Engineering Tripos Part IIA THIRD YEAR

Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2010 Tripos Paper

1. **Additive Algebraic Reconstruction Technique**

(a) AART is an iterative reconstruction technique for computed tomography. The slice to be calculated is divided into pixels and these are assigned initial values, perhaps all zero.

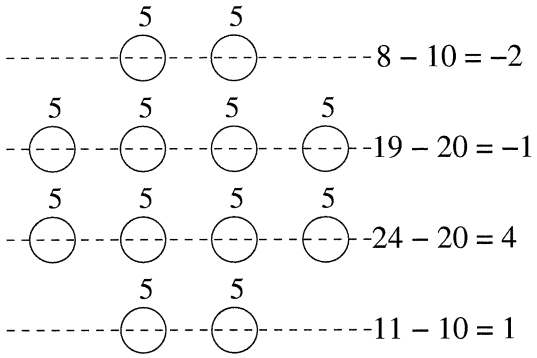
For each point in each projection, look at the set of pixel values that contribute to the point and calculate the error between the sum of the relevant pixels and the correct projection value. Divide this error by the number of pixels involved and multiply by the relaxation factor (usually < 1) to get a correction value. Subtract this correction value from each of the pixels contributing to the projection value under consideration.

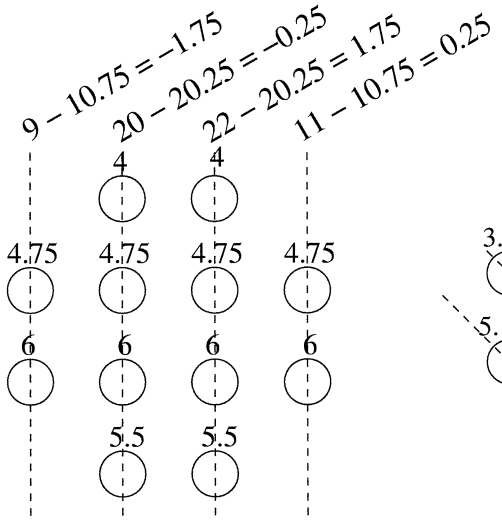
Iterate through the projections, gradually updating the values of the pixels. Given sufficient appropriate projections, the pixel values will converge to the correct solution:

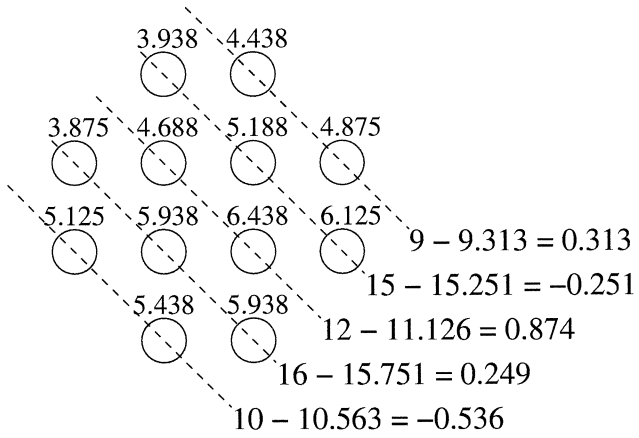
$$x_j^{(k+1)} = x_j^{(k)} + \frac{\lambda^{(k)}}{N}\left(y - \sum_{i=1}^{N} x_i^{(k)}\right)$$

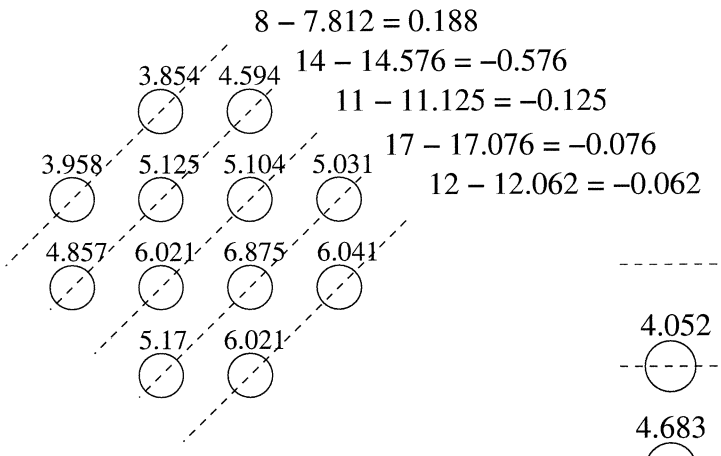
where k is the iteration index (the terms are *not* raised to the power of k), x_i is a pixel element contributing to the projection value y, N is the number of x_i and λ is the relaxation factor. When λ is 1, each iteration corrects a line of pixels to match the projection value under consideration directly. Smaller values of λ perform only a partial correction. Smaller λ's lead to slower convergence, but less risk of oscillation in the solution. [25%]

