Machine Vision: Theory, Algorithms, Practicalities

Solutions to selected problems

The ‘solutions’ provided here are intended to include analysis, methods, hints, constraints and ideas that are relevant to the set problems. As appropriate in a research and application environment, they are not intended as complete unequivocal solutions, such as might be found in a school text, or a text for an exact subject such as mathematics. They are provided in good faith in the belief they will be helpful to the reader in his or her task of exploring the wide-ranging subject of Machine Vision. Likewise, problems and solutions haven’t been provided slavishly for all chapters or all topics. Rather, they have been produced as aids to learning, where this seemed appropriate to the development of the subject and to the detailed subject matter.

Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>35</td>
</tr>
<tr>
<td>17</td>
<td>39</td>
</tr>
<tr>
<td>18</td>
<td>41</td>
</tr>
<tr>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>20</td>
<td>45</td>
</tr>
<tr>
<td>21</td>
<td>47</td>
</tr>
<tr>
<td>24</td>
<td>49</td>
</tr>
<tr>
<td>A</td>
<td>50</td>
</tr>
</tbody>
</table>

E.R. Davies  
September 2005
2.1

The required edge location algorithm is:

shrink objects in image;
subtract shrunken objects from original image;
// the result is to leave the edges

To prove this, note that the EDGE table in Chapter 2, p. 34 has a single 1 in the lower left of the output section. On the other hand, the corresponding SHRINK table would have a single 1 in the lower right of the output section. Combining the two outputs would yield 1’s in both of the lower outputs, but leave 0’s in both of the upper outputs. Thus the total output is equal to A0, i.e. the same as for the original image. Hence the edge is the original image minus the shrunken image.

2.2

(a) For shrinking, it is probably best to take the off-image values as 1’s, so that objects that are partly occluded at the boundary of the image remain contiguous with the boundary.

(b) For expanding, it is best to take the off-image values as 0’s, so that apparent object areas do not move inwards from the boundary with each expansion phase. (The two cases of shrinking and expanding ought in principle to be duals of each other. However, it seems better not to follow this line of thought because in Chapter 2 we envisage that images will have a number of small, separate binary picture objects with value 1 against a 0’s background, and the chosen strategies need to give the least error in this type of situation.)

(c) For a blurring convolution, taking the off-image values as 0’s or 1’s, or any other fixed values, would mix significantly different values into the current pixel value, and the local grey-scale would change abruptly. The smoothest variation arises when the current pixel value and/or its nearest neighbors within the image are used to estimate the new current pixel value. I.e. it is better to seek a locally acceptable value than to impose a fixed value that will be a bad choice over much of the image boundary. The simplest choice is to use the intensity of the nearest pixel within the image. Of course, some averaging of neighboring pixel intensities might serve to minimize noise.

2.3

The NOIZE algorithm looks for a 1 with a single 1 adjacent to it, and allows it to be changed to 0. However, the 1 could be at the end of a line of single pixel width. Hence the algorithm could remove the end 1, and subsequently keep on removing further 1’s until the whole line had been eliminated. This cannot happen with the NOISE algorithm, since each 1 that is removed is isolated, and therefore a ‘chain reaction’ cannot occur.
3.3

This type of filter emulates the mode filter, and as such will enhance images by taking a strong decision at each pixel as to which side of an edge the pixel lies. However, the two extreme values in a distribution are very poorly defined because they are outliers, and have absolutely no immunity to impulse noise. They are also subject to the extremes of Gaussian noise, and as such they are again inaccurately defined, and poor examples to use as paradigm intensity values for the new output.

3.5

(a) See the main text in Chapter 3.

(b) Clear the whole histogram initially; then clear only the elements that have been filled, after finding each median. Find the window minimum; then set sum equal to the minimum before proceeding to find the median. Not having to clear the whole histogram essentially saves 256 operations, and there are still $128 \times 2$ to be done, so there is a saving by a factor two. Setting sum at the window minimum value can at best save $\frac{1}{2} \times 128 \times 2$ operations, but in fact the saving will typically be ~half of this figure.

(c) See the main text in Chapter 3.

(d) original: 1 2 1 1 2 3 0 2 2 3 1 1 2 2 9 2 2 8 8 8 7 8 8 7 9 9 9
3 × 3 median: ? 1 1 1 2 2 2 2 1 1 2 2 2 2 8 8 8 8 8 8 8 9 9 ?
5 × 5 median: ? ? 1 2 1 2 2 2 2 2 2 2 2 2 2 8 8 8 8 8 8 8 8 8 9 ?

The runs of constant value are obvious. The spike of value 9 has shifted the 5 × 5 median but not the 3 × 3 median. The rule is that shifts occur when a spike and a point adjacent to an edge appear within an $n$-element window.

3.6

(a) On the background side of any edge, the background intensity dominates, so the mode filter comes up with a central background intensity value. Similarly, it comes up with a central foreground value on the foreground side of any edge. There is a critical point on any edge where the dominant peak in the local intensity distribution changes over from the background peak to the foreground peak or vice versa. Thus the mode filter tends to enhance edges. The mean filter blurs edges since it produces an intensity profile in which background and foreground intensities are mixed together, thereby reducing the local contrast between the two regions.

(b) When a max filter is applied to an image, light background regions expand, and will tend to reduce the size of objects, or even to eradicate small objects. It will also tend to produce regions of constant (high) intensity in the image. In general, the mode filter will tend to preserve edges and not move them substantially to reduce the size of objects. On the other hand, at the corners of objects, or on high curvature boundaries, the mode filter will tend to cut into objects. However, unlike the max filter, it will act symmetrically
between foreground and background, so it will also cut into small regions of background on high curvature boundaries.

(c) The purpose of the median filter is to suppress noise by eliminating local outliers in intensity. In addition, it aims (successfully) to achieve this while not producing any local image blurring. The median filter is highly computation intensive, with computation proportional (at least) to the area of the window it operates in: to offset this, it is common to implement a 2-D median filter as two 1-D filters, with a result that is commonly almost 100% effective.

(d) orig: 0 1 1 2 3 2 2 0 2 3 9 3 2 4 4 6 5 6 7 0 8 8 9 1 1 8 9  
mean: ?? 1 2 2 2 2 2 3 3 4 4 4 4 4 5 6 5 5 6 6 5 5 5 5 ??  
max: ?? 3 3 3 3 3 3 9 9 9 9 9 9 6 6 6 7 7 8 8 9 9 9 9 9 ??  
med1: ?? 1 2 2 2 2 2 3 3 4 4 4 4 4 5 6 6 6 7 8 8 8 8 8 ??  
med2: ?? ?? ?? 2 2 2 2 3 3 3 4 4 4 4 5 6 6 6 7 8 8 8 8 8 ??  
med3: ?? ?? ?? ?? 2 2 2 2 3 3 3 4 4 4 4 5 6 6 6 7 8 8 8 8 8 ??  

Clearly, the mean filter averages all the noisy values into the signal, while the median excludes them.

(e) Further applications of a median filter in this case have no further effect. The max filter spreads all high values over a wide range and also broadens any light spots.

3.7

(a) The 3 × 3 median filter removes the noise points (including the bump) and also removes one corner point. The 5 × 5 median filter removes the noise points (including the bump) and three corner points.

(b) To obtain a corner detector, first use the 3 × 3 median filter to remove the noise points; then use the 5 × 5 median filter to find the corner points. This method has numerous variations. It does not depend on orientation so is intrinsically better than template matching. However, it requires significant computation in its own right.

3.8

(a) Mean filters average and return the mean intensity in the window; median filters return the median intensity in the window. Mean filters average in all values blindly, without discriminating against outliers. Median filters ignore outliers and hence don’t blur images.

original: 1 1 1 1 2 1 1 2 3 4 4 0 4 4 4 5 6 7 6 5 4 3 3  
median: 1 1 1 1 1 1 1 2 3 4 4 4 4 4 5 6 7 6 5 4 3 3  

(b) The median algorithm is bookwork (see the main text in Chapter 3). It operates slowly mainly because it has to go steadily through the histogram entries for something like half of the 256 possible intensity values before it arrives at the median.
(c) By cascading max operations, the maximum value can be found in 8 operations. One operation then sets the maximum to zero. Repeating this a further 4 times, but not in the last case setting the result to zero, gives the median. This involves a total of $9 \times 5 - 1 = 44$ operations. This should be compared with $\sim 9 + 256/2 + 9 = 146$ operations by the standard method, so the max approach is much faster.

(d) However, finding the max in a $1 \times 3$ window takes 2 operations, and then finding it again takes 1 operation, after setting the first max to zero. Thus the overall operation takes 6 operations—vastly quicker than 146 operations. The splitting up of the median does not affect its capability for eliminating single intensity impulses, though things get more complicated when their density is high.

3.9

(a) Results:

(i) 

<table>
<thead>
<tr>
<th>0 0 0 0 0 1 0 1 1 1 1 1 1 1</th>
<th>(input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 1 1 1 1 1 1 1 1</td>
<td>(output)</td>
</tr>
</tbody>
</table>

(ii) 

<table>
<thead>
<tr>
<th>2 1 2 3 2 1 2 2 3 2 4 3 3 4</th>
<th>(input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>? 2 2 2 2 2 2 2 2 3 3 3 3 ?</td>
<td>(output)</td>
</tr>
</tbody>
</table>

(iii) 

<table>
<thead>
<tr>
<th>1 1 2 3 3 4 5 8 6 6 7 8 9 9</th>
<th>(input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 2 3 3 4 5 6 6 7 8 9 9</td>
<td>(output)</td>
</tr>
</tbody>
</table>

(b) General lessons:

(i) Median filters can eliminate impulse noise, but may introduce edge shifts. This also applies to 2D images.

