is to observe that $\mathbf{R^T R} = \mathbf{I_3}$, the identity matrix. Therefore $\mathbf{EE^T} = \mathbf{BRR^T B^T} = \mathbf{BB^T}$, and **B** is isolated. Appendix 4, §A4.1 enlarges on this point.

3.3 Examples of epipolar calculations

I propose to illustrate the steps in this elegant algorithm by using computer-created point objects and their image positions. This allows the effects of translation in x, y and z to be isolated, and eliminates measurement errors and aberrations in a camera lens. Accordingly I wrote a short program to generate the (x, y, z) co-ordinates of random point objects and calculate the image positions according to Eq 1, allowing for offset of camera 2 with respect to camera 1 by x_0, y_0 and/or z_0, with or without rotation. The cases below examine recovery of the translation vector, **b**. The method for recovering the 3×3 matrices **B** and **R** from **E** are explained in more detail with a numerical example in Appendix 4.

Example 1: Translation in x and y. Table 6 lists 12 points in the scene and their image co-ordinates in pixels as seen from cameras 1 and 2, for $d = 1700$. The shift has been only in the x and y directions, by $0 \cdot 9$ and $-0 \cdot 3$ units respectively. We will calculate **E** by using the eigenvalue-vector method in Appendix 2. Expand Eq 10. Each point contributes one row to a 12×9 matrix **M** with the form

$$(u_1u_2 \quad u_2v_1 \quad u_2 \quad u_1v_2 \quad v_1v_2 \quad v_2 \quad u_1 \quad v_1 \quad 1).(e_{11} \quad e_{12} \quad e_{13} \quad e_{21} \quad ..... \quad e_{33})^T = 0. \tag{11}$$

To put vectors $\mathbf{u_1}$, $\mathbf{u_2}$ into projective co-ordinates with third co-ordinate 1, each row is divided by $d = 1700$, making its elements tangents of the angles away from the forwards direction to

Pt	1	2	3	4	5	6	7	8	9	10	11	12
x	-0.379	0.28	-0.333	-0.03	-0.99	-0.07	0.022	0.063	-0.992	-0.908	-0.32	0.782
y	-0.801	-0.19	0.311	-0.73	0.283	0.263	0.136	0.855	-0.151	-0.827	0.688	0.115
z	4.40	4.92	3.55	4.26	2.26	3.19	0.89	3.28	1.32	3.17	3.94	1.08
u_1	-147	97	-159	-10	-744	-36	41	32	-1279	-486	-138	1234
v_1	-310	-67	149	-290	213	140	260	443	-195	-443	297	182
u_2	-495	-214	-591	-369	-1420	-515	-1677	-434	-2439	-968	-526	-187
v_2	-194	37	293	-171	438	300	833	599	192	-282	426	655

Table 6: 12 objects seen by cameras 1 and 2, with camera 2 displaced $0 \cdot 9$ units in x, $-0 \cdot 3$ in y.

the point objects. Thus the first row is

$$(0\cdot0252 \quad 0\cdot0531 \quad -0\cdot291 \quad 0\cdot00987 \quad 0\cdot0208 \quad -0\cdot114 \quad -0\cdot0865 \quad -0\cdot182 \quad 1)$$

Following Appendix 2, form the symmetric 9×9 matrix $\mathbf{M^T M}$ and find its smallest eigenvalue. Since the numbers in this example have been created and not read experimentally, the error is merely rounding error and so the smallest eigenvalue is only $1\cdot4\times10^{-7}$, essentially zero. Its eigenvector is the elements of $\mathbf{E}$ making

$$\mathbf{E} = \begin{pmatrix} -0\cdot001 & 0\cdot005 & -0\cdot224 \\ -0\cdot003 & 0\cdot001 & -0\cdot670 \\ 0\cdot224 & 0\cdot671 & -0\cdot000 \end{pmatrix}.$$

This is almost skew-symmetric; the small deviations are probably due to rounding errors in recording the image positions to ±1 pixel. It is clear even at this stage that this matrix could result only if the rotation were zero. Nevertheless we press on with the algorithm and form $\mathbf{EE^T} = \mathbf{BB^T}$:

$$\begin{pmatrix} 0\cdot050 & 0\cdot150 & 0\cdot003 \\ 0\cdot150 & 0\cdot449 & -0\cdot000 \\ 0\cdot003 & -0\cdot000 & 0\cdot501 \end{pmatrix} \rightarrow \begin{pmatrix} 0\cdot10 & 0\cdot30 & 0\cdot01 \\ 0\cdot30 & 0\cdot90 & -0\cdot00 \\ 0\cdot01 & -0\cdot00 & 1\cdot00 \end{pmatrix} = \begin{pmatrix} b_y^2+b_z^2 & -b_xb_y & -b_xb_z \\ -b_xb_y & b_x^2+b_z^2 & -b_yb_z \\ -b_xb_z & -b_yb_z & b_x^2+b_y^2 \end{pmatrix}. \tag{12}$$

The middle matrix is merely the left one multiplied by 2 to tidy the numbers. The components of $\mathbf{B}$ can be recovered either by solving the three simultaneous equations from the diagonal elements, or the three from the off-diagonal elements. In this case the off-diagonals do not give a reliable result because of the rounding errors, but the diagonals give $b_x^2 = 0\cdot450$, $b_y^2 = 0\cdot0512$, $b_z^2 = -0\cdot0001$ which would be satisfactory if it were not for the negative sign. It suggests that b_z is really zero. The ratio of b_x^2 to b_y^2 is $8\cdot78$, or $b_x = \pm2\cdot96b_y$, consistent with the actual shifts of $0\cdot9$ and $-0\cdot3$ units in x and y.

If we assume that $\mathbf{E}$ in this case is actually skew-symmetric, the average $(\mathbf{E}-\mathbf{E}^T)/2$ can be used. This gives a better conditioned version of $\mathbf{BB^T}$. Solution using the diagonals gives $b_x^2 = 0\cdot450$, $b_y^2 = 0\cdot0501$, $b_z^2 = +0\cdot00001$, and solution using off-diagonals gives $(0\cdot671, -0\cdot224, 0\cdot004)$, confirming that the y displacement is negative and $b_x = -2\cdot994b_y$, an accurate result. I cannot explain why the displacement in z is not exactly zero. It may be rounding errors or a consequence of using only 12 points.

As pointed out in Appendix 3, §A3.4, the Moore-Penrose pseudo-inverse of a version of matrix $\mathbf{M}$ can used as an alternative way of calculating $\mathbf{E}$. I have examined this is this case and obtained very close values.

Example 2: Translation in z only. I created 20 object points randomly over the range $\pm1\cdot2$ units in x and y, and $1\cdot3$ to $5\cdot5$ in z, and calculated their images when the second camera is moved forwards by $0\cdot4$ in z. Translation forwards in z increases all u and v values. Proceeding as in Example 1, the smallest eigenvalue of $\mathbf{M^T M}$ is essentially zero and its eigenvector gives

$$\mathbf{E} = \begin{pmatrix} 0\cdot000 & -0\cdot707 & 0\cdot005 \\ 0\cdot707 & -0\cdot001 & 0\cdot000 \\ -0\cdot006 & -0\cdot001 & -0\cdot000 \end{pmatrix}, \qquad \mathbf{EE^T} = \begin{pmatrix} 0\cdot500 & 0\cdot001 & 0\cdot000 \\ 0\cdot001 & 0\cdot500 & -0\cdot004 \\ 0\cdot000 & -0\cdot004 & 0\cdot000 \end{pmatrix}.$$

These values show that there is no rotation and that z is the only displacement, though its sign is ambiguous and must be checked by applying the matrix $\mathbf{B}$ to a couple of test points. Again the averaged skew-symmetric matrix $(\mathbf{E} - \mathbf{E}^{\mathbf{T}})/2$ could be examined.

Example 3: Rotation about y with translation in x and z. The rotation is about the axis of camera 2 as shown in Figure 8. The co-rdinates of $\mathbf{b}$ depend on the camera frame to which it is referred. Both are illustrated in Figure 8.

$$x_2 = (x_1 - b_{x1})\cos\theta + (z_1 - b_{z1})\sin\theta, \qquad z_2 = -(x_1 - b_{x1})\sin\theta + (z_1 - b_{z1})\cos\theta. \tag{13}$$

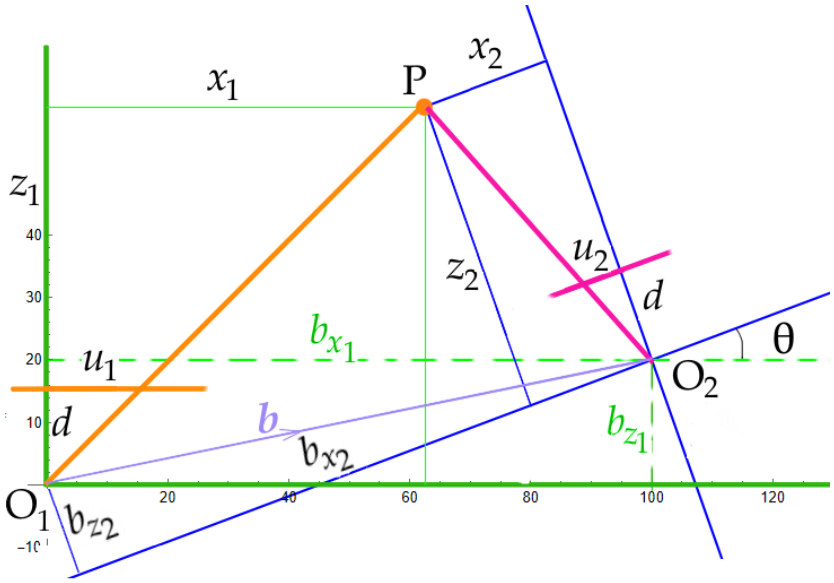

Figure 8: Co-ordinates with camera 2 moved by $\mathbf{b}$ and turned through θ about y as axis.

Pnt	1	2	3	4	5	6	7	8	9	10	11	12
x_1	−0·692	−0·035	0·726	1·076	0·158	0·036	1·116	−0·551	0·266	−1·075	0·794	0·762
y_1	0·527	−0·122	0·755	0·190	1·141	−0·933	0·537	−0·886	−0·360	−1·010	−0·283	−0·723
z_1	1·269	3·857	3·105	6·733	3·893	2·952	2·491	3·490	4·048	3·692	2·282	4·182
u_1	−927	−15	398	272	69	21	762	−269	112	−495	592	310
v_1	706	−54	413	48	498	−537	367	−431	−151	−465	−211	−294
x_2	−1·224	0·278	0·736	2·306	0·472	0·036	0·893	−0·333	0·626	−0·755	0·519	1·138
y_2	0·527	−0·122	0·755	0·190	1·141	−0·933	0·537	−0·886	−0·360	−1·010	−0·283	−0·723
z_2	1·583	3·790	2·823	6·112	3·759	2·915	2·113	3·622	3·867	3·991	2·026	3·823
u_2	−1315	125	443	641	213	21	718	−156	275	−322	435	506
v_2	566	−55	455	53	516	−544	432	−416	−158	−430	−237	−321

Table 7

Table 7 lists the co-ordinates of 12 object points in the frames of cameras 1 and 2, when camera 2 is moved 1 unit to the right and forwards 0·2 units, and rotated by +20° as in Figure 8. From these (u, v) values we form the matrix $\mathbf{M}^{\mathbf{T}}\mathbf{M}$, as before, solve for its smallest eigenvalue $(2 \cdot 6 \times 10^{-7} \approx 0)$, find its eigenvector (for which I used Mathematica), and hence construct $\mathbf{E}$ and $\mathbf{E}\mathbf{E}^{\mathbf{T}}$. I show in Appendix 4 that this measured $\mathbf{E}$ should be a multiple of

the theoretical $\mathbf{E}$ for this translation and rotation, which is

$$\mathbf{E}_{theory} = \begin{pmatrix} 0 & 0\cdot1541 & 0 \\ 0\cdot20 & 0 & -1\cdot00 \\ 0 & 1\cdot0081 & 0 \end{pmatrix} \qquad \mathbf{EE^T}_{theory} = \begin{pmatrix} 0\cdot02374 & 0 & 0\cdot1553 \\ 0 & 1\cdot04 & 0 \\ 0\cdot1553 & 0 & 1\cdot01626 \end{pmatrix}.$$

I have therefore normalised $\mathbf{E}$ from the calculated eigenvector so that element 2,3 is −1.

$$\mathbf{E} = \begin{pmatrix} 0\cdot001 & 0\cdot158 & 0\cdot000 \\ 0\cdot194 & -0\cdot000 & -1\cdot000 \\ -0\cdot000 & 1\cdot008 & 0\cdot000 \end{pmatrix}, \qquad \mathbf{EE^T} = \begin{pmatrix} 0\cdot024 & -0\cdot000 & 0\cdot152 \\ -0\cdot000 & 0\cdot992 & -0\cdot000 \\ 0\cdot152 & -0\cdot000 & 0\cdot972 \end{pmatrix}.$$

This 'measured' matrix is sufficiently close to the theoretical for $\mathbf{B}$ to be recovered. It may be that taking more than 12 points improves the fit. The example gives a warning of the care needed in measuring positions from an image, and of the errors to be expected.