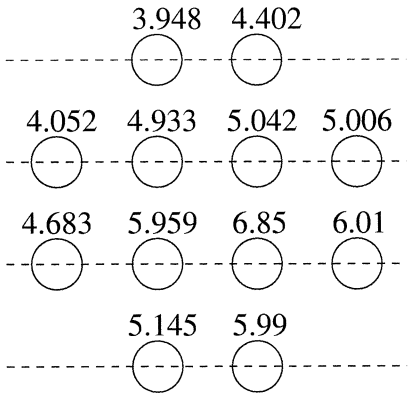
(b) (i) For efficient convergence, it is generally best to change the direction of the projections used as much as possible between each set of updates. So, if the first set of updates are based on projections in a horizontal direction, the next ones should be vertical rather than taking the projections at 45 degrees. [10%]


5 5
8 − 10 = −2
5 5 5 5
19 − 20 = −1
5 5 5 5
24 − 20 = 4
5 5
11 − 10 = 1


9 − 10.75 = −1.75
20 − 20.25 = −0.25
22 − 20.25 = 1.75
11 − 10.75 = 0.25
4 4
4.75 4.75 4.75 4.75
6 6 6 6
5.5 5.5


3.938 4.438
3.875 4.688 5.188 4.875
5.125 5.938 6.438 6.125
5.438 5.938
9 − 9.313 = 0.313
15 − 15.251 = −0.251
12 − 11.126 = 0.874
16 − 15.751 = 0.249
10 − 10.563 = −0.536


8 − 7.812 = 0.188
14 − 14.576 = −0.576
11 − 11.125 = −0.125
17 − 17.076 = −0.076
12 − 12.062 = −0.062
3.854 4.594
3.958 5.125 5.104 5.031
4.857 6.021 6.875 6.041
5.17 6.021


3.948 4.402
4.052 4.933 5.042 5.006
4.683 5.959 6.85 6.01
5.145 5.99

(b) (ii) Add the values in the rows to give 10, 20, 20 and 10. Compare these values with the row totals in Fig. 1 to give four errors:

$$\begin{aligned} 8-10 &= -2 \\ 19-20 &= -1 \\ 24-20 &= 4 \\ 11-10 &= 1 \end{aligned}$$

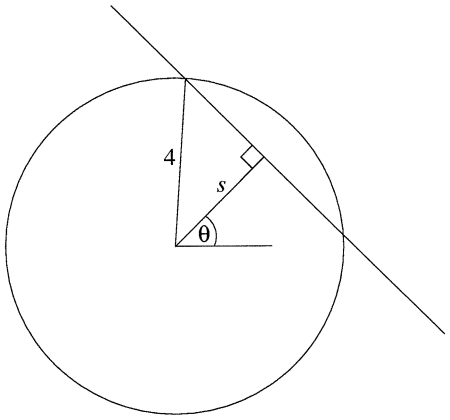
Divide these errors by the number of bars in each row to give row update values of -1, $-1/4$, 1 and $1/2$. Thus subtract 1 from all the top row, subtract $1/4$ from all the second row, add 1 to all the third row and add $1/4$ to all the bottom row. Continue as shown below, doing the columns next and then the diagonals. [65%]

Assessor's remarks: This question tested the candidates' understanding of the Additive Algebraic Reconstruction Technique. The first part of the question asked for a description of the algorithm and the second part of the question required candidates to demonstrate its operation on a simple two-dimensional example. Most candidates understood principles behind the technique and could begin to illustrate its operation. Surprisingly, slightly fewer students were able to quote the update equation and described the full operation of the algorithm correctly for the first part of the question.

2. **Radon Transform and MRI Spin Echo Sequence**

(a) We wish to calculate the radon transform, $\mathcal{R}[\cdot]$ of the two-dimensional function

$$f(x,y) = \begin{cases} 1 & \text{if } \sqrt{x^2+y^2} \leq 4, \\ 0 & \text{otherwise.} \end{cases}$$



There are two cases.
Case 1: $s > 4 \Rightarrow \mathcal{R}[f(x,y)] = 0$
Case 2: $s \leq 4 \Rightarrow \mathcal{R}[f(x,y)] = 2\sqrt{16-s^2}$

The radon transform is independent of θ. [20%]

(b)