(ii) There is a tendency to produce runs of constant value.

(iii) Apart from removing impulse noise, median filters tend to leave monotonically increasing/decreasing signals unchanged.

(c) See the main text in Chapter 3.
4.4

Two solutions exist because one solution represents a threshold in the area of overlap between the two Gaussians; the other solution is necessary mathematically and lies either at very high or very low intensities. It is the latter solution that disappears when the two Gaussians have equal variance, as the distributions clearly never cross again. In any case, it seems unlikely that the distributions being modeled would in practice approximate so well to Gaussians that the non-central solution could ever be important—that is, it is essentially a mathematical fiction that needs to be eliminated from consideration.
6.4

The definition of a North point is that it is a 1 with a 0 to the North of it, and a 1 to the South of it. This means that removing a North point cannot disconnect the point North of it. Disconnection with the points NW and NE of it is prevented by the crossing number conditions existing initially, coupled with the fact that if they are themselves North points and are removed, the points below them will still maintain connectedness. Disconnection with the points W, E, SW or SE of it is prevented because the point directly South of it maintains the connection.

6.6

The basic idea for locating, labelling and counting is to scan the image systematically until the first part of the first object is located; then go into propagation mode, propagating a constant label over the whole object; then resume scanning, and repeat until all objects have been located. Labels are incremented as every object is found. The method has the problem that a systematic scan identifies objects with several spurs, humps or lumps as being separate objects, so either multiple passes are required to consolidate this, or a coexistence table has to be processed to create a self-consistent set of labels.

Instead, we can skeletonize all objects; then prune the skeletons down to the last central point; then label all these points in one scan. This is a far tidier method and avoids all the problems of the other method, unless objects have holes. However, the latter problem can be resolved by breaking each remaining circuit in turn. The only disadvantage of this general approach is its high computational cost. On the other hand, very complex shape topologies will make both methods more complex and there may be no clear winner. What we have then is two totally distinct representations, each of which will probably have advantages with different sets of shapes.

6.7

(a) The one-pass sequential algorithm is:

\[
N = 0; \\
\text{for all pixels in image do } \{ \\
\quad \text{if (in an object } \& \& \text{ max adjacent value } == 0) \{ \\
\quad\quad N = N + 1; \\
\quad\quad \text{set current value equal to } N; \\
\quad\} \\
\quad \text{else if (in an object) } \{ \\
\quad\quad \text{set current value equal to max adjacent value; } \\
\quad\} \\
\}\]

// Note: this algorithm assumes two image spaces, one containing the initial (binary) image and one containing the output (labelled) image.

With convex blobs there is no problem. The main problem is that U-shaped objects acquire two labels down to the join, and similarly for other upwards-pointing bifurcated
shapes, as will be seen in the following figure. Snake shapes and spirals give similar trouble.

(b) The solution is to accept the labels that are obtained by a single simple-minded labelling scan, but to record all label adjacencies in a table; then to analyse the table, noting all labels that are equivalent to each other, give the minimum (or maximum) value to each such set; then relabel the objects.

We can also take it that the table must have diagonal entries, so we set up the table (the one below is a minimal representation of a U, a W and an I) in the following way:

```
1 1 0 0 0 0
1 2 0 0 0 0
0 0 3 0 0 3
0 0 4 0 4
0 0 0 5 0
0 0 3 4 0 6
```

The off-diagonal elements show where connectedness between branches of objects occurs, and in limited memory space represents this fact. Therefore, it is intrinsically faster to process this table than it is to process the original image. We must systematically propagate values through this table, by horizontal and vertical motions, until all relevant diagonal elements have had the chance of being linked. After one horizontal minimization we get:

```
1 1 0 0 0 0
1 1 0 0 0 0
0 0 3 0 0 3
0 0 4 0 4
0 0 0 5 0
0 0 3 4 0 6
```

One vertical minimization now gives:

```
1 1 0 0 0 0
1 1 0 0 0 0
0 0 3 0 0 3
0 0 0 3 0 3
0 0 3 0 3
```
This is the final result, except for the need to do a final relabelling to give the correct count.

The number of iterations is in principle the same as in an image, except that a minimum number of parameters (pixels or table entries) need to be changed on any given iteration. The real gain is therefore in adapting the complex topological processing to a more efficient representation for the purpose.

6.8

(a) It produces a rectangular convex hull around each object.

(b) Detail of do until facility:

\[
\begin{align*}
\text{do} & \{ \\
\text{finished} &= \text{true}; \\
\text{[[ sum} &= (A1 \&\& A3) + (A3 \&\& A5) + (A5 \&\& A7) + (A7 \&\& A1) ; \\
\text{if} (\text{sum} > 0) B0 = 1; \text{else} B0 = A0; \\
\text{if} (B0 != A0) \text{finished} = \text{false}; ]]; \\
\text{[[ A0} &= B0; ]]
\}\text{ until finished; } // \text{in pure C++, until } \rightarrow \text{while not}
\end{align*}
\]

6.9

(a) The following algorithm generates a rectangular convex hull around any object:

\[
\begin{align*}
\text{do} & \{ \\
\text{finished} &= \text{true}; \\
\text{[[ sum} &= (A1 \&\& A3) + (A3 \&\& A5) + (A5 \&\& A7) + (A7 \&\& A1) ; \\
\text{if} (\text{sum} > 0) B0 = 1; \text{else} B0 = A0; \\
\text{if} (B0 != A0) \text{finished} = \text{false}; ]]; \\
\text{[[ A0} &= B0; ]]
\}\text{ until finished; } // \text{in pure C++, until } \rightarrow \text{while not}
\end{align*}
\]

(b) An accurate algorithm basically requires all pairs of points in an object to be joined. However, we can save computation by just joining all pairs of points on its boundary. Hence tracking is needed. Tracking must proceed once around the boundary stopping when passing through the first point in the same sense (or the first two points in the same order).

To find how to track clockwise at any point, look back along the curve, then rotate gaze clockwise until the first object point is reached, and move to it. The simplest way of implementing this is via a lookup table: for a 3 \times 3 window we only have eight neighbors, so there are only 256 possibilities, and we can look up the direction to move in one go.

(c) To implement the convex hull algorithm, travel around the boundary clockwise with a 2-point tester. Keep the first point fixed, then move the second point onwards until it
becomes a tangent (or just ceases to be a chord at another position); then move the first point onwards until it becomes a tangent. The procedure terminates when there is no pair of test points for which the chord cuts the boundary more than twice. The algorithm cannot take longer than the time needed to consider all pairs of boundary points, i.e. it requires $O(b^2)$ computation.

6.11

(a) The distance function is a version of the binarized image in which each pixel is assigned a number indicating its distance from the boundary of the object.

(b) Local maxima are found by the algorithm:

\[
\begin{align*}
\text{maximum} &= \max(Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8); \\
\text{if } (Q0 > 0 \&\& Q0 >= \text{maximum}) \text{ B0 = 1; else B0 = 0; ]];}
\end{align*}
\]

The **simplest** reconstruction algorithm is:

\[
\begin{align*}
\text{for } (i = 1; i <= N; i++) \\
[ [ \text{Q0 = max(Q0 + 1, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8) – 1; ]];}
\end{align*}
\]

(c) For Figure 6.P1, no. of local maxima pixels is 32, and no. of boundary points is 55. Therefore the answers (in bits) for a 256 × 256 image are:

(i) $2 \times 32 \times 8 + 1 \times 32 \times 7 = 512 + 224 = 736$

(ii) $2 \times 55 \times 8 = 880$

(iii) $2 \times 1 \times 8 + (55 – 1) \times 3 = 16 + 162 = 178$

6.12

(a) A distance function shows the distance from the background to every point in a binary picture object, as indicated in the following figure:

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1
1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1
1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 2 3 4 3 2 1
1 2 3 4 3 2 1
1 2 3 4 3 2 1
1 2 3 4 3 2 1
1 2 3 3 3 2 1
1 2 3 3 3 2 1
1 1 1 1 1 1 1 1
```

The number of passes required by a parallel algorithm is equal to half the maximum width of the widest object in the image, and equals the highest value in the distance function. The number of passes required by a sequential algorithm is just two.
(b) The sequential algorithm is:

\[
\begin{align*}
P_0 &= A_0 \times 255; \quad \text{(initialization pass only)} \\
(+\ P_0 &= \min(P_0 - 1, P_2, P_3, P_4, P_5) + 1; \quad +\) \\
(-\ P_0 &= \min(P_0 - 1, P_6, P_7, P_8, P_1) + 1; \quad -)
\end{align*}
\]

This algorithm produces a left-going and downward-going wedge inside objects in the first pass, and consolidates it to include a right-going and upward-going wedge on the second pass. The parallel algorithm is more straightforward but somewhat more tedious to write out.

(c) The local maxima are marked in the following figure:

```
+ + + + + + + + + + + + + + + + \\
+ + + + + + + + + + + + + + + + + \\
+ + 3 3 3 3 3 3 3 3 3 3 3 3 3 + + + \\
+ + + 3 3 3 3 3 3 3 3 3 3 3 3 3 + + + \\
+ + + + + + + + + + + + + + + + + \\
+ + + + + + + + + + + + + + + + + \\
+ + + 4 + + +
```

The image would be reconstituted by a process of downward propagation from the local maxima—very much the opposite of the upwards propagation process used to find the distance function.

(d) The compression factor for transmitting the local maxima is:

\[\eta = \frac{128^2}{(33 \times 3 \times 8)} \approx 20.\]

The minimized set of local maxima is:

```
+ + + + + + + + + + + + + + + + \\
+ + + + + + + + + + + + + + + + + \\
+ + 3 + + + + 3 + + + + 3 + 3 + + + \\
+ + + 3 + + + + 3 + + + + 3 + 3 + + + \\
+ + + + + + + + + + + + + + + + 3 + + + \\
+ + + + + + + + + + + + + + + + + + + \\
+ + + 4 + + +
```

This set gives a compression factor: \[\eta = \frac{128^2}{(13 \times 3 \times 8)} \approx 50.\]

6.13

(a) If the local maxima were defined as in (i), very few pixels would remain labelled, and there would be insufficient information for reproducing the original shapes. If they are defined as in (ii), clusters of pixels will be labelled with the same values. (ii) is suitable for reproducing the original shape, (i) being too sparse to give a complete description. The reason that (ii) works is that one need only propagate downwards towards the outsides of the objects: at each stage there is sufficient information to go to
the next stage downwards.

(b) A suitable algorithm is:

\[
[+ \text{ maxminusone } = \max(P0 + 1, P2, P3, P4, P5) - 1; \\
\text{ if } (\text{maxminusone } \geq 0) \ P0 = \text{maxminusone}; \text{ else 0}; \ +]];
\]

\[
[- \text{ maxminusone } = \max(P0 + 1, P6, P7, P8, P1) - 1; \\
\text{ if } (\text{maxminusone } \geq 0) \ P0 = \text{maxminusone}; \text{ else 0}; \ -]]
\]

This operates by going downwards by one for each step away from a local maximum, on the first pass, and similarly in the opposite direction on the reverse pass.

(c) Run-length encoding counts the numbers of pixels of the same intensity value (0 or 1) and puts out these numbers in sequence. For an image with few objects, few numbers need be put out and the method is very efficient. The local maxima method is efficient for large objects but fussy for small objects as lots of 1’s have to be put out.

6.14

(a) When propagating a distance function using a parallel algorithm, the pixels in the object are initially given a high value, such as 255. Then, pixels adjacent to 0’s are given the value 1, pixels adjacent to 1’s are given the value 2, and so on, until no 255’s are left. After initialization, the sequential algorithm operates in two passes, a down-and-to-the-right pass, and an up-and-to-the-left pass:

\[
[[ Q0 = A0 * 255; ]]; \\
[[+ Q0 = \min(Q0 – 1, Q2, Q3, Q4, Q5) + 1; \ +]];
\]

\[
[[– Q0 = \min(Q0 – 1, Q6, Q7, Q8, Q1) + 1; \ -]];
\]

(b) One pass of a 4-pass algorithm:

\[
[[+ Q0 = \min(Q0 – 1, Q5) + 1; \ +]];
\]

(c) The parallel algorithm will take a time \( \sim N^2 \times \frac{N}{2} \times 8 = N^3 \times 4 \), because it will need a number of passes equal to at most half the width of the largest object, and each pass will involve windows with 8 pixels apart from the centre pixel. The 2-pass sequential algorithm will take \( \sim N^2 \times 2 \times (5 \times 2) = N^2 \times 20 \) operations, as there are 4 active pixels in the windows but they involve conditional statements, so multiply by an additional 2. The 4-pass sequential algorithm will take \( \sim N^2 \times 4 \times (2 \times 2) = N^2 \times 16 \) operations. Thus it is doubtful whether the 4-pass algorithm will be any faster. In addition, it doesn’t run exactly the same algorithm, i.e. it will give a 4-connected rather than an 8-connected distance function.
6.16

In this question the key to locating the insect in the distance function is to ignore all local maxima larger than the half-width of the insect. Any values significantly smaller than this value can also be ignored. This means that the most of the local maxima in the image will be eliminated, apart from some isolated points within and between grains, and of course those along the middles of the insects. Applying an isolated point removal algorithm we can now keep only the insect maxima, and then we can downwards propagate to recover the insect boundaries. Any breaks in the edges will in general not cause breaks in the loci of local maxima, as slight sideways propagation will fill these, albeit giving slightly lower distance function values, which will, however, not upset the remainder of the algorithm.

6.17

The tracker must scan until it meets an object; then it must track around it, always going clockwise (say), until it arrives back at the starting point, moving in the same direction; then it must continue scanning.

At the given $\chi = 2$ point, it must take the direction of a 1 that has 0 adjacent to it in an anticlockwise sense. Thus we have:

$$\text{if } (A2 == 1 \text{ && } A3 == 0) \text{ direction } = 2, \text{ etc.}$$

In cases where $\chi \neq 2$, the tracker must move clockwise from the previous direction reversed until it finds the first 1. The important point is that it must remember the direction it has come from, whereas for the $\chi = 2$ case, it can work out which direction to go solely from the current window.

6.18

(a) It must go right round each object, which means that the first point must be passed again going in the same direction.

The algorithm is: go in the direction corresponding to the next direction clockwise from the reverse of the previous direction. It has to be coded correctly, which tends to mean calculating modulo 8 on the direction values minus 1 (to bring them into the range 0–7). However, note that this can also be done using a look-up table!

(b) With a simple image, such as a moderate sized circle and no noise in a large image, run length coding would do quite well, local maxima would do very well, and subsets of the local maxima could do really well (though detailed analysis of the situation is by no means trivial). Chain code is a must rather than an option for (i) and (ii).

(c) Noise makes the problem far worse, and a realization and careful discussion of this is required here. In fact this is probably the most important issue in the data compression part of the question.
7.2

If a boundary has already been found using the EDGE routine, it will be no more than 2 pixels wide, but will be 4-connected. Hence, disregarding junctions, it is only necessary to thin it to 8-connected form. We can achieve this by examining pixels sequentially within $3 \times 3$ windows, and eliminating any that are adjacent only to pairs of diagonally adjacent points. The algorithm for achieving this is:

\[
\text{if } (\text{sigma} == 2) \&\& \left((\text{A1} \&\& \text{A3}) \vee (\text{A3} \&\& \text{A5}) \vee (\text{A5} \&\& \text{A7}) \vee (\text{A7} \&\& \text{A1})) \right) \text{ then } \text{A0} = 0;
\]

It remains as an exercise to determine whether this algorithm solves the complete problem, or whether further pixels need to be eliminated from the boundary.

7.4

(a)–(c) The first part is bookwork and fairly short (see the main text in Chapter 7): just a few sketches are required. A circle or square is obtained direct from the centroidal profile by analysis. More exotic shapes have to be recognized by comparison with a 1D template.

(d) Full resolution comparison with a template takes $360^2$ steps.
Low resolution comparison with a template takes $360^2/n^2$ steps.
Full resolution refinement of orientation takes $n$ steps.

\[ \therefore \text{total load is } L = 360^2/n^2 + 360n \]
\[ \therefore \frac{dL}{dn} = -2 \times 360^2/n^3 + 360 \]
which is zero for $n = (2 \times 360)^{1/3} \approx 9$.

7.5

(a) Algorithm for eliminating salt and pepper noise:

\[
\text{noise: } \left[ \text{sigma} = \text{A1} + \text{A2} + \text{A3} + \text{A4} + \text{A5} + \text{A6} + \text{A7} + \text{A8}; \right. \\
\left. \text{if (sigma == 0) B0 = 0}; \right. \\
\left. \text{else if (sigma == 8) B0 = 1}; \right. \\
\left. \text{else B0 = A0}; \right. \\
\]

To eliminate short spurs, repeat the following modified version as necessary:

\[
\text{noize: } \left[ \text{sigma} = \text{A1} + \text{A2} + \text{A3} + \text{A4} + \text{A5} + \text{A6} + \text{A7} + \text{A8}; \right. \\
\left. \text{if (sigma <= 1) B0 = 0}; \right. \\
\left. \text{else if (sigma >= 7) B0 = 1}; \right. \\
\left. \text{else B0 = A0}; \right. \\
\]
(b) Shrinking once will eliminate the short spurs, but then expanding once will be necessary to try to restore the remaining parts of objects to their former sizes and shapes. In fact, there will be definite changes in shape, an obvious example being a dumbbell which will become disconnected into two ball shapes. The rule is that any part of an object which is thinner than \(2n+1\) pixels will be eliminated by \(n\) shrink operations.

(c) The \((r, \theta)\) graph method for describing shapes is bookwork (see the main text in Chapter 7): a circle is represented by a constant line at \(r = a\). A 3-element 1-D median filter will eliminate single element notches in the \((r, \theta)\) graph. A 5-element 1-D median filter will eliminate 2-element notches; and so on. Performance will perhaps be the same as for shrinking and expanding, except that it is undefined when the \((r, \theta)\) graph is not single-valued, as happens when there are holes or inlets in the shape. However, shrinking and expanding does not eschew such extreme cases. Thus we might say that the \((r, \theta)\) graph method is less effective except when boundaries are already quite smooth.

7.6

(a) The \((r, \theta)\) graph method for approximating a convex hull is:

