Engineering Tripos Part IIB FOURTH YEAR

Module 4F9: Medical Imaging & 3D Computer Graphics

Solutions to 2005 Tripos Paper

1. **Iterative reconstruction techniques**

(a) q_i are noisy observations of the attenuated projections along projection line i. λ_j is the radioactivity in pixel j. c_{ij} is the sensitivity of projection i to radioactivity in pixel j. It accounts for arbitrary collimation and attenuation. $L(Q|\Lambda)$ is the log-likelihood of observing a set of measurements Q given activity Λ. For a PET ring with 20 detector modules, we may have $256 \times 256 = 65536$ pixels j, and ${}^{20}C_2 = 190$ projection lines i. [20%]

(b) Three ways to suppress noise in a ML-EM reconstruction.

(i) Smooth the reconstructed image.

(ii) Find the maximum a-posteriori (MAP) solution instead of the ML solution. Using Bayes's rule, the a-posteriori probability is given by

$$p(\Lambda|Q) = \frac{P(Q|\Lambda)p(\Lambda)}{p(Q)}$$

When maximising $p(\Lambda|Q)$, $p(Q)$ is constant and can be ignored. The trick is to come up with a prior probability $p(\Lambda)$ which favours smooth solutions.

(iii) Interrupt the EM algorithm before it has converged. ML-EM has the handy characteristic that low frequencies converge faster than higher ones. [20%]

(c) In X-ray computed tomography, the X-rays pass through the body and are attenuated by the tissue through which they pass. There are two main mechanisms for this attenuation: photoelectric absorption and Compton scattering.

Photoelectric absorption: The energy of the X-ray photon is completely absorbed as it ejects an electron from an inner shell. The excess energy of the photon over the binding energy of the electron is carried off as kinetic energy by the ejected electron. Lower energy characteristic radiation is emitted, in the same direction as the original photon, as an electron from an outer shell falls into the hole.

Compton scattering: The X-ray photon collides with a weakly bound, outer shell electron, which escapes from the atom with some kinetic energy. The remaining energy is carried away by a scattered X-ray photon. At diagnostic imaging energies ($\sim$100 keV), the scattering is fairly isotropic.

In both types of interaction, low energy radiation is emitted as the ejected electron loses it kinetic energy in collisions with other electrons. [20%]

(d) (i) In general, both algebraic reconstruction techniques use the following strategy.

(a) Start with an initial guess at the values of all the pixels. Perhaps set all pixel values to zero.

(b) For a particular projection, modify all the pixels that contribute to this projection in the *same* way, so they add up to the correct value.

(c) Repeat for all the other projections.

(d) Repeat the complete iteration until the pixel values converge.

In additive ART the average error is added to each pixel leading to an update formula as follows.

$$x_j^{(k+1)} = x_j^{(k)} + \frac{\lambda^{(k)}}{N}\left(y - \sum_{i=1}^{N} x_i^{(k)}\right)$$

k Iteration index.

$\mathbf{x}$ Vector of pixel values.

x_i A pixel in the $\mathbf{x}$ vector.

N The number of elements in the $\mathbf{x}$ vector.

