Vision and Graphics **Revision Notes for Summative Exam** shi[AT]ebornet.com

Disclaimer:

This is a collection of definitions and simple procedures that were taught in the Vision and Graphics (VIGR) module, initially produced for my own revision needs. This is by no means complete, may contain errors, and you should revise through all materials given in the module. **I could not be held responsible for any deviations of this collection from the original course content**. However, if you spot any problems in this collection, please do contact me through the email address on the top-right corner, so I can correct them for future readers.

This collection is based extensively on (public) lecture and practical materials produced by module leaders of VIGR, most recently by Dr W. Smith and Dr A. Bors. You should consult copyright holders and the department before using it for any purposes other than personal academic revisions.

Source: 2015-2016 materials from http://www-module.cs.york.ac.uk/vigr/ .

Transformations
Translation: vector plus vector.
Scaling: vector multiplied by a scalar [[x,0],[0,y]].
Rotation: vector multiplied by matrix [[cos*x*,-sin*x*],[sin*x*,cos*x*]].
To get the inverse of a transformation, transpose the transformation matrix ($R^T=R^{-1}$).
Affine transformations: transformations that can be expressed as a matrix multiplication and an addition.
y = A**x** + b, A:2x2,b:2x1 in 2D; A: 3x3, b:3x1 in 3D.
Order: $T_1 \rightarrow T_2 \rightarrow T_3$ on **x** = $T_3T_2T_1$**x**

Homogeneous Coordinates
All affine transformations can be represented by a *n*x*n* matrix, where *n* is one plus number of dimensions of the transformation. In 2D transformations, [x,y] → [x,y,1].
Translation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Two homogeneous points are equal if they differ only by a scale factor.

Shear in *x* direction:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear in y direction:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation in 3D: defined with respect to a specific axis: 4x4 matrix, always 0 in 4th row and column, except for always 1 in bottom-right corner; upper left 3x3 of matrix: always 0 in the x (first) / y (second) / z (third) column and row, except for their intersection, which is always 1, for none-zero rows and columns, apply [[cos*x*,-sin*x*],[sin*x*,cos*x*]].
E.g.:

$$R_x(\theta) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Image

Rectangular grid of pixels.
Histograms: represent global statistical information from images. Single-dimensional array for grey-level, multi-dimensional array for multiple colour channels.
Histogram stretching: a mapping of the histogram aiming to improve the image contrast.
Volumetric image: constructed from layers of images.
Faces (polygons formed by vectors) and vertices (vectors) are used to store graphical objects.
Pinhole camera: projecting through an indefinitely small hole,
Advantages: continuous depth sharpness, wide angle range, exact perspective projection.
Disadvantage: dark image (not enough light).
More exposure time = motion blur, larger pinhole = light spread blur; need to use lens to concentrate light.
Lens focal distance *f* affected by: curvature on two faces, material.
View plane: the place where the image is formed. Photo-reactive chemicals; CCD; Retina.

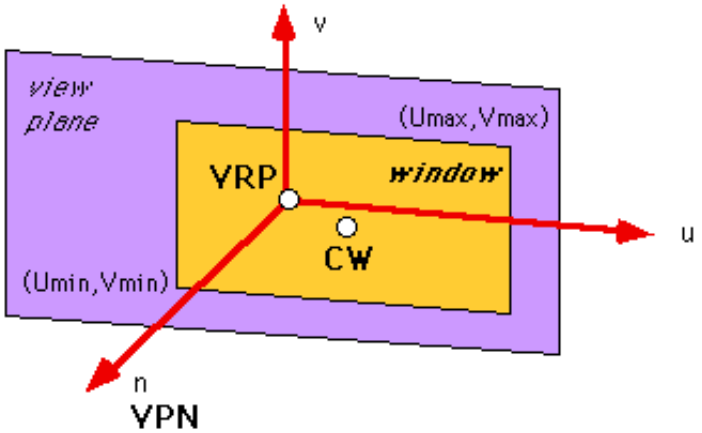
Viewing System
In computer graphics, objects are fixed, the viewer moves to the appropriate positions to achieve the desired view of object projected on image planes.