```plaintext
repeat
    finished = true;
    [[ sigma = A1 + A2 + A3 + A4 + A5 + A6 + A7 + A8
        + (A1 && ! A2 && A3) + (A3 && ! A4 && A5)
        + (A5 && ! A6 && A7) + (A7 && ! A8 && A1);
        if (A0 == 0 && sigma > 3) {
            B0 = 1;
            finished = false;
        }
        else B0 = A0; ]];
    [[ A0 = B0; ]];
until finished;
```

This algorithm produces a crude octagonal approximation. If it is desired to produce a smooth analogue approximation, why not try using the \((r, \theta)\) representation. However, the mismatch produced by a straight line becoming a sec curve prevents this from being exact. Thus a straight line in the \((r, \theta)\) representation becomes almost a circular arc in the image representation (it is a circular arc if the straight line is at constant \(r\)). However, we could start with this approximation, and remembering where it has any effect, go back to the image and refine it by moving chords around until they are tangents ...
7.7

A one-pixel rotation of the square, i.e. an angle of \( \arctan(1/20) \), gives a sudden increase in boundary length, from 4, of \( 4(\sqrt{2} - 1) \), when the actual increase in boundary length should be \( 4/20 = 1/5 \). Thus there is gross distortion in the measurement, and this also occurs near 45°. The effect averages out somewhat over all angles, and the true average gives an overestimate of the boundary length of about 6%. This shows that something funny is happening, so the \( 1:\sqrt{2} \) ratio is at best misleading at these orientations. See the main text in Chapter 7 for more detail; also, refer to Kulpa (1977) and/or Davies (1991c). To tackle this problem, it is necessary to code the distances in larger windows, e.g. \( 3 \times 3 \) rather than \( 2 \times 2 \), as for the \( 1:\sqrt{2} \) approximation.
8.1

For proof, see the figures below—where \((A \oplus B)\) is labelled as C, and \((A \ominus B)\) is labelled as D. \(g\) is just the derivative of the original edge profile.

binary case

grey-scale case
9.1

(a) The first part is simple coordinate geometry:

\[ g_{y}/g_{x} = y_{f}/x_{f} = a, \text{ and } (x - x_{f})x_{f} + (y - y_{f})y_{f} = 0 \]

Eliminating \( y_{f} \) gives:

\[ xx_{f} + ayx_{f} = x_{f}^{2}(1 + a^{2}) \]
\[ \therefore \quad x_{f} = (x + ay)/(1 + a^{2}) = g_{x} \times (xg_{x} + yg_{y})/(g_{x}^{2} + g_{y}^{2}) \]
and
\[ y_{f} = g_{y} \times (xg_{x} + yg_{y})/(g_{x}^{2} + g_{y}^{2}) \]

(b) When a line goes through the top left corner of an image at an angle of about 60°, the foot of normal to it is outside the image. It lies on a semicircle on the image centre to top left corner as diameter, and passing through the mid-point of the left side (obvious from the fact that the x-axis hits the LH side normally).

(c) This form of the HT is equally as robust as the conventional Hough transform. It is actually faster since the \( \arctan \) function does not have to be computed; also no square roots have to be found.

9.2

(a) See Problem 11.1(b).

(b) Less computation is needed since only the boundary has to be searched for peaks and not the whole image. Hence (a) computation is proportional to \( N \) rather than \( N^{2} \) for an \( N \times N \) image. Also (b), for a case of \( p \) lines, the number of peaks is \( 2p \) rather than \( p \), but the number of possible pairs of peaks to be considered is \( \binom{2p}{2} \approx 2p^{2} \), which is much larger than \( p \).

(c) This method could involve a lot of computation for checking between the 1-D histograms for correct solutions, and the result may be less robust if a great many line segments appear in the image.

9.3

(a) The Hough transform is a means of rapidly searching images for particular features, including circles, ellipses and straight lines. It is especially robust as it looks for evidence only: it ignores missing data such as would occur with partial occlusion. In this way it is far better than the centroidal profile approach, which takes all the data as given and is easily misled by partial occlusions, touching objects and so on. Specifically, it refers all its measurements to the centroid of the shape, thus building on a shaky foundation if part of the shape is missing or part of another object is added. For circles, the Hough
transform is found by moving inwards from every edge point and accumulating a point in parameter space at a distance equal to $R$; then peaks in parameter space correspond to circle centres or centres of whole circles of which only parts may be visible in the image.

(b) To locate straight edges, each edge segment in the image is extended and the distance $\rho$ from the origin is measured, together with the direction $\theta$ of the edge normal. Then a point $(\rho, \theta)$ is accumulated in $\rho, \theta$ parameter space. When all such points have been accumulated, peaks are found, and each one is taken to correspond to a line in the original image. To find each line, the $\rho, \theta$ values are used to move from the origin by a distance $\rho$ in a direction $\theta$ and draw a line through this point perpendicular to the direction $\theta$. If curved edges are present in the image, these lead to lots of isolated points in parameter space, and should not give rise to additional peaks. Thus there is only a very low chance of erroneous interpretations.

(c) When a square object is present, this will give four peaks in parameter space, and these will be at predictable positions: specifically, one pair will have the same $\theta = \theta_1$, and the other pair will have equal $\theta = \theta_2 = \theta_1 \pm 90^\circ$, and the peaks will actually fall at the corners of a parallelogram. The dimensions of the parallelogram lead to values for the size and orientation of the square, and also its position. The algorithm would contain the following elements: clear parameter space, accumulate parameter space, find peaks in parameter space, deduce the presence of a square from the identification of the parallelogram corners, work out the size and orientation of the square, confirm it is a square from the equality of the various $\rho$ difference values, work out the positions of the sides of the square, and finally, work out the positions of the corners if necessary.

(d) (i) If parts of some sides of the square were occluded, this would not matter, as the Hough transform peak positions would still reveal the presence of the square. (ii) If one side of the square were missing, there would be enough information to confirm the presence of an occluded square and to work out its size; if another side were missing things would get more difficult; we then have to ask questions as to whether these sides are completely or partly missing and whether the remaining sides are guaranteed completely intact, but it becomes increasingly likely that we could only infer that a rectangle is present. (iii) If several squares appeared in the image, the Hough transform approach should be able to find and identify them all, in general without confusion. (iv) If several of these complications occur together, the algorithm would be certain of some of the squares being present but doubtful about others.

(e) It is crucial to this type of algorithm to have edge detectors that are capable of accurately determining edge orientation—say to $\sim 1^\circ$. The Sobel edge detector is capable of achieving this. This is a vector operator edge detector, and finds direction as $\arctan(g_y/g_x)$. 

© E.R. Davies 2005
10.1

In Fig. 10.7, when D is close to C, PD will be approximately normal to the boundary at P, and similarly CA will be approximately normal to PD. Therefore AD will be approximately normal to CA. Hence A will lie on, or extremely close to, a small circle whose diameter is CD.

10.2

(a) The Hough transform is valuable because it is inherently robust against noise, defects and occlusions. Description of the Hough transform for circle detection is bookwork (see the main text in Chapter 10); so is ellipse detection (see Chapter 12). However, the latter involves use of pairs of edge points, and as a result it is automatically sensitive to object size. Since a circle is a special type of ellipse, we can use ellipse detection to detect all sizes of circle: no specific change is needed in the algorithm.

(b) A Hough transform is merely a parameter space in which data is accumulated, so an accumulating histogram is simply a 1-D Hough transform. The problem with this approach is that the two 1-D transforms are disconnected and independent, and it is not known which solution of the one corresponds to a solution of the other. Thus the Hough transforms lead to hypotheses which in this case must be checked. In addition, the method is less robust as it requires special pairs of edge points to be taken into account. Finally, the method does lead to an advantage, in that it can be made much faster, both by sampling and by use of simpler edge detectors. (See the main text in Chapter 10.)

(c) In this case take the ends of the chord as being \((x_1, y)\) and \((x_2, y)\), and the length of the chord as \(d\):

Assuming that the circle radius is known to be \(R\), Pythagoras’ theorem gives:

\[
R^2 = h^2 + \left(\frac{d}{2}\right)^2
\]

where \(d = x_2 - x_1\). We can now deduce the circle centre as \((x_c, y_c)\) where:

\[
x_c = \frac{x_1 + x_2}{2}
\]

\[
y_c = y \pm h = y \pm \left[R^2 - (x_2 - x_1)^2\right]^{1/2}
\]

This is in many ways a cleaner method than (b), and doesn’t depend on relating 1-D
transforms in the \( x \) and \( y \) directions, which could be tedious in complex images with many edge points. Thus it could well be faster, in spite of the need for square root computations. It could also be more robust. The sign ambiguity is far less ambiguous than that inherent in method (b), and would be coped with easily by the Hough transform voting technique. Of course, any method that depends on finding chords could be upset by texture (see also the main text in Chapter 10), and the final outcome for any such algorithm must depend on the detailed characteristics of the image data.
11.1

(a) The Hough transform operates by taking a parameter space (often congruent to image space), clearing it, accumulating votes to build up evidence that particular sets of parameter values are significant and represent objects in the image, and analysing the parameter space to locate significant peaks representing objects in the image, and finally deciding that these objects are present (perhaps after further checking). The Hough transform operates by building up evidence for particular solutions: it ignores irrelevant information. If there is partial information, it accepts it. Thus it is able to infer the presence of objects from information that is restricted—whether because of noise, object distortions, occlusions, etc. In this sense the method is robust. Above all, the robustness arises because of the transform’s capability for ignoring irrelevant information.

(b) It is said that the Hough transform only leads to hypotheses about the presence of objects in images, and that they should all be checked independently before making a final decision about the contents of any image. This statement is entirely accurate, in principle. In practice the risk involved in not checking may be negligible (you only find widgets on a widget line).
12.2

(a) The diameter-bisection method will detect hyperbolas only if both branches of the hyperbola are present. The chord–tangent method cannot be used for this purpose. The GHT can be used.

