Engineering Tripos Part IIA THIRD YEAR

Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2016 Tripos Paper

1. **Ultrasound and the Radon transform**

(a) (i) Ultrasound signals that are backscattered from features deep in the body are more greatly attenuated than signals backscattered from features near the surface of the skin, because the ultrasound has to pass a greater distance through attenuating tissue as it travels from the probe to the backscattering feature and back to the probe again. Similarly, the ultrasound signal from features that are deeper in the body takes longer to get back to the probe. It is therefore the case that the longer it takes a backscattered signal to get back to the probe, the more it will be necessary to amplify that signal to compensate for the attenuation caused by the tissue along the path it has travelled. Ultrasound machines therefore have an automatic system that increases the amplification of the received signal as a function of the time after the ultrasound pulse was transmitted. This is called time-gain compensation. [15%]

(ii) The axial resolution of an ultrasound machine is about half the width of the transmitted pulse. The pulse must contain at least one wavelength at the probe centre frequency, so the resolution is improved for shorter wavelengths, which means for higher frequencies. At medical imaging frequencies, ultrasound is attenuated by tissue at a rate that is proportional to the frequency. So higher frequencies lead to greater attenuation and hence smaller scanning depths. The result is that higher frequencies give better resolution but can only be used to scan more superficial structures (e.g. scanning the thyroid at 8 MHz). Lower frequencies offer lower resolution but can achieve greater scanning depths (e.g. scanning the abdomen at 3 MHz). [20%]

(iii) Ultrasound backscatter is caused mostly by variations in the specific acoustic impedance of the material:

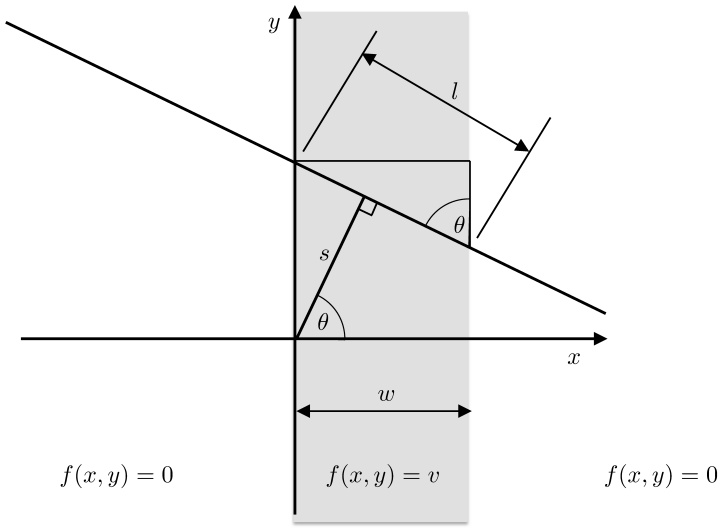
$$Z = \rho c = \sqrt{K_s \rho}$$

where Z is the specific acoustic impedance of the material, ρ is the density, c is the sound speed, and K_s is the adiabatic bulk modulus. Hence backscatter is caused by variations in the product of the adiabatic bulk modulus and the density. [15%]

(b) (i) The Radon transform of a function which has values of zero everywhere except for on a strip of width w, where it takes the value v, is given by the length of the line integral passing through the strip multiplied by v. Let l be the length of the line integral passing through the strip for which the function takes a non-zero value. From the diagram below, we can see that $w/l = \sin\theta$. Hence $l = w/\sin\theta$. Therefore, provided $\theta \neq 0$, the Radon transform of the function is given by

$$\frac{wv}{\sin\theta}$$

For f we have $w = 1$ and $v = 2$. For g we have $w = 2$ and $v = 1$. So, as long as $\theta \neq 0$, the Radon transforms of the function $f(x, y)$ and the function $g(x, y)$ are both given by $2\,\text{cosec}\theta$ for all values of s.