4 Feature extraction with SIFT

Both 3D reconstruction from parallax and panoramic image stitching first require that small features in overlapping photographs be identified and matched picture-to-picture. What is involved? A 'feature' or 'key point' will be a small patch of the picture which is distinctive on account of its shape, colour change or contrast. An edge is not a good feature because it is much the same along its length, but a small kink or dent in an edge might be good. In my examples in Figures 3 and 5 I picked out key points by eye and hand, and that is how things were done with aerial reconnaissance photographs in World War II and during the Cold War. To automate this is challenging; the algorithm should be able to recognise the same feature at different enlargements, since changes of scale will occur as the view point changes, and also at different rotations. The Scale-Invariant Feature Transform (SIFT) is a powerful algorithm for picking out distinctive features in an image, and has these highly desirable properties of being insensitive to scale and rotation and indeed to image brightness and grain. It is widely used in computer vision – for example, for recognising a flat object in a collection of other objects, even when the target object is partly covered by other items[5].

In SIFT each feature is identified as a circular patch, called a 'blob', drawn over the photograph and covering a small region which has distinctive characteristics. SIFT has four main stages:

1. scanning the image to detect candidate features, then drawing blobs around them,
2. sorting them so that weak, spurious and misleading ones are eliminated,
3. determining a principal orientation for each blob,
4. labelling each blob with a signature descriptor which, nominally, is unique.

[5] SIFT works less well with convex and concave objects because of the large change in appearance with angle of view.

A clear account for the lay reader of all these stages is given by Edmund Weitz at *https:/weitz.de/sift* and a more detailed mathematical account is by Ives Oteo and Mauricio Delbracio[6]. There is an excellent series of YouTube videos by Prof. Shree Nayar of Columbia University as part of a course on computer vision. Here I give only a brief account.

The idea behind feature-detecting algorithms is to compare two versions of the same image, one slightly modified, and find the differences. For instance, if image 2 is shifted in the u direction by a few pixels and subtracted from the unshifted image 1, most of the result will be zero (black) and only edges perpendicular to the u axis will show as grey or coloured lines in the v direction. Since a feature will generally be distinctive in all directions, not just u, the shift at each point should also be in all directions. For many years the favoured modification has been convolution with the second derivative of a 2D Gaussian (normal) function (its Laplacian). This has radial symmetry; it smooths out noise and gives a narrow peak centred where the contrast in greyness or colour is greatest. In practice convolution is performed instead with two Gaussians of different widths σ and their difference calculated. This is because the difference of two Gaussians is a fair approximation to the Laplacian as explained in Appendix 5, §A5.1, and is faster to calculate. Convolution with a Gaussian is called 'Gaussian blurring'. Subtracting a blurred image from the same image convoluted with smaller σ will reveal edges irrespective of their direction, as well as other features which differ sharply from the background. The 'difference of Gaussians', DoG, algorithm is implemented as an edge-finding filter in GIMP and other image manipulation programs.

SIFT builds scale invariance into feature extraction with DoG by blurring the image many times using a range of σ values. The process is to increase σ in say 6 steps, then resample the image on a coarser scale, increase σ again in several steps, again resample to shrink the image, and so on. This resampling is called 'down-sampling' because is removes the finer detail, decreasing the resolution and therefore causing the algorithm to look over a larger scale. Down-sampling continues through 4 or 5 'octaves', with increments in σ at each octave, until the image has been reduced to just a few pixels. A matrix of differenced images is thereby generated on many length scales. Images in the matrix are chosen in turn and examined pixel-by pixel to locate local maxima and minima in brightness, which indicate the centres of candidate features. The process is to compare each pixel with its 8 neighbours in the same image and with the corresponding pixels in the two adjacent blurred images in the same row of the matrix. If that pixel's maximum or minimum exceeds that of all its neighbours, the pixel is accepted as the centre of a blob and a circle drawn around with radius proportional to σ for that image. The location of the blob is the position of the maximal pixel. Blobs vary significantly in size as given by σ, which is a measure of the extent of that feature.

The sorting of blobs to eliminate weak ones is done in part by setting a threshold on the extrema, either whiteness or blackness, and so retaining only the most marked blobs. This eliminates most of near-grey extremal points which are probably due to spatial noise. A sophisticated algorithm is also applied to eliminate blobs along an edge. Also the RANSAC algorithm may be used to sort the significant from the spurious (see §5).

[6] 'Anatomy of the SIFT method', Image Processing On Line, *https://www.ipol.im/pub/art/2014/82/*

Assigning an orientation and a signature to each blob are done using the gradient of changes between one sub-patch of the blob and its neighbours. Details are given in Appendix 2. Four 8×8 square grids are laid over the circular blob and at each cell the gradient vector of change – in whiteness on a grey scale, and in each colour on an RGB image – is calculated as a finite difference. The predominant gradient direction becomes the blob's assigned direction. Blobs which do not have a dominant orientation are discarded. From this stage onward assigning a signature or 'descriptor' to the blob, and comparisons with its counterparts in another images, are made relative to the blob's principal orientation. This makes SIFT insensitive to the orientation of the whole object.

The magnitudes of the gradient vectors are disregarded. It is their directions over the 8×8 grids which are important. These are recorded on a quantized scale as a bar chart. Numbers corresponding to the levels in the columns in this bar chart or histogram provide a digital character. Suitably normalised, this is the signature of the blob. Normalisation includes referencing the local gradient directions to the assigned principal direction of the blob. The signature descriptor, therefore, is built entirely of digitised local gradients orientations in a small patch around a position of strong change. Blobs in overlapping images are matched by matching their signatures.

Through these stages SIFT is structured to be insensitive to the scale and orientation of an image, to spatial noise, to image brightness, and to confusion from edges. The algorithm performs well, for instance, in recognising a book scattered randomly amongst other books, by matching against a reference picture of its cover.

There are several other algorithms, working on somewhat different principles, for extracting and labelling features. One worth mentioning is AKAZE (Accelerated KAZE, a Japanese word meaning 'wind') which uses the DoH – 'Difference of Hessians' – operation instead of difference of Gaussian blurs. I refer the interested reader to technical articles on the internet. AKAZE is available in Meshroom as an optional alternative to SIFT.

5 Removing spurious matches with RANSAC

Random Sample Consensus (RANSAC) is an algorithm widely used in many data interpretation situations to distinguish genuine data points from 'outliers' – points which are spurious, erroneous or otherwise do not match the user's expectations. Having worked for many years with experimental data, however, I am cautious about labelling any datum an outlier. To my mind such points almost always deserve checking to see why they have the values recorded. There could be many causes – human errors in reading the measuring instruments, errors in calibration, errors during transcription of the numbers, or faults in the samples being studied. However there could be a genuine and unexpected phenomenon, possibly even the discovery of a new effect. The history of science and technology has examples of important discoveries being made because an assiduous scientist thoroughly investigated a supposed outlier – penicillin and radioactivity are two.

That said we are concerned here with matching points P, Q, R, ... identified in one photograph with P′, Q′, R′, ... in a second. Experience of automated methods like SIFT is that many points will be incorrectly paired simply because the respective tiny surrounding areas in the two images happen to look closely similar. RANSAC works by proposing many models of the image pairing in turn, calculating for all object points in one image how they map to the other image under each model, and counting for each model the number which lie within a tolerance of the true image point in the second photograph.

At its simplest the 'model' could be merely the translation vector between any chosen pair of matched points, R, R′ say, specified by displacements in u, v or by distance and angle. The same displacement vector would be applied to P, Q, S, T ... and their mapped positions compared with the paired points P′, Q′, S′, T′ ... respectively. Allowing for parallax one might expect shifts between pairs to differ by x% of screen width, so this could be one criterion for deciding whether the mapped positions were close enough to be accepted. Call the number of accepted point-pairs n for that displacement model R → R′. Perhaps 100 or more other point-pairs would be taken to define 100 or more models, and the vectors calculated and applied in turn. A different set of 'inliers' would probably be obtained at each trial and a different n. The 'consensus' in RANSAC is to accept the model with the highest n. It is that simple in principle.

More sophisticated models can be prescribed. For instance, sets of four points could be taken at a time, and the map from the quadrilateral defined by their vertices in image 1 to image 2 formed (see Appendix 1). All points in image 1 would be mapped under this matrix and the discrepancy from the observed points in image 2 calculated. Again a count of closely adjacent positions would be made and RANSAC would accept the transformation matrix with the largest count. After sufficient trial models all point-pairs will have been categorised as either inliers or outliers. Further analysis of the images would be made only with the inliers.

Commercial implementations of RANSAC contain statistical criteria for how many models must be postulated to obtain high confidence in the categorisation. This depends on the relative number of outliers. The required number of trials increases rapidly once the fraction of outliers exceeds about 20% of the total point-pairs, but the algorithm can be pressed to identify correct data sets amongst over 50% of outliers, an impressive performance.

RANSAC is a type of optimisation algorithm in that it votes for the set of model parameters which maximises n. Optimisation algorithms of all types depend on being able to specify a function which attains its global minimum or maximum at the exact solution, *and* which remains close to that minimum value over a sufficiently wide neighbourhood for the search to be guided by the gradients of the local 'terrain'. An example of a 2-D pathological function is plotted in Figure 9. The requirement is to find its global minimum, but this is like searching for a mineshaft in a hilly landscape. The `FindMinimum` function of Mathematica converges on the broad minimum at $(1 \cdot 034, 0 \cdot 10)$ unless the seeded value is greater than $1 \cdot 82$, very close to the true minimum of $(2, 0)$.

We are most fortunate to have Longuet-Higgin's epipolar geometry to discover the separation in distance and angle of two camera stations, for (in my view) it would be extremely difficult to discover this by some optimisation scheme. One might suppose that, in searching for the best fit of camera co-ordinates to produce the observed paired image points, we have an optimisation problem. It would involved searching for the best fit of camera positions and angles to the all the observed image positions, and this might be attained by minimising a function of the camera's co-ordinates expressed in a function $\mathcal{F}(...)$ of all positions and rotational co-ordinates. $\mathcal{F}(...)$ would be built by aggregating discrepancies between intersecting rays, and be exactly zero when the rays traced backwards from the two cameras intersected precisely at each and every object point. For all but the exactly correct camera positions the rays from each camera to point P would intersect only pairwise, so the value of $\mathcal{F}$ would be greater than zero. However, a few moments trying to adjust two fans of rays drawn on tracing paper will soon persuade the reader that the distance between intersections is smallest when the cameras are turned towards each other such their rays cross at very small range z. An algorithm with minimize $\mathcal{F}(...)$ would gravitate towards spurious and wildly incorrect points of close convergence just in front of the camera. The situation would be a multidimensional version of Figure 9.

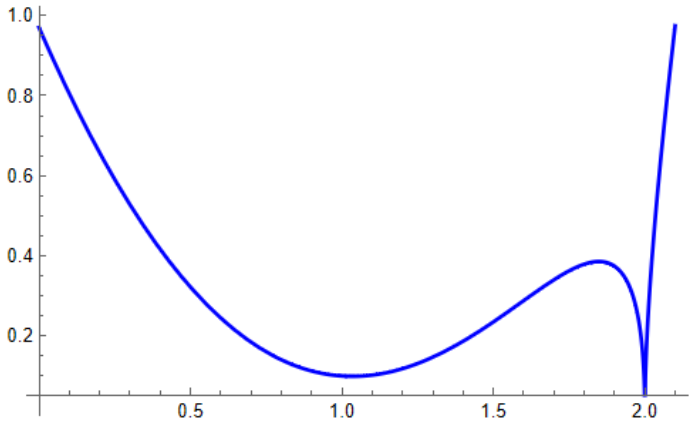

Figure 9: A challenging function to minimize. $(x - \frac{1}{2})^2 + 2 \cdot 1\sqrt{|x-2|} - 2 \cdot 25$.

John Coffey, November 2024.

Appendix 1: Mapping quadrilaterals

In this Appendix I illustrate a theorem in projective geometry that any four projective points A, B, C, D can be mapped into any other four projective points P, Q, R, S provided no three are projectively collinear. This theorem underpins the mapping of one image onto an overlapping one, so fundamental to much of computer vision.

The word 'projective' is important. Recall that in projective geometry a 'p-point' (my name for a projective point) is identical with a line through the origin O in Euclidean space, and a p-line is a Euclidean plane through O. This correspondence arises because in a perspective view of the world, all object points along the same line of sight appear to the viewer behind each other and so are represented in the image by the same point. Projective points are the rays from point objects in the scene through the focal point of the camera. Similarly a p-line in the image is the representation of all object points which lie in a single plane which passes through the focus. A p-point is represented by a ray vector $\mathbf{p}$ and any point on that ray will have the form $\sigma\mathbf{p}$ where σ is a scale factor. A p-point will therefore have the co-ordinate form $(\sigma p_u, \sigma p_v, \sigma)$ and will be recorded as point $(u, v) \equiv (p_u, p_v)$ in the image. (We noted that d, the image-focus distance in Figure 1, is a scaling factor.)

The map taking A to P, B to Q, C to R and D to S is effected by an invertible 3×3 matrix. Recall that a 2×2 matrix can effect rotation about the origin, reflection, scaling and shear but not translation. However, the matrix

$$\begin{pmatrix} 1 & 0 & b_u \\ 0 & 1 & b_v \\ 0 & 0 & 1 \end{pmatrix}$$

will map $(u, v, 1)$ to $(u + b_u, v + b_v, 1)$. An invertible 3×3 matrix adds translation to the operations of its 2×2 counterpart and also adds morphing of a rectangle into a trapezium.

Figure 10: Two views of a greenhouse, from below and above, with 9 common features marked.