(b) The diameter-bisection method will work in this case. So will the GHT. The chord–tangent method cannot be used.

(c) Only the GHT will work in this case.

12.4

The diameter-bisection method involves making a list of all the edge points in the image, and searching through this for instances of antiparallel edge points. The bisectors of lines joining such pairs of edge points are accumulated in parameter space as candidate centre locations. The chord–tangent method involves listing all edge points again, and finding the tangents and chord bisectors of pairs, and then joining, producing and accumulating the lines from the tangent junctions to the chord bisectors. It is justified for circles by symmetry, as the accumulating line must pass through the circle centre. In the case of ellipses, simple orthographic projection of the circle case proves it sufficiently, because points, straight lines, chords, tangents, midpoints, and conic sections all project into these same respective entities. (Note that perspective projection does not lead to a viable proof, as midpoints do not project into midpoints.)

12.5

The chord–tangent method can be used with circles as well as ellipses, and has the advantage that it works for all sizes of circle. However, the usual circle Hough transform does not, so it normally has to be augmented by using a 3-D \((x, y, r)\) parameter space. In this case the latter option will be better, as only a few radius values will be relevant (only a limited number of coins can be inserted into a vending machine!).

12.6

The diameter-bisection method involves making a list of all the edge points in the image, and searching through this for instances of antiparallel edge points. The bisectors of lines joining such pairs of edge points are accumulated in parameter space as candidate centre locations. The chord–tangent method involves listing all edge points again, and finding the tangents and chord bisectors of pairs, and then joining, producing and accumulating the lines from the tangent junctions to the chord bisectors. It is justified for circles by symmetry, as the accumulating line must pass through the circle centre. In the case of ellipses, simple orthographic projection of the circle case proves it sufficiently, because points, straight lines, chords, tangents, midpoints, and conic sections all project into these same respective entities. (Note that perspective projection does not lead to a viable proof, as midpoints do not project into midpoints.)
12.7

The diameter-bisection method would be speeded up so that execution time is approximately proportional to $N$ rather than $N^2$.

12.8

When several identical equally orientated ellipses appear in the image, centres will be found half way between them. These may be suppressed by only permitting centre votes to appear by moving *inwards* from any edge point. Any symmetrical shape is detected by this method. To avoid alternative shapes, specific tests need to be made for ellipses. The safest such test is to superimpose the deduced ellipse and see whether the number of coincidences with edge points is sufficient.
13.3

For an image of $N \times N$ pixels, each histogram bin is accumulated from $N$ pixel intensity values. Assuming values of $N$ much larger than the hole size, the total signal in each histogram bin will remain constant, but the noise (error) level will be proportional to $\sqrt{N}$. Hence the signal-to-noise ratio and probability of detection will gradually decrease with increase in $N$. In addition, accuracy will be reduced as $N$ increases. While amplitude noise is not the same as position noise, it seems likely that the standard deviation in estimating hole position will vary in proportion to $\sqrt{N}$, at least for moderate values of $N$ (for high values of $N$, the detection sensitivity will fall to zero well before the position error approaches the width of a hole). More important, the limiting factor will be image clutter rather than noise, and the probability of competing events in the clutter will increase in proportion to $N$. To improve the situation, the subimage approach should be used (see the main text in Chapter 13).

13.5

(a) The computational load of template matching is proportional to the area of each template and also to the number of times it has to be applied to test for different orientations, which is a greater number of times for large templates. Hence the templates have to be reduced in size for practicality. Then there is the problem of inferring the presence of the objects from the features.

(b) For corner detection, a typical mask is:

\[
\begin{bmatrix}
-4 & 5 & 5 \\
-4 & 5 & 5 \\
-4 & -4 & -4
\end{bmatrix}
\]

The more obvious 5’s mask (with all the –4’s being zeros) would have a positive sum which would give a response strongly dependent on the average intensity of the image. To avoid image average intensity from affecting the response, zero-mean masks have to be used. Eight masks are needed for corner detection.

(c, d) Clearly, a diagonal line of 1’s on a 0’s background will give a positive response with the given mask. Eight masks are needed in principle, but four of these would be identical to the other four, so only four are needed.

(e) A hole detection mask, with an 8 surrounded by eight –1’s, would give a positive response ($8-1-1 = 6$) for a diagonal line of 1’s. Hence line segments can be detected by a single hole detection mask, giving a saving in computation. Similarly, a hole consisting of an 8 surrounded by eight –1’s will give a signal of $16-4-4+1+1+1+1+1=14$ with the above line detection mask, so the latter will also detect holes.

(f) The reason why masks may detect objects which they are not specifically designed to detect is that the total response from all the pixels is quite likely not to cancel to zero. However, they will normally pick up more irrelevant noise. Indeed, the principle of matched filtering indicates that the optimum (in a signal-to-noise ratio sense) mask is the
one having the same profile as the signal it is to detect. While SNR is not everything, it reflects a principle of discriminability against clutter or other types of object, and again, the best mask to use will be one that is matched to the signal in question.
14.1
This is similar to the case of a semicircle covered by Davies (1989c). Here the localization point is taken to be the centre of the circular part of the boundary. The remaining two sides are perpendicular to each other and to the adjacent parts of the boundary, so the minimum exterior angle (excluding the infinite number of short sides around the circular section) is $\pi/2$, which is exactly the same as for the semicircle. Thus four planes are required in parameter space to obtain optimal sensitivity. A full analysis of this type of problem is given by Davies (1989c).

14.2
The cases of the parallelogram and the rhombus are the same, insofar as the angles of the polygons are made to be the same: the relative lengths of the sides differ, but this has no effect on the number of planes needed, the number being governed by the exterior angles of the polygons. In both cases, the order $G$ of the group of identity rotations of the polygons is 2, and thus the minimum number of planes needed in parameter space for optimal detection is given by:

$$M \geq \frac{(2\pi G)}{\psi_{\text{min}}} = \frac{\pi}{\psi_{\text{min}}}$$

where $\psi_{\text{min}}$ is the minimum exterior angle.

For clarification of these ideas, see the main text in Chapter 14, and the original paper (Davies, 1989c).

14.4
In this case a trick is used to eliminate the smallest two exterior angles from consideration, by choosing the localization point to be on both of the interior angle bisectors. Then the number of planes needed in parameter space is determined by the next largest exterior angle which essentially becomes $\psi_{\text{min}}$. Thus the formula for the minimum exterior angle of an almost regular polygon essentially changes from $2\pi/(N+1)$ to $2\pi/(N-1)$. This means that the number of planes required in parameter space falls from $N+1$ to $N-1$.

In the case of a semicircle, the localization point can be taken to be the centre of the circular part of the boundary. The remaining two sides are perpendicular to the adjacent parts of the boundary, so the minimum exterior angle (excluding the infinite number of short sides around the circular section) is $\pi/2$. Thus four planes are required in parameter space to obtain optimal sensitivity. A full analysis of this type of problem is given by Davies (1989c).
15.1

Figure 15.S1 shows (a) an isosceles triangle template; (b) a basic example image; (c) the match graph; (d) the symmetry-reduced match graph; (e) a kite template; (f) a basic example image; (g) the match graph; (h) the symmetry-reduced match graph. For (d), the labels on the template have been changed from A to $\alpha$, and from B and C to $\beta$. For (h), the labels on the template have been changed from A to $\alpha$, from B and C to $\beta$, and from D to $\delta$.

In Fig. 15.S1c there are two large maximal cliques representing the same solution, while in Fig. 15.S1d there is only one large maximal clique representing a unique solution, as required by reality. A similar situation applies for Figs. 15.S1g and h. (Note that in Fig. 15.S1g, only the two large maximal cliques are shown, as the complete set would be too confusing.)
15.4

(a) The maximal clique method is bookwork (see the main text in Chapter 15), but this is a genuine problem. Each match graph will have $4 \times (4 - 1) = 12$ nodes. The template for the flange is labelled A, B, C, D (as in Figure 15.S4a), and the image holes are labelled 1, 2, 3: it is not known to the computer which is which of the set 1–3 in any of the four cases. Note that there are a lot of matches of hole pairs because several inter-hole distances are identical.

Take the four cases in turn, with the missing holes being respectively A, B, C, D:

(i) A missing: no problem.
(ii) B missing: ambiguity—we don’t know which way round the flange is.
(iii) C missing: no problem.
(iv) D missing: ambiguity—we don’t know which way round the flange is.

(b) The maximal clique method will immediately show the two problems by coming up with two cliques of size 3 in the two cases (see Figure 15.S4b). No errors can occur, only ambiguities. The results tally exactly with those for human perception. Ambiguity can be resolved by taking the solutions as hypotheses and testing them, e.g. by looking for the corners of the bar. Alternatively (more robust but less efficient), the corners could be included in the initial analysis.

![Figure 15.S4](image-url)
15.5

(a) A clique is a set whose elements are all compatible with each other in some sense. In matching problems the criterion is mutual compatibility. A maximal clique is one that cannot be enlarged by including further vertices. The largest maximal clique amongst a set with known compatibility relations is the one having the largest number of agreements within it, and hence the one with greatest levels of confidence. Hence this is the one supporting the strongest interpretative hypothesis about the system.

(b) When symmetry occurs, the match graph containing all the compatibility relations is richer and thus more complex to analyse. However, the symmetry permits relabelling which can simplify the system. For a rectangle, each corner has the same potential labelling A, so the number of feature label matches can be reduced from $4 \times 4$ to $1 \times 4$. For a parallelogram, there are two symmetry labels, A and C both becoming A, and B and D both becoming B. Thus the number of feature label matches can be reduced from $4 \times 4$ to $2 \times 4$—still a substantial reduction.

