Engineering Tripos Part IIA THIRD YEAR

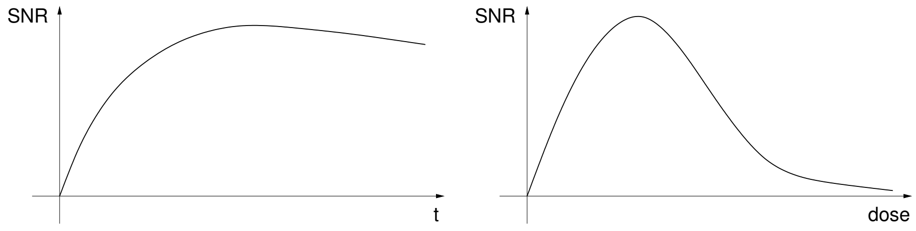
Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2024 Tripos Paper

1. **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)



(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%]

(c) $N(d_1, d_2)$ is the number of photon pairs per second arriving at detectors d_1 and d_2. λ is the distribution of tracer activity, and μ is the attenuation coefficient, along the line joining d_1 and d_2. The measured photon count $N(d_1, d_2)$ is a simple scaling of the required projection $\int_{-\infty}^{+\infty} \lambda(s)\, ds$. So if we know the total attenuation between d_1 and d_2, we can recover the radioactivity distribution by filtered backprojection (although iterative reconstruction would be better, to deal with the Poisson noise). The attenuation can be measured in a combined PET/CT scanner. [20%]

(d) Referring back to (b)(ii), reducing τ will improve the SNR, since fewer measurements will need to be rejected when more than two photons hit the ring within the coincidence window τ. There will also be fewer *randoms*, which are photon pairs that do not originate from the same positron but nevertheless happen to strike the detector array at much the same time, further degrading the SNR. However, τ cannot be reduced all the way down to 0.5 ns, since this will lead to many valid photon pairs being missed. This is because of time-of-flight within the patient: unless the annihilation event happened right at the centre of the ring, the two photons will be detected at different times since they need to travel different distances to the detectors, taking 1 ns to travel 30 cm. [20%]

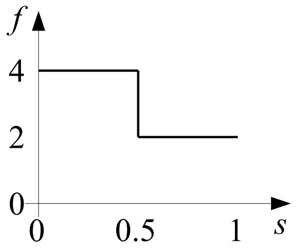
(e) The diagram in the question shows a photon passing directly through the centre of the lower detector's aperture, but being detected by the upper detector. Hence, whereas the line of response should end at the lower detector, it will instead end at the upper detector, causing the event to be mispositioned. Since photon-crystal interactions are random, the next photon to arrive may very well be stopped by the lower detector, the randomness resulting in a blurred image. The loss of resolution is not the same at all locations within the ring, affecting the edge more than the middle. Photons emitted at the centre of the ring travel directly along the axis of each detector, with no probability of being detected by a neighbour. [This so-called *depth of interaction effect* causes an approximate 40% loss of resolution at a distance of 10 cm from the centre of the ring.] [20%]

Assessor's remarks: This question tested candidates' understanding of nuclear medicine imaging and the detection of PET photon pairs. There were several perfect answers, but also quite a few very weak answers where the candidates appeared to have no knowledge of the issues in play, despite this being one of the themes covered in detail through worked examples. For those candidates who had evidently engaged with the course at some level, most demonstrated a good understanding of the fundamentals in (a), though some did not touch on the central issue of sensitivity/collimation. Most candidates answered (b) well, though some thought that detector saturation (at high radionuclide dose) was an issue for PET scanners only, whereas SPECT detectors are similarly unable to deal with multiple photons hitting the crystal within the scintillation time. Most candidates understood the attenuation equation in (c), though not all emphasised its separability and the need for a CT reconstruction to measure μ. In (d), not all candidates identified time-of-flight issues. In (e), while all appreciated that imaging resolution would be compromised, surprisingly few realised that depth of interaction artefacts would affect the edges more than the centre of the ring.