Perspective Projection
The further away an object, the smaller it is.
Lines not parallel to the viewplane converge to a vanish point.
Principle vanishing points for lines parallel to one of the object's principle axes.
COP: centre of projection.

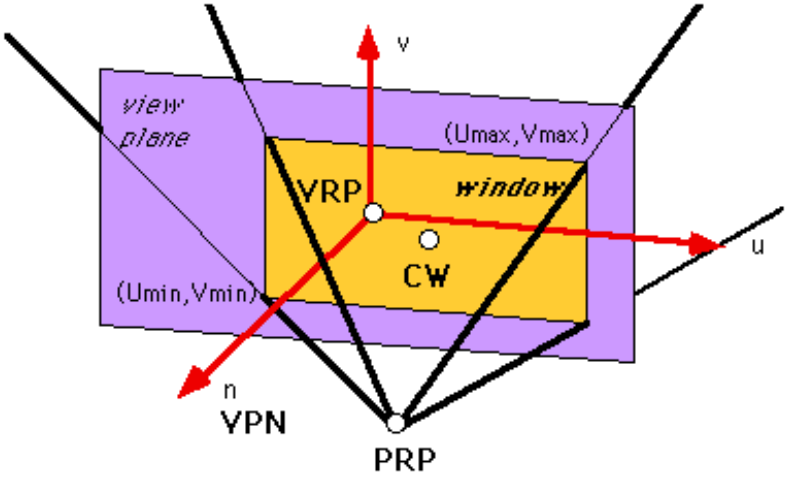
Parallel Projection
COP at infinity.
DOP: direction of projection.
Viewplanes aligned with axes or DOP's.
Types of parallel projection:
Orthographic: direction of projection is normal to view plane. Objects have similar size despite the depth, used for computer aided design (CAD).
Oblique: direction of projection is not normal to view plane.

View Plane
Anywhere in the world-space.
Defined by point on the plane (view reference point, **VRP**), and normal to the plane pointing to COP. (view-plane normal, **VPN**).

Viewing Reference Coordinate system (VRC)
Axes u,v,n.
Origin: VRP.
n-axis: VPN.
v-axis: **VUP** (view-up vector).
u-axis: defined to form a right-hand coordinate system with n and v.
A window is a part of the infinite viewplane, defined by (**Umin**, **Vmin**) to (**Umax**, **Vmax**) .
VRP may be off-center or even not within the window.



CW: centre of projection window.
Projection Reference Point (**PRP**) is relative to the VRC.
PRP = COP in perspective projection.
DOP = PRP-CW in parallel projection.



View Volume
A specific region of the scene which is viewed in the current image.
Back clipping plane – rectangular image – projection rays.
Cone in perspective projection, cuboid in parallel projection.

Focal Length
Can model the effect of the distance to the focal plane and the density of the receptors like CCD.

Axis — Focal Length Params — Skew Param Usually 0 — Offsets from Center

$$x = \frac{\phi_x u + \gamma v}{w} + \delta_x$$

$$y = \frac{\phi_y v}{w} + \delta_y.$$

Intrinsic Parameters formed by with focal length(s), skew and offset.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Extrinsic Parameters from position in real world.

Point in frame of camera | Extrinsic Matrix | Point in frame of world | Translation

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

Extrinsic parameters formed from $R_xR_yR_z$ in 4D homogeneous space with transformations on 3 axes.

Camera Calibration

Scale in Image | 2D homogeneous coordinates | Intrinsic Camera Parameters | Extrinsic Camera Parameters | 3D homogeneous coordinates

