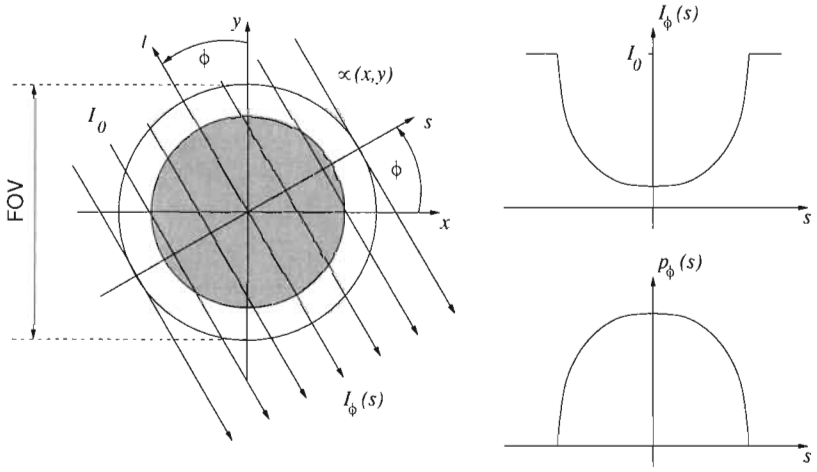
Module 3G4: Medical Imaging & 3D Computer Graphics

Solutions to 2014 Tripos Paper

1. **Sinograms and the Radon transform**

(a) A projection at angle ϕ is the set of all line integrals through the function perpendicular to a line which makes an angle ϕ with the x axis.



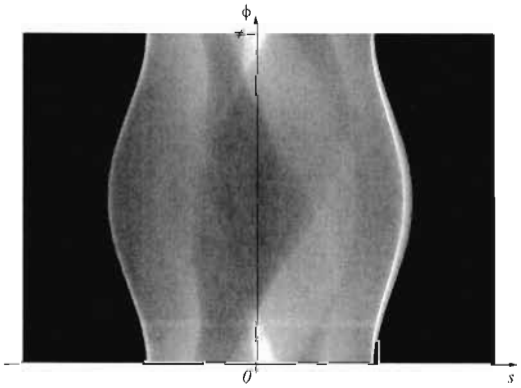
If $\mu(x, y)$ is a function defined on the (x, y) plane, then

$$p_\phi(s) = \int_{-\infty}^{+\infty} \mu(s\cos\phi - l\sin\phi, s\sin\phi + l\cos\phi)\, dl$$

is the projection of μ at an angle ϕ.

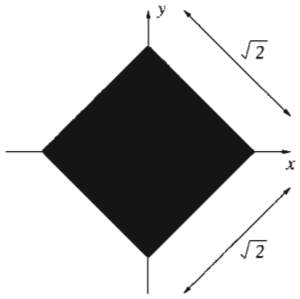
A projection may be displayed with amplitude encoded as brightness.

A sinogram is the set of all the projections of a single function, at different angles, stacked up one on top of another.

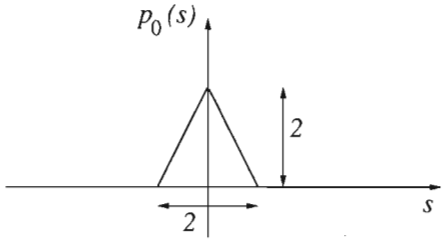


[20%]

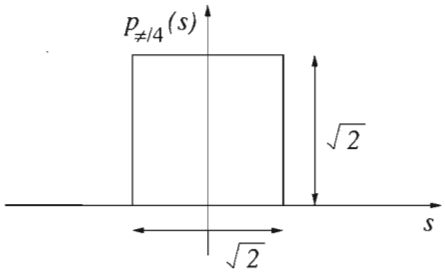
(b) First we sketch the shape of the function f.



The projection at $\phi = 0$ ramps linearly up to a maximum value of 2.

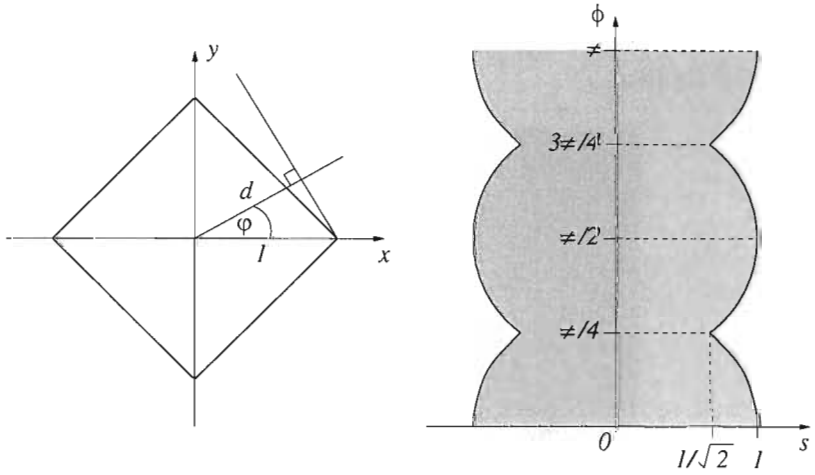


The projection at $\phi = \pi/4$ has a constant value as the object is of constant thickness in this direction.