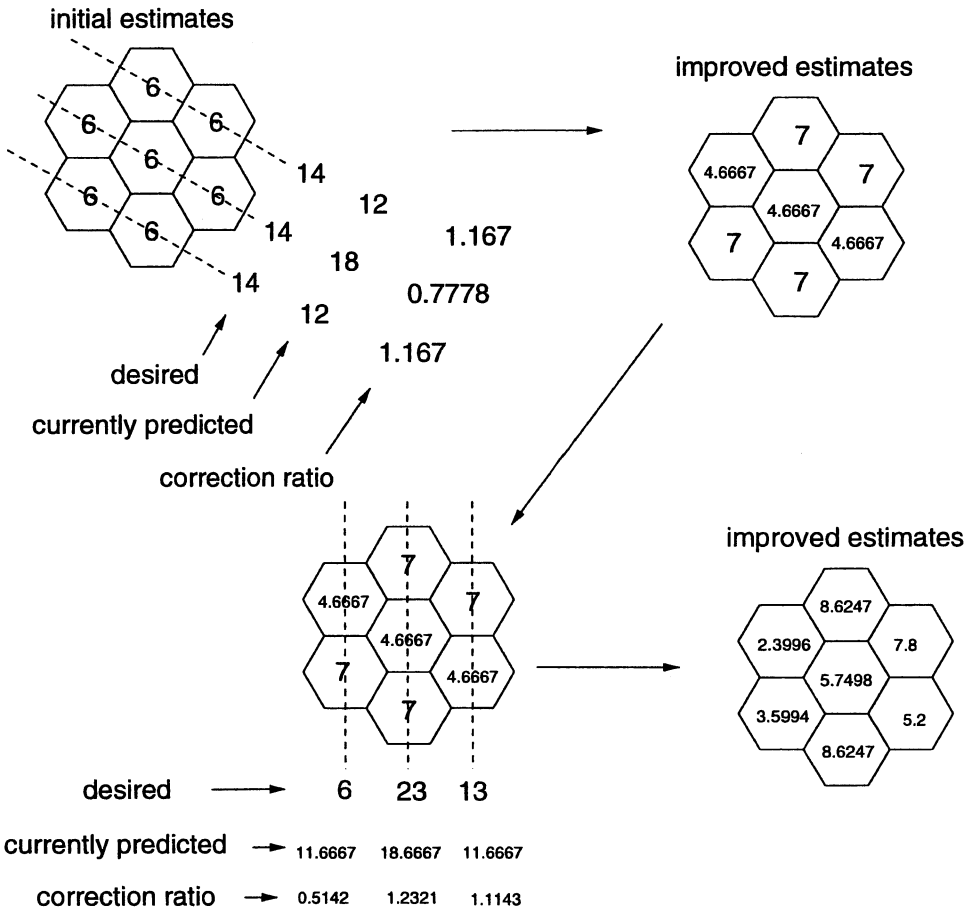
y Element of a projection corresponding to $\mathbf{x}$.

λ Relaxation factor.

In multiplicative ART all the pixels are modified by the same ratio, leading to an update formula for the form.

$$x_j^{(k+1)} = x_j^{(k)} \times \left(\frac{y}{\sum_{i=1}^{N} x_i^{(k)}}\right) \qquad [20\%]$$

(ii) The solution process is illustrated in the figure below. We start out by assuming the X-ray linear attenuation coefficients of all the rods are 6. We then calculate the X-ray transforms that would be observed if this were true. We find the ratios between these assumed values and the real measured values, and then correct each assumed value by the corresponding ratio. This gives us the improved estimates shown on the right. We repeat this procedure, using the other two sets of Q values, each time updating the assumed values to produce a further improved estimate.



[20%]

Assessors' remarks: This question tested candidates understanding of the computed tomography reconstruction techniques used in PET and SPECT systems. This was an unpopular question. Several of the candidates who chose to answer it produced sketchy or weak answers. This resulted in a low average mark for the question. Candidates showed good understanding of algebraic reconstruction techniques and many got the final numerical part of the question correct. The descriptive parts of the question about the expectation maximisation algorithm and the physics of Compton scattering and photoelectric were generally less well answered.

2. **Ultrasonic imaging**

(a) *Speckle* is an effect observed when the resolution cell of the ultrasound imaging system is large enough to contain a number of weak scatterers with random phase for which the central limit theorem holds. Thus it occurs when there are about 30 scatterers in the resolution cell. The result of this is that the sum of backscattered signal from the scatterers forms a 2D Gaussian in phase space, resulting in a Rayleigh amplitude distribution. The speckle effect is therefore multiplicative corruption of the signal with a fixed mean divided by standard deviation of 1.9. Speckle gives rise to diffuse variations of texture in the ultrasound image.

Specular reflection is reflection from a single strong scatterer or scattering structure. It gives rise to clear features in the ultrasound image. [20%]

(b) In most tissue, ultrasound attenuation is proportional to the frequency of the wave passing through it. The attenuation factor α_0 is quoted in dB/(cm MHz). Thus the 3.5 MHz probe will have better range (eg 14cm) than the 10 MHz one (eg 8cm).