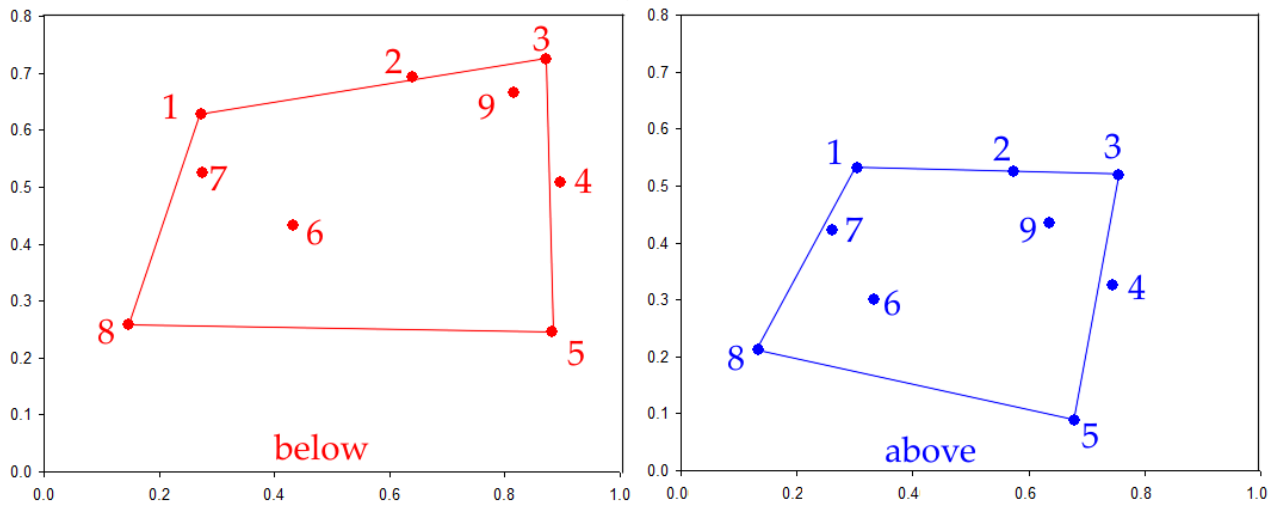

Figure 11: Feature points 1 to 9 in the two greenhouse photographs in the Mathematica convention.

	Screen				Mathematica			
	below		above		below		above	
Point	u	v	u	v	u	v	u	v
1	707	279	742	389	0·273	0·628	0·304	0·531
2	1122	206	1047	395	0·639	0·692	0·573	0·526
3	1385	169	1254	404	0·871	0·725	0·756	0·518
4	1413	416	1242	623	0·896	0·507	0·745	0·325
5	1397	713	1168	890	0·882	0·245	0·680	0·089
6	888	501	776	651	0·433	0·432	0·334	0·300
7	710	395	694	513	0·276	0·526	0·262	0·422
8	564	699	548	750	0·147	0·258	0·133	0·213
9	1322	236	1119	498	0·816	0·666	0·637	0·435

Table 8 : Co-ordinates of the nine feature points in the two greenhouse photographs. Left: in pixels relative to the screen. Right: Scaled and shifted to Mathematica's convention.

Below are three ways to calculate the matrix which maps the corners of a non-degenerate p-quadrilateral ABCD into another, PQRS. The illustration is made by reference to the two photographs of a greenhouse in Figure 10, one from the left and near the ground (the 'below' picture), the other from about 2m above ground to the right (the 'above'). Nine points have been selected and labelled, as listed in Table 8. Their coordinates were read from the screen by placing the mouse cursor over them and noting the screen co-ordinates in pixels using MPos, a useful tool from Sourceforge showing mouse position. The screen has (0,0) at its top left. Screen co-ordinates (u, v) are listed in the left side of Table 8 for both the 'below' and 'above' photographs. Later I use Mathematics to display the morphed photograph and for this it has been necessary to convert the screen co-ordinates to Mathematica's convention, which has $(0, 0)$ at the bottom left of the image, and the image scaled to 1 unit wide and the aspect ratio high, 4/5 in this case. The converted co-ordinates are listed in the right half of Table 8. Figure 11 shows the isolated image points plus the quadrilaterals with vertices 1, 3, 5, 8. The aim is to map the red quadrilateral to the blue.

Method 1: Stepwise algebraic solution In homogeneous co-ordinates let Point 1 in the 'below' photograph be called A = $(a_u, a_v, 1)$, and let it map to Point 1 in the 'above' photograph as P = $(\sigma_p p_u, \sigma_p p_v, \sigma_p)$. A maps to P by matrix **H** according to

$$\begin{pmatrix} \sigma_p p_u \\ \sigma_p p_v \\ \sigma_p \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} a_u \\ a_v \\ 1 \end{pmatrix}$$

and similarly for the pairs B → Q, C → R, D → S. There are 12 unknowns: the 9 elements in **H** and three of the scale factors σ. We can take $\sigma_p = 1$ and determine σ_q, σ_r and σ_s. Method 1 is the tedious algebra of multiplying out the matrix into three simultaneous equations for each pair of points, and solving for one unknown at a time. Thus from pair D:S

$$h_{13} = 0\cdot133\sigma_d - 0\cdot147h_{11} - 0\cdot258h_{12}, \qquad h_{23} = 0\cdot213\sigma_d - 0\cdot147h_{21} - 0\cdot258h_{22}$$

$$h_{33} = \sigma_d - 0\cdot147h_{31} - 0\cdot258h_{32}.$$

Pressing along the chain of substitutions with the help of algebraic software we arrive at

$$\mathbf{H} = \begin{pmatrix} 0\cdot760 & 0\cdot211 & -0\cdot0363 \\ -0\cdot145 & 0\cdot922 & -0\cdot00853 \\ 0\cdot0443 & 0\cdot0475 & 0\cdot958 \end{pmatrix} \quad \text{giving}$$

$$P = \begin{pmatrix} 0\cdot304 \\ 0\cdot531 \\ 1 \end{pmatrix}, \qquad Q = \begin{pmatrix} 0\cdot779 \\ 0\cdot534 \\ 1\cdot031 \end{pmatrix}, \qquad R = \begin{pmatrix} 0\cdot686 \\ 0\cdot090 \\ 1\cdot009 \end{pmatrix}, \qquad S = \begin{pmatrix} 0\cdot130 \\ 0\cdot208 \\ 0\cdot977 \end{pmatrix}.$$

Converting from homogeneous to Cartesian co-ordinates, P = $(0\cdot304, 0\cdot531)$, Q = $(0\cdot756, 0\cdot518)$, R = $(0\cdot680, 0\cdot089)$, S = $(0\cdot133, 0\cdot213)$ exactly as in Table 8 for Points 1, 3, 5, 8.

Using the `ImagePerspectiveTransform` function in Mathematica I have morphed the 'below' photograph with **H** to obtain the left picture in Figure 12. Points 1, 3, 5, 8 do indeed fall exactly over those in the 'above' photograph, and there is now a fair amount of similarity between the pictures, especially along the roof line. However, as we should expect, many features such as Point 9 on the door frame remain far from their corresponding ones.

Method 2: Two-stage approach This approach takes advantage of the fact that the matrix **J** which maps three points A, B, C to the triangle with vertices $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ is constructed simply of **a**, **b**, **c** as its columns. Similarly suppose **K** maps P, Q, R to $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ respectively. Then $\mathbf{KJ^{-1}}$ will map A to P, B to Q, C to R. The fourth pair D → S is used to find the three unknown scaling factors, and it does this via the intermediate point $(1,1,1)$. I do not claim that the computational effort in this method is significantly less that in Method 1, but some stages are simpler because the inverse of a 3×3 matrix is not complicated and readily coded.

Take the same four points 1, 3, 5, 8 as for Method 1 and form the matrix from them as its columns, weighted by scale factors η_j:

$$\mathbf{J} = \begin{pmatrix} 0\cdot273\,\eta_a & 0\cdot871\,\eta_b & 0\cdot882\,\eta_c \\ 0\cdot628\,\eta_a & 0\cdot725\,\eta_b & 0\cdot245\,\eta_c \\ \eta_a & \eta_b & \eta_c \end{pmatrix}.$$

Figure 12: The 'below' picture of Figure 9 morphed (left) to match the 'above' photograph (right) at points 1, 3, 5, 8.

This maps $(1,0,0)$ to A, $(0,1,0)$ to B, $(0,0,1)$ to C in homogeneous co-ordinates. Now choose the various η_j so that $\mathbf{J}.(1\ 1\ 1)^T$ is point D. This can be written as $\mathbf{F}.\eta_j = \mathbf{d}$ or, in full,

$$\begin{pmatrix} 0\cdot273 & 0\cdot871 & 0\cdot882 \\ 0\cdot628 & 0\cdot725 & 0\cdot245 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \eta_a \\ \eta_b \\ \eta_c \end{pmatrix} = \begin{pmatrix} 0\cdot147 \\ 0\cdot258 \\ 1 \end{pmatrix}.$$

It is straightforward to invert the 3×3 matrix $\mathbf{F}$ and apply it to $\mathbf{d}$ to obtain the three η. $\mathbf{F^{-1}}$ and the η_j are

$$\begin{pmatrix} -1\cdot667 & -0\cdot037 & 1\cdot48 \\ 1\cdot330 & 2\cdot113 & -1\cdot691 \\ 0\cdot337 & -2\cdot077 & 1\cdot212 \end{pmatrix} \begin{pmatrix} 0\cdot147 \\ 0\cdot258 \\ 1 \end{pmatrix} = \begin{pmatrix} 1\cdot224 \\ -0\cdot951 \\ 0\cdot727 \end{pmatrix}.$$

With these weightings, $\mathbf{J}$ is

$$\begin{pmatrix} 0\cdot335 & -0\cdot828 & 0\cdot641 \\ 0\cdot769 & -0\cdot689 & 0\cdot178 \\ 1\cdot224 & -0\cdot951 & 0\cdot727 \end{pmatrix}.$$

We now carry out exactly the same procedure with the target points P, Q, R, S to find a matrix $\mathbf{K}$ which maps $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ to P, Q and R respectively, and then $(1,1,1)$ to the fourth point S. The result is

$$\mathbf{K} = \begin{pmatrix} 0\cdot381 & -0\cdot758 & 0\cdot510 \\ 0\cdot665 & -0\cdot520 & 0\cdot0669 \\ 1\cdot253 & -1\cdot003 & 0\cdot751 \end{pmatrix}.$$

The required map from the 'below' photograph to the 'above' at points 1, 3, 5, 8 is

$$\mathbf{KJ^{-1}} = \begin{pmatrix} 0\cdot778 & 0\cdot216 & -0\cdot037 \\ -0\cdot148 & 0\cdot944 & -0\cdot009 \\ 0\cdot045 & 0\cdot049 & 0\cdot981 \end{pmatrix}.$$

This has different elements from **H** calculated by Method 1, but one is merely a scalar multiple of the other so both represent the same points in homogeneous co-ordinates.

Method 3: Matrix row reduction This method carries out Method 1 using matrices. Its importance lies in its capacity for extending to many more than four pairs of points, as outlined in Appendix 2, and thereby giving a least squares best fit matrix for morphing one whole image into another.

Let **H** again be the transformation matrix of A → P, B → Q, C → R, D → S. Let $p_u = p_1/p_3,\ p_v = p_2/p_3$ in homogeneous co-ordinates and similarly for the vectors representing Q, R and S. For point P, expand the relation $\mathbf{p} = \mathbf{Ha}$ and use $p_3 = a_u h_{31} + a_v h_{31} + h_{31}$ to obtain the following two rows of a matrix multiplying the vector $\mathbf{h} = (h_{11}, h_{12}, h_{13}, h_{21}, ...h_{33})^T$:

$$\begin{pmatrix} a_u & a_v & 1 & 0 & 0 & 0 & -p_u a_u & -p_u a_v & -p_u \\ 0 & 0 & 0 & a_u & a_v & 1 & -p_v a_u & -p_v a_v & -p_v \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The equivalent rows for the other three mapped points stack together to form an 8×9 column matrix **M** satisfying $\mathbf{M}.\mathbf{h} = 0$. In our case **M** is

$$\begin{pmatrix} 0\cdot273 & 0\cdot628 & 1 & 0 & 0 & 0 & -0\cdot083 & -0\cdot191 & -0\cdot304 \\ 0 & 0 & 0 & 0\cdot273 & 0\cdot628 & 1 & -0\cdot145 & -0\cdot333 & -0\cdot531 \\ 0\cdot871 & 0\cdot725 & 1 & 0 & 0 & 0 & -0\cdot658 & -0\cdot548 & -0\cdot756 \\ 0 & 0 & 0 & 0\cdot871 & 0\cdot725 & 1 & -0\cdot451 & -0\cdot375 & -0\cdot518 \\ 0\cdot882 & 0\cdot245 & 1 & 0 & 0 & 0 & -0\cdot600 & -0\cdot167 & -0\cdot680 \\ 0 & 0 & 0 & 0\cdot882 & 0\cdot245 & 1 & -0\cdot079 & -0\cdot022 & -0\cdot089 \\ 0\cdot147 & 0\cdot258 & 1 & 0 & 0 & 0 & -0\cdot020 & -0\cdot034 & -0\cdot133 \\ 0 & 0 & 0 & 0\cdot147 & 0\cdot258 & 1 & -0\cdot031 & -0\cdot055 & -0\cdot213 \end{pmatrix}$$

This can be row reduced to echelon form by the standard pivoting method, a procedure which is readily coded for machine numerical implementation. The reduced form gives the values of eight elements h_{11} to h_{32} in terms of h_{33}, which can be set to 1. We obtain

$$\mathbf{H} = \begin{pmatrix} 0\cdot793 & 0\cdot221 & -0\cdot0379 \\ -0\cdot151 & 0\cdot963 & -0\cdot0089 \\ 0\cdot0463 & 0\cdot0496 & 1 \end{pmatrix}.$$

This is a scaled version of the matrix produced by both methods 1 and 2.