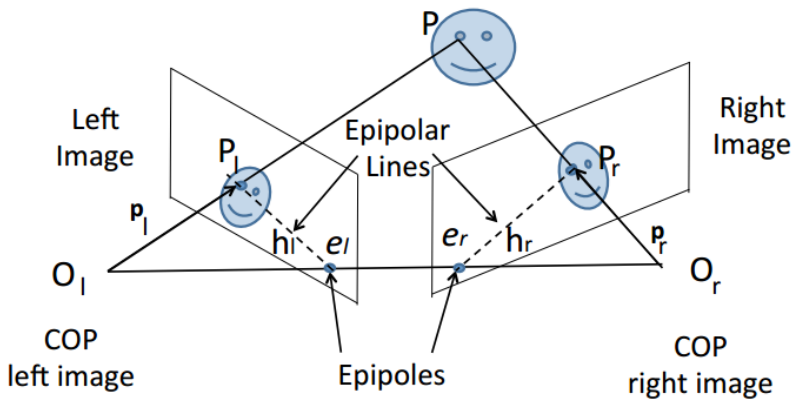
$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}$$

Iterative update: solve a system of equations with camera parameters as unknowns. Make assumptions, estimate error, then improve camera parameter estimations again.

Epipolar Geometry

Used to connect images of the same object taken from different perspectives, or search for one point from one image in the other image.

Projection of object P, Two images *l*, *r* have COP's O*l* and O*r*. p*l* = *Ol* P, p*r* = O*r* P. Connect *Ol*-*Or*. Image *l* intersects *Ol*-*Or* at e*l*, intersects p*l* at P*l* ; image *r* intersects *Ol*-*Or* at e*r*, intersects p*r* at P*r*. Connect h*l* = P*l*-e*l*, h*r* = P*r*-e*r*. h*l* and h*r* are the corresponding epipolar lines.



The left and right image are connected by means of a matrix representing rotation **R** in the plane PO*l*O*r* and translation ***t***.

E- is called fundamental matrix

$$\mathbf{p}_r \mathbf{E} \mathbf{p}_l = 0 \quad \text{with} \quad \mathbf{E} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{R}$$

rank (**E**) =2

If displacement between images is parallel, epipolar lines are in parallel and epipoles (e*l*, e*r*) are at infinity.
Image rectification: parallel the epipolar lines by multiplying them with the transformation matrix.

Depth Map
Represents the distance from the image plane to the actual 3D scene.
Estimated from disparities between pairs of images in Computer Vision, calculated as distance from 3D scene to image plane in Computer Graphics.
Zero disparity for objects far away – they appear in identical places. Displacements for objects near the camera, according to their depth.
Block Matching for Depth Maps:
Split an image in blocks of pixels, cannot be too small (confusions) or too large (computationally expensive).
Split left image in blocks of pixels (B). Define a search region (S) in right image around B's position in left image. S > B. Find a block of B's size most similar to B in S, which is B'. Vector BB' is the difference of their locations.
Works well with feature-rich textures, does not work well with constant colour or regular textures.
Local errors when estimating from real images: left and right images affected by camera noise or change in illumination, cause wrong matches and hence depth map errors.

Space Carving of 3D Imaging
Voxel information preserved if it projects onto any image, otherwise carved away.
If both camera sees the same voxel, it is preserved. Statistical estimation for voxel seen by many cameras.
Voxels used for construction in space carving.
Errors in construction due to confusions caused by illumination change, noise and incorrect camera geometries.
Procedure: Start with a large, full cube filling the 3D space. Carve away each individual voxels of the cube when required. If a voxel is not carved away, estimate its colour from colours of its projected pixels.

Photometric Processes
Recording light (electromagnetic waves) arriving at a camera from the scene.
Shorter the wavelength, better the resolution.

Infra-red lights have longer wavelength, so IR cameras have worse resolution.
Irradiance: power incident on a surface (per unit area).
Radiance: power from a source (per unit solid angle per unit source area).
Image brightness has no physical meaning, just a computational value.