As the strip is of infinite extent in the direction $\theta = 0$, the Radon transform in this direction is infinite for both f and g when $0 \leq s < w$ and zero otherwise. As they have different values of w, they will have different parts of the Radon transform that tend to infinity. [30%]

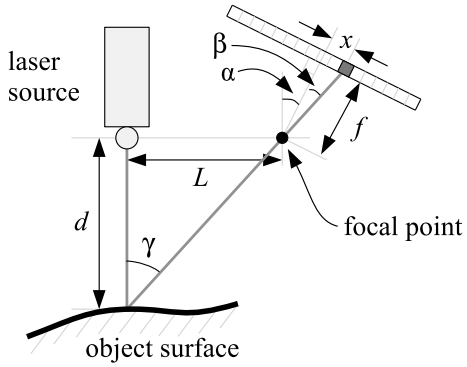
(ii) First, it is important to note that the functions f and g both have non-zero values stretching to infinity in both the positive and negative y directions. Objects of infinite extent do not occur in reality, so stripes in computed tomography subjects are going to produce projections that are different to the abstract results calculated here.

The two Radon transforms are different when $\theta = 0$ and this provides clear information to determine the width of the stripe. For a non-infinite feature, the value of the transforms when $\theta = 0$ will be finite even when $0 \leq s < w$. There will thus be no difficulty calculating unique values for w and v. Furthermore, for a practical stripe feature of non-infinite extent, there will be a difference between the projections of f and g for non-zero values of θ, when θ is close to zero such that the line of integration crosses an end of the feature. This will provide even more distinction between the two Radon transforms. Hence, in reality, the calculation in (i) does not indicate any problem with the tomographic reconstruction of stripe features. [20%]

Assessors' remarks: In the exam, candidates were generally more successful if they understood the principle behind the Radon transform (i.e. the fact that it is a line integral at a given angle and distance from the origin), and drew a diagram to relate this to the problem in the question. Those candidates that worked from the formula for the Radon transform alone found it harder to produce a correct result.

2. **Laser range scanning and cubic B-splines**

(a) (i)



From the figure above:

$$\frac{L}{d} = \tan\gamma = \tan(\alpha+\beta) = \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta}$$

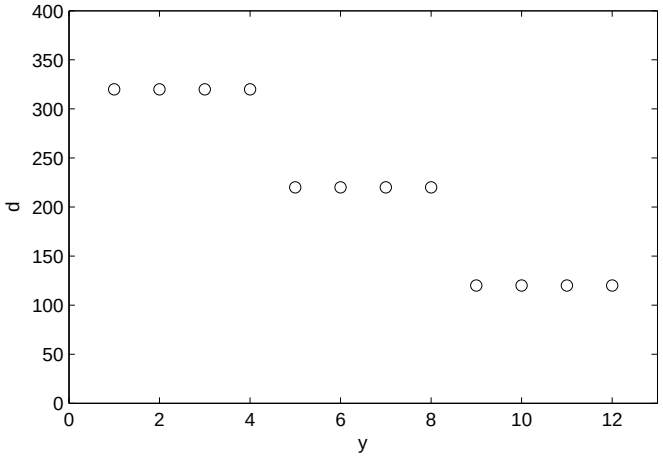
Also,

$$\tan\beta = \frac{x}{f}$$

Rearranging and combining:

$$d = \frac{L(1 - \frac{x}{f}\tan\alpha)}{\tan\alpha + \frac{x}{f}} = \frac{L(f - x\tan\alpha)}{x + f\tan\alpha} \qquad (1) \quad [20\%]$$

(ii) There are three values, $x \in \{0, 2, 6\}$, given by multiplying the pixel locations by the pixel width. Substituting each of these in turn into equation (1) gives $d \in \{320, 220, 120\}$ respectively. Hence the depth plot is as follows:

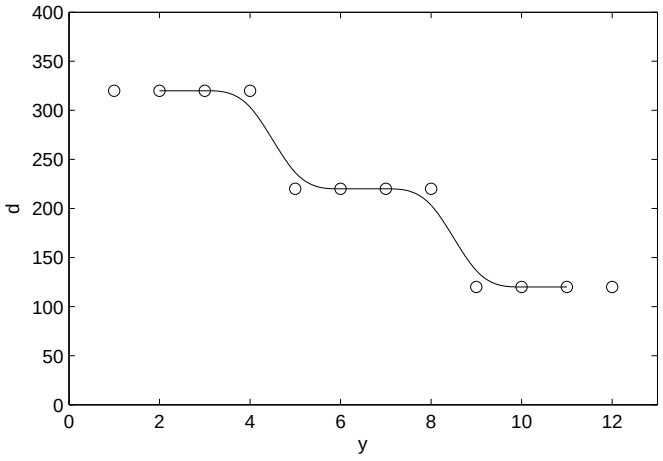


[15%]

