Engineering Tripos Part IIA THIRD YEAR

Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2022 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:

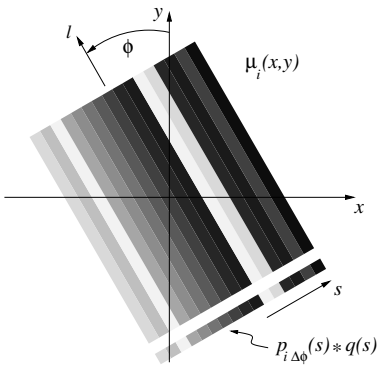
$$\mu(x, y) = \sum_{i=0}^{n-1} [p_{i\Delta\phi}(s) * q(s)] = \sum_{i=0}^{n-1} \mu_i(x, y)$$

where

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

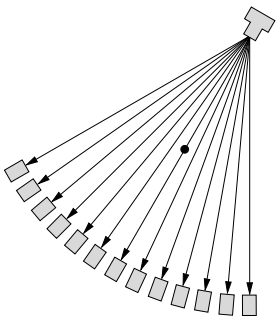
So the attenuation image $\mu(x, y)$ is reconstructed by summing a number of sub-images $\mu_i(x, y)$, each derived from a filtered projection $p_{i\Delta\phi}(s) * q(s)$. Notice how $\mu_i(x, y)$ varies only in the s direction (perpendicular to the X-rays) and not in the l direction (parallel to the 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) Every $\Delta\theta$ rotation provides one further vertical projection, each a little further to the right, until the rightmost edge of the fan beam is vertical after $n - 1$ rotations:

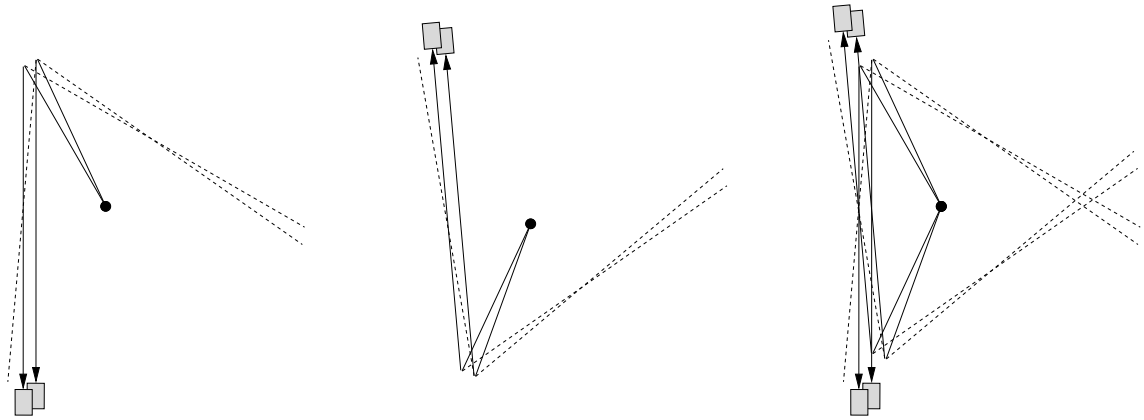


After this, there will be no further vertical projections until we start to get 180° repeats. So we need $n - 1$ rotations in total.

[10%]

(ii) Examination of Fig. 1(d) reveals that the leftmost vertical projection is a distance $R\sin(\phi/2)$ from the centre of rotation. By symmetry, the rightmost vertical projection, shown above, will also be $R\sin(\phi/2)$ from the centre of rotation. So the width of the field of view is $2R\sin(\phi/2)$. [10%]

(iii) In the diagram below, (a) shows the starting position and its neighbour, along with the two vertical projections they produce at the left of the field of view. (b) shows the situation after rotating through $180°$, plus the fan beam width, minus one $\Delta\theta$ step. This produces the two projections at the left of the field of view that are oriented $\Delta\theta$ anticlockwise from the vertical and that would not have been acquired earlier in the rotation sequence.



(a) (b) (c)

(c) shows (a) and (b) superimposed. One step later, the two projections in (b) will be vertical repeats of those already acquired in (a). So (b) represents the finish position, requiring a rotation of $180° + \phi - \Delta\theta$ in total, or $k + n - 2$ steps of $\Delta\theta$.