Appendix 2: Panoramic stitching

It is quite popular nowadays to combine two or more photographs of a holiday scene into one wide panorama to capture its extent and grandeur. The process requires that the images be distorted to fit well together. Adjustments in brightness and contrast are also usually needed to smooth the transition, though I do not discuss this aspect here. This Appendix extends Method 3 of Appendix 1, on mapping any projective quadrilateral into another, to allow many more pairs of points to be included. No paired points will fit exactly, but rather a least-squares best fit is obtained across all points.

Two images taken with the same camera satisfy the condition that all rays from objects points have passed through the focal point. Each pair of (u, v) co-ordinates is expressed in homogeneous co-ordinates (u^*, v^*, w^*) such that $u = u^*/w^*$, $v = v^*/w^*$. The set of points in the source image $\mathbf{U} = (u_j, v_j)$ are mapped to the corresponding points $\mathbf{W} = (s_j, t_j)$ in the second image by a 3×3 invertible matrix $\mathbf{H}$ with elements h_{ij}. Thus in principle $\mathbf{W} - \mathbf{HU} = \mathbf{0}$, but errors of measurements and the near impossibility of matching many pairs exactly means that in the general case $\mathbf{W} - \mathbf{HU} = \mathbf{\Delta} \neq \mathbf{0}$. The task is to determine the h_{ij} to minimise $|\mathbf{\Delta}|$ subject to the constraint $|\mathbf{H}^2| = 1$, so as to avoid the trivial solution $\mathbf{H} \equiv 0$. This type of problem occurs in many places in physics, engineering, finance, and is called a 'constrained least squares' problem. It is solved by the method of Lagrange multipliers as follows.

A2.1 Constrained optimisation with Lagrange multiplier

As in Method 3 of Appendix 1 the auxiliary matrix $\mathbf{M}$ is constructed from the equation $\mathbf{W} = \mathbf{HU}$ such that $\mathbf{Mh} = \mathbf{0}$. $\mathbf{h}$ is the column vector formed from the 9 elements of $\mathbf{H}$, and $\mathbf{M}$ is an $2n \times 9$ matrix whose elements are built from the co-ordinates of the n matching pairs. For a least squares optimisation we must minimize $|\mathbf{Mh}|^2$ whilst constraining $|\mathbf{h}^2| = 1$. Lagrange observed that a general function f of variables $s, t, ...x, y, z$ subject to the constraint $g(s, t, ...x, y, z) = K$, K a constant, will have stationary points – usually maxima or minima – wherever f and g vary together, in the same way, as the variables change values. Geometrically, curves representing $f(..)$ and $g(..)$ become tangent to each other at the maxima and minima of f[7]. Tangency occurs where the gradients are equal, or at least proportional. Lagrange introduced a parameter λ so to express this proportionality for each partial derivative. Thus $\partial f/\partial s = \lambda\, \partial g/\partial s$, ... $\partial f/\partial z = \lambda\, \partial g/\partial z$. The simultaneous equations so created are solved for λ, and the variables at the stationary points evaluated from the relation $g(..) = K$. The method is profound, ingenious and has been widely used since Lagrange first proposed it over 200 years ago.

In our case the variables are h_{ij}, the function is $f = |\mathbf{Mh}|^2$ and the constraint is $g = |\mathbf{h}^2| - 1 = 0$. Though the partial differentiation can be done symbolically, I will spell out the steps because differentiating with respect to a vector is not obvious. Let ^{T} denote a matrix

[7] You can persuade yourself of this by observing that the largest and smallest values of the line $f = mx - y$ subject to $x^2 + y^2 = 1$ occur where $m = -x/y$; that is, where the gradient of the line equals the gradient of the circle.

transpose and take the derivatives of $|\mathbf{h^2}|$.

$$|\mathbf{h^2}| = \mathbf{h^T h}, \qquad \frac{\partial(\mathbf{h^T h})}{\partial h_{11}} = \mathbf{h^T}\frac{\partial \mathbf{h}}{dh_{11}} + \frac{\partial \mathbf{h^T}}{dh_{11}}\mathbf{h} = \mathbf{h^T}.\mathbf{I_1} + \mathbf{I_1^T}.\mathbf{h} = h_{11} + h_{11}$$

where $\mathbf{I_1}$ is the column matrix $(1\ 0\ 0\ ...0\ 0)^T$ with the 1 in the position corresponding to h_{11}. If the relations for the other h_{ij} are stacked, the result can be written as

$$\frac{d|\mathbf{h^2}|}{d\mathbf{h}} = 2\mathbf{h},$$

an equation we might have guessed by analogy with scalar calculus. It is to be interpreted as differentiation with respect to all elements of $\mathbf{h}$ down the column vector.

The $f(...)$ is differentiated similarly:

$$||\mathbf{Mh}|^\mathbf{2} = (\mathbf{Mh})^\mathbf{T}(\mathbf{Mh}) = \mathbf{h^T M^T Mh},$$

$$\frac{\partial(\mathbf{h^T M^T Mh})}{\partial h_{11}} = \mathbf{h^T M^T M}\frac{\partial \mathbf{h}}{dh_{11}} + \frac{\partial \mathbf{h^T}}{dh_{11}}\mathbf{M^T Mh} = \mathbf{h^T M^T M}.\mathbf{I_1} + \mathbf{I_1^T}.\mathbf{M^T Mh}.$$

Now $\mathbf{h^T M^T M}.\mathbf{I_1}$ is the first row of $\mathbf{h^T M^T M}$ and $\mathbf{I_1^T}.\mathbf{M^T Mh}$ is the first row of $\mathbf{M^T Mh}$. Both terns are scalars and they are equal. Differentiation with respect to h_{ij} picks out the equivalent rows in $\mathbf{M^T Mh}$. By stacking these we may notate the result as

$$\frac{\partial|\mathbf{Mh}|^2}{\partial h} = 2\mathbf{M^T Mh}.$$

Again we might have guessed this. The Lagrange equation of partial derivatives becomes

$$(\mathbf{M^T M})\mathbf{h} = \lambda\mathbf{h}$$

which is an eigenvalue equation. The eigenvalues will give the various stationary points of $|\mathbf{Mh}|^2$. Which eigenvalue do we choose? The aim is to minimise $|\mathbf{Mh}|^\mathbf{2} = \mathbf{h^T M^T Mh}$ subject to $\mathbf{h^T h} = 1$. If $(\mathbf{M^T M})\mathbf{h} = \lambda\mathbf{h}$, then $\mathbf{h^T}(\mathbf{M^T Mh}) = \mathbf{h^T}\lambda\mathbf{h} = \lambda$. We therefore choose the smallest λ. $\lambda_{min} = \Delta$, the least squares error. The components of its eigenvector are the required h_{ij}.

This establishes the projective transformation, though still leaves finding the smallest eigenvalue of a 9×9 matrix. To illustrate the mapping in action we find the best matrix to morph the 'below' image of the greenhouse, Figure 10 of Appendix 1, to the 'above' using all nine points in Table 8.

A2.2 Example of image mapping with many points

Before using all nine points, let us first use only Points 1, 3, 5, 8 as in Appendix 1, Method 3. Since an exact fit can be made, we expect the smallest eigenvalue to be zero. Matrix $\mathbf{M}$ is given under Method 3, Appendix 1. $\mathbf{M^T M}$ is 9×9. Mathematica gives the following eigenvalues:

$$9 \cdot 00,\quad 6 \cdot 19,\quad 0 \cdot 727,\quad 0 \cdot 397,\quad 0 \cdot 260,\quad 0 \cdot 167,\quad 0 \cdot 011,\quad 0 \cdot 0053,\quad 6 \times 10^{-17}.$$

The second and third smallest eigenvalues must each correspond to a fairly good fit. The eigenvector of the zero eigenvalue gives the elements of $\mathbf{H_4}$ according to the four points:

$$\mathbf{H_4} = \begin{pmatrix} 0\cdot489 & 0\cdot136 & -0\cdot0234 \\ -0\cdot0931 & 0\cdot593 & -0\cdot00549 \\ 0\cdot0285 & 0\cdot0306 & 0\cdot616 \end{pmatrix},$$

and this is precisely a scaled version of the $\mathbf{H}$ found in Appendix 1.

We now form the 18-row matrix $\mathbf{M}$ using the nine points in Mathematica notation in Table 8. I will not list this large matrix. Mathematica gives the eigenvalues of $\mathbf{M^T M}$ to be:

$$21\cdot6,\quad 14\cdot7,\quad 1\cdot15,\quad 0\cdot618,\quad 0\cdot338,\quad 0\cdot224,\quad 0\cdot0153,\quad 0\cdot00825,\quad 0\cdot00292$$

The eigenvector for the smallest eigenvalue gives the transformation matrix

$$\mathbf{H_9} = \begin{pmatrix} 0\cdot437 & 0\cdot111 & -0\cdot00455 \\ -0\cdot126 & 0\cdot564 & 0\cdot0173 \\ -0\cdot0776 & 0\cdot00747 & 0\cdot676 \end{pmatrix}.$$

It is not very different from $\mathbf{H_4}$ above. The right panel of Figure 13 is the 'below' image of the greenhouse morphed by this, to be compared with the left image using only the four points of Figure 12 left.

Figure 13: Left: 'below' photograph morphed using $\mathbf{H_4}$. Right: morphed using $\mathbf{H_9}$.

In another article[8] on *mathstudio.co.uk* I discuss numerical methods for finding the eigenvalues of a matrix. 9×9 is not a particularly large matrix. John Francis's shift algorithm, which builds on QR decomposition, is perhaps the fastest method for real eigenvalues. There is a numerical example of its application to a 5×5 matrix in §10.2 of that article.

[8] 'Iterative numerical methods for real eigenvalues and eigenvectors of matrices.'

Appendix 3: Matrix SVD and pseudo-inverse

I do not make direct use of SVD in this article, but it is allied to matrix techniques which are used. SVD (Singular Value Decomposition) is a powerful and widely used procedure in linear algebra to factor a matrix – any matrix – into three matrix factors. It has many uses, but one major appeal is that the columns of the factor matrices have a hierarchy in which the gross, coarse features of the given matrix are encoded in the first columns with the finer details following in subsequent ones. This hierarchical structure means that most of the relevant information held in a large matrix can be captured in only a modest percentage of its columns, so allowing a vast matrix, as might come from a digitized photograph, to be represented by only the most significant fraction of its SVD columns. Thus a large data matrix can be boiled down to its essentials, and stored and manipulated as a modest sub-matrix of its decomposition. The method has been likened to Fourier series decomposition of a periodic or quasi-periodic function into a sum of trigonometry functions with decreasing wavelength. SVD splits the given matrix into a decreasing sequence of rank 1 matrices which can be truncated without much loss of detail. The procedure is applied commercially to data reduction including the compression of photographic images, to pattern matching and face recognition. It is also one way to calculate the pseudo-inverse, also called the Morse-Penrose inverse, of a rectangular matrix, and this is used widely in least squares fitting of models to large data sets.

SVD is well described in textbooks and on the internet so I will limit my explanation to two worked examples after the following introduction.

A3.1 Similarity transformations

Some of the motivation for SVD comes from the familiar factorisation of a square matrix into a diagonal matrix pre- and post-multiplied by orthogonal matrices in a similarity transformation $\mathbf{A} = \mathbf{PDP^{-1}}$. Here $\mathbf{D}$ is diagonal with the eigenvalues of $\mathbf{A}$ down the diagonal and $\mathbf{P}$ has as its columns the eigenvectors of $\mathbf{A}$. I propose to build on this analogy with a similarity transformation $\mathbf{P^{-1}AP} = \mathbf{D}$, so here is an example.

$$\text{Take} \quad \mathbf{A} = \begin{pmatrix} 3 & 2 & -3 \\ 1 & -2 & 0 \\ 2 & -6 & 1 \end{pmatrix}.$$

Subtracting $\lambda\mathbf{I_3}$ and forming the determinant gives the characteristic equation $\lambda^3 - 2\lambda^2 - \lambda + 2 = 0$, with roots 1, 2, −1. Let the eigenvector be the column $\mathbf{h} = (h_1, h_2, h_3)^T$. Substitute the three λ in turn into $\mathbf{Ah} = \mathbf{0}$ and solve the three simultaneous equations for relations between the h_j. The eigenvectors are found to be $(4, 1, 2)^T$, $(9, 3, 8)^T$ and $(1, 1, 2)^T$ corresponding to eigenvalues 2, 1 and −1 respectively. Note that they are not orthogonal – only a real symmetric matrix has orthogonal eigenvectors. Form them into the matrix $\mathbf{P}$ and take its inverse:

$$\mathbf{P} = \begin{pmatrix} 4 & 9 & 1 \\ 1 & 3 & 1 \\ 2 & 8 & 2 \end{pmatrix}, \qquad \mathbf{P^{-1}} = \frac{1}{6}\begin{pmatrix} 2 & 10 & -6 \\ 0 & -6 & 3 \\ -2 & 14 & -3 \end{pmatrix},$$

and indeed $\mathbf{P^{-1}AP = D}$ is the diagonal matrix with elements $2, 1, -1$ in that order. The effect of this factorisation of $\mathbf{A}$ is to rotate the quantity which $\mathbf{A}$ represents, stretch or shrink it by its eigenvalues as scale factors along its principal axes, then rotate it back to its original orientation by applying the same rotation in reverse.