(c) The case in Figure 15.P2 allows symmetry to be applied to features A–D, these being relabelled as A, B, while the E, when re-included, can remain as E. Thus we have three types of labels, so overall the original $5 \times 5$ is reduced to $3 \times 5$. However, even if in the end the original labelling is reinstated (as really it should be), while the number of compatibilities to be considered is the same as it would originally have been, nonetheless, the solution is largely known, so a far less wide-ranging search need be carried out to find the final maximal cliques. The disadvantage of the approach lies in its reduced resistance to occlusion and breakage, because initially the benefit of the E (or other features in a more complex case) is not felt.

15.6

(a) Template matching involves impractically large numbers of operations for large objects, so it is better to use small features and then infer the presence of objects. Edge and line segments, corners, holes, circle segments and other salient features are normally used for feature detection.

(b) Corners are typically detected by masks of the type:

\[-4 & -4 & 5 & -4 & 5 & 5 \\
-4 & 5 & 5 & -4 & 5 & 5 \\
-4 & -4 & 5 & -4 & -4 & -4 \]

Holes are typically detected by masks of the type:

\[-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1 \]

(c) This problem uses the standard maximal clique method which is bookwork (see the main text in Chapter 15). Detecting the blade from its four corners is a straightforward
application of a standard example. Similarly, locating it from its three holes is standard. Except that in both cases it should be noted that the object is symmetrical, so the template has only two labels in the first case and only two in the second.

(d) When overlaps occur, some of the blade will be missing and there will be ambiguity in either of the above cases if either two corners are missing or one hole is missing. In extreme cases we need all the features we can get. So consider all the corners and holes equally as point features. This gives seven features in the ideal case, both in the template and in the image, so the match graph has 49 feature assignments. However, symmetry reduces the situation to four template labels giving just 28 feature assignments, and resulting in one maximal clique of size seven. The situation is actually simpler when occlusion occurs, as the number of features in the image may then be just four, so the match graph contains 16 feature assignments, and there will be just one maximal clique of size four. Finally, if say two corners remain unoccluded, extra information can be obtained from corner orientations which can help confirm the overall solution. This will be especially relevant if noise points appear like holes and we need to know whether the data to hand is completely consistent.

15.7

(a, b) (i) A standard if tedious solution to the maximal clique problem (Figure 15.S7a): note the extra compatibilities caused by the fact that distance $AE = DE$.

(ii) A much simpler solution (Figure 15.S7b).

(iii) An even simpler (and trivial) solution (Figure 15.S7c).

(iv) A more complex situation, but manageable if dealt with systematically (Figure 15.S7d).

In fact (iv) is rather nice, as it is only necessary to include exactly the right feature associations in the match graph. Hence we get a fully general solution with less effort than the standard solution (i).

(c) As a result, (iv) is faster than (i), though (ii) and a fortiori (iii) are very fast. On the other hand (ii) is not very robust, and (iii) is very non-robust. (i) could possibly give rise to additional wrong solutions, whereas (iv) is ideal really.

(d) The time taken to build a match graph is proportional to $MN$ where $M$ is the number of features on an ideal object that are actually used and $N$ is the number of points in the image that are actually used. Hence the respective build times are basically: $6 \times 6$, $4 \times 4$, $2 \times 2$, $6 \times 6$, the first case having a lot more extraneous small cliques which require additional build time (trivial). Let’s assume that the time taken to find a maximal clique of up to $m$ nodes in a match graph of $n$ nodes (where $m \leq n$), is approximately $t_0 \exp(mn)$, where $t_0$ is a constant of the system. In these simple problems with no occlusions or irrelevant features, we have $m = M$ and $n = MN$. Thus the clique finding times are basically: $\exp(6 \times 36)$, $\exp(4 \times 16)$, $\exp(2 \times 4)$, $\exp(6 \times 36)$. More detailed calculation or argument will show that (iv) is faster than (i).
For reasons of clarity, not all compatibility lines are shown in this figure: in (a) a good many lines have been omitted, and in (d) only the main maximal clique is shown. (b) and (c) are complete.

15.8

(a) Numbering the image features in forward raster scan order as 1, 2, 3, 4, 5, 6, and the template features as A, B, C, D in the same order as those in the first object in the image, we find the following compatibilities:

A1–B2, A2–B1; B2–C3, B3–C2; C3–D4, C4–D3; D4–A1, D1–A4; A1–C3, A3–C1; B2–D4, B4–D2; A5–C6, A6–C5.

This can be regrouped as the following maximal cliques:
A1–B2–C3–D4; A2–B1; B3–C2; C4–D3; D1–A4; A3–C1; B4–D2; A5–C6; A6–C5.

The 4-element maximal clique wins over the first six 2-element cliques, leaving the last two 2-element maximal cliques, which constitute equally probable solutions at this level.

Thus we have an almost certain 4-feature object, and a possible 2-feature object which is partly occluded or defective and which may be either way around.

(b) If the objects have an axis of symmetry, we will get more compatibilities, as the two features B, C related by symmetry cannot be distinguished. The final result is that the two objects each have twice as many maximal clique solutions:

A1–B2–C3–D4; A1–C2–B3–D4; A5–C6; A6–C5; A5–B6; A6–B5.

We can overcome this complication by labelling B and C as $\beta$, in which case there is just the following set of solutions:

A1–$\beta$2–$\beta$3–D4; A5–$\beta$6; A6–$\beta$5.

In addition, the size of the match graph goes down from $4 \times 6$ to $3 \times 6$, saving storage and computation, and arriving at only the meaningful answers.

(c) For the case where there are three sizes of hole, the extraneous 2-element cliques will mostly not arise, the only one that will remain will be D1–A4. Thus the final solutions are:

A1–B2–C3–D4; D1–A4; A5–C6

which is entirely satisfactory.

(d) However, if the further matching strategy is employed, the equal sized holes in the image are features 1, 4, 5, and the equal holes in the template are A and D. This means that we get just the following subset of compatibilities:


This leads to the following cliques:


Both the 3-element cliques represent the same object, but there is only one clique to represent the occluded object. Thus the technique locates both the objects, in spite of the occlusion, though some means must be adopted for choosing between the two solutions for the same object.

The disadvantage of this method is that it takes in less evidence about the objects, and thus is less robust. Thus it could occasionally miss objects or interpret them as arising from noise features. In general, it is not a good strategy, but may be justified in a more complicated problem in that it zones in on the right solutions quickly, after which one can revert to normal but accelerated clique finding.
(e) An optimal object identification strategy is one which uses all possible information to prevent unnecessary compatibilities from getting into the match graph and thus complicating clique finding. However, as indicated above, cutting down the information artificially even more than this may make the process of clique finding even quicker, though ultimately it makes the method less robust and less sensitive, so in the end we have to revert to exactly what the image tells us. ‘Optimal’ means quick and sensitive and robust in being able to overcome the effects of occlusions and other artefacts: there will be a tradeoff between these parameters.

15.9

(a) The first stage in carrying out the maximal clique technique is to draw a match graph showing how the object (feature points 1–4 in the image) matches to a template (feature points A–D). Feature pairs are considered to see if they are the same distance apart in the two cases: if so a compatibility line is drawn between the appropriate combination, A1 and B2, but also between A2 and B1, as these are the same distance and the object could be the other way around. After all compatibilities have been marked between the vertices (e.g. A1, B2, A2, B1, C3, D4, ...) in the match graph, we look for maximum sets of linkages—cliques—and indeed the maximal cliques, i.e. those not contained in other cliques, and of these the largest maximal cliques represent the most likely solutions, as like Hough transform peaks they offer the most votes for objects. If an object is partially obscured, still the largest maximal cliques will reveal what is consistent about the evidence for the objects and they will be found robustly.

(b) The basic algorithm will not distinguish between widgets that are normally presented from those that are upside down, as it is only looking for evidence arising from distances apart of the various features: being upside down makes no difference to this. However, if we look at the orientations of the lines joining the features, and see whether there are positive or negative angles between crucial lines, we will find which ways up the objects are.

(c) If the camera used to view the widgets is accidentally jarred and then reset at a different height, the inter-feature distances will mostly not match the ideal template. However, we can make a list of inter-feature distances in the image and the template, and make them match by scaling. We can then run the maximum clique algorithm. This will work if all the objects are widgets. If this is not the case, we take the two lists and then see if we can find ratios of distances which match. This will locate the true feature distances, and then we can proceed as before. There could be problems in practice, but at least a number of reasonable methods seem to be available for trial.

(d) If the camera is set at an unknown angle to the vertical, examination of a circular calibration object would permit it to be identified as an ellipse and the axis and amount of rotation worked out. Then the whole image could be expanded by the right amount in the right direction, in order to eliminate the distortion, and analysis could proceed as before. This assumes that perspective distortion is not produced by excessive rotation of the camera.
16.1

Consider a point P in image 1 whose centre of projection is C. The epipolar line L for image 2 occurs where plane PC_{1}C_{2} crosses image plane 2. For a separate point Q in image 1, the epipolar line M in image 2 occurs where plane QC_{1}C_{2} crosses image plane 2. All these planes (which cross image plane 2) contain line C_{1}C_{2}. This line passes though image plane 2 at the (single) point R. As C_{1} lies on C_{2}R, by definition R is the image of projection point C_{1} in image 2. (Note that R may not actually lie within the confines of image 2, but on the extended plane containing it.)

16.2