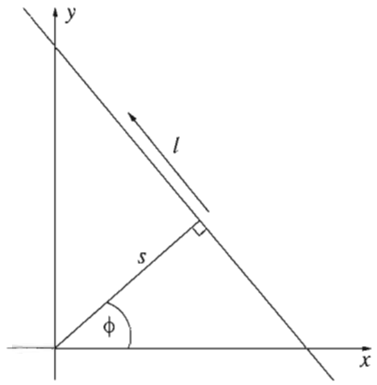
[20%]

(c) For $0 \leq \phi \leq \pi/4$, the half-width d of the silhouette is $\cos(\phi)$ (see diagram, below left).



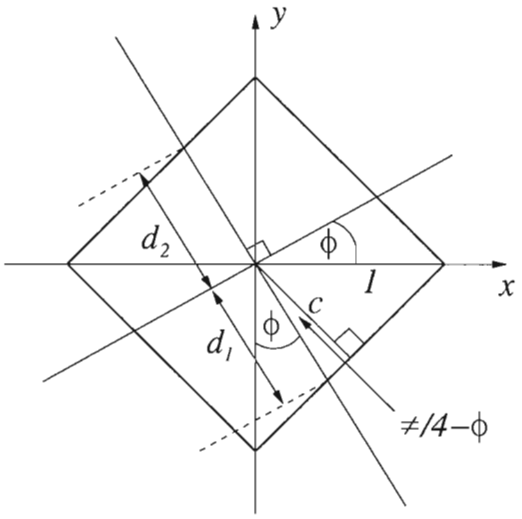
By symmetry, the half-width then increases as a mirrored cosine in the range $\pi/4 \leq \phi \leq \pi/2$, returning to 1 when $\phi = \pi/2$. The whole pattern then repeats for $\pi/2 \leq \phi \leq \pi$. [20%]

(d) The 2D Radon transform maps a function $f(x, y)$ to the set of its integrals over lines at perpendicular angles ϕ and distances s from the origin.



$$\mathcal{R}[f(x,y)] = \int_{-\infty}^{+\infty} f(s\cos\phi - l\sin\phi, s\sin\phi + l\cos\phi)\,\mathrm{d}l$$ [20%]

(e)



When $s = 0$, the Radon transform $\mathcal{R}[f(x, y)] = (d_1 + d_2)$, where $d_1 = d_2$. So we just have to work out d_1 and substitute in the required value of ϕ. Examining the diagram we see

$$d_1 = \frac{c}{\cos(\pi/4 - \phi)} = \frac{1/\sqrt{2}}{\cos(\pi/4 - \phi)}$$

Hence, if $\phi = \pi/8$ we have

$$d_1 + d_2 = 2d_1 = \frac{\sqrt{2}}{\cos(\pi/8)} = 1.5307$$ [20%]

Assessors' remarks: This question tested the candidates understanding of the Radon transform and the idea of a projection, as used in computed tomography reconstruction. Most candidates were able to explain the ideas of a projection and a sinogram in (a) but fewer were able to get the shape of the projections required in (b). A surprisingly large number of marks were lost in this part due to elementary errors in geometry. Most candidates got the approximate shape and period for the sinogram outline in (c), but very few got it exactly correct. Candidates showed good knowledge of the definition of the Radon transform for (d) and many got the numerical calculation in (e) correct which was particularly pleasing.

2. **Laser range scanning and polygon meshes**

(a) Scanning issues 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. This typically generates errors in the distance of each point. 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.