Conversely, the resolution in the ultrasound image depends on the wavelength of the pulse. Thus the 10 MHz probe will have better resolution than the 3.5 MHz one. [20%]

(c) (i) Filtering: removes high frequency noise. (ii) Envelope detection removes the radio frequency component leaving just the envelope of the received signal. (iii) Time gain compensation attempts to correct for the attenuation. Backscattering from deeper in the body will have been subject to more attenuation and will therefore need to amplified more that signals from more superficial scattering. (iv) Log compression reduces the dynamic range of the signal so it can usefully be displayed as a brightness signal in a B-scan image. Note that the envelope detection and the log compression involve loss of information. [20%]

(d) Continuous wave Doppler measures the frequency shift of a continuous wave transmitted into the body. It is unable to determine the location of velocities within the beam. Pulsed wave Doppler measures velocity at a particular location. It does not use the Doppler effect, but fires a short sequence of pulses along the same line and measures the phase difference in the received signals. Pulsed wave Doppler is often used to construct colour flow images in which moving tissue is colour coded either red or blue. [20%]

(e) A small volume of fluid dV moves with variable velocity u under the influence of variable pressure p. The net force on the volume is given by

$$\begin{aligned} F &= \text{net pressure} \times \text{cross-sectional area} \\ &= \left[p - \left(p + \frac{\partial p}{\partial x}\,dx\right)\right] dy\,dz = -\frac{\partial p}{\partial x}\,dV \end{aligned}$$

Now apply Newton's second law:

$$\begin{aligned} F &= \text{rate of change of momentum of } dV \\ &= \frac{\partial}{\partial t}\left(dV\,\rho u\right) = dV\,\frac{\partial(\rho u)}{\partial t} \end{aligned}$$

Combining the two expressions for F:

$$-\frac{\partial p}{\partial x} = \frac{\partial(\rho u)}{\partial t}$$

This is the one-dimensional **inviscid force equation**.

To linearise the equation, we assume the sound wave induces a *small* disturbance of the static pressure, density and particle velocity fields:

$$\begin{aligned} p &= p_0 + \delta p, \text{ where } \delta p \ll p \\ \rho &= \rho_0 + \delta\rho, \text{ where } \delta\rho \ll \rho \\ u &= u_0 + \delta u, \text{ where } u_0 = 0 \end{aligned}$$

We can neglect products of the δ quantities. Thus

$$\rho u = (\rho_0 + \delta\rho)\delta u \approx \rho_0\, \delta u = \rho_0 u$$

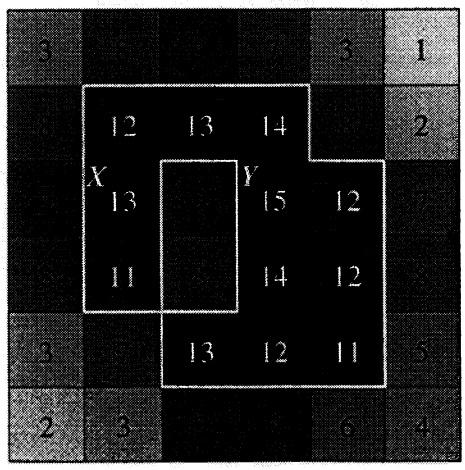
The linearised 1D inviscid force equation is then $-\frac{\partial p}{\partial x} = \rho_0 \frac{\partial u}{\partial t}$. [20%]

Assessors' remarks: This question tested candidates understanding of ultrasonic imaging. It was more popular than question 1 and most candidates who attempted it produced reasonable answers. The explanations of the strengths and weaknesses of probes with different centre frequencies were particularly good, and many candidates gave satisfactory descriptions of speckle, although few were able to describe the process by which it is formed. Surprisingly few candidates were able to remember the stages in the formation of a B-scan line from radio-frequency data. Some candidates also had a very muddled view of the two types of Doppler ultrasound. Generally a satisfactory question, with a pleasing number of good answers.

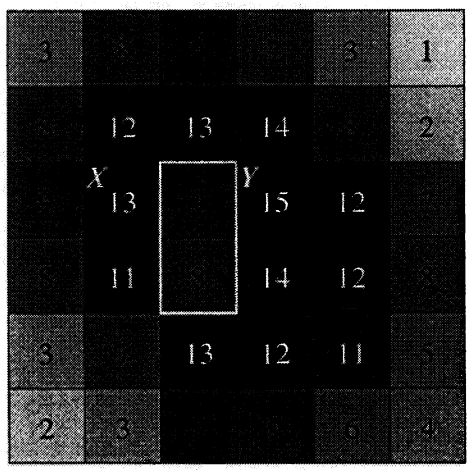
3. **Interpolation and contour extraction**

(a) (i) The results of the contour following process are affected by *choice of connectedness* and *boundary conditions*. A contour following algorithm can be 4-connected or 8-connected: this determines whether objects are allowed to connect diagonally across pixels. Where objects reach the border of the image data, they can either be connected along the border (i.e. we assume that outside the image is also outside the object), or left disconnected at the border. In the latter case, perimeter and area measurements are no longer possible, but surfaces can still be visualised. Alternatively, we can refuse to follow any contour which is not completely contained by the image, but this may potentially exclude useful information. [20%]