The straight line in gradient space corresponds to the case when the cone angle is 90° and thus is completely flat—i.e. a plane. In that case it intersects with the z = 1 plane in a straight line, namely the line that is under discussion here. But why should the cone be flattened out? Or rather, when should it be flattened out? This occurs when the angle of incidence $i$ is 90°, so that cos $i = 0$, and zero light emerges from the surface. This means that the straight line in gradient space corresponds to $R = 0$.

16.6

An alien with three eyes is still subject to equation (16.5) and any derivative of it that is used to estimate accuracy, and hence is in the same general situation as a human. On the other hand, this equation only applies for every position that is seen in at least two eyes. Hence, with three eyes, the alien should be able to see more and to get a slightly better depth view of the world. A lot also depends on the value of $b$ that is used (the baseline between a pair of eyes). Slightly more averaging is also possible, giving slightly greater accuracy for points that are visible in all three eyes.

16.8

(a) The two equations are deduced by examining the similar triangles in a simple 2-D cross-section of the scene:

$$(X + b/2)/x_{1} = Z/f$$
$$(X - b/2)/x_{2} = Z/f$$

(b) They are solved by cross-multiplying:

$$(X + b/2) = x_{1}Z/f$$
$$(X - b/2) = x_{2}Z/f$$

then subtracting the second from the first, getting the formula:

$$b = (x_{1} - x_{2})Z/f$$
This gives the disparity $D$:

$$D = x_1 - x_2 = bf/Z$$

$$\therefore Z = bf/D$$

16.9

From the main text in Chapter 16, the standard formulae for depth from disparity are:

$$f/Z = x_1/(X + b/2) = x_2/(X - b/2)$$

$$\therefore D = x_1 - x_2 = [(X + b/2) - (X - b/2)] \times f/Z = bf/Z$$

$$\therefore Z = bf/D$$

$$\therefore dZ/dD = -bf/D^2$$

so \((1/Z)dZ/dD = -1/D = -Z/bf\)

$$\therefore |\delta Z/Z| = |\delta D| \times Z/bf = (|\delta D|/D) \times (D/f) \times (Z/b) = (|\delta D_p|/D_p) \times (D/f) \times (Z/b)$$

where the p suffices denote measurement in pixels. Hence the error in measuring the fractional depth in a scene is (i) proportional to pixel size, (ii) proportional to $Z$, and (iii) inversely proportional to $b$.

16.10

(a) A simple but clearly drawn figure will show immediately that the ordering $A_1, B_1, C_1, D_1$ in the first image is the same $A_2, B_2, C_2, D_2$ as in the second image, and the same $A, B, C, D$ as on the object—assuming that all these points are visible in both images. For an interior point $F$, the ordering of $F_1$ and $F_2$ could be anywhere amongst $A_1, B_1, C_1, D_1$ and $A_2, B_2, C_2, D_2$ respectively. However, it is not unlikely that we will get a response like:

$$A_1, B_1, F_1, C_1, D_1$$

and

$$A_2, B_2, C_2, F_2, D_2$$

(b) Such a response becomes increasingly likely as the number of observed features increases. Obviously this is ultimately the same effect as parallax. Note that ordering is sufficient to show what is going on, but measurement of disparity is still required to gauge the relative depths.
16.11

(a) These terms are bookwork (see the main text in Chapter 16). Ambiguities arise because one must infer what’s happening in 3-D from limited 2-D data. This requires assumptions which may not be justified.

Lambertian surfaces have reflectivities which depend only on the angle of incidence \( i \). Furthermore, the variation is as \( \cos i \). This is incredibly simple leading to the surface normal lying in a cone of directions around the incident light direction.

Two such cones intersect in two lines defining two directions in space:

A third cone is needed to make the surface normal completely unambiguous. If the absolute reflectance \( R \) of the surface is unknown, it might reasonably be thought that the cones must be grown until they lead to a single solution, as above. However, the first solution need not be unique. To disambiguate the situation, one might think that at least four cones would be needed, but this is not so: e.g. in the case of three lights in mutually perpendicular directions, the sums of the squares of the cosines of the angles relative to the axes must add to unity, showing that only two of the angles are independent; hence the three sets of data can determine \( R \) and two independent angles, and thus lead to a full solution. Another relevant factor is that any element of specular reflection could be discerned and allowed for by using more than three sources.

(b) Photometric stereo (as above) leads hopefully to a unique set of surface orientations. These must be integrated over space to obtain depth maps. Hence binocular vision is to be preferred. However, for smooth featureless surfaces binocular vision is ineffective.

16.12

(a) Matte surfaces reflect light diffusely and ideally do not give any purely specular components of reflection. The best model of matt surfaces is the Lambertian model, which (i) reflects light in such a way that it is only a function of the angle of incidence \( i \), and (ii) reflects light such that the intensity is proportional to \( \cos i \) \times \text{a constant reflection factor (the albedo)}.\n
(b) As the light reflected by a Lambertian surface depends only on \( \cos i \), for a given reflected light intensity \( I \), we will only know that the angle of incidence is given by \( \cos i = \text{some constant} \times I \). Hence \( i = \text{constant} \), so the direction of the surface normal must lie in a cone whose half angle is \( i \), centred around the incident light direction.
(c) With three lights, each surface normal must lie in each of three cones having a common apex: two cones intersect in two lines, and the third cone, *in general position*, will narrow this down to a single line, giving a unique interpretation and estimate of the direction of the surface normal. If any points are hidden from any of the lights, an element of ambiguity will remain. The three lights must not lie in a straight line, and should be spread out widely relative to the surface, without being occluded from any of it. Four lights would help in practical cases by indicating any inconsistent interpretations, which may for instance be due to spurious specular reflections.

Case of two lights: situation for three or four lights is similar.

(d) The surface map that is obtained is an orientation map, which distinguishes it from that for binocular vision which leads to a depth map (though in principle integrating an orientation map will yield a depth map, after much computation). The two approaches are best applied to different situations: binocular vision is best applied to textured or highly featured surfaces, while shape from shading is best applied to plain surfaces. Structured light can generate a coarse surface texture which will permit binocular vision to work on plain surfaces, but not really to the resolution available for shape from shading.

(e) Either of these approaches only provides information about surface shape. Recognition requires further processing to identify likely candidates for specific objects: e.g. the existence of an egg can only be revealed after modeling the shape as an oval shape rather than a set of coordinates (this is similar to what the Hough transform does during circle location).
17.1

The complete table of pose ambiguities that arise for *weak* perspective projection, for various numbers of object points identified in the image, is:

<table>
<thead>
<tr>
<th>arrangement of the points</th>
<th>$N$</th>
<th>no. of ambiguities with WPP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 coplanar</td>
<td>$\leq 2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\geq 5$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>2</td>
</tr>
<tr>
<td>2 non-coplanar</td>
<td>$\leq 2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\geq 5$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>1</td>
</tr>
</tbody>
</table>

All the 2’s that occur in the coplanar case correspond to perspective inversion. Any non-coplanar points (above the minimum of 3 corresponding to the planar case) add sufficient information to break the ambiguity.

17.2

Full perspective projection is the normal case of perspective projection, and weak perspective projection is the special case when a particular object is viewed and the depth of the object $\Delta Z$ is much less than its depth $Z$ in the scene. In this situation, the image can be thought of as being the result of orthographic projection in which the depth information is simply wiped out, together with the application of a scale factor which brings the object to the observed size. Of the named cases, (a)–(e) are reproduced exactly under weak perspective projection, but circles become ellipses: specifically, mid-points remain mid-points and parallel lines remain parallel lines. However, for full perspective projection, mid-points do not remain mid-points and parallel lines do not remain parallel. (f) The centres of circles remain as centres of the resulting ellipses only for weak perspective projection.
17.3

In (a) the ambiguity arises as positive and negative inclinations to the viewing direction cannot be distinguished because the profile is the same. In the end this comes down to the fact that \( \cos(-\alpha) = \cos \alpha \). In (b) the fourth point adds no additional information as its position is deducible from the other three, but for (d) this is not so, so the ambiguity is then eliminated—as a perspective figure illustrates. In (c) the fourth point does distinguish \(-\alpha\) from \(\alpha\), so the ambiguity is eliminated.

\[ d \]
18.1

If more sources of error are applied to the same data, the variances have to be added. Furthermore, if each source contributes equal amounts of error, the variance has to be multiplied by the number of error sources $M$. In the inverse situation, if more data is supplied and the error sources are unchanged, the variances are divided by the total number of data points $N$. Thus there is a natural ratio $M/N$ governing the overall error. In a considerable proportion of the applications of statistics, variances are combined by addition, because various sources of error have to be combined: here the question states what many might feel is the usual situation—that variances are normally combined by addition. However, this ignores the contrary situation where averaging is used to reduce errors: arguably, people are so used to averaging to reduce error that they don’t get around to thinking that the result is expressed not merely by an average but by an associated variance of reduced size. Moreover, when the data points have to be given unequal weightings, rather fewer people will know quite how this affects the variance: this is essentially what is being discussed in relation to equation (18.56).
19.3

With no loss of generality we can take the conic to be an ellipse. If the two points are inside the ellipse, draw a line joining the two points and extend it to cut the ellipse in two further points. Use all four points to give a cross ratio. The same method applies if one of the initial points is outside the ellipse.

If the two points are outside the ellipse, draw two tangents from each of the points, and consider the four points of contact. Apply Chasles’ theorem to these four points to give a cross ratio.

19.4

(a) If the two conics intersect in four points, apply Chasles’ theorem to one of the ellipses and use the four points to give a cross ratio.

(b) If the two conics intersect in two points, draw the tangents to one ellipse at the points of intersection, and find where they intersect the other ellipse again. We now have four points on the latter ellipse and we can use Chasles’ theorem to give a cross ratio.

