Part IIA 2013

Module 3G4: Medical Imaging & 3D Computer Graphics

Numerical Answers

2. (d) (i) $\Delta x = c\Delta t/2$, (ii) 3.75 cm

5. (b) (iii) 37.5

   (c) 36.25

6. (b) (ii) $(0, 0, 0.75)$, (iii) 25%

Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2013 Tripos Paper

1. CT imaging and reconstruction

(a) (i) $\mu(x, y)$ is the linear attenuation coefficient at the point (x, y) in the imaging plane. $p_\phi(s)$ is the projection of $\mu(x, y)$ at angle ϕ. $q(s)$ is the reconstruction filter, theoretically a Ram-Lak filter which is the inverse Fourier transform of $|\omega|$. [10%]

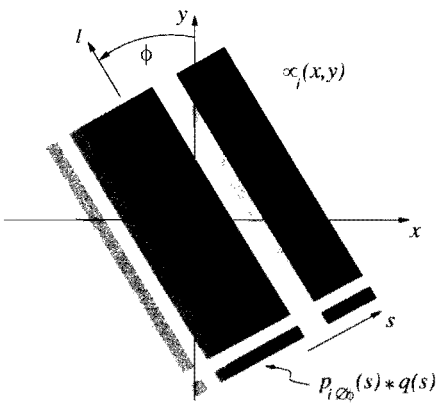
(ii) In a practical implementation, the integral is replaced by a finite summation:

$$\mu(x, y) = \sum_{i=0}^{n-1} [p_{i\Delta\phi}(s) * q(s)] = \sum_{i=0}^{n-1} \mu_i(x, y)$$

where

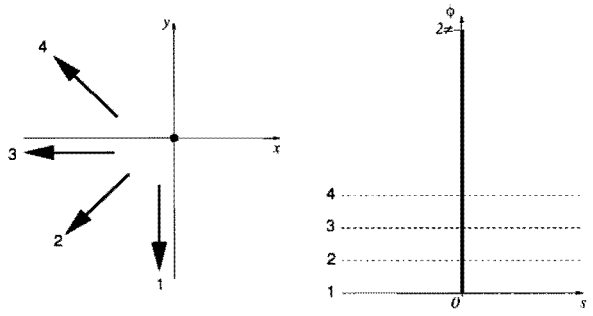
$$\mu_i(x, y) = p_{i\Delta\phi}(s) * q(s)\,, \;\; \Delta\phi = \pi/n$$

So the attenuation image $\mu(x, y)$ is reconstructed by summing a number of sub-images $\mu_i(x, y)$, each derived from a filtered projection $p_{i\Delta\phi}(s) * q(s)$. Notice how $\mu_i(x, y)$ varies only in the s direction (perpendicular to the X-rays) and not in the l direction (parallel to the X-rays). This is where the "backprojection" part of the name comes from.



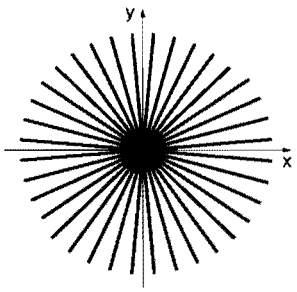
- Take n projections $p_\phi(s)$ of the object.
- Convolve each projection with $q(s)$.
- Backproject each filtered projection in the l direction.
- Accumulate the backprojections. [20%]

(b) (i) The point projects to $s = 0$ for every value of ϕ.



Its sinogram is therefore a straight line. [10%]

(ii) At each value of ϕ, we backproject a line through the origin.



The reconstruction therefore appears as an ensemble of bright spokes radiating from the origin. [10%]

(iii) In the limit of infinitely many projections, the reconstruction of the point will approach a $1/r$ distribution, where r is the distance from the origin (since the "spokes" pass through a circumference of $2\pi r$ at radius r, so the spoke density is proportional to $1/r$). Viewed as a linear system, the image of a point has turned out to be a circularly symmetric $1/r$ distribution, so this is the system's impulse response. Since the system is shift-invariant, it follows that its response to an arbitrary attenuation distribution will be the correct distribution convolved with a circularly symmetric kernel of magnitude $1/r$. [20%]