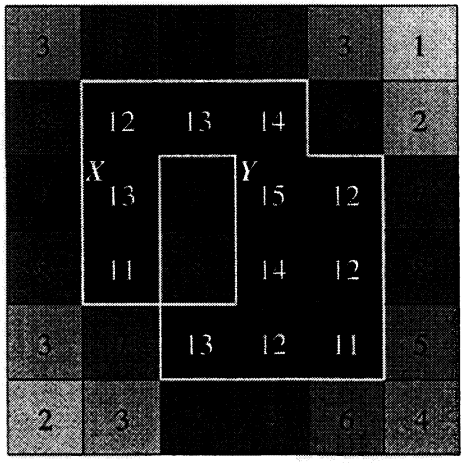
(ii)



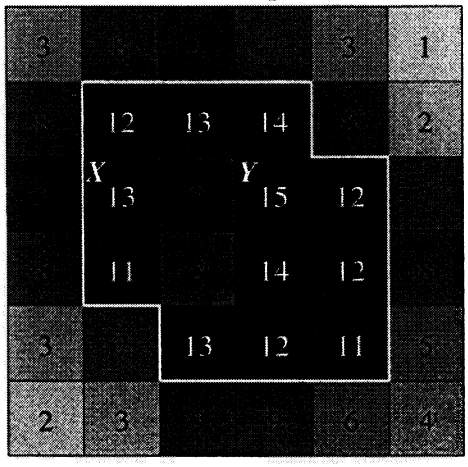
4–connected, starting at X.
Perimeter = 22 pixels



8–connected, starting at X.
Perimeter = 6 pixels



4–connected, starting at Y.
Perimeter = 22 pixels



8–connected, starting at Y.
Perimeter = 16 pixels

[30%]

Note that candidates were given credit for the more accurate perimeter estimates based on using diagonals at borders, in which case the 4-connected perimeters were $10 + 6\sqrt{2}$*, and the others* $2 + 2\sqrt{2}$ *and* $8 + 4\sqrt{2}$*.*

(iii) Clearly in this case the 4-connected region gives consistent (correct) results, whereas the 8-connected region does not. This is *not* really the issue though: the underlying problem is that the contour following algorithm should have been run twice, once initialised after searching right from the initial pixel, and once after searching left, until *all* bordering edges were found. In this case, all four options would generate a perimeter estimate of 22 pixels. [10%]

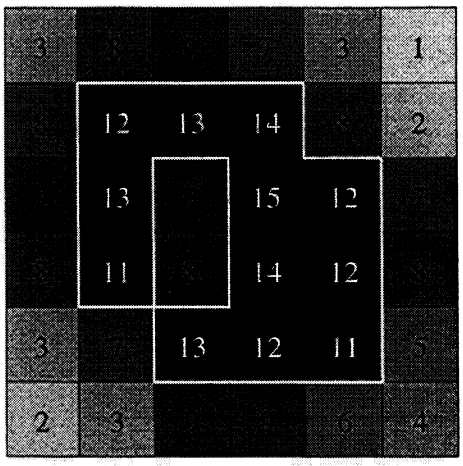
(b) (i) Nearest-neighbour interpolation sets each zoomed pixel to the value of the closest pixel in the original image. This is very simple and fast, but results in effectively zooming each original pixel to 100 times the size, without any smoothing,

i.e. not even C_0 continuous. However, it is the only algorithm to stay completely faithful to the original data.

Bi-linear interpolation sets the data value for each zoomed pixel to the bi-linear interpolant from the four nearest pixels in the original image. This is also fast and fairly simple to implement, and results in an image with some smoothing — the smoothing is however, only C_0 continuous.

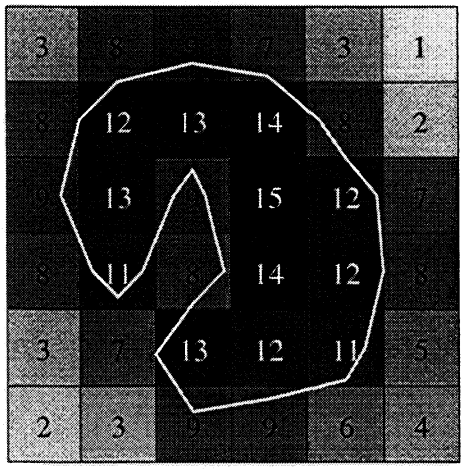
B-spline interpolation preserves C_2 continuity by using the 16 nearest original data values in a 4 x 4 grid to create new values for each new zoomed pixel. This will create a much more convincing (though slightly blurred) zoomed image, but is clearly more complex and time consuming. [20%]