(iii) The pixel width contributes an error of ± 0.5 pixels $= \frac{1}{3}$ mm. At the zeroth pixel, putting $-\frac{1}{3}$ into equation (1), and subtracting 320 gives an error of $23\frac{9}{17} = 23.53$ mm. At the 9th pixel, putting $6\frac{1}{3}$ into this equation and subtracting from 120 gives an error of $5\frac{15}{37} = 5.41$ mm.

Alternatively, a very rough approximation to these errors can be determined by noting that 0.5 pixels $= \frac{1}{6}$ of the change in depth for the first transition, since this covers three pixels. Since the change in depth is 100, then the error is roughly $\frac{100}{6} = 16\frac{2}{3}$ mm. By a similar argument, the minimum error is then $\frac{100}{12} = 8\frac{1}{3}$ mm. [15%]

(b) (i)



[15%]

(ii) The maximum error is at any of the pixels next to the depth transitions, for instance the fourth pixel. Using the normal formulation and considering the $t = 0$ start of the curve from the fourth to the fifth pixel, we have:

$$
\begin{aligned}
d &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 320 \\ 320 \\ 220 \\ 220 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 320 \\ 320 \\ 220 \\ 220 \end{bmatrix} \\
&= \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 320 \\ 320 \\ 220 \\ 220 \end{bmatrix} \\
&= 320 \times \frac{5}{6} + 220 \times \frac{1}{6} = 320 - \frac{50}{3}
\end{aligned}
$$

Hence the error is $\frac{50}{3} = 16\frac{2}{3}$ mm. [25%]

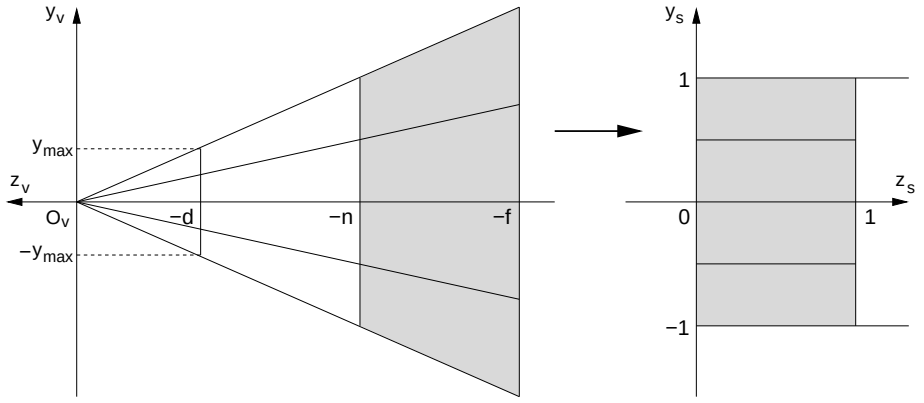
[It might be suggested that the maximum error is exactly at the midpoint of the step, where the curve passes through 270, giving an error of 50 mm. However, this assumes that the surface is piecewise continuous, whereas it might be smooth at this boundary, we simply do not know. So the error should really be evaluated at an actual measurement point.]

(iii) The error in (b)(ii) is in-between those calculated in (a)(iii). Hence, for one of the transitions (between 8th and 9th pixels), the cubic approximation is clearly introducing additional error. For the other transition (between 4th and 5th pixels), it might be acting to reduce the error, depending on whether it is a good assumption that the surface being scanned is actually smooth or not. [10%]

Assessors' remarks: This question concerned the analysis of laser scanners and also interpolation of data using piecewise B-splines. While there were many correct answers to the simple trigonometric proof in (a)(i), there were also surprisingly many problems, given that this was a simplified version of the derivation in the lecture notes. Equally surprising was the number of candidates making numerical mistakes in (a)(ii), which only involved evaluating the given equation. Understanding of the potential errors in (a)(iii) was better: candidates were given credit for any of a number of possible answers so long as the arguments were sound. The B-spline sketches in (b)(i) were mostly good, and there were several perfect answers for the B-spline error in (b)(ii). Many discussions in (b)(iii) showed a good general understanding of the technique.