(c) Reconstruction (b) looks like the correct distribution convolved with a $1/r$ filter, so this must be unfiltered backprojection. Reconstruction (c) is a typical Ram-Lak filtered backprojection, with characteristic aliasing artefacts caused by sparse sampling. In reconstruction (d), the artefacts have been suppressed by applying a smoothing window (Hamming) to the Ram-Lak filter. In reconstruction (e), the resolution has clearly increased with no evident artefacts: this suggests narrower X-ray beams and more detectors. Reconstruction (f) exhibits classic cupping caused by beam hardening. [30%]

Assessors' remarks: This question examined candidates' understanding of CT imaging and reconstruction. Part (a) was book work and well answered by most candidates. In (b), most candidates deduced the correct sinogram for a point attenuator and also worked out

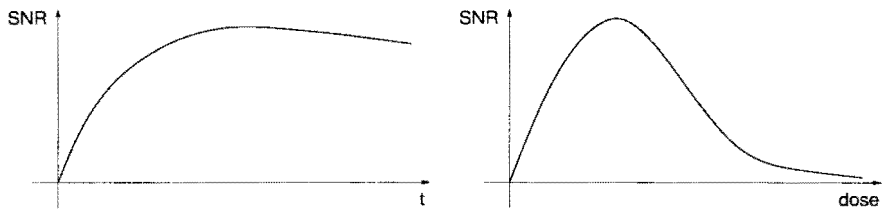
its unfiltered backprojection, though relating this to a $1/r$ convolution proved surprisingly difficult. In (c), there were a couple of excellent answers, though most candidates had clearly not experimented with the virtual CT simulator (made available to all students taking the course) which was used to produce Fig. 1.

2. **Nuclear medicine imaging**

(a) SPECT stands for single photon emission computed tomography. PET stands for positron emission tomography. SPECT has the advantage of relatively cheap scanners and ready availability of suitable radiopharmaceuticals without the need for an on-site cyclotron. PET scanners are orders of magnitude more sensitive than SPECT scanners, since they use electronic rather than mechanical collimation: this allows reduced patient doses, shorter acquisition times and better signal to noise ratios. However, the scanners are expensive and the radionuclides short-lived, requiring an on-site cyclotron production facility. [20%]

(b) While filtered backprojection is the standard for CT image reconstruction, SPECT/PET reconstruction differs in its requirements for (i) attenuation correction and (ii) Poisson noise suppression. Iterative reconstruction algorithms allow both of these issues to be addressed: attenuation factors contribute to the c_{ij} sensitivity coefficients, while a proper Poisson model inside a ML-EM (with early stopping) or MAP formulation deals with the Poission noise. In contrast, filtered backprojection takes no account of Poisson noise, and attenuation correction pre-processing is only feasible in the case of PET. [20%]

(c)



(i) At low measurement time, Poisson noise will dominate and the SNR will follow that of the Poisson distribution $\sqrt{t}$. At longer measurement time, Poisson noise will no longer be the dominant factor and other factors (electronic noise, patient motion etc.) will limit the SNR. In fact, patient motion is likely to reduce the SNR at large t.

(ii) With low radionuclide dose, we have high Poisson noise and therefore low SNR. The SNR will improve with higher dose, but then turn a sharp corner when the counts become so large that photon detection starts to fail due to near-coincident arrival times from separate decay events (the probability of successfully detecting a photon decreases exponentially with increased radioactivity). [20%]

(d) (i) Assume the two photons are ejected from the location of the emitter (i.e. the positron doesn't travel far) and then travel at the speed of light to the detectors with no scattering.

Let the distance travelled by the first photon be x_1 and the distance travelled by the second photon be x_2. The difference between the two detection times is

$$t = \frac{x_1 - x_2}{c}$$

Now imagine an uncertainty Δx in the emitter's location, increasing x_1 and decreasing x_2. The difference between the two detection times is now

$$t + \Delta t = \frac{(x_1 + \Delta x) - (x_2 - \Delta x)}{c} = \frac{x_1 - x_2}{c} + \frac{2\Delta x}{c} = t + \frac{2\Delta x}{c}$$

Hence $\Delta x = c\Delta t/2$. [20%]