(ii)



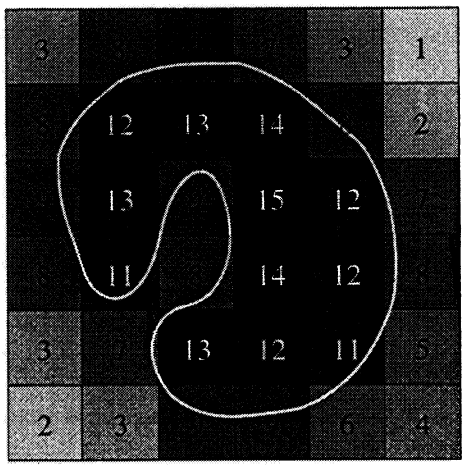
Contour after nearest–neighbour interpolation

Zooming the image using nearest-neighbour interpolation has no effect whatsoever on the form of the extracted contour.



Contour after bi–linear interpolation

Using bi-linear interpolation will generate a polyline as above.



Contour after b–spline interpolation

Using B-spline interpolation will generate a smoother contour as above.

Note that in all the diagrams above, the actual contour would actually have a stair-step appearance at the resolution of the zoomed pixels. [20%]

Assessors' remarks: This question was answered well. Candidates demonstrated a good working knowledge of the issues surrounding contour extraction and simple forms of interpolation. Several candidates had the answers to (a)(i) swapped around, however.

4. **Laser-stripe scanning**

(a) Sources of error include:

Surface properties Laser scanning relies on a single, clear, reflection of the laser off the surface. Surfaces which are highly specular, or translucent, or have fine features (e.g. hairs) will give multiple or distorted laser reflections. It is sometimes possible to make the surface more diffuse and simpler, for instance by wearing tight fitting clothes or dousing the surface in white powder.

Camera pixel accuracy The depth resolution is determined by the pixel size in the camera image — more pixels give better resolution. The resolution also reduces with distance from the camera and laser, so scanning surfaces as close as possible reduces this error.

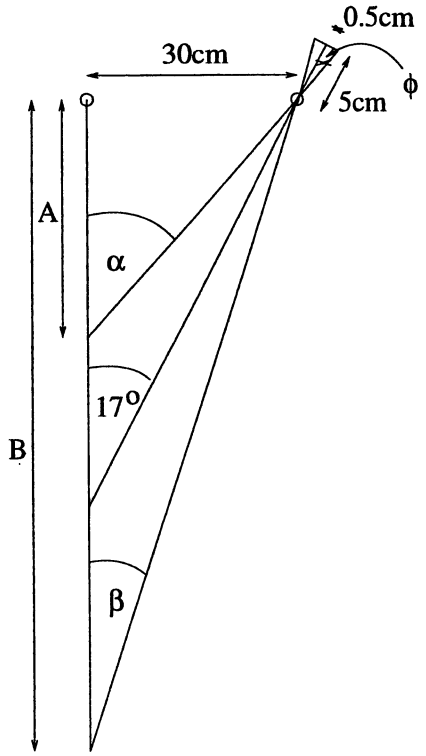
Object movement It generally takes several minutes to scan most objects and the object must remain stationary during that time. Obviously, the faster the scan, the lower this error is likely to be. It is also better to rotate the laser scanner rather than the object.

Obscured features In order to scan the surface, the laser light must reach it, and the camera see the reflection. Complex surfaces therefore often contain regions which cannot be scanned. It may be possible to chop up the object into several parts to get around this. Otherwise, missing surface parts have to be approximated in the reconstruction stage.

Laser thickness The laser stripe has a finite thickness. When triangulating, we look for the centre of the laser reflection in the camera image. If scanning sharp corners, only part of the laser stripe may be reflected, and the apparent centre will not be correct, resulting in depth errors. This is known as edge curl. Little can be done to improve this.