2. **Data interpolation**

(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. [15%]

(b) (i) Nearest neighbour interpolation simply uses the closest value as the interpolant. This is 4 for $s < \frac{1}{2}$ and 2 otherwise:

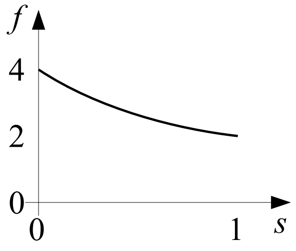


[10%]

(ii) In bi-linear interpolation, the interpolated data is the weighted sum of each surrounding data value, weighted by one minus the (normalised) distance (in each of the s and t directions) to that data point. We could directly calculate that sum using all four data values — but it is simpler to note that if we subtract 2 from all of them, three values become zero, and we can then ignore them in the summation and just use the remaining point, with value 2 at $(0, 0)$. We then add 2 to the final interpolated result. So, if the interpolated data is f, we have:

$$\begin{aligned} f &= 2(1-s)(1-t)+2 \\ &= 2(1-s)(1-\frac{s}{2})+2 \\ &= 2(\frac{s^2}{2}-\frac{3s}{2}+1)+2 \\ &= s^2-3s+4 \end{aligned}$$

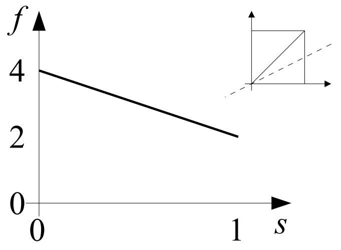
This passes through 4 at $s = 0$ and 2 at $s = 1$ (as it must, since it is an interpolant) but it is not linear, having a minimum at $s = \frac{3}{2}$:



[25%]

(c) (i) The Delaunay Triangulation proceeds by ensuring that the circumcircle of each triangle does not contain any points from any other triangles. Since the four points at the corner of the square from $(0,0)$ to $(1,1)$ are all on the same circumcircle, there are two different sets of triangles which are both correct Delaunay Triangulations. These both contain two triangles, but in one case the diagonal is from $(0,0)$ to $(1,1)$ and in the other it is from $(0,1)$ to $(1,0)$. These are shown at the top right of each of the figures in (ii). [10%]

(ii) Unlike in bi-linear interpolation, interpolation over a triangle is completely planar, but there are two triangles so the interpolant over the line will only be piecewise linear. In one case, the line lies entirely within one triangle, so the interpolant will be a line from 4 to 2, since it starts at the vertex $(0,0)$ with value 4 and ends on an edge whose vertices both have value 2:



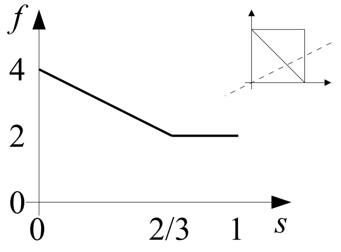
In the other case, the line passes through two triangles, so we need to find its intersection with the triangle edges at $t = 1 - s$. This intersection is hence at:

$$1 - s = \frac{s}{2}$$

with solution:

$$s = \frac{2}{3}$$

Within the first triangle, the interpolant changes linearly from 4 to 2 as in the previous case. As the line passes through the second triangle (starting at $s = \frac{2}{3}$), it stays at 2 since all vertices of this triangle have data value 2. The full sketch is hence:



[20%]

(d) Whilst the data could be approximated using RBFs, so long as the resulting matrices were solved iteratively and the procedure halted before convergence, the most obvious technique to use here is piecewise cubic splines. There are several approximating splines, but a B-spline, as taught in lectures, is the smoothest and hence most appropriate version.

The difficulty with this technique is that it needs more data: to interpolate the data inside the four points, it actually needs a 4×4 grid of 16 points surrounding these data values. Since we do not have this data, we will need to make it up in order to use an approximating technique. There are several options here. We could set all other values to 0: but this does not seem appropriate for the data since all the values we have are at least 2. Setting all the other values to 2 would be a better choice, but this presumes that the data value at $(0, 0)$ is isolated and not, for instance, the corner of a region with larger values.