(ii) Substituting $\Delta t = 250\,\text{ps}$, we find $\Delta x = 3.75\,\text{cm}$. So the time-of-flight calculation can locate the emitter to within a range of $\pm 3.75 = 7.5\,\text{cm}$. On its own, this is not going to be sufficient to form a useful image, so we will still need to gather data along multiple projections to localise the event more precisely. However, for each projection we can now say "I know the event happened somewhere in this 7.5 cm span", which is better than "I know the event happened somewhere along this line joining the two detectors." This extra information can be exploited in the reconstruction algorithm to improve the image SNR. [20%]

Assessors' remarks: This question was about nuclear medicine imaging. Parts (a)–(c) were book work covering standard PET/SPECT imaging and reconstruction issues. They were well answered by most candidates, though several had clearly not understood the first thing about these imaging modalities, despite similar tripos questions being asked in the past and cribs being readily available. Part (d) was a straightforward but unfamiliar quantitative analysis of time-of-flight PET. Most candidates made a decent attempt at relating Δt to Δx, though the factor of two was sometimes missing and sometimes on the wrong side of the equation. While almost all candidates commented that Δx was too large for x to be used on its own, only a handful realised that $x \pm \Delta x$ could nevertheless improve the accuracy of conventional reconstruction algorithms.

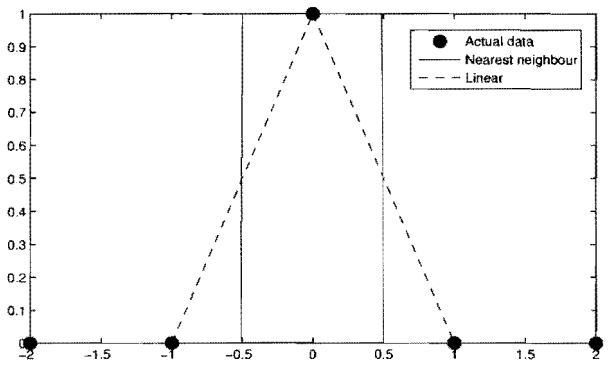
3. **Interpolation and approximation of scalar data**

(a) Both interpolation and approximation are techniques for creating a continuous function which is used to represent discrete (sampled) data. Such a function can then be used for visualisation or other purposes. This is called *interpolation* if the function passes through the sampled data, and *approximation* if the function merely comes close to the sampled data. Interpolants have the advantage of exactly representing the recorded data at the sample locations, which means they are more faithful to the data at these points. However, away from the sampled points they do not necessarily represent the most plausible underlying data variation - this depends on the technique. Also, if the data is noisy, an interpolant can amplify this noise in-between the samples.

An *approximation* will not exactly represent the data but it can more easily be made to obey certain characteristics - for instance smoothness - away from these sample locations. This is a better thing to do if the medical data it represents is also known to be smooth in

reality. An approximating function can also be made to smooth over noise in the data - so long as the amount of noise is known. [20%]

(b) The two interpolants are shown below, along with the original data. Both should pass through the original data points. The change in the nearest neighbour interpolant is exactly at the -0.5 and 0.5 points.



[10%]

(c) (i) We need to look at the combination of the parameter matrix $[t^3\ t^2\ t\ 1]$ and $\mathbf{M}$ when $t = 0$ (the beginning of each segment) and $t = 1$ (the end of each segment). For the former we want this to return $[0\ 1\ 0\ 0]$ so that $S_x(0) = F(x)$; for the latter we want $[0\ 0\ 1\ 0]$ so that $S_x(1) = F(x+1)$. Hence at $t = 0$:

$$S_x(0) = [0\ \ 0\ \ 0\ \ 1]\,\mathbf{M} \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

and

$$\mathbf{M} = a \begin{bmatrix} -1 & -1 & 1 & 1 \\ 2 & 1 & -2 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{b}{6} \begin{bmatrix} -1 & -9 & 9 & 1 \\ 3 & 12 & -15 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & -2 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 & -2 & 0 \\ 0 & -3 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

hence

$$S_x(0) = \left(\frac{b}{6} [1\ \ -2\ \ 1\ \ 0] + [0\ \ 1\ \ 0\ \ 0] \right) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

which will return $F(x)$ if $\frac{b}{6} = 0$ and $1 - \frac{2b}{6} = 1$, both of which lead to the condition that $b = 0$ for interpolation.