SVD generalises this to non-square matrices $\mathbf{A}$. In order for the number of rows and columns to conform to matrix multiplication it uses two *different* pre- and post- orthonormal matrices. The standard notation is that the given matrix $\mathbf{A}$ is factored into $\mathbf{U\Sigma V^T}$. $\mathbf{A}$ can be square or $m \times n$, $m < n$ or $m > n$. $\mathbf{U}$ and $\mathbf{V^T}$ are the orthonormal matrices and $\mathbf{\Sigma}$ is a diagonal matrix constructed from eigenvalues. The following examples illustrate the procedure.

A3.2 Example 1: Square non-degenerate matrix

Let $\mathbf{A}$ be the matrix used above. $\mathbf{U}$ and $\mathbf{V}$ will be derived from the symmetric matrices $\mathbf{AA^T}$ and $\mathbf{A^TA}$ respectively. First $\mathbf{V}$ is isolated and determined, then $\mathbf{U}$. $\mathbf{V}$ is calculated from $\mathbf{A^TA = V\Sigma^T U^T U\Sigma V^T = VDV^T}$ since $\mathbf{U^T = U^{-1}}$ for an orthogonal matrix. This is a similarity transformation like the one above, but uses the fact that a symmetric matrix has orthogonal eigenvectors, and the inverse of $\mathbf{V}$ (which takes the role of $\mathbf{P}$ above) equals its transpose, greatly simplifying the calculation. The characteristic equation, eigenvalues and eigenvectors of $\mathbf{A^TA}$ are readily found:

$$\mathbf{A^TA} = \begin{pmatrix} 14 & -8 & -7 \\ -8 & 44 & -12 \\ -7 & -12 & 10 \end{pmatrix}, \qquad \lambda^3 - 68\lambda^2 + 939\lambda - 4 = 0, \quad \lambda = 48 \cdot 73,\ 19 \cdot 26,\ 0 \cdot 00426.$$

Interestingly, one eigenvalue is almost zero, making the matrix almost degenerate. $\mathbf{AA^T}$ has the same eigenvalues but different eigenvectors. After finding the eigenvectors, they are normalised to unit modulus and formed into $\mathbf{V}$ and $\mathbf{V}^T$:

$$\mathbf{V^T} = \begin{pmatrix} 0 \cdot 166 & -0 \cdot 950 & 0 \cdot 264 \\ -0 \cdot 811 & 0 \cdot 0215 & 0 \cdot 585 \\ 0 \cdot 561 & 0 \cdot 311 & 0 \cdot 767 \end{pmatrix}.$$

This is orthonormal; its rows form an orthogonal set, and so do its columns. As expected $\mathbf{V^TA^TAV = D}$, where $\mathbf{D}$ is diagonal with the eigenvalues in order of decreasing magnitude. For SVD, however, we need only $\mathbf{V^T}$.

The diagonal elements of $\mathbf{\Sigma}$ are the square roots of the eigenvalues λ. This might be suspected because of the squaring contained in $\mathbf{A^TA}$. Symbolically $\mathbf{\Sigma} = \sqrt{\mathbf{D}}$.

$$\mathbf{\Sigma} = \begin{pmatrix} 6 \cdot 981 & 0 & 0 \\ 0 & 4 \cdot 389 & 0 \\ 0 & 0 & 0 \cdot 06528 \end{pmatrix}.$$

In $\mathbf{\Sigma}$ the eigenvalues are placed in decreasing order and in $\mathbf{V}$ the eigenvectors are in the same order. Thus the top row of $\mathbf{V^T}$ corresponds with the largest diagonal element in $\mathbf{\Sigma}$.

By a similar procedure we find $\mathbf{U}$, using this time $\mathbf{AA^T}$ to eliminate $\mathbf{V}$:

$$\mathbf{AA^T} = \begin{pmatrix} 22 & -1 & -9 \\ -1 & 5 & 14 \\ -9 & 14 & 41 \end{pmatrix}.$$

As already noted, this has the same eigenvalues as $\mathbf{A^TA}$ since for any two matrices $\mathbf{BC}$ and $\mathbf{CB}$ have the same eigenvalues. On determining its eigenvectors and normalising them whilst retaining the decreasing order, we arrive at the SVD factorisation

$$\begin{pmatrix} -0\cdot315 & -0\cdot944 & 0\cdot0969 \\ 0\cdot296 & -0\cdot195 & -0\cdot935 \\ 0\cdot902 & -0\cdot266 & 0\cdot341 \end{pmatrix} \begin{pmatrix} 6\cdot981 & 0 & 0 \\ 0 & 4\cdot389 & 0 \\ 0 & 0 & 0\cdot0653 \end{pmatrix} \begin{pmatrix} 0\cdot166 & -0\cdot950 & 0\cdot264 \\ -0\cdot811 & 0\cdot0215 & 0\cdot585 \\ 0\cdot561 & 0\cdot311 & 0\cdot767 \end{pmatrix}.$$

This multiplies out to $\mathbf{A}$, though there is the subtlety that the signs of the columns have to be correctly chosen for this to be the case. This factorisation is quite different from the similarity transformation $\mathbf{PDP^{-1}}$ of §A3.1.

To be clear, the left column of $\mathbf{U}$ is associated with the largest eigenvalue in $\mathbf{\Sigma}$, as is the top row in $\mathbf{V^T}$. $\mathbf{\Sigma}$ can be split into the sum of three matrices, each containing just one diagonal element:

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} 6\cdot981 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{V^T} + \mathbf{U}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4\cdot389 & 0 \\ 0 & 0 & 0 \end{pmatrix}\mathbf{V^T} + \mathbf{U}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0\cdot0653 \end{pmatrix}\mathbf{V^T}.$$

The last of these products contributes less than 5% to the elements of $\mathbf{A}$. The contribution from the largest eigenvalue is

$$\begin{pmatrix} -0.36 & 2.09 & -0.58 \\ 0.34 & -1.96 & 0.54 \\ 1.04 & -5.98 & 1.66 \end{pmatrix}.$$

This and the other two matrix terms are rank 1, as their rows are merely multiples of each other. This illustrates how SVD separates the given matrix into a sequence of rank 1 terms, with each contributing less to the whole than its previous term. Thus 80% of the content and function of a matrix might be stored in 20% or less of its terms.

A3.3 Example 2: More rows than columns

$$\text{Let} \quad \mathbf{A} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \\ -2 & 1 \\ 1 & 3 \end{pmatrix}.$$

As above, calculate $\mathbf{V}$ from $\mathbf{A^TA} = \mathbf{V\Sigma^T U^T U\Sigma V^T} = \mathbf{VDV^T}$ since $\mathbf{U^T} = \mathbf{U^{-1}}$ for an orthogonal matrix. The characteristic equation, eigenvalues and eigenvectors are

$$\mathbf{A^TA} = \begin{pmatrix} 10 & 2 \\ 2 & 11 \end{pmatrix}, \qquad \lambda^2 - 21\lambda + 106 = 0, \quad \lambda = \frac{1}{2}(21 \pm \sqrt{17}), \qquad \begin{pmatrix} 4 & -4 \\ \sqrt{17}+1 & \sqrt{17}-1 \end{pmatrix}.$$

The diagonal elements of $\mathbf{\Sigma}$ are $\sqrt{\frac{1}{2}(21 \pm \sqrt{17})} = \sqrt{12 \cdot 562} = 3 \cdot 544$ and $\sqrt{8 \cdot 438} = 2 \cdot 905$. $\mathbf{V}$ is the normalisation of the matrix on the right above; its columns are the eigenvectors. To avoid carrying square roots I will use decimal approximations. The magnitudes of the eigenvectors are $6 \cdot 500$ and $5 \cdot 075$. Taking the transpose of $\mathbf{V}$

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} 3 \cdot 544 & 0 \\ 0 & 2 \cdot 905 \end{pmatrix}\begin{pmatrix} 0 \cdot 615 & 0 \cdot 788 \\ -0 \cdot 788 & 0 \cdot 615 \end{pmatrix}.$$

Again in $\mathbf{\Sigma}$ the eigenvalues are placed in strictly decreasing order.

By a similar procedure we find $\mathbf{A}$, using this time $\mathbf{AA^T}$ to eliminate $\mathbf{V}$:

$$\mathbf{AA^T} = \begin{pmatrix} 2 & 2 & -1 & 4 \\ 2 & 4 & -4 & 2 \\ -1 & -4 & 5 & 1 \\ 4 & 2 & 1 & 10 \end{pmatrix},$$

It will be clear that this matrix will have four eigenvalues, and that two of them must be $\frac{1}{2}(21 \pm \sqrt{17})$. The other two must be zero. This is consistent with the determinant of the above 4×4 matrix being zero. $\mathbf{AA^T}$ has characteristic polynomial $\lambda^2(\lambda^2 - 21\lambda + 106) = 0$, λ^2 times that of $\mathbf{A^T A}$.

We should pause at this stage to understand what the two zero eigenvalues mean. They arise because the rank of the original matrix $\mathbf{A}$ is only 2, as is readily seen by row reduction. Suppose we retain $\mathbf{U}$ as a 4×4 orthogonal matrix by finding eigenvectors for the two zero eigenvalues. These can be found by solving the simultaneous equations implied by $(\mathbf{AA^T}).\mathbf{h} = \mathbf{0}$. If $\mathbf{h} = (h_1, h_2, h_3, h_4)$.

$$h_1 + h_3 + 3h_4 = 0, \qquad 2h_2 - 3h_3 - 2h_4 = 0$$

where h_3, h_4 are undefined and free to be chosen subject to the two vectors being orthogonal. If $h_3 = 0$, $h_4 = 1$, we have $h_1 = -3$ $h_2 = 1$. The other eigenvector is calculated to be normal to this: $(5, 24, 22, -9)^T$ The eigenvectors corresponding to $12 \cdot 562$ and $8 \cdot 438$ are $(0 \cdot 471, 0 \cdot 413, -0 \cdot 149, 1)^T$ and $(-0 \cdot 163, -1 \cdot 490, 2 \cdot 072, 1)^T$ respectively. Taking these as the columns of $\mathbf{U}$, normalising each column to 1, and maintaining the order of decreasing eigenvalues

$$\mathbf{U} = \begin{pmatrix} 0 \cdot 396 & -0 \cdot 0595 & -0 \cdot 905 & 0 \cdot 146 \\ 0 \cdot 347 & -0 \cdot 543 & 0 \cdot 302 & 0 \cdot 703 \\ -0 \cdot 125 & 0 \cdot 755 & 0 & 0 \cdot 644 \\ 0 \cdot 841 & 0 \cdot 364 & 0 \cdot 302 & -0 \cdot 264 \end{pmatrix}.$$

Now we observe that it is not possible to multiply this $\mathbf{U}$ with $\mathbf{\Sigma}$ on account of their rows and columns not conforming. The solution is simply to extend $\mathbf{\Sigma}$ with two extra rows of zeros. The singular value decomposition is

$$\mathbf{A} = \begin{pmatrix} 0 \cdot 396 & -0 \cdot 0595 & -0 \cdot 905 & 0 \cdot 146 \\ 0 \cdot 347 & -0 \cdot 543 & 0 \cdot 302 & 0 \cdot 703 \\ -0 \cdot 125 & 0 \cdot 755 & 0 & 0 \cdot 644 \\ 0 \cdot 841 & 0 \cdot 364 & 0 \cdot 302 & -0 \cdot 264 \end{pmatrix}.\begin{pmatrix} 3 \cdot 544 & 0 \\ 0 & 2 \cdot 905 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 \cdot 615 & 0 \cdot 788 \\ -0 \cdot 788 & 0 \cdot 615 \end{pmatrix}.$$

However, it is clear that the extra rows of zeros in $\mathbf{\Sigma}$ make null any contribution from the last two columns of $\mathbf{U}$, so there was no need to determine the eigenvectors corresponding to its two zero eigenvalues. In fact, the last two columns of $\mathbf{U}$ and the two extra rows of $\mathbf{\Sigma}$ can be deleted. We therefore define a truncated SVD

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 2 & 0 \\ -2 & 1 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 0\cdot396 & -0\cdot0595 \\ 0\cdot347 & -0\cdot543 \\ -0\cdot125 & 0\cdot755 \\ 0\cdot841 & 0\cdot364 \end{pmatrix} \begin{pmatrix} 3\cdot544 & 0 \\ 0 & 2\cdot905 \end{pmatrix} \begin{pmatrix} 0\cdot615 & 0\cdot788 \\ -0\cdot788 & 0\cdot615 \end{pmatrix}.$$

The test is to multiply this out, and indeed it is correct. Truncation is possible whenever the rank of the given matrix is less than its number of rows or columns.