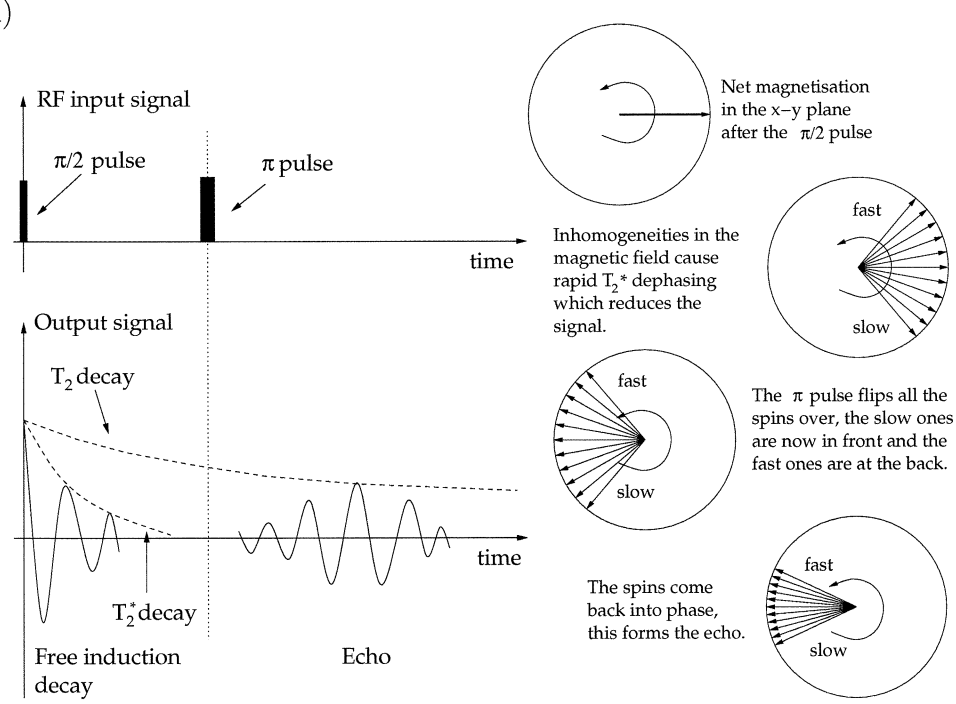
T_1 is the 'longitudinal' or 'spin-lattice' relaxation time. It governs the recovery of the net magnetisation vector in the z direction. Spins change from the higher energy, pointing against the applied magnetic field, to the lower energy, i.e. aligned. Energy is dissipated from the spin system into the atomic and molecular environment — the lattice. Generally, material with a higher proportion of free water has a longer T_1.

T_2 is the 'transverse' or 'spin-spin' relaxation. It governs the decay of the net transverse (x–y) magnetisation to zero. Individual spinning protons generate tiny magnetic fields that affect the spin speeds of their neighbours. This causes the net x–y magnetisation vector to de-phase and reduce. Water bound to the surface of large molecules has a shorter T_2 than free water.

PD is the density of detectable protons (Hydrogen atoms) in the material. This affects the magnitude of the magnetic resonance signal. [20%]

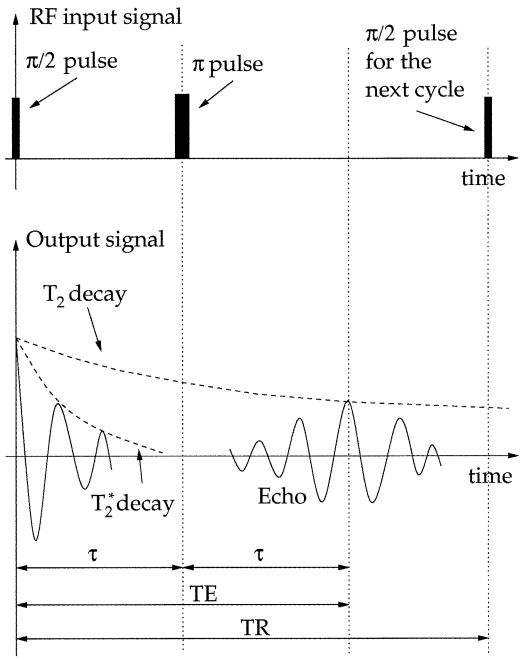
(b)(i) In free induction decay, the output signal de-phases rapidly in the lateral (x–y), direction because of (a) small scale effects from the gradient coils, (b) imperfections in the main magnet and (c) the magnetic susceptibility of the patient. This is called T_2^* and it is generally of the order of a few milli-seconds which is considerably faster than T_2. The signal we are trying to measure therefore dies away quickly before we can measure it. [20%]

(b)(ii)



The spin echo sequence involves two excitations. First a $\pi/2$ radio-frequency pulse is used to initiate free induction decay. Once the spin vectors have dephased, a π pulse is applied to change their directions. The faster vectors will now be at the back of the pack and the slow ones will be at the front. The vectors will therefore come back into phase and produce an "echo" signal from which T_2 can be measured. [20%]

(b)(iii)



T_2 weighted image: A long repetition time, TR (1000–2000 ms) is used, plus a relatively long time to echo, TE (90–140 ms). The long repetition time enables all the spins to return to close to their equilibrium position before the next spin-echo sequence begins. The relatively long TE enables the measurement of T_2. Water and CSF appear brighter than fat. [20%]

Assessor's remarks: This question tested the candidates understanding of some of the basic principles behind magnetic resonance imaging. It was less popular than question 1, but answered well by most of the candidates that attempted it. There were several very good descriptions of T1, T2 and PD, but fewer candidates were able to describe how to configure for a T2-weighted image. When describing the Spin-Echo Sequence, most candidates confused the "time to echo" (TE) with the time between $\pi/2$ and π pulses.