Alternatively, consider that we need to record k sets of parallel projections. To record the first set, we require $n - 1$ rotations of $\Delta\theta$. Each additional rotation of $\Delta\theta$ prodices a new set of parallel projections. So the total number of rotations required is $(n-1)+(k-1) = k + n - 2$. [20%]

(iv) Yes, having unscrambled the projections into parallel sets we could use the parallel geometry filtered backprojection algorithm for reconstruction. However, note that the rays in each parallel set are not uniformly spaced. For example, in Fig. 1(d) it is clear that the vertical rays are positioned $R\sin(\phi/2-i\Delta\theta), i \in \{0 \ldots n-1\}$ horizontally from the centre of rotation. So we would either need to resample the projections onto regular intervals, or account for the nonuniform spacing in the backprojection algorithm. [15%]

(v) Narrow, cylindrical collimation is feasible in third generation scanners since the relative positions of the X-ray source and detectors are fixed, so each detector only needs to "look" in a particular direction towards the source. Furthermore, collimation is desirable since it prevents Compton-scattered photons from reaching the detectors, thereby reducing the noise in the reconstructed images. [15%]

Assessors' remarks: This question tested candidates' understanding of parallel-beam CT reconstruction in (a), before asking candidates to extrapolate their knowledge to fan-beam reconstruction in (b). Most candidates answered (a) very well, though a few confused filtered backprojection with direct Fourier reconstruction. Many candidates demonstrated a good appreciation of the complexities of fan-beam reconstruction in (b), even though this had not been explicitly covered in the course. Attention to detail was often lacking though, with errors made in simple trigonometric calculations and counting of projections. Nevertheless, it was pleasing to see several candidates obtain the correct expressions in (b)(i)–(iii) and identify the irregular sampling issue in (b)(iv). Most candidates correctly referred to Compton scattering in (b)(v).

2. **Resampling scalar data**

(a) A cubic parametric curve is defined as a cubic function of a parameter t, which is 0 at the start of each segment and 1 at the end. The curve is found by multiplying a row vector parameter matrix $[t^3\ t^2\ t\ 1]$ by a 4×4 basis matrix, and a 4×3 geometry matrix. The geometry matrix contains four points which define the location of the curve, and for multi-segment versions, the new curve is created between the second and third points, by varying the parameter t between zero and one.

The same formulation can be applied to scalar data, but the geometry matrix is now a single 4×1 column of scalar data values rather than a multi-column matrix of locations. New data between the second and third values in this scalar geometry matrix is generated by again varying the parameter t between zero and one.

For neighbouring data, the geometry matrix is changed so that the first entry is thrown away, the next three scalar data values are retained but moved up by one element, and one new value is inserted at the bottom. The process is then repeated with t again varying between zero and one to generate the next part of the interpolant. [20%]

(b) (i) The weights w_i are generated by multiplying the parameter matrix by the basis matrix. So for the Catmull-Rom basis:

$$\mathbf{w} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

For z_1 at $x_z = 0$ we have $t = 0$, and hence:

$$\mathbf{w} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

For z_2 at $x_z = 0.5$ we have $t = 0.5$, and hence:

$$\mathbf{w} = \begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} = \frac{1}{16} \begin{bmatrix} -1 \\ 9 \\ 9 \\ -1 \end{bmatrix}$$

By symmetry, the value for z_3 at $x_z = 1$ is:

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad [15\%]$$

(ii) Following the same argument as in (i) but with the B-spline basis matrix, the weights for $z = \{z_1,\ z_2,\ z_3\}$ are:

$$\mathbf{w} = \frac{1}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \\ 0 \end{bmatrix},\ \frac{1}{48} \begin{bmatrix} 1 \\ 23 \\ 23 \\ 1 \end{bmatrix},\ \frac{1}{6} \begin{bmatrix} 0 \\ 1 \\ 4 \\ 1 \end{bmatrix} \qquad [15\%]$$

(iii) If the data is linearly interpolated, then the end points $z_1 = y_2$ and $z_3 = y_3$, since this is an interpolant. The mid point z_2 is just the average of the two neighbouring points, since the location is exactly in the middle. Hence the weights are:

$$\mathbf{w} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},\ \frac{1}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix},\ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

For the equivalent basis matrix, consider that the result is just the linear interpolation of the surrounding points y_2 and y_3, which is given by:

$$z = (1 - t)y_2 + ty_3$$

This does not involve t^3 or t^2: hence the first two rows of the basis matrix will be zeros. Neither does it involve y_1 nor y_4: hence the first and last columns of the basis matrix will also be zeros. Hence:

$$z = (1 - t)y_2 + ty_3 = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

where the middle 4×4 matrix is the required basis matrix for linear interpolation. [25%]

(iv) The end points z_1 and z_3 should be considered separately from the midpoint z_2.