Charge-coupled Device
Square array of solid-state capacitors based on the photo-electric effect of photons knocking out electrons.
Electron capture "wells" allow charge to accumulate.
Capacitor values go through shift registers into ADC.
CCD Response: gain x input + bias + noise – noise from dark currents (thermal), photon noise (quantum), pixel quantisation and amplifier.
Saturated electron capture well means maximum brightness value (255 in 8-bit image).
Cameras apply algorithms to achieve non-linear response to illuminations for better looing images.
Signal-to-Noise Ratio (SNR):

$$\text{SNR (dB)} = 10 \log_{10} \frac{\text{signal power}}{\text{noise power}}$$

Quantisation noise: ADC errors, follow a uniform distribution.
SNR (dB) = $10\log_{10}(2^{B-1} * \text{sqrt}(12))$
Photon noise (shot noise) due to random emission of photons, significant SNR (23dB for mid-brightness levels). Follows Poisson distribution.
Dark current due to thermal energy within the CCD. Electrons in the wells without light falling on the sensor, builds up over time and increases with temperature. Related to manufacturing quality. Can be limitedly removed with dark frame subtraction. More significant at low light levels, overall not as significant as photon noise. Follows Poisson distribution.
To record colour images, CCD can:
Take 3 separate images with different filters – motion blur problem;
Use beamsplitter with 3 CCDs with own filters – expensive;
Mosaic pattern of individual pixel filters – worse actual resolution, cheap, commonly used.
Bayer Mosaic: differently coloured filters for each CCD cell, more G than R/B due to human vision has higher G resolution than that of R and B. Must be interpolated (demosaicing). Lower effective CCD resolution.

Exposure
Relationship between pixel brightness of an image taken and the scene radiance of where the image is taken. Specific to each setting of each camera.

Exposure Illuminance Time

$$H = E \cdot t$$

To change the amount of light coming into the camera, change irradiance (light arriving at CCD, in relation to **aperture size**) or **shutter speed**.
Aperture size: the cone angle of a bundle of rays that come to focus in the image plane, measured in "stops" (a discrete set of aperture sizes).
Smaller aperture (larger f-number) = less light, sharper focus; Bigger aperture (smaller f-

number) = more light, sharp focus for limited ray focal lengths only.

$$\text{f-number} = \frac{f}{D}$$

where *f* is the focal length, and D is the aperture diameter.

To halve the aperture area, f-number needs to be multiplied by 1.4. Hence the set of stops for f-number is 1, 1.4, 2, 2.8, etc.

The larger the f-number, the less light reaches into the aperture onto the CCD.

Shutter speed: slower the speed (longer the exposure), more light coming in; however more motion blur and dark current noise.

Gain: Image brightness can also be adjusted by amplifying the analogue signal from CCD. Gain is measured in ISO scale, baseline ISO 100. However, cannot alter saturated pixels.

HDR Imaging

Varying shutter speed is the most practical for HDR Imaging.

HDR captures a sequence of images with different exposures, and combine images to capture much wider dynamic range.

Need to fit a highly dynamic range of real word irradiance with in a limited CCD value range.

$$\log(E_i) = \log(H_{ij}) - \log(t_j)$$

H is brightness of pixel *i* in image *j*, *t* is known and varying shutter speed. E is unknown radiance at a pixel.

Apply varying weight to brightness values, estimate is unreliable for bright pixels (close to 255), dark pixels (close to 0, too noisy).

$$w(H) = \begin{cases} H & \text{if } H < 127.5 \\ 255 - H & \text{if } H > 127.5 \end{cases}$$

For each pixel: take a weighted average of the estimates from each image. Excluding saturated pixels. Estimate radiance of a pixel from images where exposure for that image is appropriate for the scene radiance.

Process colour channels separately for colour images.

To display these images, use tonemapping (gamma compression). Tonemapping can be global (same function applied to all pixels), but applied to local regions around each pixel in practise.

$$I_{\text{tonemapped}} = cE^{\gamma}$$

where c is a constant scaling (c > 0) and *γ* is a constant that determines the contrast of the image (0 < *γ* < 1).