3. **Interpolation of scalar data**

(a) Nearest Neighbour interpolation is very fast, since it simply consists of setting each value to that of the nearest data sample. It also makes no assumptions about the underlying data, and does not generate any new values, so in a sense is the most

faithful to the sampled data. However, the interpolant is not even C_0 continuous and hence looks very blocky, particularly when viewing the data at a large magnification.

Tri-linear interpolation is also relatively fast, and generates a C_0 continuous interpolant, which looks reasonably smooth even when using a large magnification, so long as the original data was fairly well sampled. The interpolant degrades significantly when the data is less well sampled, and this sort of interpolation can not be used for calculating gradients in the data. It is particularly poor at interpolating diagonal features in data sets.

B-spline approximation generates data which is C_2 continuous and hence will look very smooth to the eye even when using a large magnification. It can also be used to produce good gradient estimates in all directions. However, since it is an approximation rather than an interpolation, it is less faithful to the original data and will in general return different values than the original data even at sample locations. It also assumes that the underlying data is itself smooth, so may not be the appropriate choice where this is not in fact the case. It is slower than tri-linear interpolation.

Radial Basis Function interpolation is capable of generating an interpolant with a variety of continuities (dependent on the chosen basis) from completely unstructured samples. However, it is far harder to implement, particularly on large data sets, since it involves the inversion of a matrix of the same dimension as the number of data samples. Although this matrix inversion can be performed very efficiently, this is not a method suitable for real time data interpolation. [30%]

(b) (i) The Catmull-Rom spline is an interpolating spline with C_1 continuity. Hence there will be a wiggle in S which goes below 0 in the $2 \leq x \leq 3$ segment and above 1 in the $4 \leq x \leq 5$ segment. For the former case, the function is given by:

$$S(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{2}\left(t^3 - t^2\right)$$

This has a minima at $t = \frac{2}{3}$, in which case $S = \frac{-2}{27}$. By symmetry, there will also be a maxima in the $4 \leq x \leq 5$ segment of $S = \frac{29}{27}$.

The B-spline is an approximating spline with the convex hull property. In this case S clearly starts at 0 and ends at 1, and the convex hull property ensures it can not go beyond the bounds of the data values, so these are the maxima and minima. [20%]

(ii) For the gradient, we consider the segment $3 \leq x \leq 4$. In this case, the derivative of the Catmull-Rom function is given by:

$$S(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = -3t^2 + 3t + \frac{1}{2}$$

To get $x = 3.5$ we set $t = \frac{1}{2}$, in which case the gradient is $\frac{5}{4}$.

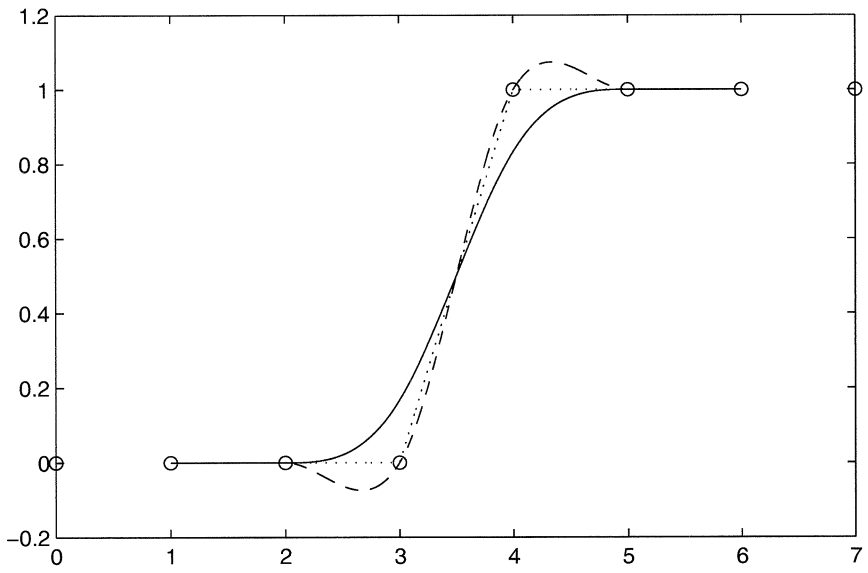
Similarly for the B-spline, we have:

$$S(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = -t^2 + t + \frac{1}{2}$$

To get $x = 3.5$ we once again set $t = \frac{1}{2}$, in which case the gradient is $\frac{3}{4}$.