3. **View volumes and projection matrices**

(a)



[20%]

(b) (i) By inspection of the diagram above, it is evident that the field of view in the y direction is $2\tan^{-1}(y_{\text{max}}/d)$. For this specific projection, this is $2\tan^{-1}(1) = 90°$. Likewise, the field of view in the x direction is $2\tan^{-1}(x_{\text{max}}/d) = 2\tan^{-1}(4/3) = 106.3°$. [15%]

(ii) Dividing the two non-zero elements in the third row of the projection matrix, we find $n = 10$. It follows, manipulating either of the non-zero expressions in the third row of the matrix, that $f = 1000$. [10%]

(iii) The aspect ratio of the image plane is x_{max}: y_{max} = 4:3. Assuming square pixels, a 640×480 window has the same aspect ratio and there will be no geometrical distortion. In contrast, a 1920×1080 window has a mismatched aspect ratio of 16:9, so objects would appear stretched in the x direction by a factor of 4/3. [15%]

(c) (i) Instead of the current 640×480 window, we now need the view volume to project to a 2×2 pixel square region at the centre of this window, so that anything more than one pixel away from the click point is outside the view volume. Hence, compared with the on-screen projection, the off-screen projection needs to have x_{max} reduced by a factor of $640/2 = 320$, and y_{max} reduced by a factor of $480/2 = 240$. The off-screen projection matrix is therefore

$$\begin{bmatrix} 240 & 0 & 0 & 0 \\ 0 & 240 & 0 & 0 \\ 0 & 0 & -100/99 & -1000/99 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad [20\%]$$

(ii) Had the click point not been at the centre of the window, the off-screen view volume would not be symmetrically disposed around the view plane normal. Such view volumes are supported by OpenGL, but the resulting projection matrices have a different structure to the 3G4 simplification that underpins this question. [10%]

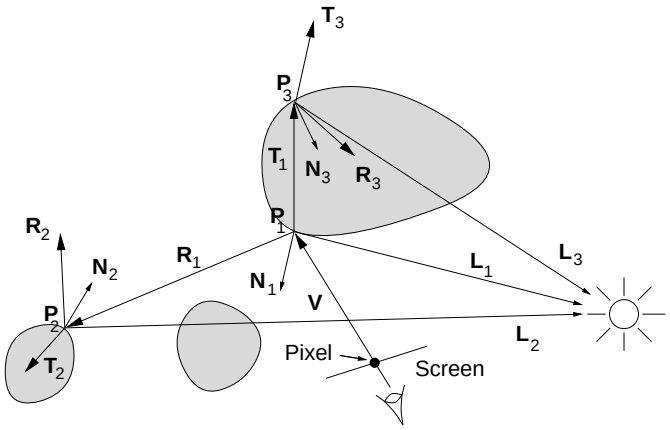
(iii) The important point here is to realise that there might be triangles occluding other triangles, and the user presumably wanted to select the visible triangle nearest to the viewpoint. So, after narrowing the selection down to those triangles covering the click point in the x and y directions, we should choose the one with the smallest depth value. [10%]

Assessors' remarks: This question tested candidates' understanding of perspective projection and clipping. Almost all candidates offered good sketches of the view volumes in (a), and most were fine calculating the various parameters in (b), though some missed the obvious point in (b)(iii) and instead discussed aliasing artefacts and perspective compression of depth values. Answers to (c) were more variable, with only around a quarter of candidates arriving at a plausible off-screen projection matrix in (i), and nobody pointing out that the given projection matrix is invalid in (ii). Answers to (c)(iii) were better, with most candidates suggesting some form of depth sorting.

4. **Ray tracing and intersection tests**

(a) Recursive ray tracing is a rendering technique which accounts for a degree of indirect illumination. It elegantly combines hidden surface removal, transparency effects and shadow computation into a single model.

Simple ray tracing algorithms work in world coordinates. A ray $\mathbf{V}$ is fired from the centre of projection through every pixel on the screen. When the ray strikes an object at $\mathbf{P}_1$, further rays are spawned. One ray, $\mathbf{L}_1$, heads for the ith light source. If $\mathbf{L}_1$ passes through other objects on the way, then the illumination intensity is attenuated by a shadow factor S_i, depending on the number and opacity of objects in the way.