[20%]

(b) (i)

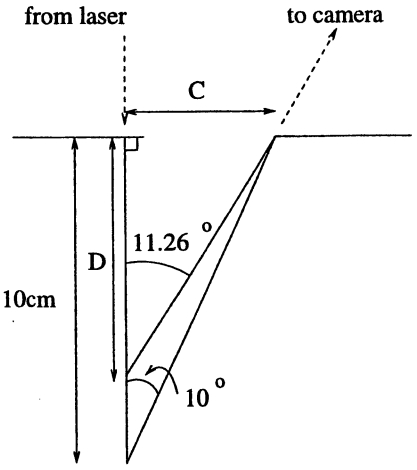


The depth range is determined by whether the laser reflection can be seen by the camera. Hence:

$$\begin{aligned}
\phi &= \tan^{-1}\frac{0.5}{5} \\
&= 5.7106^\circ \\
A &= \frac{30}{\tan\alpha} \\
&= \frac{30}{\tan(17+\phi)} \\
&= 71.68\text{cm} \\
B &= \frac{30}{\tan\beta} \\
&= \frac{30}{\tan(17-\phi)} \\
&= 150.28\text{cm}
\end{aligned}$$

[20%]

(ii)



The surface must be seen by both the laser and the camera. The minimum angle is at the maximum range from the scanner, 150.28cm, i.e. 11.29°. Hence:

$$\begin{aligned} C &= 10\tan 10 \\ &= 1.763\text{cm} \\ D &= \frac{C}{\tan 11.29} \\ &= 8.832\text{cm} \end{aligned}$$

[20%]

(iii) To get the maximum depth resolution, we need to scan the top of the surface at the closest possible range from the scanner, whereas the deepest part must be scanned at the farthest range. So we simply need the minimum and maximum depth resolution of the scanner. Following the same calculation as in part (i), but using a slightly reduced image size of 0.5cm $\times \frac{127}{128}$, gives a range from 71.84cm to 149.68cm. Hence the depth resolution varies from (71.68-71.84) = 0.16cm to (150.28-149.68) = 0.60cm, i.e. by a factor of 3.75 from the top of the hole to the deepest scan-able part.

[20%]

(c) Laser scanning generates a point cloud of surface data. If the laser stripe data is structured, then we can form triangle strips directly from these points, otherwise we must use Delaunay triangulation, or RBF interpolation to form the surface. We may also need to register surface patches from different sweeps of the laser scanner. It is likely that we will also need to use mesh decimation to reduce the number of polygons and improve the quality of the mesh, so that we can render it dynamically.

[20%]

Assessors' remarks: Surprisingly, candidates demonstrated a good knowledge of the basic operation of laser cameras in terms of the maths in part (b), but lost marks on the descriptive sections (a) and (c). Very few candidates mentioned the camera resolution as a source of error in (a), despite this being the focus of the remainder of the question. Few candidates were able to answer (b)(iii) correctly.

5. **Volume rendering**

(a) Volume rendering is the term used to describe the direct visualisation of a 3D voxel array without explicit segmentation into surfaces. Rays of light (usually parallel) are cast through the volume where they are shaded and attenuated by the individual voxels. The rays exiting the volume impinge on an image plane to create the rendering, in a manner analogous to taking an X-ray. The colour and opacity of the voxels are set using classification techniques, often heuristic in nature. The

volume may be re-oriented with respect to the image plane to construct views from different viewpoints, clipping planes can be used to look behind occluding objects, and the colours and opacities can be adjusted interactively to emphasise different aspects of the volumetric data. [20%]

(b) (i) α and c_λ ($\lambda \in \{r, g, b\}$) are, respectively the opacity and colour of a voxel. α^{in} and I_λ^{in} are, respectively, the accumulated opacity and intensity of the ray entering the voxel. α^{out} and I_λ^{out} are the same attributes when the ray leaves the voxel. The final value of I_λ^{out} determines the colour of the pixel in the volume rendering. [10%]

(ii) I_λ^{in} and α^{in} are both initially set to zero. A transparent voxel ($\alpha = 0$) has absolutely no influence on the rendering. A totally opaque voxel ($\alpha = 1$) stops any contribution from voxels further down the ray. [15%]