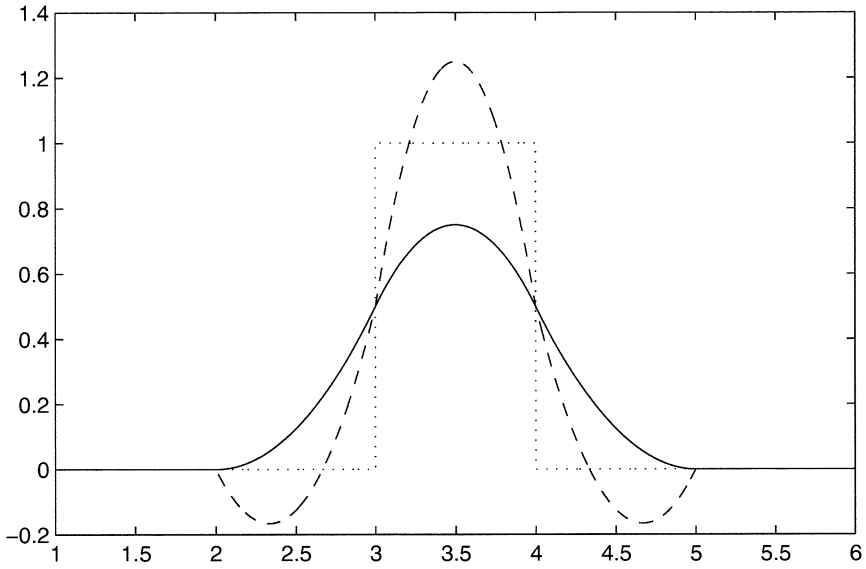
The gradient for linear interpolation is 1. [20%]

(iii) A sketch of the curves is given in the figure below. The solid line is the B-spline, the dashed line is the Catmull-Rom spline, and the dotted line is linear interpolation. The circles show the original samples.



[20%]

For completeness (*this is not required in the question*) the following is the gradient of each interpolant:

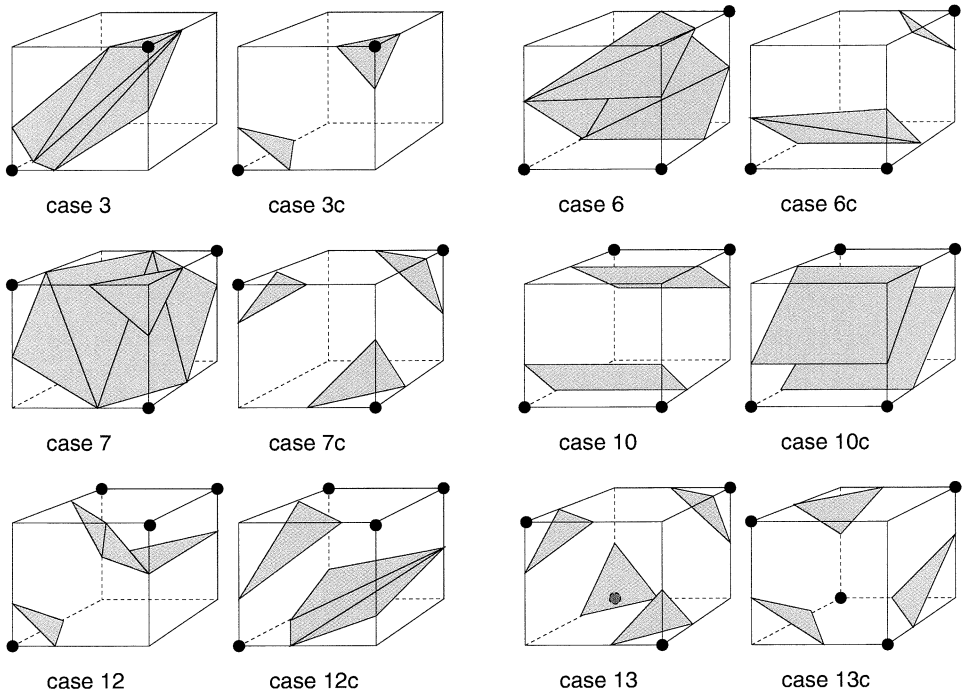


(c) B-splines generate smoother data than Catmull-Rom splines, but do not interpolate values; Catmull-Rom splines interpolate values fairly smoothly, but can generate over-shoots, as in the case above. Catmull-Rom splines will give sharper edges in the data (higher gradients than linear interpolation) whereas B-splines tend to over-smooth real edges in the data (lower gradients than linear interpolation). [10%]

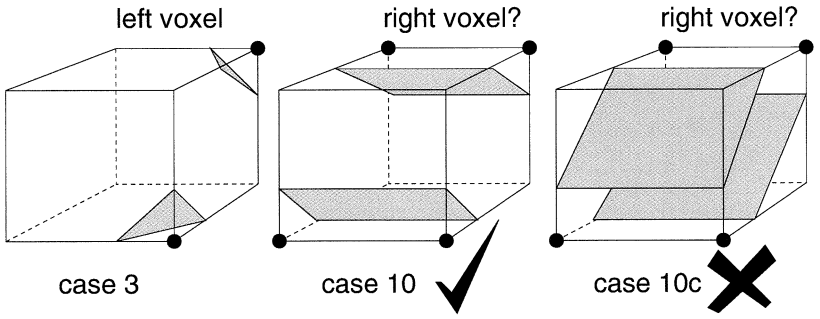
Assessor's remarks: This question investigated the use of various interpolating and approximating functions on scalar data. Most candidates gave good answers to the comparison in (a). Several candidates had difficulty finding the maxima and minima for the two spline functions in (b)(i), though some correctly noted that the convex hull property of B-splines made this part of the question potentially very straight forward. Most candidates gave correct answers for the gradients in (b)(ii). Many of the sketches in (b)(iii) were quite poor, neither exhibiting smoothness nor sometimes even correct interpolation or approximation characteristics, though there were exceptions. Only a few candidates mentioned the effect of the interpolant on gradients in (c).