A3.4 Moore-Penrose pseudo-inverse and least squares

One valuable corollary to the SVD algorithm is the construction of a type of inverse for a non-square matrix, which cannot have a regular inverse. The Moore-Penrose pseudo-inverse is $\mathbf{V\Sigma^{-1}U^T}$. This is because $\mathbf{V\Sigma^{-1}U^T}.\mathbf{U\Sigma V^T} = \mathbf{I}$, the identity matrix. Moreover $\mathbf{U\Sigma V^T}.\mathbf{V\Sigma^{-1}U^T} = \mathbf{I}$ also. $\mathbf{\Sigma^{-1}}$ is simply the diagonal matrix whose elements are the reciprocals of the respective elements in $\mathbf{\Sigma}$. The M-P pseudo-inverse of any matrix $\mathbf{F}$ is written $\mathbf{F}^+$. The over-determined equation $\mathbf{Fx} = \mathbf{y}$ has its least square solution $\mathbf{x_0}$ given by $\mathbf{F}^+\mathbf{y}$. I will demonstrate by example that this is the same optimal solution as obtained by the traditional approach to least squares, and go on to consider the relation of this method to the constrained optimisation using Lagrange multipliers and the eigenvalue equation in Appendix 2.

The example is to find the least squares fit of a parabola $y = f(x) = a_0 + a_1x + a_2x^2$ to these seven points:

$$(0\cdot0, 0\cdot9),\ \ (0\cdot2, 0\cdot5),\ \ (0\cdot4, 0\cdot1),\ \ (0\cdot6, 0\cdot0),\ \ (0\cdot8, -0\cdot1),\ \ (1\cdot1, 0\cdot2),\ \ (1\cdot2, 0\cdot2)$$

by choosing a_0, a_1, a_2. The matrix equation is $\mathbf{Xa} = \mathbf{y}$, or in full

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0\cdot2 & 0\cdot04 \\ 1 & 0\cdot4 & 0\cdot16 \\ 1 & 0\cdot6 & 0\cdot36 \\ 1 & 0\cdot8 & 0\cdot64 \\ 1 & 1\cdot1 & 1\cdot21 \\ 1 & 1\cdot2 & 1\cdot44 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.5 \\ 0.1 \\ 0 \\ -0.1 \\ 0.2 \\ 0.2 \end{pmatrix}.$$

The traditional derivation is to minimise $F = \sum_k^n [f(x_k) - y_k]^2$ using calculus. This requires the partial derivatives with respect to a_0, a_1 and a_2 to be zero.

$$\frac{\partial F}{\partial a_0} = \sum [f(x_k) - y_k], \qquad \frac{\partial F}{\partial a_1} = \sum x_k [f(x_k) - y_k], \qquad \frac{\partial F}{\partial a_2} = \sum x_K^2 [f(x_k) - y_k].$$

This leads to the equation

$$\begin{pmatrix} N & \sum x & \sum x^2 \\ \sum x & \sum x^2 & \sum x^3 \\ \sum x^2 & \sum x^3 & \sum x^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum y \\ \sum xy \\ \sum x^2 y \end{pmatrix}, \qquad \mathbf{Ma} = \mathbf{p}$$

Note that the square matrix $\mathbf{M}$ is symmetric. For the numerical example this is

$$\begin{pmatrix} 7 & 4\cdot3 & 3\cdot85 \\ 4\cdot3 & 3\cdot85 & 3\cdot859 \\ 3\cdot85 & 3\cdot859 & 4\cdot1041 \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1\cdot8 \\ 0\cdot52 \\ 0\cdot502 \end{pmatrix}.$$

This is a well conditioned set of three simultaneous equations for three unknowns which can be found by row reduction or by taking the inverse of $\mathbf{M}$:

$$\mathbf{M^{-1}p} = \begin{pmatrix} 0\cdot769 & -2\cdot360 & 1\cdot498 \\ -2\cdot360 & 11\cdot76 & -8\cdot844 \\ 1\cdot498 & -8\cdot844 & 7\cdot155 \end{pmatrix}\begin{pmatrix} 1\cdot8 \\ 0\cdot52 \\ 0\cdot502 \end{pmatrix} = \begin{pmatrix} 0\cdot908 \\ -2\cdot572 \\ 1\cdot689 \end{pmatrix} = \mathbf{a}.$$

So $a_0 = 0\cdot908,\ a_1 = -2\cdot572,\ a_2 = 1\cdot689$. Note that $\mathbf{X^TX} = \mathbf{M}$ and $\mathbf{X^Ty} = \mathbf{p}$. Therefore $\mathbf{Xa} = \mathbf{y}$ has been converted to $\mathbf{X^T(Xa} = \mathbf{y}) \equiv \mathbf{Ma} = \mathbf{p}$.

Now use the pseudo-inverse to carry out the solution. From SVD or otherwise we have the pseudo-inverse

$$\mathbf{M}^+ = \begin{pmatrix} 0\cdot769 & 0\cdot357 & 0\cdot064 & -0\cdot108 & -0\cdot161 & -0\cdot015 & 0\cdot094 \\ -2\cdot360 & -0\cdot362 & 0\cdot929 & 1\cdot513 & 1\cdot388 & -0\cdot125 & -0\cdot983 \\ 1\cdot498 & 0\cdot015 & -0\cdot895 & -1\cdot233 & -0\cdot999 & 0\cdot426 & 1\cdot187 \end{pmatrix}.$$

Its first column is identical with the first in $\mathbf{M^{-1}}$ above. The product $\mathbf{M^+y}$ is indeed exactly the $\mathbf{a}$ obtained from the traditional method above. This illustrates the least squares property of the pseudo-inverse.

Appendix 2, §A2.1 gives an account of a similar approach to finding the least-error set of parameters $\mathbf{h}$ for the equation $\mathbf{Mh} = \mathbf{0}$ subject to the constraint that $\mathbf{h} \neq 0$. We might expect that approach also to find the best-fit parabola in our example. Accordingly augment $\mathbf{X}$ with the column $-\mathbf{y}$, and augment $\mathbf{a}$ to $\hat{\mathbf{a}}$ with 1 to represent the coefficient of y, as follows:

$$\mathbf{Q\hat{a}} = \begin{pmatrix} 1 & 0 & 0 & -0\cdot9 \\ 1 & 0\cdot2 & 0\cdot04 & -0\cdot5 \\ 1 & 0\cdot4 & 0\cdot16 & -0\cdot1 \\ 1 & 0\cdot6 & 0\cdot36 & 0 \\ 1 & 0\cdot8 & 0\cdot64 & 0\cdot1 \\ 1 & 1\cdot1 & 1\cdot21 & -0\cdot2 \\ 1 & 1\cdot2 & 1\cdot44 & -0\cdot2 \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The eigenvalue equation is $(\mathbf{Q^TQ})\hat{\mathbf{a}} = \lambda\hat{\mathbf{a}}$.

$$\mathbf{Q^TQ} = \begin{pmatrix} 7\cdot & 4\cdot3 & 3\cdot85 & -1\cdot8 \\ 4\cdot3 & 3\cdot85 & 3\cdot859 & -0\cdot52 \\ 3\cdot85 & 3\cdot859 & 4\cdot1041 & -0\cdot502 \\ -1\cdot8 & -0\cdot52 & -0\cdot502 & 1\cdot16 \end{pmatrix}.$$

The smallest eigenvalue of $\mathbf{Q^TQ}$ is $0\cdot001268$. It represents the degree to which the given points depart from the closest parabola, and would be zero were they all to lie exactly on the

curve. Its eigenvector is

$$(-0\cdot2680, 0\cdot7668, -0\cdot5054, -0\cdot2911)^T \quad \text{which normalises to} \quad (0\cdot9205, -2\cdot634, 1\cdot736, 1)^T.$$

We obtain $a_0 = 0\cdot921,\ a_1 = -2\cdot634,\ a_2 = 1\cdot736$. These are slightly different from those above. It appears that the eigenvalue method is not exactly the same least-squares procedure, but something close.

The given points and the two fitted parabolas are plotted in Figure 14. Their difference is negligible and one is not to be preferred over the other.

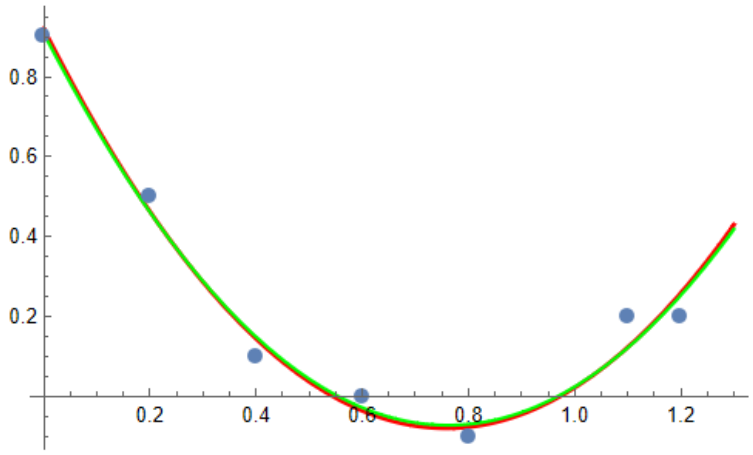

Figure 14: Seven given points and two fitted parabolas. Red: traditional and pseudo-inverse. Green: Lagrange-eigenvalue method.

The pseudo-inverse, therefore, is a second method by which a best fit can be made to an over-determined set of linear equations. In the epipolar calculations of §3.3 the row at Eq 11 can be replaced by

$$(u_1u_2 \quad u_2v_1 \quad u_2 \quad u_1v_2 \quad v_1v_2 \quad v_2 \quad u_1 \quad v_1 \quad 1).(e_{11} \quad e_{12} \quad e_{13} \ldots. \quad e_{31} \quad e_{32})^T = -1$$

since one of the elements, here e_{33}, is arbitrary. A matrix similar to $\mathbf{M}$ is assembled by stacking the n rows for the n point-pairs, and its pseudo-inverse formed. Multiplying the $n \times 1$ matrix $(-1, -1, ..., -1)^T$ by the pseudo-inverse gives the essential matrix $\mathbf{E}$ directly.

Appendix 4: Epipolar geometry

A4.1 Product of 3×3 skew and orthonormal matrices

This product type occurs in epipolar geometry. Let $\mathbf{A} = \mathbf{K}.\mathbf{N}$ where $\mathbf{K}$ is skew symmetric and $\mathbf{N}$ is orthonormal with $\mathbf{N}\mathbf{N}^\mathbf{T} = 1$. We wish to recover $\mathbf{K}$ and $\mathbf{N}$ separately from $\mathbf{A}$. We will see that this can be solved without the full SVD, but it does call on operations used in SVD.

First notice that $\mathbf{K}^\mathbf{T} = -\mathbf{K}$, and that $\mathbf{K}$, though 3×3, is only of rank 2. Its determinant is zero and so it does not have an inverse. The general form is

$$\mathbf{K} = \begin{pmatrix} 0 & p & q \\ -p & 0 & r \\ -q & -r & 0 \end{pmatrix}, \qquad \mathbf{L} = \mathbf{K}.\mathbf{K}^\mathbf{T} = \mathbf{K}^\mathbf{T}.\mathbf{K} = \begin{pmatrix} p^2+q^2 & qr & -pr \\ qr & p^2+r^2 & pq \\ -pr & pq & q^2+r^2 \end{pmatrix}. \qquad (A4.1)$$

The eigenvalues of $\mathbf{K}.\mathbf{K}^\mathbf{T}$ are two of $p^2+q^2+r^2$ and one of 0. If $\mathbf{K}.\mathbf{K}^\mathbf{T}$ can be calculated, it can be solved for $p,\ q,\ r$ to within $\pm$ sign; that is, either $\mathbf{K}$ or $\mathbf{K}^\mathbf{T}$ would be found. One route would be to take the equations for qr, $-pr$ and pq being equal to their respective matrix element values, and solve simultaneously to obtain $p^2 = -L_{13}L_{23}/L_{12},\ q^2 = -L_{12}L_{23}/L_{13},\ r^2 = -L_{12}L_{13}/L_{23}$. Alternatively one could just take ratios of matrix elements to find the ratios p/q, etc.

We have an example of an orthonormal matrix $\mathbf{N}$ in $\mathbf{V}^\mathbf{T}$ in Example 1 of Appendix 3. Such matrices arise where there are rotations about three orthogonal axes. In §3.2.1. of my article on perspective on *www.mathstudio.co.uk* I show how an orthonormal matrix arises by rotation by γ about the z axis, then β about y and finally by α about x. The general form is

$$\begin{pmatrix} cb.cg & cg.sa.sb - ca.sg & ca.cg.sb + sa.sg \\ cb.sg & ca.cg + sa.sb.sg & -cg.sa + ca.sb.sg \\ -sb & cb.sa & ca.cb \end{pmatrix}, \quad sa = \sin\alpha,\ ca = \cos\alpha, etc.$$

I have been unable to reconcile $\mathbf{V}^\mathbf{T}$ in Example 1 with any set of three rotations. However, the following matrix is the product of a rotation by $2\pi/5$ about z, $-\pi/11$ about y and $1\pi/7$ about x in that order:

$$\mathbf{N} = \begin{pmatrix} 0\cdot296 & -0\cdot913 & -0\cdot282 \\ 0\cdot525 & 0\cdot402 & -0\cdot750 \\ 0\cdot798 & 0\cdot0745 & 0\cdot598 \end{pmatrix}.$$

I will work through a numerical example, taking $p = 1,\ q = -2,\ r = 3$:

$$\mathbf{A} = \mathbf{K}.\mathbf{N} = \begin{pmatrix} -1\cdot071 & 0\cdot2531 & -1\cdot947 \\ 2\cdot0971 & 1\cdot136 & 2\cdot076 \\ -0\cdot9817 & -3\cdot032 & 1\cdot687 \end{pmatrix}.$$

$$\mathbf{A}^\mathbf{T}\mathbf{A} = \begin{pmatrix} 6\cdot508 & 5\cdot088 & 4\cdot783 \\ 5\cdot088 & 10\cdot55 & -3\cdot248 \\ 4\cdot783 & -3\cdot248 & 10\cdot95 \end{pmatrix}, \qquad \mathbf{A}\mathbf{A}^\mathbf{T} = \begin{pmatrix} 5 & -6 & -3 \\ -6 & 10 & -2 \\ -3 & -2 & 13 \end{pmatrix} = \mathbf{K}\mathbf{K}^\mathbf{T}.$$

The eigenvalues of both of these product matrices are 14, 14, 0, and $14 = p^2+q^2+r^2$ so $\mathbf{A}^\mathbf{T}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^\mathbf{T}$ have the same eigenvalues as $\mathbf{K}^\mathbf{T}\mathbf{K}$. From the second matrix above and the form of $\mathbf{K}$, $pq = -2,\ qr = -6,\ pr = 3$ so $p = \pm1,\ q = \mp2,\ r = \pm3$ and we now know $\pm\mathbf{K}$.