(iii) Looking at the equation for I_λ^{out}, we require

$$\alpha\left(1 - \alpha^{\text{in}}\right) = k \tag{1}$$

for some constant k. Substituting into the equation for α^{out}, we get

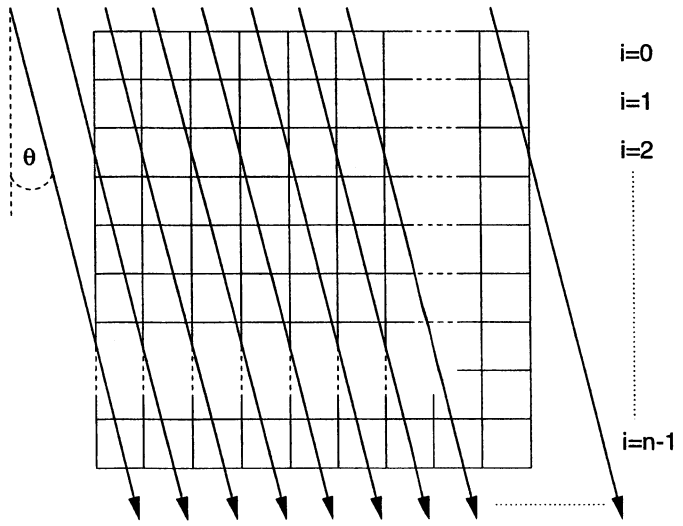
$$\alpha^{\text{out}} = \alpha^{\text{in}} + k$$

and it is therefore clear that $\alpha^{\text{in}} = ki$, where i is the index of the voxel along the ray. Substituting back into equation (1) we get

$$\alpha = \frac{k}{1 - ki}$$

This will work for any value of k in the range $0 < k \leq 1/n$. [25%]

(c) (i) Consider rotating the voxel array around an axis normal to the page, and casting a ray through the centre of each voxel in the top plane.



The intersection of the rays with the voxels is exactly the same as in Figure 6: the rays pass through the centre of every fourth voxel. Thus, shifting the voxel planes is equivalent to rotating the voxel array. If the rays make an angle θ with the vertical and pass through the centre of every $1/f$ voxels, then

$$\tan\theta = \frac{1}{1/f} \Leftrightarrow f = \tan\theta$$

Note that the perpendicular separation of the rays is reduced. Since each ray maps to a single pixel in the volume rendering, this affects the scale of the image. The greater the angle θ, the more the rendering will appear to be stretched. This effect can be corrected by a post-processing stage which shrinks the image appropriately. [20%]

(ii) This is a convenient way to generate volume renderings at oblique angles, though some sort of interpolation is required to determine α and c_λ at each voxel plane i. Nearest neighbour interpolation would result in significant rendering artifacts (until the ray crosses a voxel boundary, which happens at $i = 2$ in the example above, it would be as if the voxel planes hadn't shifted at all). Linear interpolation would be preferable. Higher order interpolation is unlikely to make much difference. [10%]

Assessors' remarks: This question covered familiar topics in volume rendering, before asking the candidates to analyse some unfamiliar extensions. The book work was almost universally well answered, as was the interpolation discussion in (c)(ii). About half the candidates managed the voxel shifting/rotating extension in (c)(i), but only one candidate derived a correct expression for α in (b)(iii).

6. **View volumes, 3D screen coordinates and z-buffers**

(a) The view volume, which for perspective projection is usually a truncated rectangular pyramid with its apex at the centre of projection, defines the visible portion of the scene. The clipping stage of the surface rendering pipeline clips all polygons to the view volume. The sides of the pyramid are clearly required to remove those polygons which are off the edge of the screen. One purpose of the near clipping plane is to remove polygons which are behind the viewpoint. The front and back clipping planes have other uses too:

- We can selectively cut out portions of the scene we are not interested in. We can even look behind objects by pushing the front clipping plane further back.
- The far clipping plane can be used to cut out distant objects which would be very small in the image: such objects often produce aliasing artifacts.
- Cutting out objects we are not interested in can drastically reduce the number of polygons to be rendered, increasing the speed of rendering. [20%]

(b) Under perspective projection, the point (x_v, y_v, z_v) projects to the view plane point $(x_p, y_p, -d)$, where

$$x_p = \frac{-dx_v}{z_v}, \quad y_p = \frac{-dy_v}{z_v}$$

If we define

$$x_s = \frac{x_p}{x_{\max}} = \frac{-dx_v}{x_{\max} z_v} \ , \quad y_s = \frac{y_p}{y_{\max}} = \frac{-dy_v}{y_{\max} z_v} \ , \quad z_s = \frac{f(1+n/z_v)}{f-n}$$

then points within the view volume satisfy the inequalities given in the question. [20%]

(c) (i) Substituting $x_v = y_v = 1$, $x_{\max} = y_{\max} = 1$, $n = d$ and $f = kd$ into the above expressions gives

$$x_s = \frac{-d}{z_v} \ , \quad y_s = \frac{-d}{z_v} \ , \quad z_s = \frac{k(1+d/z_v)}{k-1}$$