4. **Marching cubes and laser scanning**

(a) Marching cubes creates a polygonal iso-surface at a specific threshold, in patches based on eight data points at the vertices of a cube. Unfortunately, there are frequently two different ways to construct a polygonal surface which divides vertices on either side of the chosen threshold.



The figure above shows the six possible pairs of alternate surface patches for different combinations of vertices. Only one pair is needed to provide an example of the issue. This causes two problems: firstly, the polygonal surface topology is ambiguous, and may change if we swap the above / below threshold sense of the vertices.



However, topological ambiguity is not the only issue. The figure above shows what happens if the wrong cases are used next to each other: when all the triangulated cubes are connected, a gap remains in the surface. [30%]

(b) Sources of error during laser scanning 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.

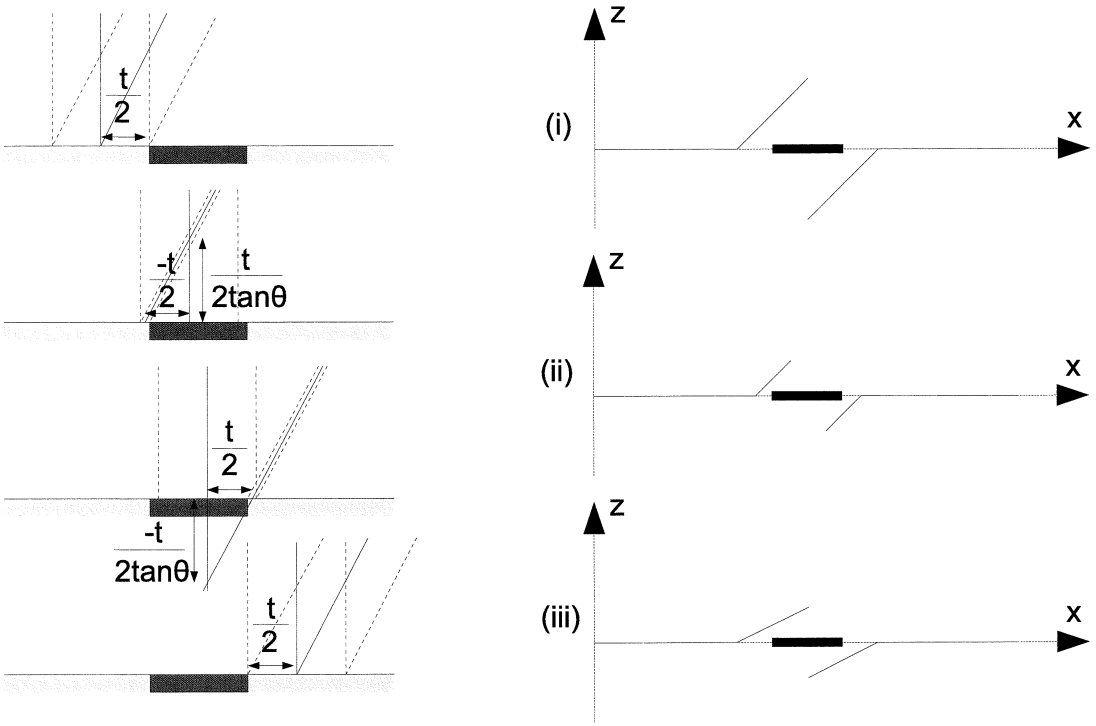
Camera pixel accuracy The depth resolution is determined by the pixel size in the camera image — more pixels give better resolution. The resolution also reduces with distance from the camera and laser, so scanning surfaces as close as possible reduces this error.

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, or changes in reflectance properties, 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. A narrower laser beam will improve this or possible greater camera angle will improve this. [20%]

(c)



The figure above shows what happens as the laser passes over the surface patch. At a distance in x of $\frac{t}{2}$ away from the patch, the z coordinate is correct. As the laser beam nearly covers the patch, the x estimate of the reflected light is wrong by $\frac{t}{2}$ and this causes an error in the z coordinate of $\frac{t}{2\tan\theta}$. The other side of the patch, the error has the same magnitude but opposite sign. The error varies linearly between these values. The sketches for (i), (ii) and (iii) are also given above.