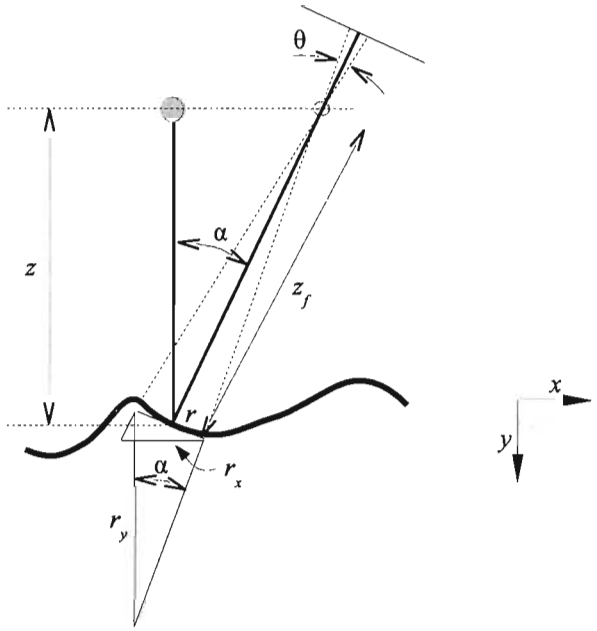
Object movement It generally takes several minutes to scan most objects and the object must remain stationary during that time. This leads to point sets which are not consistent with each other, and combining such sets can be difficult. Obviously, the faster the scan, the lower this error is likely to 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, leading to missing data. 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.

[30%]

(b) If we define r and z_f as in the following figure, and r_x and r_y as the required resolutions:



Then:

$$z_f = \frac{z}{\cos\alpha}\ ,\quad r \approx z_f\theta\ ,\quad r_x \approx \frac{r}{\cos\alpha}\ ,\quad r_y \approx \frac{r}{\sin\alpha}$$

Putting this all together and tidying up, we have:

$$r_x \approx \frac{2z\theta}{1+\cos 2\alpha}\ ,\quad r_y \approx \frac{2z\theta}{\sin 2\alpha}$$

This means that both r_x and r_y scale with the depth z and the camera resolution θ, but r_y is much more sensitive to the camera angle α than r_x. When scanning a large object, parts which are nearer the scanner will have better resolution than parts which are further away. But also, parts which are further to each side of the scanner (smaller α) will also have a poorer depth (r_y) resolution than parts which are immediately in front of the scanner (larger α). This difference in r_y with sideways location is more extreme the closer we are to the scanner. [30%]

(c) (i) The area of a triangle is given by half the base length multiplied by the height. If we align the base with the side from **a** to **b**, then this is given by:

$$\text{area} = \frac{1}{2}(\mathbf{b}-\mathbf{a})(\mathbf{c}-\mathbf{a})\sin A$$

where A is the interior angle at **a**. This can be expressed by a cross product:

$$\begin{aligned}\text{area} &= \frac{1}{2}(\mathbf{b}-\mathbf{a})\times(\mathbf{c}-\mathbf{a})\\ &= \frac{1}{2}(\mathbf{b}\times\mathbf{c}-\mathbf{b}\times\mathbf{a}-\mathbf{a}\times\mathbf{c}+\mathbf{a}\times\mathbf{a})\\ &= \frac{1}{2}(\mathbf{a}\times\mathbf{b}+\mathbf{b}\times\mathbf{c}+\mathbf{c}\times\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{3}\mathbf{v}_i\times\mathbf{v}_{i\oplus 1}\end{aligned}$$

where $v_i = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, $i \in \{1, 2, 3\}$. [20%]

(ii) The direction of $\mathbf{p}$ is the normal to the triangle, or more generally the normal to an average plane passing through all the polygon points. In mesh decimation we are aiming to remove as many points as possible without affecting the geometric accuracy of the surface. This is done by testing for each point whether removing it (and re-triangulating) will make any difference to the surface. So for each point we form an average plane through the surrounding points, then test how close the point is to that plane: if it is close to it, then it can safely be removed. Hence if we calculate $\mathbf{p}$ from these surrounding points, the normalized value $\hat{\mathbf{p}}$ gives the normal to this plane which, along with the plane offset, allows us to perform this test. [20%]

Assessors' remarks: This question investigated laser scanning and mesh properties. (a) was generally answered very well, though only a few candidates mentioned issues with the resolution of the camera. Answers to (b) were more variable, with several candidates not taking advantage of small angles early enough in their working. (c)(i) was generally well answered and (ii) less so: few candidates mentioned actual issues with mesh simplification, opting instead to discuss mesh visualisation.

3. **Spline surface patches and sub-division**

(a) All such splines are defined by a basis matrix and a geometry matrix that, when multiplied together, give the weightings of the cubic polynomial in the curve parameter t. The way in which the geometry matrix is constructed varies for each type of spline, and hence the basis matrix will also be different.