For large k, we can write

$$z_s = (1 + d/z_v)$$

The equation of the line of intersection of the planes is therefore

$$x_s = y_s \ , \quad z_s = 1 - x_s = 1 - y_s$$ [20%]

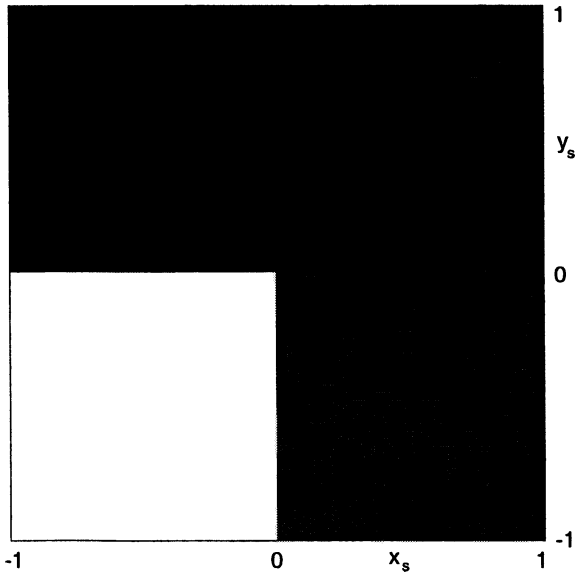
(ii) The line of intersection is only visible for

$$-1 \le x_s \le 1 \ , \quad -1 \le y_s \le 1 \ , \quad 0 \le z_s \le 1 \ .$$

The latter constraint can be written as

$$0 \le 1 - x_s \le 1 \ \Leftrightarrow \ 0 \le x_s \le 1$$

The appearance of the planes is therefore as sketched below.

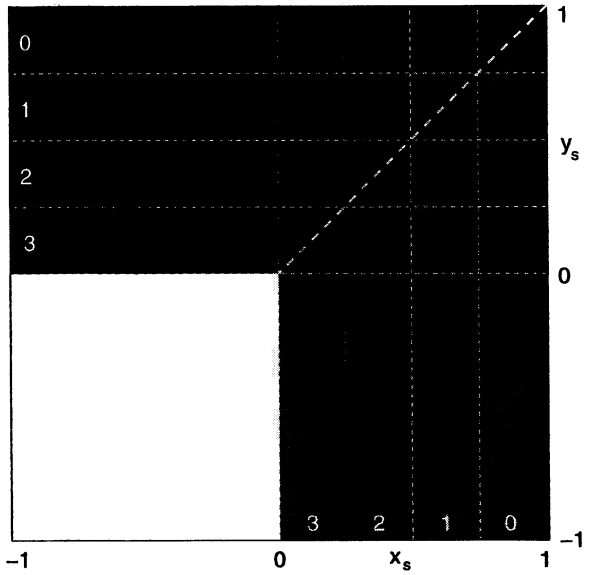


Note the vanishing point (the point where the planes meet on the horizon) at the centre of the image. [20%]

(iii) The 2-bit z-buffer discriminates between four ranges of z_s as follows:

range	value written in z-buffer
$0 \leq z_s < 0.25$	0
$0.25 \leq z_s < 0.5$	1
$0.5 \leq z_s < 0.75$	2
$0.75 \leq z_s < 1.0$	3

Since $z_s = 1 - x_s = 1 - y_s$ (ie. a linear relationship), the four z_s ranges map to equal-length segments of the line of intersection in 3D screen space (but not in view space). The sketch below shows the revised appearance of the two planes.



We have assumed that the $x_v = 1$ plane is rendered after the $y_v = 1$ plane, and that the z-buffer test is "render if less than or equal to". In the sketch, each plane has been divided into strips of constant z-buffer value in the range 0 ... 3. Portions of the $x_v = 1$ plane above the dotted line of intersection, although strictly behind the $y_v = 1$ plane, share the same z-buffer value and are therefore rendered. [20%]

Assessors' remarks: This question asked candidates to explain the purpose of a view volume and derive the relationship between 3D screen coordinates and view coordinates: all well answered. Remarkably few candidates, however, were able to calculate the line of intersection of the two infinite planes in 3D screen coordinates. Despite this, many used qualitative arguments to correctly sketch the planes in (c)(ii), and there was generally good understanding of the effects of limited z-buffer resolution in (c)(iii).

Andrew Gee, Richard Prager & Graham Treece
May 2005