(i) The maximum error is $\frac{t}{2\tan\theta}$. [20%]

(ii) The disadvantages of reducing t are that there is a limit to how thin the laser beam can be and yet still provide enough intensity. Also, the thinner laser beam causes a patch of surface to be completely missed in this case. [15%]

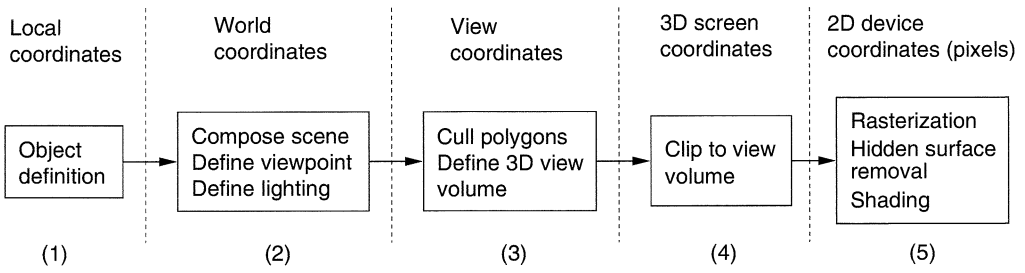
(iii) The disadvantages of increasing θ are that it becomes increasingly difficult to mount the camera at such a large angle whilst still having a reasonable view volume — the camera has to be increasingly distant from the laser source. Also, the larger the angle, the more likely parts of the surface will occlude the reflection and the camera won't be able to see the laser light at all. [15%]

Assessor's remarks: This question tested the candidates knowledge of constructing surfaces using marching cubes and laser surface scanning. The bookwork in (a) and (b) was answered well, with most candidates correctly identifying sources of error in (b). The answers to the laser scanning problem in (c) were more mixed. Few candidates noted that the depth error swapped sign at each side of the non-reflecting

patch, and few noted that the error was linear in x, generally drawing a curved error function instead. However, the impact of the changes in (c)(ii) and (c)(iii) was better understood, and candidates gained marks in these sections so long as the change from the error in (c)(i) was correct, even if the error graph in (c)(i) was itself not correct.

5. **Surface rendering**

(a)



[25%]

(b) The purpose of both *hidden surface removal* and *back face culling* is to suppress parts of the scene that are not visible from the viewpoint. However, they operate at different stages of the pipeline and on different graphics primitives. Back face culling eliminates *whole polygons* that are facing away from the viewpoint. These will be occluded only if they are part of a solid, opaque polyhedron. Hence, back face culling should be enabled only for such polyhedra. Since back face culling happens quite early in the pipeline, and eliminates whole polygons at a time, the computational savings can be significant. In contrast, hidden surface removal happens at the end of the pipeline, as part of the final rasterisation stage, and eliminates occluded *pixels*, typically using a z-buffer algorithm. [25%]

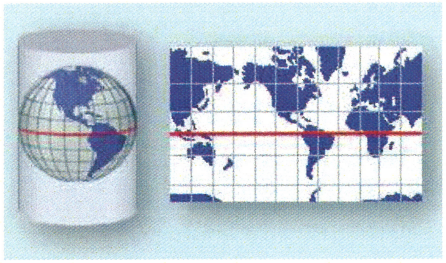
(c) (i) This is most likely inappropriate use of back face culling. The parachute appears to be modelled using four quadrilaterals, so it is important that both the front and back faces are rendered. Enabling back face culling would produce precisely the error in (b): the two missing panels are the two back facing ones. [10%]

(ii) Since the rest of the scene is correctly shaded, this is probably not a problem with lighting. More likely, the surface normals of the lander have been defined the wrong way round. All the lander's polygons are facing away from the illumination and are therefore shaded using the Phong ambient term only, producing the characteristic flat silhouette. It is also possible that the material parameters for the cone have been set incorrectly, such that only the ambient reflection coefficient is non-zero. [10%]

(iii) The lander being a solid polyhedron, a competent programmer would have enabled back face culling (for the lander itself, not the parachute). But it is important to define the front and back faces correctly, by ordering the vertices of the polygons anticlockwise when viewed from the front. The most likely cause of the error in (d) is that the polygons making up the sides of the cone have been defined the wrong way round, so back face culling has eliminated the front faces, not the back ones. It

is also possible, though unikely, that the near and far clipping planes have been set inappropriately far apart such that there is a loss in z-buffer resolution which causes both sides of the cone to have the same z value. [10%]

(d) The image in (b) is a 2D texture that is mapped onto a sphere to produce the rendering in (a). The texture would appear to have been constructed by projecting the planetary terrain onto a cylinder, and then unwrapping the cylinder to produce a flat image, as illustrated on the right.



When rendering the sphere with 2D texture mapping, the programmer uses the same projection to associate the planet's polygon vertices (x_l, y_l, z_l) with points (s, t) in the texture image. Within each polygon, the graphics pipeline then applies perspective interpolation to map each rasterised pixel to a point in the texture image. [20%]

Assessor's remarks: This question tested the candidates knowledge of surface rendering. The bookwork in parts (a) and (b) was answered very well. However, many candidates failed to note the distinction between hidden surface removal operating on pixels, and back face culling on polygons, or the different computational gains in each case. The answers to the rendering problems in (c) were generally good, with candidates offering several interesting explanations as to how these renderings errors might have occurred. However, many candidates suggested incorrect placement of clipping planes, which clearly could not be the case since the planet is rendered correctly. Most candidates noted the use of texture mapping in (d), but few gave sufficiently detailed explanations of the process.