(c) If the two conics do not intersect, take their four common tangents, and apply Chasles’ theorem to the four points of contact on either conic to give a cross ratio.

19.5

(a) Following the notation of Figure 19.S5, and applying the sine rule four times, we find:

\[ A = \frac{d}{\sin \theta} = \frac{a}{\sin \alpha} \]
\[ B = \frac{e}{\sin \theta} = \frac{(a + b)}{\sin(\alpha + \beta)} \]
\[ C = \frac{d}{\sin \phi} = \frac{(b + c)}{\sin(\beta + \gamma)} \]
\[ D = \frac{e}{\sin \phi} = \frac{c}{\sin \gamma} \]

To eliminate \( d, e, \theta \) and \( \phi \), we form the ratio:
\[ \frac{AD}{BC} = 1 = \frac{ac}{\sin \alpha \sin \gamma} \frac{1}{(a + b)(b + c) \sin(\alpha + \beta) \sin(\beta + \gamma)} \]

Hence we find:

\[ \frac{ac}{(a + b)(b + c)} = \frac{\sin \alpha \sin \gamma}{\sin(\alpha + \beta) \sin(\beta + \gamma)} \]

as required.

(b) The derived equation shows that the cross ratio on the left would be the same for any line crossing the pencil of four lines through O. It also shows that there is an equivalent cross ratio for angles.

19.6

(a) Invariants are measures that are invariant to certain transformations such as translation and rotation. They are able to help characterize objects uniquely irrespective of position and orientation—especially important in complex 3-D situations. Thinning reduces 2-D objects to unit width thereby introducing a degree of normalization and aiding invariant (to thickness) recognition.

(b) The ratio idea is to overcome problems of scale in imaging by finding ratios of distances. The ratio of ratios idea is to overcome problems of projection angle by finding ratios of ratios in some sense, the way this is realized being by means of the cross ratio. The cross ratio is a ratio of ratios, and using it gives a number that is independent of position and orientation of observation point. To prove that labelling the points in reverse order makes no difference, just make the substitutions and rearrange!

(c) Weak perspective projection is the special case of full perspective projection in the limit \( \Delta Z \ll Z \), so that the cross-ratio concept must still be valid. Under weak perspective projection, the full ratio of ratios concept is unnecessary, as parallel projection permits ratios of lengths on the same straight line to be identical on projection. This can be simply proved by similar triangles.

(d) The same concept will carry over to lengths measured on parallel lines, as the orientations of both lines are identical in 3D space, so again projection does not affect the ratios of the lengths. This means that the two parallel sides of a flat lino-cutter blade always appear in the same ratio. Thus it can be identified from any orientation in 3D by measuring the lengths of its sides.
19.7

(a) This is bookwork (see the main text in Chapter 19).

(b) When the scene is viewed by a camera, the spatial relations between the various feature points at various depths in the scene can be calculated from the angles they make with each other: this means that the different depths in the scene and the fact that the feature points being viewed do not originate on neat planes are irrelevant for any single image. As a result, measurements can be made using scene angles to obtain cross ratios. However, these angles intersect in the camera image plane in such a way that the cross ratios can instead be measured by the distances between features in the image plane. If now the camera is moved and rotated, the cross ratios will in general be different. But note that if the centre of projection of the camera is kept in the same position, the camera being rotated around it, the angles between the feature directions in the scene will remain unchanged. Hence the angular cross ratios will remain unchanged, and the distance cross ratios in the rotated image planes will be identical. Thus the condition for problem-free stitching is that the camera be rotated about the centre of projection. Nevertheless, there will be a fair amount of computation involved in forming the right correspondences between feature points and relating the various cross ratio values.
20.1

A full 3-D diagram can be quite complex and confusing, by the time the image plane is included as well as the ground plane and various construction lines. Instead, we here draw merely the lateral diagram corresponding to Fig. 20.4:

The pair of similar triangles immediately leads to:

\[
\frac{X}{Z} = \frac{x}{f}
\]

Recalling equation (20.2):

\[
Z = \frac{Hf}{y}
\]

we can eliminate \( Z \). Hence:

\[
X = \frac{Hx}{y}
\]

This proves equation (20.3).

Assumptions made: that the camera is optically perfect, that it is aligned parallel to the ground plane, and that the latter is perfectly flat. Perspective distortions are completely allowed for by the careful setup of the geometry.
20.2

The figure shows that equation (20.10) can be written in the following forms for respective vanishing points $V_1$, $V_2$ distant $d_1$, $d_2$ along the ellipse main axes directions:

$$\varepsilon_1 = \frac{b^2}{d_1}$$
$$\varepsilon_2 = \frac{a^2}{d_2}$$

We can now see that the transformed centre $T$ has been moved a distance $\varepsilon$ from the ellipse centre $C$, where:

$$\varepsilon = \left(\varepsilon_1^2 + \varepsilon_2^2\right)^{1/2}$$

If the ellipse major axis is inclined at an angle $\alpha$ to the image $x$-axis, the direction of $T$ relative to the image $x$-axis must be:

$$\xi = \alpha + \arctan(\varepsilon_1/\varepsilon_2)$$
21.1

From the main text in Chapter 16, the standard formulae for depth from disparity are:

\[ \frac{f}{Z} = \frac{x_1}{X + b/2} = \frac{x_2}{X - b/2} \]

\[ \therefore D = x_1 - x_2 = \left[ (X + b/2) - (X - b/2) \right] \times \frac{f}{Z} = \frac{bf}{Z} \]

\[ \therefore Z = \frac{bf}{D} \]

\[ \therefore \frac{dZ}{dD} = - \frac{bf}{D^2} = - \frac{Z}{D} \]

Hence:

\[ \frac{\delta Z}{Z} = - \frac{\delta D}{D} \]

so the numerical values of the fractional errors in \( Z \) and \( D \) are equal.

This means that the fractional error in depth is strongly limited by the pixel size, and further, that the fractional error is reduced in proportion to pixel size, or equivalently in inverse proportion to the size of \( D \) in pixels. To find the constant of proportionality, note that:

\[ D = f \times \left( \frac{b}{Z} \right) \]

This means that the size of \( D \) in pixels equals the size of \( f \) in pixels, scaled (downwards) by the factor \( b/Z \) which is dimensionless. \( b \) and \( Z \) can be regarded as world variables (which are both measured in metres).
21.2

The basic geometry appears in the figure on the left, and the details of the error $\delta$ appear in the figure on the right:

We now have:

$$\delta = a \sec \left( \frac{\alpha}{2} \right) - a \tan \left( \frac{\alpha}{2} \right) = \frac{b}{2Z}$$

Substituting for $\alpha$, we find:

$$\delta = a \left[ 1 + \left( \frac{b}{2Z} \right)^2 \right]^{1/2} - a \approx \frac{ab^2}{8Z^2}$$

Thus the error is proportional both to $a$ and to $Z^{-2}$, as stated in the main text of Chapter 21.

21.3

To reduce the ambiguities as far as possible, and to ensure accuracy of location of any feature, the aim should be to generate two epipolar lines that are as close to perpendicular as possible in at least one image space. This will uniquely define any feature. Once a feature is defined uniquely, no further loss of ambiguity can result from use of additional cameras. However, additional cameras may produce further corroborative data, and via averaging processes, help to reduce location errors even further.

The centre of projection of the third camera should not be in line with those of the other two cameras: otherwise the epipolar lines will be collinear and no reduction in ambiguity can result from employing a third camera. On the other hand, further cameras would yield slight gains in accuracy and range of 3-D depth perception.

(These conclusions make it somewhat curious that three-eyed animals don’t appear in nature.)
24.1

Equation (24.15) gives \( L(C_i|C_j) = \delta_{ij} \), the Kronecker delta. Substituting in equation (24.13) simplifies the summation hugely, as all but one of the terms is zero, leaving the result:

\[ R(C_i|x) = P(C_i|x) \]

Substituting into equation (24.14) now immediately yields the original probability-based decision rule, relation (24.5).

24.2

In the cross-over region (cf. Fig. 24.3b), the overlap region represents errors. Near the very centre of this region, where we might instigate rejection rather than incur a high error rate, the incidence of errors is approximately equal to the incidence of correct decisions. Thus rejection incurs approximately the same reduction in correct decisions as in erroneous decisions. Therefore reducing the errors by \( R \) in fact requires \( 2R \) test patterns to be rejected, as \( R \) correct patterns are also rejected. As \( R \) increases, the situation becomes worse, as more correct patterns than incorrect patterns are rejected. In the end, to reduce the number of incorrect patterns to zero, all the correct patterns have to be rejected: this is because the pattern distributions tend to have wings whose extent is unlimited.

24.3

The point nearest to the origin on the ROC curve is the position where the function

\[ P_n = (P_{FP}^2 + P_{FN}^2)^{1/2} \]

is a minimum. However, the probability of error is:

\[ P_E = P_{FP} + P_{FN} \]

This means that:

\[ P_{FP} = P_E - P_{FN} \]

Differentiating, we find:

\[ \delta P_{FP} = \delta P_E - \delta P_{FN} \]

For a minimum, \( \delta P_E = 0 \).

It is now clear that this occurs where the gradient of the ROC curve is equal to \(-1\).

While this result may seem trivial, it reflects an important truth about error analysis that applies for many pattern classification tasks.
A.1

(a) The breakdown point is the proportion of a distribution on one side or the other which can be disregarded by a data analysis method and hence the proportion of the distribution on either side which can contain outliers or false peaks, etc. It is related to robustness since robustness is the capability not to be confused by outliers, e.g. in the form of noise, clutter, breakages or occlusions. (It is worth considering whether this statement represents a complete definition of robustness.)

Accuracy is finally governed by the proportion of valid data points which are averaged to improve signal-to-noise ratio. If the proportion is cut down to improve robustness, accuracy is impaired. Thus there is a trade-off between accuracy and robustness.

(i) The mean has 0% breakdown point and 100% efficiency.
(ii) The median has 50% breakdown point and \((2/\pi) \times 100\% \approx 63.7\%\) efficiency.
(iii) The Hampel is variable but lies more or less between these two extremes, though it is able to totally ignore distant points. (A detailed solution to the problem would involve arguments based on the shape of the Hampel influence function.)

(b) It is possible straight away to write down that \((1 - \varepsilon)^2 = 0.5\), since the probability of a correct solution is the square of the probability of one point *not* being an outlier.

\[ 1 - \varepsilon = \sqrt{0.5} = \sqrt{(1/2)} = 0.7071 \]
\[ \therefore \varepsilon = 1 - 0.7071 = 0.2929 \]

Thus the breakdown point is 29.3%. This is definitely not the most robust approach. (However, it does not follow that the trade-off mentioned above makes this approach more accurate—in fact, this approach throws away robustness and gains nothing in accuracy, except via the outliers that are eliminated.)