Knowing $\mathbf{A}$ and $\pm\mathbf{K}$ should allow $\mathbf{N}$ to be determined. Since $\mathbf{K}$ is not invertible, the solution can be found by taking a general algebraic form for a 3×3 matrix, pre-multiplying it by the known $\mathbf{K}$ and solving for each element in turn. Thus

$$\mathbf{A} \;=\; \begin{pmatrix} 0 & p & q \\ -p & 0 & r \\ -q & -r & 0 \end{pmatrix}.\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix} \;=\; \begin{pmatrix} dp+gq & ep+hq & fp+jq \\ -ap+gr & -bp+hr & -cp+jr \\ -aq-dr & -bq-er & -cq-fr \end{pmatrix}. \qquad (A4.2)$$

It is necessary to solve for only 6 elements. I have found that $\mathbf{N}$ is fully recovered, though its sign, + or −, is determined by the choice of sign for $\mathbf{K}$. If necessary, and assuming that $\mathbf{K}$ does describe a rotation, the angles about each axis can in principle be found by comparison with the general rotation matrix involving $\sin\alpha$, $\cos\alpha$, etc. as above (or equivalent matrices when the order of axes is different from z then y then x).

A4.2 Details of Longuet-Higgins' matrix algebra

There are some subtleties in epipolar geometry related to the translation between the two cameras being a vector $\mathbf{b}$ but referred to the distinct co-ordinates frames of cameras 1 and 2. In these frames $\mathbf{b}$ is represented by its co-ordinates as the elements of row or column matrices $\mathbf{b_1}$ and $\mathbf{b_2}$. Here I spell out the details and illustrate it with Example 3 from §3.2.

For rotation about y only, Eq 13 gives the co-ordinates of a point in the frame of camera 2 in terms of frame 1 by

$$x_2 \;=\; (x_1 - b_{x1})\cos\theta \;+\; (z_1 - b_{z1})\sin\theta, \qquad z_2 \;=\; -(x_1 - b_{x1})\sin\theta \;+\; (z_1 - b_{z1})\cos\theta. \qquad (13)$$

with $y_2 = y_1$. In this example the translation vector is $\mathbf{b_1} = (b_{x1}, b_{y1}, b_{z1}) = (1\cdot0,\ 0,\ 0\cdot2)$. With $\theta = 20^\circ$ about the y axis, $\mathbf{b_2} = (b_{x2}, b_{y2}, b_{z2})^T = (1\cdot0081,\ 0,\ -0\cdot1541)^T$. The signs are so that $\mathbf{b}$ is directed from O_1 to O_2. $\mathbf{b_1}$ and $\mathbf{b_2}$ are physically the same; the subscript is only to denotes the frame in which its components are stated. With $\mathbf{p_1} = (x_1, y_1, z_1)$, $\mathbf{p_2} = (x_2, y_2, z_2)$, Eq 13 can be stated in matrix terms as

$$\mathbf{p_2} = \mathbf{R}\mathbf{p_1} - \mathbf{R}\mathbf{b_1} \qquad \text{with} \qquad \mathbf{b_2} = \mathbf{R}\mathbf{b_1}, \qquad \mathbf{R} \;=\; \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}.$$

The normal to the epipolar plane at O_1 is simply a translation of the normal at O_2. Therefore

$$\mathbf{n_1} \;=\; \mathbf{b_1}\times\mathbf{p_1} \;=\; \mathbf{n_2} \;=\; \mathbf{b_2}\times\mathbf{p_2}.$$

Since $\mathbf{p_2}$ is perpendicular to $\mathbf{n_2}$,

$$\mathbf{p_2}\cdot\mathbf{n_2} \;=\; \mathbf{p_2}\cdot[\mathbf{b_2}\times\mathbf{p_2}] \;=\; \mathbf{0}, \qquad \text{so} \qquad \mathbf{p_2}\cdot[\mathbf{b_2}\times(\mathbf{R}\mathbf{p_1} - \mathbf{R}\mathbf{b_1})] = \mathbf{0}.$$

But $\mathbf{R}\mathbf{b_1} = \mathbf{b_2}$ and $\mathbf{b_2}\times\mathbf{b_2} = \mathbf{0}$, so

$$\mathbf{p_2}[\mathbf{b_2}\times\mathbf{R}]\mathbf{p_1} \;=\; \mathbf{0}. \qquad (A4.3)$$

The next step is to translate the cross product operation $\mathbf{b_2}\times$ into matrix form and obtain the 'essential matrix' $\mathbf{E}$

$$\mathbf{E} \;=\; \mathbf{b_2}\times\mathbf{R} \;\equiv\; \mathbf{B_2R} \;=\; \begin{pmatrix} 0 & -b_{z2} & b_{y2} \\ b_{z2} & 0 & -b_{x2} \\ -b_{y2} & b_{x2} & 0 \end{pmatrix}\mathbf{R} \qquad (A4.4)$$

which in this example is

$$\begin{pmatrix} 0 & -b_{z2} & 0 \\ b_{z2}\cos\theta + b_{x2}\sin\theta & 0 & -b_{x2}\cos\theta + b_{z2}\sin\theta \\ 0 & b_{x2} & 0 \end{pmatrix} \;=\; \begin{pmatrix} 0 & -b_{z2} & 0 \\ b_{z1} & 0 & -b_{x1} \\ 0 & b_{x2} & 0 \end{pmatrix}.$$

Here use has been made of the inverse of Eq 13. The above makes plain that $\mathbf{E}$ depends only on the displacement between the two camera stations and the angle of rotation.

As a numerical check, Point 1 in Table 7 is at $(-0{\cdot}6917,\ 0{\cdot}5272,\ 1{\cdot}2687)$ from camera 1 and $(-1\cdot2242,\ 0\cdot5272,\ 1\cdot5829)$ from camera 2, giving

$$\mathbf{p_2Ep_1} \;=\; \begin{pmatrix} -1\cdot2242 & 0\cdot5272 & 1\cdot5829 \end{pmatrix}\begin{pmatrix} 0 & 0\cdot1541 & 0 \\ 0\cdot20 & 0 & -1\cdot00 \\ 0 & 1\cdot0081 & 0 \end{pmatrix}\begin{pmatrix} -0\cdot6917 \\ 0\cdot5272 \\ 1\cdot2687 \end{pmatrix} \qquad (A4.5)$$

$$= \begin{pmatrix} -1\cdot2242 & 0\cdot5272 & 1\cdot5829 \end{pmatrix}\begin{pmatrix} 0\cdot0812 \\ -1\cdot4070 \\ 0\cdot5314 \end{pmatrix} \;=\; 0\cdot000.$$

To recover $\mathbf{b_2}$ form

$$\mathbf{EE^T} \;=\; \begin{pmatrix} b_{z2}^2 & 0 & -b_{x2}b_{z2} \\ 0 & b_{x1}^2 + b_{z1}^2 & 0 \\ -b_{x2}b_{z2} & 0 & b_{x2}^2 \end{pmatrix} \;=\; \begin{pmatrix} 0\cdot02374 & 0 & 0\cdot1553 \\ 0 & 1\cdot04 & 0 \\ 0\cdot1553 & 0 & 1\cdot01626 \end{pmatrix}.$$

Compare this with Eq A4.1 and solve the diagonals for $b_{x2}^2,\ b_{y2}^2,\ b_{z2}^2$. The result is $1{\cdot}0163,\ 0,\ 0{\cdot}02374$ respectively. Taking their square roots gives $b_{x2} = \pm1{\cdot}0081,\ b_{z2} = \mp0{\cdot}15408$, which indeed is vector $\mathbf{b}$ in the frame of camera 2. Element 3,3 shows that b_{x2} and b_{z2} have opposite signs. The reconstruction of $\mathbf{B}$ (or its transpose) is

$$\begin{pmatrix} 0 & 0\cdot1541 & 0 \\ -0\cdot1541 & 0 & -1\cdot0081 \\ 0 & 1\cdot0081 & 0 \end{pmatrix}.$$

Now recover $\mathbf{R}$ from $\mathbf{B}$ and $\mathbf{E}$. In practice $\mathbf{E}$ would have been determined from the co-ordinates of 8 or more image point-pairs. I propose at Eq A4.2 that this is done by solving a sequence of simultaneous equation. Using the dummy matrix at Eq A4.2

$$\mathbf{B_2}\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & j \end{pmatrix} \;=\; \begin{pmatrix} 0 & 0\cdot1541 & 0 \\ 0\cdot20 & 0 & -1\cdot00 \\ 0 & 1\cdot0081 & 0 \end{pmatrix} \;=\; \mathbf{E}$$

$$= \begin{pmatrix} 0\cdot1541\,d & 0\cdot1541\,e & 0\cdot15401\,f \\ -0\cdot1541\,a - 1\cdot0081\,g & -0\cdot1541\,b - 1\cdot0081\,h & -0\cdot1541\,c - 1\cdot0081\,j \\ 1\cdot0081\,d & 1\cdot0081\,e & 1\cdot0081\,f \end{pmatrix}.$$

Immediately $d = f = 0$, $e = 1$. Matching other elements gives $g = -0\cdot1528\,a - 0\cdot1984$, $h = -0\cdot1528\,b$, $j = -0\cdot1528\,c + 0\cdot9920$, at which stage the dummy matrix is

$$\begin{pmatrix} a & b & c \\ 0 & 1 & 0 \\ -0\cdot1528\,a - 0\cdot1984 & -0\cdot1528\,b & -0\cdot1528\,c + 0\cdot9920 \end{pmatrix}.$$

Then $b = 0$. The final two unknown are found by normalising the columns to unity, since this is a property of rotation matrices. Two solutions result: $a = 0\cdot9397$, $c = 0\cdot3420$ and $a = -0\cdot9990$, $c = -0\cdot0457$. The first pair are $\cos 20°$, $\sin 20°$ respectively, while the second pair can be rejected because of the large angle they imply. $\mathbf{R}$ has been recovered and can be checked with a few data points.

It remains to show that the essential matrix can be found from the image positions rather than the actual object positions as used so far. From Eq 1

$$\mathbf{p_1} = \frac{z_1}{d}\mathbf{u_1}, \quad \mathbf{u_1} = (u_1,\, v_1,\, d)$$

and similarly for $\mathbf{p_2}$. Eq A4.3 gives

$$z_2\mathbf{u_2}\,\mathbf{E}\,\mathbf{u_1}z_1 = 0, \qquad \text{implying} \qquad \mathbf{u_2}\,\mathbf{E}\,\mathbf{u_1} = 0. \tag{A4.6}$$

The image positions of Point 1 in Table 7 are $(-927,\ 706)$ in camera 1 and $(-1315,\ 566)$ in camera 2. With image-focus distance $d = 1700$ and $\mathbf{E}$ from Eq A4.5

$$\mathbf{u_2}\,\mathbf{E}\,\mathbf{u_1} = \begin{pmatrix} -1315 & 566 & 1700 \end{pmatrix} \begin{pmatrix} 0 & 0\cdot1541 & 0 \\ 0\cdot20 & 0 & -1\cdot00 \\ 0 & 1\cdot0081 & 0 \end{pmatrix} \begin{pmatrix} -927 \\ 706 \\ 1700 \end{pmatrix}.$$

This evaluates to 278, not zero, but this is because of the large numbers when measured in pixels, and because of accumulated rounding errors. If as is usual with projective points, we divide $\mathbf{u_1}$, $\mathbf{u_2}$ by d, the result is

$$\begin{pmatrix} -0\cdot7735 & 0\cdot3329 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0\cdot1541 & 0 \\ 0\cdot20 & 0 & -1\cdot00 \\ 0 & 1\cdot0081 & 0 \end{pmatrix} \begin{pmatrix} -0\cdot5453 \\ 0\cdot4153 \\ 1 \end{pmatrix} = 0\cdot000.$$

Therefore the relative position and rotation of the two cameras can be found from the images alone.