Each of these choices will affect how the approximation behaves *throughout* the unit square. Probably it would be better to simply accept that we do not have enough data here and interpolate with one of the simpler techniques mentioned above. [20%]

Assessor's remarks: This was a popular question on interpolation and approximation of scalar data. It was mostly very well answered. However, a surprising number of students did lose marks on parts (b) and (c)(ii) by not actually sketching the interpolant, as requested, even if they worked out the correct mathematical description. Understanding of bilinear interpolation and the Delaunay triangulation was clearly good, and many students suggested alternative techniques for (d), though few pointed out the necessity to make up additional points if using bicubic interpolation.

3. **Generic splines and sub-division**

(a) **T** is the parameter matrix, where

$$\mathbf{T} = \begin{bmatrix} t^3 \; t^2 \; t \; 1 \end{bmatrix}$$

and t usually varies from 0 to 1 along the length of the curve. **M** is the basis matrix, which is a 4×4 matrix of constants, varying with each type of spline. **G** is the geometry matrix, which is a 4×3 matrix of physically meaningful points, usually defining the locations of part of the curve, or the end points or gradients.

For a spline surface patch, we need an additional parameter s and parameter matrix **S** which has exactly the same form as **T**, then the x coordinate of the spline surface is given by:

$$q_x(s,t) = \mathbf{S}\mathbf{M}\mathbf{Q}_x\mathbf{M}^T\mathbf{T}^T$$

with similar equations for the y and z coordinates. $\mathbf{Q}_x$ is a 4×4 matrix which is like the geometry matrix, but contains only the x coordinates of the various physical points which define the surface patch. [20%]

(b) Sub-division is the process of splitting a spline curve in two by creating seven control points from the original four, which define two new curve segments consistent with the original curve. The matrix **L** can be used to create new control points (i.e. a new geometry matrix **G**), but this only works for a Bézier spline, in which case the new (left hand) control points are given by

$$\mathbf{G}_L = \mathbf{L}\mathbf{G}$$

When sub-dividing a Bézier surface patch, the sixteen control points are first grouped into four rows of four, and each row is sub-divided to give four rows of seven control points. Then the seven columns are sub-divided to finally give 49 control points. These new control points form four Bézier surface patches which will match the original surface. [20%]

(c) (i) For $t = 0$, the parameter matrix $\mathbf{T}$ is:

$$\mathbf{T}_{t=0} = \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}\right]$$

When multiplying this into $\mathbf{M}$, only the last row remains: so the position of the non-zero term in the last row determines which row of $\mathbf{G}$ will be used. For $\mathbf{M}_1$, the start location is entirely determined by the first row. For $\mathbf{M}_2$ it is the second row.

Considering the gradient, we need to differentiate the parameter matrix and evaluate at $t = 0$:

$$\begin{aligned} \mathbf{T}' &= \left[\begin{array}{cccc} 3t^2 & 2t & 1 & 0 \end{array}\right] \\ \mathbf{T}'_{t=0} &= \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \end{array}\right] \end{aligned}$$

Hence in this case only the penultimate row remains when multiplying into $\mathbf{M}$. Hence for $\mathbf{M}_1$, the first and second rows determine the gradient, whereas for $\mathbf{M}_2$ it is the first and third rows. [15%]

(ii) Each row of $\mathbf{G}$ defines the coordinates of a control point. There are four rows in all and hence four control points. For $\mathbf{M}_1$, which is the Bézier basis matrix, the first (and last) control points define the start and end of the curve, and the middle control points define the starting (and ending) gradients, such that the starting gradient is in the direction from the first point to the second.