For the end points, the Catmull-Rom weights are exactly the same as for linear interpolation, since both methods do actually interpolate the data, i.e. return $z = y$ when the location is the same. The weights for the B-spline are different since this is not an interpolant. In fact the B-spline weights show that this is smoothing the data.

For the midpoint, neither the Catmull-Rom nor B-spline weights match the linear interpolation weights, but the B-spline weights are very nearly the same: the mid-values are $\frac{23}{48}$ rather than $\frac{1}{2}$. This means that the midpoint of the B-spline interpolant is nearly the same as for linear interpolation, or at least closer than for the Catmull-Rom spline. [15%]

(v) The convex hull property, if held, means that the resampled data would never be larger or smaller than the four closest data values in y.

This can be deduced from the weights by checking firstly that they sum to one (which they do in all cases) and also that each weight is between zero and one (clearly not true for Catmull-Rom). So the Catmull-Rom spline does not exhibit the convex hull property. [10%]

Assessors' remarks: This question tested candidates' knowledge of cubic spline interpolation applied to scalar data. The fairly standard part (a) was surprisingly poorly answered, mostly due to candidates describing the different types of spline in general, which was not what the question asked for. (b) parts (i) and (ii), which required interpreting splines as weights, were mostly well answered, though many candidates over-complicated the analysis by considering the geometry matrix as a two-column vector including the data locations, rather than just a single column of the scalar data values. Most candidates saw what the linear weights should be in (iii): there was a pleasingly good degree of insight here, as well as several correct answers for the linear basis matrix, which was the hardest part of the question. Most candidates offered sensible answers for (iv) and (v), though it was common to forget that the weights must all be positive as well as sum to one for the convex hull property to hold.

3. **Marching cubes and distance transforms**

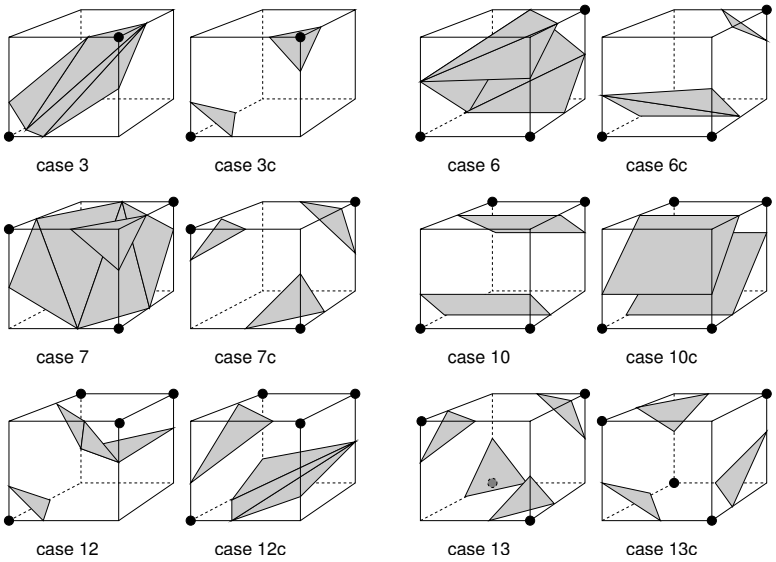
(a) *Marching Cubes* is a technique for extracting a triangulated isosurface from a regular voxel array of data. It breaks the problem down into that of triangulating a cube with one data point at each corner. The *form* of the triangles required to construct the surface within this cube is a function only of whether each data point at the corners of the cube is inside or outside the surface to be defined. This determines the *triangulation case*. Having found the case in which this cube falls, a lookup table is used to give the number and location of the triangles to form the isosurface within this cube.

This lookup table only defines which cube edge the vertices of each triangle lie on. The exact location of each triangle vertex is determined by linear interpolation of the original data values along the required cube edge. For example, if the isosurface is at the value 10, and the data values at the corners are 7 and 13, then this vertex would lie exactly half way along the edge.

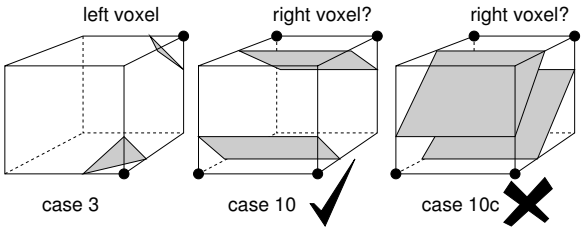
Having triangulated one such cube, we then gradually move (or *march*) the cube through the entire voxel array. The complete isosurface is then given by the set of all the triangles from each of the cubes.