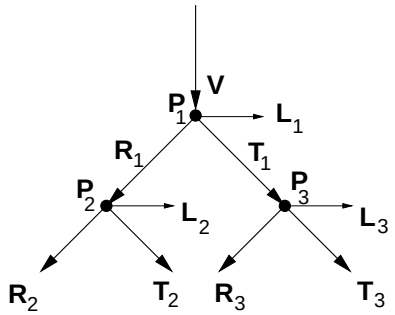


As well as these **shadow feelers**, the algorithm also spawns a **reflection ray** $\mathbf{R}_1$ and a **refraction ray** $\mathbf{T}_1$. The direction of the refraction ray is determined from the refraction indices using Snell's law. The intensity of the pixel is then

$$I_\lambda = c_\lambda I_a k_a + \sum_i S_i f_{att} I_{pi} (c_\lambda k_d \mathbf{L}_i.\mathbf{N} + k_s \left(\mathbf{R}_i.\mathbf{V}\right)^n) + k_s I_{r\lambda} + k_t I_{t\lambda}$$

where $I_{r\lambda}$ is the intensity of the reflection ray and $I_{t\lambda}$ is the intensity of the refraction ray. k_t is a **transmission coefficient** in the range 0 to 1.

Values for $I_{r\lambda}$ and $I_{t\lambda}$ are found by recursively evaluating the equation at the surfaces $\mathbf{R}_1$ and $\mathbf{T}_1$ next intersect ($\mathbf{P}_2$ and $\mathbf{P}_3$), with $\mathbf{P}_1$ as the new viewpoint. So the progress of the algorithm follows a **ray tree**:



The tree is *constructed* top down, until either a ray fails to intersect an object or some predetermined maximum depth is reached. The tree is then *evaluated* bottom-up, as each node's intensity is computed from its children's intensities. Care must be taken to prevent aliasing in the ray-traced image.

In a naive implementation, every ray in the tree needs testing for a possible intersection with every polygon. The required number of intersection tests is therefore

$$n_r n_p \left[(1+n_l) + 2(1+n_l) + 4(1+n_l) + \ldots\right] = n_r n_p (1+n_l)(2^{d+1}-1)$$

(summing the geometric progression inside the square brackets). [40%]

(b) (i)

```
while not finished
  output(X,Y);
  if next_x < next_y
    next_x = next_x + dx;
    X = X + 1;
  else
    next_y = next_y + dy;
    Y = Y + 1;
  end
end
```

[20%]

(ii) Depending on the ray's direction, we may need to decrement `X` and/or `Y` instead of incrementing. There are four increment/decrement combinations, and we would need to pick the right one depending on the quadrant in which the ray's direction lies. We would also need to handle degenerate cases where the ray is parallel to the x or y axis. [10%]

(iii) We would need to calculate increments and intersections in the z direction too, and look for the smallest of `next_x`, `next_y`, `next_z` each time around the loop. There would be eight increment/decrement combinations to choose between, depending on the direction of the ray, and there would be one more special case when the ray is parallel to the z axis. [10%]

(c) For a sensible voxel size, and an array covering the whole stadium, it is likely that the teapot will fit inside a single voxel. So we would find ourselves traversing empty voxels until we come across the one containing the teapot, and then we would end up checking for intersections with every one of the polygons that make up the teapot.

A non-uniform array would perform better in situations like this. We could have a finer subdivision around the teapot, and much larger voxels in the empty regions. The one-off task of constructing the non-uniform array and populating it with polygons (typically using a k-d tree) would take longer than the uniform case, but the subsequent intersections tests would be much quicker. [20%]

Assessors' remarks: Part (a) of this question tested candidates' understanding of basic ray tracing as presented in the lecture course. Parts (b) and (c) then asked candidates to explore a method for speeding up the ray/polygon intersection tests. While some candidates made excellent attempts at even the more speculative parts of the question, most answers were rather poor, even with regards to the book work in (a). Candidates' inability to write simple pseudo-code for voxel traversal in (b)(i) was particularly disappointing. Many attempts were obviously truncated, suggesting rushed efforts right at the end of the examination.

Andrew Gee, Richard Prager & Graham Treece
May 2016