Now checking interpolation at $t = 1$:

$$S_x(1) = [1 \;\; 1 \;\; 1 \;\; 1]\,\mathbf{M} \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

hence

$$S_x(1) = \left(\frac{b}{6}[0 \;\; 1 \;\; -2 \;\; 1] + [0 \;\; 0 \;\; 1 \;\; 0]\right) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

from which it can be seen that $b = 0$ will return $F(x+1)$ as required. [15%]

(ii) The gradient is found by taking the derivative of the parameter matrix. Hence at $t = 0$:

$$S'_x(0) = [0 \;\; 0 \;\; 1 \;\; 0]\,\mathbf{M} \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

hence

$$S'_x(0) = \left(a[-1 \;\; 0 \;\; 1 \;\; 0] + \frac{b}{2}[-1 \;\; 0 \;\; 1 \;\; 0]\right) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix} = \left(a + \frac{b}{2}\right)(F(x+1) - F(x-1))$$

For the other end $t = 1$:

$$S'_x(1) = [3 \;\; 2 \;\; 1 \;\; 0]\,\mathbf{M} \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

hence

$$S'_x(1) = \left(a[0 \;\; -1 \;\; 0 \;\; 1] + \frac{b}{2}[0 \;\; -1 \;\; 0 \;\; 1]\right) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix} = \left(a + \frac{b}{2}\right)(F(x+2) - F(x))$$

The gradient is hence always a difference of the surrounding points and S will have C1 continuity independent of a and b. [15%]

(iii) Continuing with the approach from (ii) we find that, for the second derivative:

$$S''_x(0) = (a[4\ \ 2\ \ -4\ \ -2] + b[1\ \ 4\ \ -5\ \ 0] + [0\ \ -6\ \ 6\ \ 0]) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

and

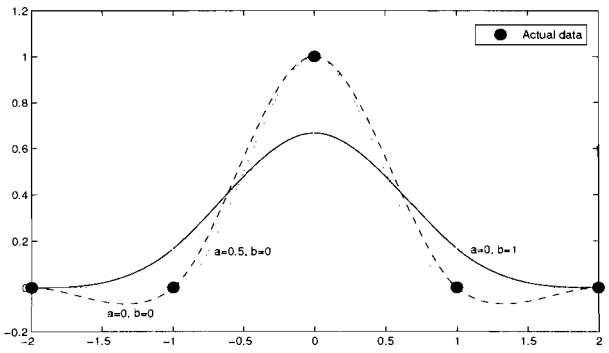
$$S''_x(1) = (a[-2\ \ -4\ \ 2\ \ 4] + b[0\ \ -5\ \ 4\ \ 1] + [0\ \ 6\ \ -6\ \ 0]) \begin{bmatrix} F(x-1) \\ F(x) \\ F(x+1) \\ F(x+2) \end{bmatrix}$$

For C2 continuity, the second derivative can only be a property of the immediately surrounding points: so $F(x+2)$ must not be involved in the former equation, nor $F(x-1)$ in the latter. Hence $a = 0$. We also require the same weighting of the surrounding points, i.e.:

$$\begin{aligned} b &= -5b + 6 \\ 4b - 6 &= 4b - 6 \\ -5b + 6 &= b \end{aligned}$$

These are satisfied when $b = 1$. Hence $a = 0, b = 1$ produces a C2 continuous curve S. [20%]

(iv) A sketch is shown below. $a = 0, b = 0$ is C2 continuous and does not interpolate the points (this is actually a B-spline), $a = \frac{1}{2}, b = 0$ does interpolate the points, and the gradient at each location is just the difference of the surrounding points (this is actually a Catmull-Rom spline). $a = 0, b = 0$ interpolates the points and has a zero gradient at each point.



[20%]

Assessors' remarks: This was a very popular question which tested candidates' knowledge of data interpolation and approximation. Parts (a) and (b) were mostly answered well,

though some candidates failed to centre the nearest-neighbour interpolant on $x = 0$. Part (c) was more patchy, though there were some near-perfect answers. Many candidates spent a lot of time on unnecessary mathematical working, for instance evaluating the whole of **M** before thinking through what the pre- and post-multiplying vectors were.