6. **Shading algorithms**

(a) I_λ is the intensity of the reflected light of colour λ, where $\lambda \in \{r, g, b\}$ for red, green and blue.

I_λ depends on several terms. First, there is the ambient reflection term, $c_\lambda I_a k_a$, which models indirect illumination of the surface. c_λ, where $0 \leq c_\lambda \leq 1$, specifies the colour of the surface. I_a is the intensity of the general background illumination, and k_a is the surface's ambient reflection coefficient.

The next two terms in the model are calculated for a point light with intensity I_p. First there is the diffuse reflection term, $c_\lambda k_d \mathbf{L}.\mathbf{N}$, which models even reflection of the light source in all directions. Diffuse reflection is greatest when the surface is pointing directly towards the light source, and tails away to zero when the surface is side-on to the light source. $\mathbf{L}$ is the unit vector from the surface point towards the light source, $\mathbf{N}$ is the unit surface normal and k_d is the surface's diffuse reflection coefficient (small for dark surfaces, high for bright surfaces).

Finally, there is the specular reflection term, $k_s\,(\mathbf{R}.\mathbf{V})^n$, which models directional reflection of the light source along the unit mirror vector $\mathbf{R}$. $\mathbf{V}$ is the unit vector

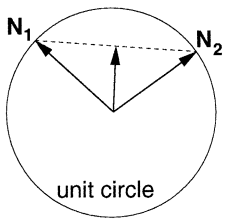
from the surface point towards the viewer. The viewer only perceives the specular highlight (or glint) when looking along the mirror direction, or at least close to it. k_s is the surface's specular reflection coefficient (small for matte surfaces, high for shiny surfaces), and n is the specular exponent that determines the tightness of the glint. n is high for a tight highlight (eg. a perfect mirror) and small for a more blurred highlight (eg. aluminium). [20%]

(b) Both Gouraud and Phong shading work with *vertex normals*, which are found by averaging the normals of all polygons incident at a vertex. Gouraud shading proceeds by calculating a colour at each vertex using the vertex normal and the Phong model. Colours for interior pixels are found by bilinear interpolation. For efficiency, the interpolation can be formulated using fast, incremental calculations.

Phong shading interpolates the normals instead of the intensities. This tends to restore the original curvature of a surface, so that highlights can be reproduced accurately. The disadvantage of Phong shading is its expense. Even though the normals can be interpolated using incremental calculations, the interpolation considers the three components independently, so the vector must be renormalised at each pixel. Then, a *separate* intensity for each pixel is calculated using the Phong model.

Gouraud shading is comparatively fast, though it produces less photo-realistic renderings. It is particularly poor with the specular component. If a highlight should impinge on a polygon but not extend to its vertices, Gouraud shading will miss the highlight. [30%]

(c) Linearly interpolating between two unit vectors will always yield a vector with magnitude less than one (see right). Using these vectors in the Phong equations will produce pixels that are too dark.



However, for small polygons with similar normals at the vertices, the error is likely to be small and tolerable, at least when it comes to the diffuse Phong term. The problem would be with the specular term, where vectors are raised to an exponent n. It is perhaps easiest to appreciate this point by considering the Blinn approximation $(\mathbf{N}.\mathbf{H})^n$ instead of $(\mathbf{R}.\mathbf{V})^n$. Suppose $\mathbf{N}$ has a magnitude of 0.95 and n is 20. Then $(\mathbf{N}.\mathbf{H})^n$ will be 0.358 $(= 0.95^{20})$ times what it should be. [20%]

(d) With a four-value rasteriser, we could interpolate depth z and the three elements of $\mathbf{N}$ from vertices to pixels, but there would be no capacity for interpolating $\mathbf{L}$ and $\mathbf{V}$ as well. This would still permit Phong shading, as long as the light source and viewer were at infinity. If this were the case, $\mathbf{L}$ and $\mathbf{V}$ would be constant across the scene and only $\mathbf{N}$ would need interpolating. The pixel shader program would receive the interpolated elements of $\mathbf{N}$, normalise them, then evaluate

$$I_\lambda = c_\lambda(I_a k_a + I_p k_d \mathbf{L}.\mathbf{N}) + I_p k_s \, (\mathbf{N}.\mathbf{H})^n$$

(i.e. the Phong model with Blinn's approximation) to produce intensity values at each pixel. Note that **L** and **H** are constants.

There are a variety of other possibilities, however, including partial calculation of the diffuse and ambient terms at each vertex, and just using interpolated normals, or even the angle between the reflection and view vector, to update the specular lighting at each pixel. [30%]

Assessor's remarks: This question looked at the Phong reflection model and Gouraud and Phong shading. The bookwork in (a) was answered well. Most candidates could distinguish between each type of shading and gave good explanations of the advantages and disadvantages of each. However few noted the cost of renormalising the normals in Phong shading, and hence failed to spot in (c) that the unnormalised surface normal was the result of linearly interpolating between normalised surface normals. The answers for (d) ranged from very poor to very interesting, plausible techniques for approximating Phong shading.

Graham Treece & Richard Prager & Andrew Gee
April 2010