A Bézier curve is defined by four control points, two of which define the ends of the curve and the other two define the end gradients. It is fairly easy to create a single curve segment, but only provides continuity between segments if neighbouring points are carefully aligned. It can easily be sub-divided, which makes it easy to refine a pre-existing curve. It is also useful for display, since the curve is within the convex hull of the control points, so if a polygon formed from the control points is outside the visible area, the curve does not need to be drawn. It can also be efficiently displayed by iterative sub-division and then joining up the resulting control points.

A Catmull-Rom curve will always pass through the control points, and a multiple segment curve inherently has first order continuity. Hence it is easy to define a line which is fairly smooth and passes through defined points. However, it does not have the same convex hull property as the Bézier nor can easily be sub-divided, so conversion is probably necessary before refinement or display. However, the multi-segment nature means moving one control point still results in the same continuity.

A B-spline curve does not pass through control points, but it does pass quite close to them and exhibits second order continuity. So it is a good choice when defining a smooth curve if it is not critical exactly where the curve is located. It does have the convex hull property, so can be displayed according to whether the control point polygon is visible or not. Again,

it is probably easier to convert to a Bézier for curve refinement, although like the Catmull-Rom the control points can easily be moved without changing the overall continuity. [30%]

(b) **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:

$$\mathbf{q}_x(s,t) = \mathbf{SMQ}_x\mathbf{M}^T\mathbf{T}^T$$

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%]

(c) (i) 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

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

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%]

(ii) The control points for the surface patch are symmetrical about the planes $x = 1.5$ and $y = 1.5$, and the z values rise towards these planes of symmetry, hence the maximum must be at the point $(1.5, 1.5)$. With the Bézier basis matrix we could substitute $t = 0.5$ and $s = 0.5$ and calculate the new value for z. However, the maxima also falls at the exact centre which would be the corner of the four new patches after sub-division. Since a Bézier surface always passes through the corner control points, all we have to do is find the location of the corner control point of the new left hand patch.

This corner point is the last point in the left hand sub-division, hence we only need the last row of the matrix **L**, $\frac{1}{8}[1\ 3\ 3\ 1]$. After sub-dividing the curves in both directions, we get the weights w (to apply to the z values of the existing control points):

$$w = \frac{1}{64}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

The z values at these locations are:

$$z = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Combining these gives $z = \frac{96}{64} = 1.5$, or a coordinate of $(1.5, 1.5, 1.5)$. [30%]

Assessors' remarks: This question regarded spline surfaces. The bookwork in (a) was well answered. Most candidates could describe the terms for a spline curve in (b) but definitions for a spline surface were more variable. Sub-division in (c)(i) was generally well described, though in many cases only for a curve rather than the required surface. There were several perfect mathematical analyses for (c)(ii), with others noting the surface symmetry but often not being able to follow up on this.

4. **Illumination and reflection, shading and hardware**

(a) The first formulation is the standard Phong model. 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. $\mathbf{L}$ 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 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 (e.g. a perfect mirror) and small for a more blurred highlight (e.g. aluminium).

The second formulation is Blinn's approximation, which has a different specular term involving the halfway vector $\mathbf{H} = (\mathbf{L} + \mathbf{V}) / |\mathbf{L} + \mathbf{V}|$. Calculating $\mathbf{R}$ involves considerable computational expense: $\mathbf{R} = 2(\mathbf{L}.\mathbf{N})\mathbf{N} - \mathbf{L}$. The advantage of the $\mathbf{N}.\mathbf{H}$ alternative (which produces very similar specular highlights) becomes apparent when the light source and viewer are both at infinity. Under these conditions, $\mathbf{L}$, $\mathbf{V}$ and $\mathbf{H}$ are constant across the scene. So calculating the specular component requires only a single dot product $\mathbf{N}.\mathbf{H}$. [40%]

(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 renormalized 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) Assuming a monochromatic surface and no texture mapping, the most hardware efficient implementation would require just a four-value rasterizer. We could then 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 $\mathbf{N}$, normalize them, then evaluate

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

(i.e. the Phong model with Blinn's approximation) to produce intensity values at each pixel. Note that $\mathbf{L}$ and $\mathbf{H}$ are constants. With a bigger rasterizer we could relax the constraints and interpolate $\mathbf{L}$ and $\mathbf{V}$ as well. [30%]

Assessors' remarks: A popular question, with the bookwork in (a) and (b) well answered by the vast majority of candidates. (c) was much more variable, with many candidates offering vague comments about graphics hardware without addressing the specific issue of accelerated Phong shading. Any sensible discussion of the *minimum* number of interpolated values would need to mention the Blinn approximation with $\mathbf{L}$ and $\mathbf{V}$ constant. There were several excellent answers, though these were few and far between.

Andrew Gee, Richard Prager & Graham Treece
May 2014