4. **Extracting surfaces and surface normals from 3D data**

(a) Real surfaces within 3D medical data may either be marked by the transition from one set of data values to another, as in computed tomography, or they may be marked by locally high data values, as in ultrasound. In the former case, a surface can be directly picked out by selecting a threshold (for instance between bone and soft tissue) then applying an algorithm like Marching Cubes to extract a polygonal representation of the surface. In the latter case, the data will need to be processed first, before it can be thresholded. In any case, medical data is almost certain to have enough noise in it to warrant a smoothing step before any attempt at extracting a surface.

Medical data is usually complex, and many polygons (usually triangles) will be needed to represent a typical surface within a data set - of the order of a million. Although this could be reduced by post-processing, with the power of modern graphics cards, we usually stick to a polygon representation for this sort of data. The size of polygon also needs to be considered - a surface with much more detail will be generated if the polygons are smaller.

If the surface is intended for visualisation, it is generally necessary to post-process in order to improve the aspect ratio of the triangles in the surface, since this makes later rendering stages work better. Surfaces which are not completely contained in the data set can also be an issue: we have to decide what to do at the point that the surface reaches the edge of the data. If we force it to be closed by considering everything outside the data set to also be outside the surface, we can go on to measure volume, but it might not be correct. If we only display surfaces actually within the data, they will not in general be closed (watertight) and volume measurement is not then possible. [30%]

(b) (i) Each vertex will be connected to several different triangles. The surface normal to one triangle can be calculated by taking the cross-product of vectors along two of the edges. If the triangles vertices are stored consistently, this cross-product can be chosen to always create an outward pointing surface normal. A normal at each *vertex* can then be formed by taking the average of each of the normals of triangles containing that vertex. (More complicated techniques are also possible which take into account the angle which the triangle sub-tends at the vertex.) [10%]

(ii) To calculate the normal, we need two vectors **a** and **b** lying on the surface: once we have these we can take the cross-product to get the surface normal. The easiest way to get these two vectors is to take the derivative in each of the s and t directions. Hence if:

$$\mathbf{a}(s,t) = [a_x(s,t) \;\; a_y(s,t) \;\; a_z(s,t)]$$

$$\mathbf{b}(s,t) = [b_x(s,t) \;\; b_y(s,t) \;\; b_z(s,t)]$$

then:

$$a_x(s,t) = [3s^2 \;\; 2s \;\; 1 \;\; 0]\mathbf{M}Q_x\mathbf{M}^T[t^3 \;\; t^2 \;\; t \;\; 1]^T$$
$$b_x(s,t) = [s^3 \;\; s^2 \;\; s \;\; 1]\mathbf{M}Q_x\mathbf{M}^T[3t^2 \;\; 2t \;\; 1 \;\; 0]^T$$

and similarly for the $y-$ and $z-$ coordinates. The surface normal $\mathbf{n}$ is then given by:

$$\mathbf{n}(s,t) = (\mathbf{a}(s,t) \widehat{\times} \mathbf{b}(s,t)) \qquad [20\%]$$

(iii) Q_x is a 4×4 matrix which contains the x-coordinates of a 4×4 grid of points defining the surface; Q_y and Q_z contain the other coordinates of these points. The patch P will be defined roughly between the middle four of these points, though the B-spline does not in general pass through any of the points. These points need to be close to the actual surface - so for instance we could define the grid of points by taking a 4×4 grid of parallel lines, intersecting these with the data set, and defining points where these lines cross the iso-surface. This would only define one surface patch: defining a complete surface would require the joining of many patches which is somewhat more complex. [10%]

(iv) Given that we are extracting an iso-surface from a 3D data set, the normals to this surface will be at the direction of maximum gradient in the data. If the data values are given by d, then the data gradient is given by ∇d. Hence if we evaluate $\widehat{\nabla d}$ at the isosurface locations, this will give a vector pointing along the surface normal. [10%]

(c) The cubic and data-based methods produce a continuously varying surface normal, whereas the polygonal method only generates discrete normals at vertices (though these can be interpolated across the triangles for visualisation).