Marching cubes will generate an isosurface with several triangles per cube through which the surface passes. The surface is a good approximation to the real location of the iso-data. [20%]

(b) Six of the 15 possible cases have alternative triangulations. These are given in the figure below (not all necessary to answer the question: just one good example).



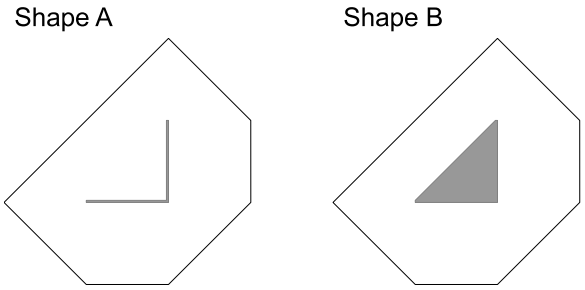
The figure below shows what happens if the wrong alternate case is used. When all the triangulated cubes are connected, a gap remains in the surface.



[10%]

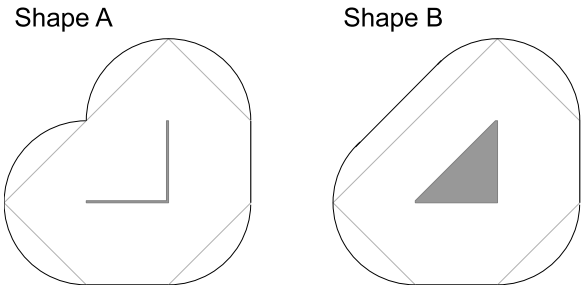
(c) Marching cubes implicitly uses linear interpolation on the underlying data to work out the vertex locations. This will generally work well within each original slice. With shape-based interpolation, each slice is first thresholded, then distance-transformed, and these slices are interpolated to generate intermediate slices. So the resulting data is more dense as well as the in-between slices being better interpolations of the desired threshold. Hence the resulting marching cubes will generate more, smaller triangles (more slices of data) at better locations (better interpolation). [10%]

(d)(i) The figure below shows the city-block contours at 5 cm, note this is exactly the same contour in each case.



[20%]

(ii) The figure below shows the Euclidean contours at 5 cm, with the city-block contour in light grey.



[20%]

(iii) The maximum estimation error is along diagonals, where a city-block distance of 1 actually represents a Euclidean distance of $\frac{1}{\sqrt{2}}$. This is an estimation error of $\sim 41\%$.

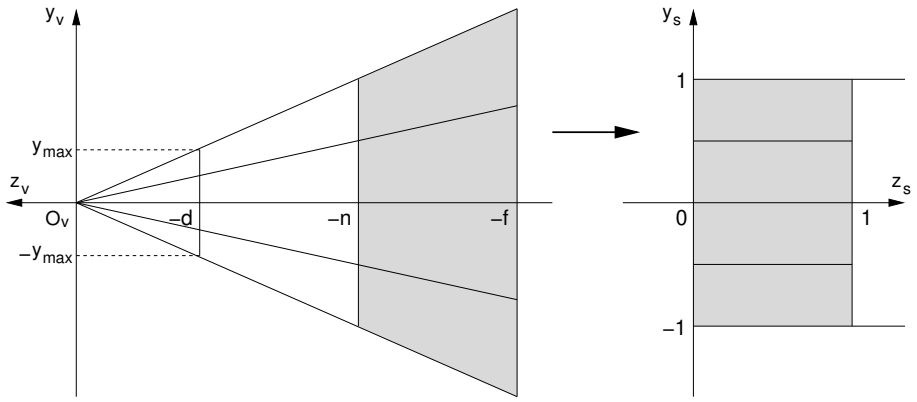
The consequence when calculating margins of error is that the city-block contour may be physically closer to the tumour than was intended, by up to the estimation error above. A larger city-block threshold may have to be used to account for this. [10%]

(iv) In shape-based interpolation, the distance transform is usually sampled fairly close to the original contour, so while the percentage error here would remain much the same, the absolute value of this error will be quite small. So it is likely to have a somewhat reduced effect on any surface extracted from the interpolated data, unless there are large differences in the contours on neighbouring slices. [10%]

Assessors' remarks: This question tested candidates' knowledge of marching cubes and distance transforms in shape-based interpolation. Once again, what should have been a straightforward explanation in (a) was often answered poorly, without much attention to the detail of the marching cubes algorithm. The question about alternative triangulations in (b) was answered much better, with credit given to candidates who noted that different triangulations of the same topological surface could lead to different shading artefacts. Most candidates offered reasonable answers to (c). The shapes in (d)(i) and (ii) were sketched well by most, though a few did not see how a city-block distance could be applied without a discrete grid of pixels (which was not given). Most candidates could calculate the city-block estimation error in (iii), but fewer noted that this did not matter much for shape-based interpolation in (iv).