For $\mathbf{M}_2$, which is the Catmul-Rom basis matrix, the middle control points define the start and end of the curve, and the first and last control points define the gradients at the start and end. In this case, the gradient at the start is the direction from the first point to the third: and the curve starts at the second and finishes at the third. [15%]

(iii) For $\mathbf{M}_1$, it is easier to use the sub-division matrix $\mathbf{L}$, since the last row multiplied into $\mathbf{G}$ directly gives the mid-point of the curve, $\mathbf{p}(0.5)$:

$$\mathbf{p}(0.5) = \frac{1}{8}\left(P_1 + 3P_2 + 3P_3 + P_4\right)$$

For $\mathbf{M}_2$ we need to evaluate the parameter matrix at $t = 0.5$:

$$\mathbf{T}_{t=0.5} = \frac{1}{8}\left[\begin{array}{cccc} 1 & 2 & 4 & 8 \end{array}\right]$$

and multiply this into the basis matrix, to get:

$$\mathbf{p}(0.5) = \frac{1}{16}\left(-P_1 + 9P_2 + 9P_3 - P_4\right)$$ [10%]

(iv) Evidently, the mid-points calculated in (iii) are not coincident. However, they are both functions of only $P_1 + P_4$ and $P_2 + P_3$ since they are symmetric in the geometry matrix (i.e you can reverse the order of the control points and the sense of t and get the same curve). Noting this, and equating the two mid-point locations, gives:

$$\begin{aligned}
\frac{1}{8}(P_1+P_4)+\frac{3}{8}(P_2+P_3) &= \frac{-1}{16}(P_1+P_4)+\frac{9}{16}(P_2+P_3)\\
2(P_1+P_4)+6(P_2+P_3) &= -1(P_1+P_4)+9(P_2+P_3)\\
3(P_1+P_4) &= 3(P_2+P_3)\\
\frac{1}{2}(P_1+P_4) &= \frac{1}{2}(P_2+P_3)
\end{aligned}$$

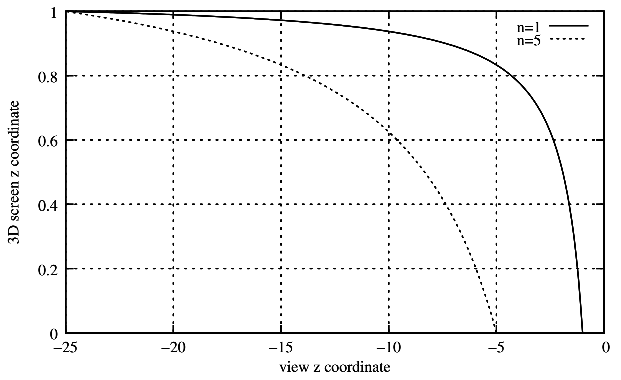
Hence if the mid-point of P_1 and P_4 is in the same location as the mid-point of P_2 and P_3, then the mid-point of each curve will also be coincident.

In that case this is also the location of the actual curve mid-point as well. [20%]

Assessor's remarks: This was another popular question on parametric cubic spline curves and surfaces. Many students provided perfect answers for the calculations in part (c), including correctly noting the type of curve in (ii) and working out the general constraint in (iv). For (iv), a few noted that making the curve a straight line (or reducing the curve in some other way) would make the midpoints coincident, which is correct but only a particular example of the more general condition. Many students lost marks due to lack of detail in the descriptive answers to (a) and (b), either by not defining all the required terms, or by only explaining the terms for either curves or surface patches alone, or by explaining the purpose and uses of subdivision in (b) rather than the process of subdivision.