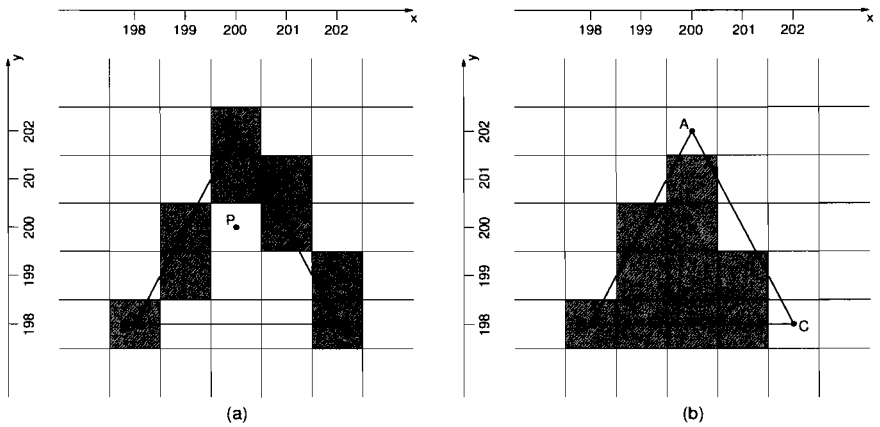
The smoothest normals are likely to result from the B-spline data, since this is a smooth approximating function, but this depends on how well the parametric surface has been extracted from the data set. Hence this method will be the least sensitive to noise. The data-based method will be the most noise-sensitive, since there is no explicit smoothing, and taking the derivative will amplify the noise in the data. The polygonal method will have some robustness to noise, mainly depending on the size of the polygons, since the polygon normals are averaged to create the normals at the vertices. [20%]

Assessors' remarks: This was an unpopular question, though the one candidate who answered it did reasonably well. Given the spread of marks on Question 3, it seems likely that many candidates would have done better to attempt this less mathematical question. The question examined techniques for extracting iso-surface and iso-surface normals from discrete 3D data.

5. **Rasterisation, shading and anti-aliasing**

(a) *Rasterisation* refers to the process of displaying graphics primitives (lines, polygons etc.) on discrete, pixelised displays. Rasterisation algorithms generate a list of pixels which are shaded in order to display the primitive. More advanced rasterisation algorithms incorporate anti-aliasing features. Rasterisation is also sometimes referred to as *scan conversion.* [10%]

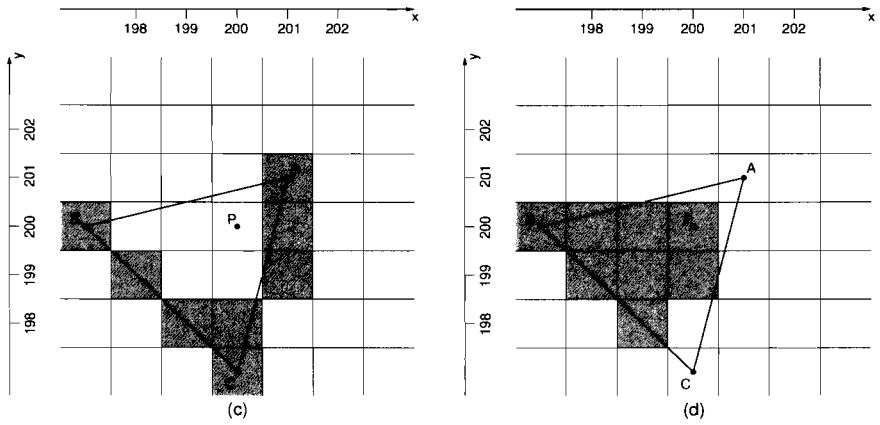
(b) (i, ii) It is important to remember that, to prevent enlarged areas, edges are shaded only up to the y-value immediately before the end vertex, and scan lines are shaded up to one pixel before the right hand edge. [20%]



(iii) Gouraud shading uses bilinear interpolation of the intensities at A, B and C to calculate the intensity at P. In this case, the scan line containing P is exactly half way along the edges BA and CA, and P is half way along this scan line, so

$$I_P = 0.5(0.5I_B + 0.5I_A) + 0.5(0.5I_C + 0.5I_A) = 0.25(I_B + I_C) + 0.5I_A = 37.5 \qquad [10\%]$$

(c) After rotation of the viewpoint and rasterisation, the new vertex, edge and shaded pixel positions are:



The vertex intensities will be the same as before, since nothing that affects the Phong equation has changed: all that has happened is a twist of the viewpoint around the principal

viewing axis. Now, the scan line containing P is 0% of the way along the edge BA and 75% of the way along edge CA, and P is 75% of the way along this scan line, so

$$I_P = 0.25(1.0I_B+0.0I_A)+0.75(0.25I_C+0.75I_A) = 0.25I_B+0.1875I_C+0.5625I_A = 36.25$$ [30%]

(d) The scene should look identical in (b) and (c), apart from a 45° rotation. And yet the area of the triangle has changed from 10 pixels to 8 pixels, and the intensity of the pixel at the centre of the image, which represents exactly the same world location in (b) and (c), has changed from 37.5 to 36.25. Switching to Phong shading is not going to fix the intensity anomaly: the problem is that the relative locations of A, B, C and P have changed, and this will affect the interpolation of normals just as it affects the interpolation of intensities. Instead, what we have here are aliasing artefacts.

Aliasing artefacts arise because images are rendered into a discrete sampling array in space (and, in the case of animations, in time). The most commonplace aliasing artefact is the ubiquitous jagged edge of a polygon or shadow. "Jaggies" are particularly noticeable in animated sequences, where edges appear to "crawl", and small or thin objects may appear and disappear depending on their orientation ("scintillate"). In (b) and (c), we saw a less severe example of scintillation, where the area of the triangle changed depending on its orientation.

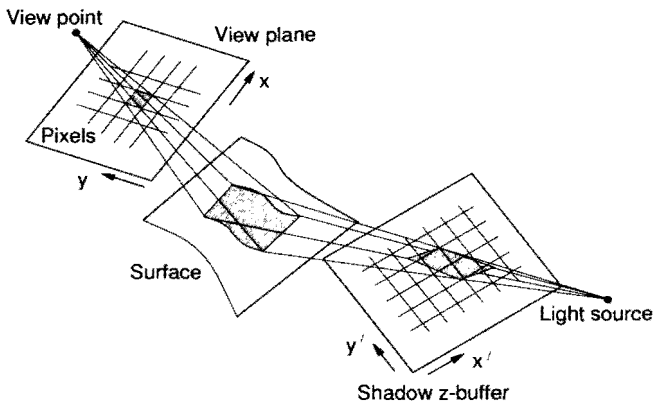
More advanced rasterisation techniques go some way towards suppressing aliasing artefacts. "Supersampling", or "postfiltering", involves rendering the image into an intermediate frame buffer at n times the display resolution. The supersampled image is then low-pass filtered at the Nyquist limit of the final display frame buffer, which is filled by subsampling the intermediate frame buffer. "Prefiltering", or "area sampling", involves rasterising each polygon's edges to sub-pixel accuracy, so that each pixel can be classified as, for example, 30% polygon A, 20% polygon B and 50% polygon C. Then the intensity of the pixel is set to a weighted average of the individual polygon's contributions. Both techniques are computationally expensive, though hardware postfiltering is now commonplace in commodity PC graphics cards. [30%]

Assessors' remarks: This question was about rasterization. The candidates were asked to work through some straightforward examples to demonstrate how this stage of the rendering pipeline might introduce geometrical and intensity errors into animated sequences. The candidates were then invited to speculate how these errors might be corrected by anti-aliasing. There were some excellent answers, but also a fair number of minimal attempts suggesting a last minute token effort: this affected the average mark. It was disappointing how many candidates resorted to lengthy, matrix multiplication calculations to rotate the vertices of a triangle through 45°, when this could be done in a few seconds by inspection.

6. **The shadow z-buffer algorithm**

(a) The shadow z-buffer algorithm is a two-stage process. The scene is first "rendered" using the light source as the viewpoint. No intensities are calculated, but a "shadow" z-buffer is filled in the usual way. At the end of this stage, the shadow z-buffer contains the distance from the light source to the *nearest* polygon along each ray.

The second stage involves rendering the scene from the viewpoint. A standard z-buffer algorithm is used, with one extra step. If a point is visible, its 3D screen coordinates (x_s, y_s, z_s), as seen from the viewpoint, are transformed to 3D screen coordinates (x'_s, y'_s, z'_s), as seen from the light source. The (x'_s, y'_s) are converted to 2D device coordinates and used to index the shadow z-buffer. The retrieved value z_b is compared with z'_s. If $z'_s > z_b$, there must be a surface between the point and the light source, and the point is rendered with a reduced illumination. Otherwise the point is rendered as normal.