Appendix 5: Some details of SIFT

A5.1 Difference of Gaussians

The bell-shaped Gaussian function in two dimensions is

$$G(u,v) = \frac{1}{2\pi\sigma}\exp\left(\frac{-r^2}{2\sigma^2}\right), \qquad r^2 = u^2 + v^2.$$

It is normalised so that its integral over the uv plane is unity. σ is the radial distance at which $G(r)$ has fallen to 61% of its peak. Its negative second derivative – its negative Laplacian – is

$$-\nabla^2(G) = \frac{(r^2 - 2\sigma^2)}{2\pi\sigma^5}\exp\left(\frac{-r^2}{2\sigma^2}\right).$$

This is preferred as the convolution function to apply to an image because it is somewhat sharper than G itself. Figure 15 shows G and $-\nabla^2 G$ as 2D plots against r for $\sigma = 1$.

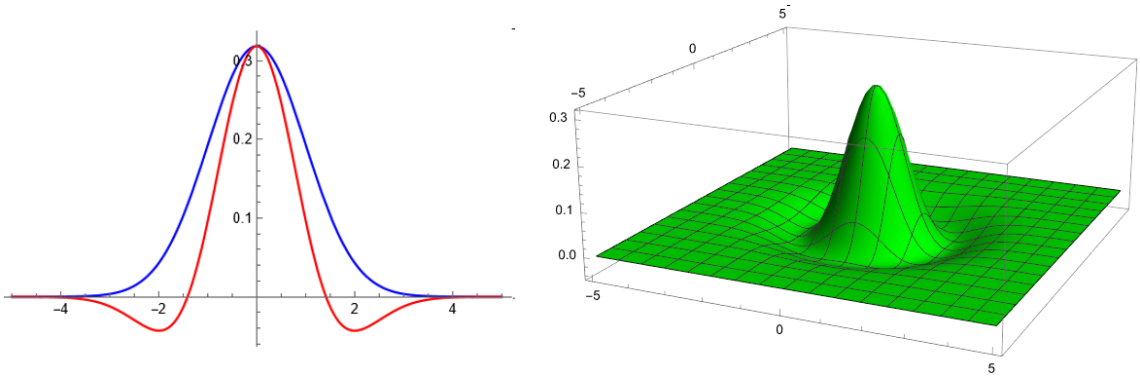

Figure 15: Left: $G(r)$ in blue, $-\nabla^2(G)$ in red. Right: Perspective representation of $-\nabla^2(G)$.

It happens that the shape of $-\nabla^2 G$ is fairly well approximated by the difference of two $G(r)$ functions with different widths, and these are faster to compute. Figure 16 compares $C(G(\sigma) - G(k\sigma))$ with $-\nabla^2(G)$ for $k = 1\cdot 9$, $C = 4\cdot 2$. The side lobes of $C(G(\sigma) - G(k\sigma))$ are larger, but this is not important as all that is required is a sharp central peak encircled by a ring where the value is zero. I find that

$$C \approx \frac{1}{-0.178k^2 + 0.774k - 0.591}$$

gives a fair fit to the central peak over the range $1 < k <= 2$.

Figure 17 shows Difference of Gaussians in action on a made-up graphic as carried out using the Edge → DoG filter in GIMP 2.10. The smaller the two radii, σ and $k\sigma$, the finer is the detail. k is usually set between about $\sqrt{2}$ and 2. I presume that the image convolution with $G(r)$ operates in parallel on each of the three RGB colour channels.

A5.2 Blob orientation and signature

In this section I illustrate how the principal orientation of a blob is calculated, and how its signature descriptor is compiled. In actual SIFT implementations the details will not be exactly as below, but the principles will be much the same.

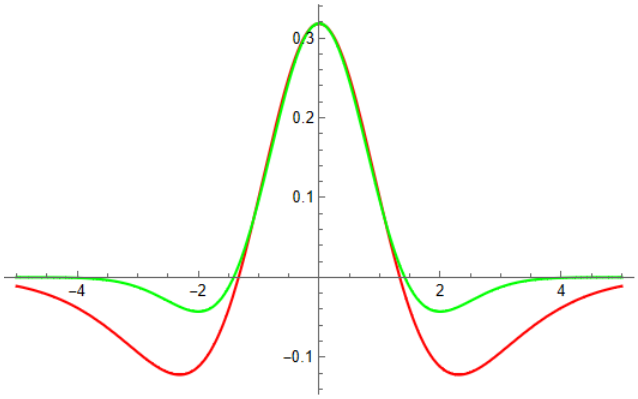

Figure 16: Green: $-\nabla^2(G)$ with $\sigma = 1$. Red: $C(G(\sigma) - G(k\sigma))$ for $k = 1 \cdot 9$, $C = 4 \cdot 22$.

Figure 17: A demonstration image and three derived images by applying DoG with radii 1:2, 3:5 and 8:13.

The left panel of Figure 18 is a small patch from the bottom right of Figure 17c. It has been down-sampled, as if from a later octave in the blurring process, to only 18×18 pixels. The central panel shows by yellow arrows the gradient at each pixel, as calculated in this example by Mathematica's gradient orientation function. To simplify the example I will use a grey-scale version, right panel, with adjusted contrast. Mathematica calculates gradients only within the interval $[-\pi/2, \pi/2]$. The histogram on the left of Figure 19 shows the distribution of these gradients for the whole 18×18 matrix. The dominant orientation is about $0 \cdot 8$ radians. This is therefore taken to be the principal orientation of this blob with respect to the frame of the original image, Figure 17. The histogram on the right plots the same data shifted by $0 \cdot 8$ and with $\pi/2$ added or subtracted to keep within $[-\pi/2, \pi/2]$.

Having determined the principal orientation, we now create a signature descriptor – a type of bar code for the blob. The 18×18 matrix, with $0 \cdot 8$ subtracted, is divided into four quadrants and a histogram of the gradient orientations within each quadrant calculated, again normalised to $[-\pi/2, \pi/2]$. These four histograms are shown in Figure 20. Each bin is $\pi/8$ wide. The signature of the blob is a list of the numbers in each bin, concatenated over the four quadrants. In this example it is

$$0, 2, 11, 4, 18, 25, 21, 0, 0, 0, \ : \ 0, 7, 5, 28, 26, 3, 12, 0, 0, 0$$

$$0, 5, 1, 16, 22, 24, 13, 0, 0, 0, \ : \ 0, 1, 0, 4, 25, 32, 19, 0, 0, 0.$$

A blob of the same feature in another image could be matched to Figure 18 by its signature. A metric would be used to assess the goodness of fit between the two signatures. For instance it could be the sum of squares of differences at each digit in the signature. Experience would suggest a threshold below which a match would be accepted.

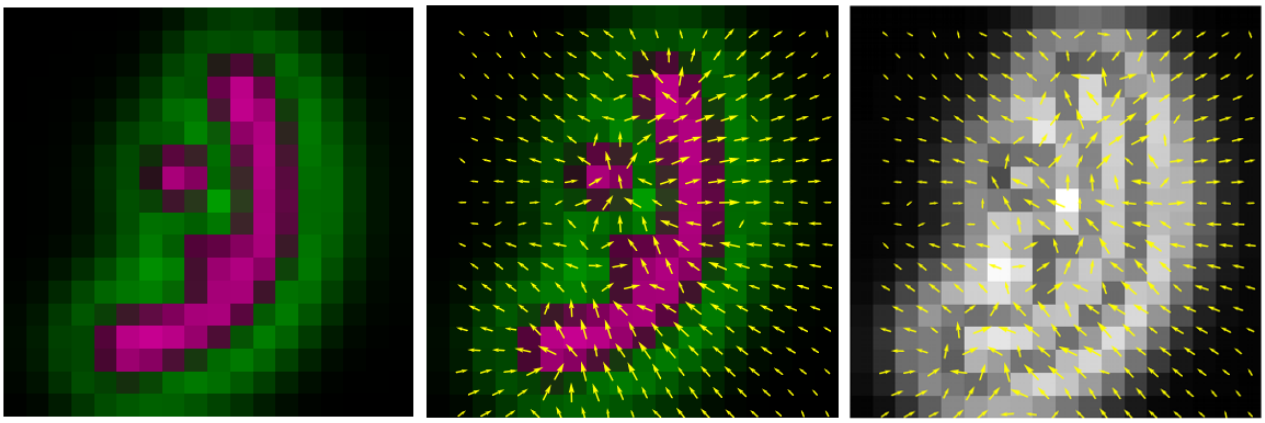

Figure 18: Left: Patch of Figure 10c 18×18 pixels. Centre:: gradient direction. Right: gradient vectors on a grey-scale version.

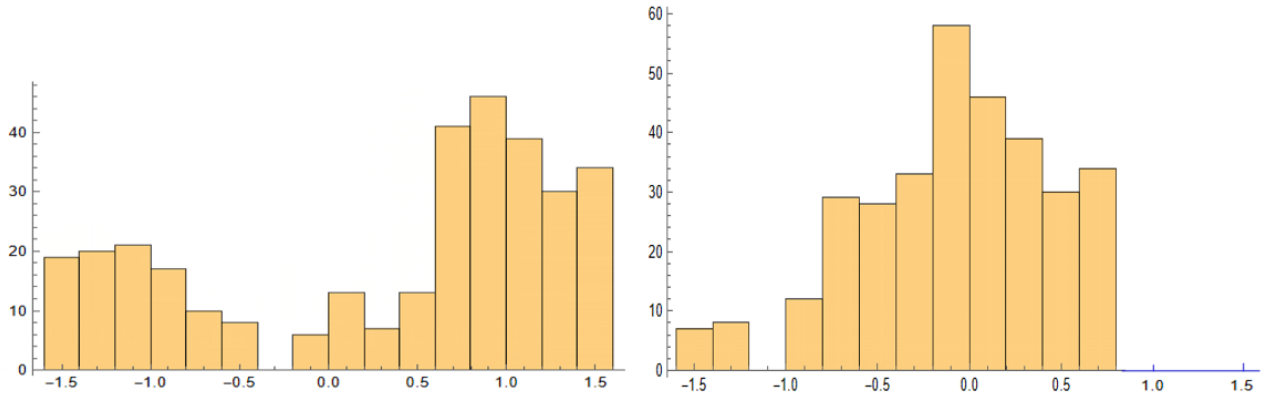

Figure 19: Histograms of gradient directions over the 18×18 grid, within interval $[-\pi/2, \pi/2]$. Left: relative to the whole image. Right: normalised to blob's principal direction.

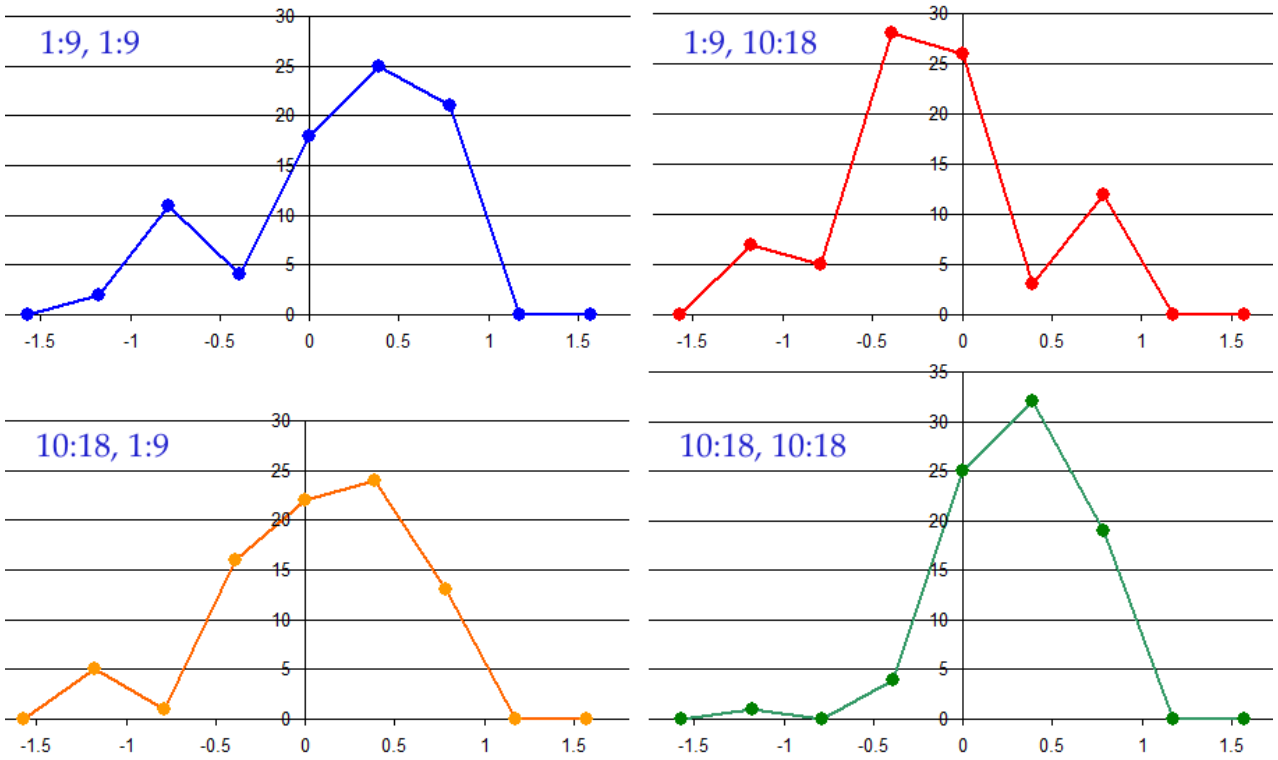

Figure 20: Histogram of distribution of gradient direction, $[-\pi/2, \pi/2]$, in the four quadrants of the gradient matrix of blob of Figure 18. Labels denote the rows and columns constituting each quadrant.

John Coffey, November 2024.