4. **3D screen space, z-buffers and hidden surface removal**

(a) (i)

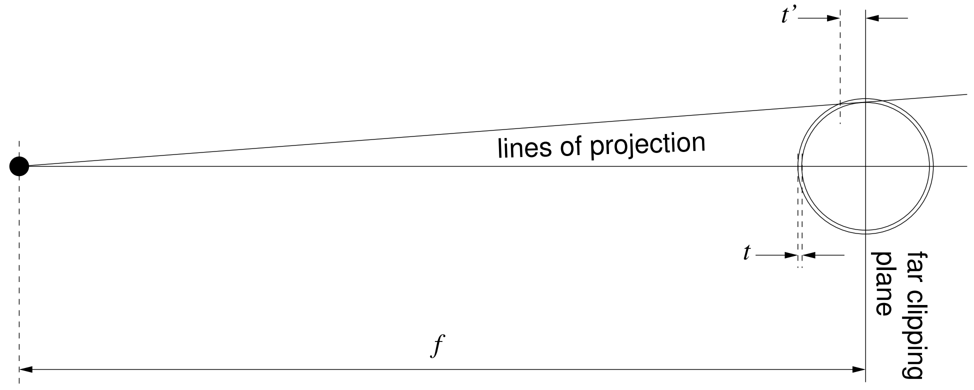


The curves above are for $f = 25$, $n \in \{1, 5\}$. [10%]

(ii) The nonlinear definition of z_s normalizes depth values to the range 0 to 1. More importantly, it is designed to ensure that lines in view space map to lines in 3D screen space, and planes in view space map to planes in 3D screen space. If this were not the case, we would not be able to use linear interpolation in 3D screen space for clipping and depth interpolation. [10%]

(iii) In hardware implementations, z_s is quantized to a finite set of values. At the back of the view volume, a significant range of z_v values will produce the same quantized z_s value, making depth discrimination impossible. This is especially the case when the ratio of the far to near clipping plane distances is large: compare the $n = 1$ and $n = 5$ curves in the example above. [10%]

(b) (i) The viewing geometry is shown below.



At the centre of the rendering, the difference between the z_v values of the outer and inner shells is just t, which is so small that they evidently map to the same quantized z_s value. The artefact does not appear to affect the visible outer edge of the shell, even though these polygons are even closer to the far clipping plane, since here the z_v values are $t' > t$ apart. [15%]

(ii) The assumption is that the z_s values for the outer and inner shells are precisely 2^{-k} apart at the centre of the rendering, where $z_v = -900$ for the outer shell and $z_v = -900.047$ for the inner shell. Hence

$$\begin{aligned} 2^{-k} &= \frac{1000(1-1/900.047)}{999} - \frac{1000(1-1/900)}{999} \\ \Leftrightarrow 2^k &= 1.72 \times 10^7 \\ \Leftrightarrow k &= 24.0 \end{aligned}$$

So the z-buffer precision appears to be around 24 bits. [25%]

(c) A floating point number of the form $m \times 2^e$, where m is the mantissa and e is the exponent, has more precision for small numbers than for large numbers. This is because the gap between successive, quantized numbers, that differ only in the least significant bit of m, scales with e. So, at first sight, a floating point depth buffer is going to make the situation even worse at the far clipping plane, since an even wider range of z_v values will map to the same quantized z_s value. However, there is a trick we can use to make the situation considerably better. If we reverse the z_v–z_s mapping so that the near clipping plane maps to $z_s = 1$ and the far clipping plane to $z_s = 0$, i.e. just flip the y-axis values in the graph in (b)(i) above, then we will benefit from the extra floating point precision where we most need it, towards the left of the graph. The choice of a floating point, reversed z-buffer is fairly common in graphics programming, especially when the scene spans a large range of depth values. [30%]

Assessor's remarks: This question tested candidates' understanding of 3D screen coordinates and the z-buffer hidden surface removal algorithm. There were many strong responses, but also quite a few very weak answers where the candidates appeared to have no knowledge of the issues in play, despite this being one of the themes covered in detail through worked examples. For those candidates who had evidently engaged with the course at some level, the book work in (a) was well understood, as was the precision calculation in (b), although a few incorrectly considered the distance to the back of the view volume rather than the distance between the inner and outer shell surfaces. In (c), very few candidates were secure in their understanding of floating point numbers, though there was one perfect answer including the need to reverse the z-buffer.

Andrew Gee & Graham Treece
May 2024