A single pixel in the view plane will not map onto a single pixel in the shadow z-buffer. This means that a pixel may be partly in shadow and partly not. If we use only one z_b to decide whether the pixel is in shadow, the rendering will suffer from jagged shadow boundaries. To avoid this, we calculate (x'_s, y'_s) for each of the four corners of the pixel. This defines a quadrilateral in the shadow z-buffer. If we now look at each z_b value covered by the quadrilateral, we can deduce the fraction of the pixel in shadow. We then use this fraction to calculate a suitable attenuation factor. The hard edge of the shadow will be softened for the pixels partly in shadow, and the aliasing artifacts suppressed. [30%]

(b) (i) The viewpoint transformation is clearly linear ($w = 1$) and is therefore an orthographic projection. The diagonal nature of the matrix reveals that there is no rigid body transformation between the local and view coordinate systems. The light source transformation is nonlinear (w depends on x_l) and is therefore a perspective transformation, with projection rays emanating from a single point: we are modelling a *point* light source. There is a rigid body transformation between the local and light coordinate systems. [15%]

(ii) The camera's transformation matrix is trivial to invert by inspection, so

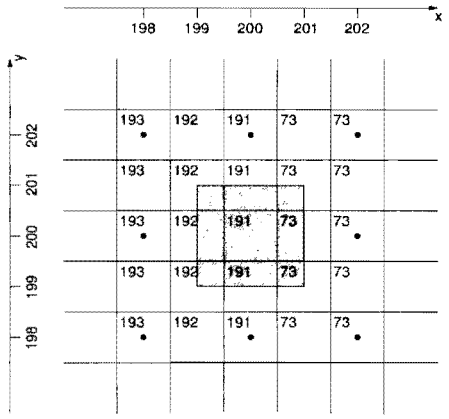
$$\begin{bmatrix} wx'_s \\ wy'_s \\ wz'_s \\ w \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -50 \\ 0 & 1 & 0 & 0 \\ -1.25 & 0 & 0 & 75 \\ -1 & 0 & 0 & 100 \end{bmatrix} \begin{bmatrix} 200 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 \\ 0 & 0 & -100 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 75 \\ 100 \end{bmatrix}$$

Therefore $(x'_s, y'_s, z'_s) = (0, 0, 0.75)$. [15%]

(iii) First convert (x'_s, y'_s) to 2D device coordinates. The 400×400 shadow z-buffer spans the range $-1 < x'_s < 1$ and $-1 < y'_s < 1$, so

$$x = 400 \times (0 - (-1))/2 = 200$$

and likewise for the y coordinate. Looking at $(200, 200)$ in the shadow z-buffer we find the value 191, which is 0.75×255, i.e. point A. So there is nothing between point A and the light source. But this is just the centre of the pixel, and we should check its corners which, with the aid of the dots, we know map roughly here:



Given that the scene comprises several smooth, continuous surfaces, we can deduce that the z-buffer values in the range 191–193 belong to point A's surface, but the sudden jump to 73 must be some other surface closer to the light source. The pixel is right on the shadow's edge, 1/4 in shadow to be precise, and we should therefore attenuate its intensity by 25%. [20%]

(iv) The negative z'_s coordinate would place point A closer to the light source than the light source's near clipping plane. Clearly, if any polygon is clipped it is not going to contribute to the shadow z-buffer and not going to cast any shadows. We need to make sure that the near clipping plane does not result in any potential shadow-casters getting clipped. In this case, we have at least one surface (the one containing point A) that is clipped, so the shadow z-buffer algorithm is not going to detect all shadows correctly. [20%]

Assessors' remarks: This question was about the shadow z-buffer algorithm. It started with 30% book work, where students were asked to simply describe how the algorithm works. The solutions fell into two categories. "Serious" solutions were generally excellent, the only shortcoming being a failure to mention aliasing. However, the average mark was dragged down by a handful of candidates who could not appreciate that the question was about shadows and instead described the standard z-buffer algorithm for hidden surface removal: very disappointing. The rest of the question asked the candidates to work through a quantitative case study with numbers contrived to be straightforward. Again, the competent candidates did this well, with just the occasional algebraic slip.

Andrew Gee & Graham Treece, May 2013