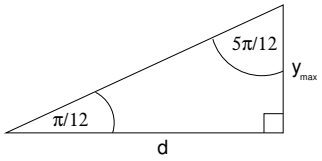
4. **Viewing systems and projection matrices**

(a)



[20%]

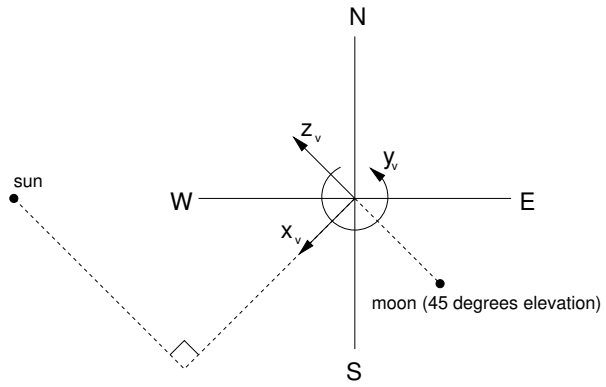
(b) (i) Comparing the general and specific matrices, it is evident that $d/y_{\max} = \tan(5\pi/12)$.



From the triangle above, it follows that the field of view in the y direction is therefore $2 \times \pi/12 = 30°$. The field of view in the x direction is the same. [15%]

(ii) Dividing the two non-zero elements in the third row of the projection matrix, we find $n = 2 \times 10^5$ km. It follows, manipulating either of the non-zero expressions in the third row of the matrix, that $f = 6 \times 10^5$ km. [10%]

(c) (i) The diagram below shows observer A's view coordinate system, along with the positions of the moon and the sun.



Considering the distances to the moon and the sun, the view coordinates of the moon are $(0\,,\,4/\sqrt{2}\,,\,-4/\sqrt{2}) \times 10^5$ km and those of the sun are $(1.5/\sqrt{2}\,,\,0\,,\,1.5/\sqrt{2}) \times 10^8$ km. [15%]

(ii) To rotate coordinates by $45°$ around the x_v axis, we use the matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 45° & \sin 45° \\ 0 & -\sin 45° & \cos 45° \end{bmatrix}$$

Applying this matrix to the coordinates obtained in (i), we find that for observer B, the view coordinates of the moon and the sun are $(0\,,\,0\,,\,-4) \times 10^5$ km and $(1.5/\sqrt{2}\,,\,0.75\,,\,0.75) \times 10^8$ km respectively. [20%]

(iii) In B's view coordinate system, the direction vector of the line from the moon to the sun is $[\ 1.5/\sqrt{2}\ 0.75\ z\ ]^t$, where $z = 0.754$. Adding this to the coordinates of the moon, we obtain an arbitrary point P on the line with view coordinates $(1.5/\sqrt{2}\,,\,0.75\,,\,z - 4 \times 10^5)$. We can find P's perspective projection onto the view plane by multiplying x_v and y_v by $-d/z_v$, obtaining $d\,(1.5/\sqrt{2}\,,\,0.75)/(4 \times 10^5 - z)$. There is no rasterization distortion (since the field of view is isotropic and the window is square), so

$$\theta = \tan^{-1}\left(\frac{0.75}{1.5/\sqrt{2}}\right) = \tan^{-1}\frac{1}{\sqrt{2}} = 35.3° \qquad [20\%]$$

Assessors' remarks: This question tested candidates' understanding of the various transformations between world, view and perspective screen coordinates. Most candidates were able to sketch correctly the view volumes in (a) and calculate the view parameters in (b). In (c)(i), most candidates were able to calculate the sun's and moon's coordinates in view coordinates, though there were many sign errors. In (c)(ii), rotating by $45°$ around the x_v axis turned out to be surprisingly difficult for many candidates. Only a handful of candidates made more than a cursory attempt at (c)(iii), though it was pleasing to see several fully correct and well explained answers.

Andrew Gee & Graham Treece
May 2022