Light and Spectra

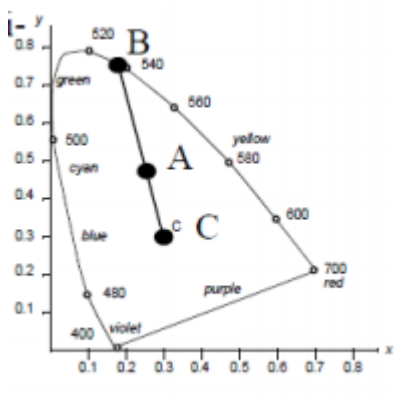
Visible light wavelength 350 nm to 780 nm. Ultraviolet just under 350 nm, infra-red (heat

generation) just above 780 nm. Spectra outside visible light can be visualised by appropriate sensors and lenses.
False colouring: images that do not appear to the human eyes as the original, mapping given by a look-up table of colours to pseudo-colours.
Hyperspectral image: a vector of various wavelengths associated to each image pixel. Constructs a volumetric image with two spatial (real) dimensions and a spectral dimension of wavelength, with brightness reflecting how surface at each pixel absorb and reflect radiations of each specific wavelength.
Illumination: The lighting conditions of the scene have a significant impact on the scene. Both the spectrum of the incident light and the light absorption/reflection/diffusion properties of the object influence the colour of an object. Illuminating light must include the visible spectra.

CIE Colour Model
CIE: 3 standard primaries x,y,z for three colours perceived by human eyes. Any visible colour is a combination of x,y,z. CIE Chromaticity Diagram is taken from x+y+z=1 plane in the CIE Diagram. Chromaticity Values use dominant wavelength and saturation, and disregard luminance and luminance-dependent colours like brown.

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z}$$

Need just (x,y) and *Y* or similar pairs to workout the chromaticity value, as x+y+z=1 => *X* = *Y* *x/y, *Z* = *Y**(1-x-y)/y. Chromaticity diagram can be used to calculate the dominant wavelength and excitation purity of light:



Find the position of colour in the chromaticity diagram (A), draw a directed line from centre (x=y=z=0.333...) C to A, crossing the U-shape at B; dominant wavelength = B, excitation purity = AC/BC * 100%.

Complementary colours are colours on a line through C with same distances to C. Mixing them will create white light.

Some colours have no dominant wavelength, line from C to their colour position will point towards the opening of U-shape.
CIE Matches monochromatic lights.

Colour Image Pipeline
Sequence for digital image processing:
CCT → Gamut Limit → White Balance → De-mosaic
Details:
1. Take output from a Bayer sensor.
2. Coordinated Colour Temperature (CCT) applied to adjust colours based on the light illuminating the source.
3. Apply colour gamut, fitting colours into the colour space that can be visualised, device-dependent.

4. Adjust the white balance.
5. De-mosaic output, mixing arrays of single colour pixels into a demosaiced image.

Colour Systems

RGB: based on the saturations of three colours: red, green, blue. Baseline values defined by NTSC standard and monitor manufacturers. Convert between colour-matched XYZ(CIE) and RGB by matrix multiplication.

YIQ: NTSC composite signal, Y is luminance, I and Q influence hue (perceptual analogue of dominant wavelength) and purity. Multiply a defined matrix with RGB vector to get YIQ.

CMY: subtractive colour model for printing, $[1,1,1]^T$ minus RGB vector for ink mixing.

Munsel: Cylinder system, hue is along the circumference of the top and the bottom, 100 eye-distinguishable hues equally separated. Saturation from none (centre) towards pure/dominant wavelength (circumference). Brightness from bottom (darkness) to top (most bright).

HSV: Pyramid, top is range of hue separated around, saturation is how far from centre towards top circumference, intensity is from bottom (none) to top (most). Convert from RGB:

$$H = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

$$V = \frac{1}{3}(R+G+B) \qquad S = 1 - \frac{3}{R+G+B}[\min(R,G,B)]$$

HSL: Double cone with luminance. Histogram can be calculated by quantising luminance into levels, and for each level calculate hue mean weighted by saturation, and saturation mean corresponding to the luminance level. Heights of histogram based on frequency, colour based on saturation-weighted hue mean. Used to distinguish discontinuity in hue.

Evaluation: RGB most used because of simplicity; HSV most intuitive; CIE Lab is most accurate for human eye; Y component of YIQ is best to represent image grey-level.

Image Manipulation

Images can be considered as 0-255 quantised results of functions of scene information.

Subsampling: remove rows and columns at equal adjacencies from an image and reconstruct (with replication – bad or interpolation – better).

Interpolation: reconstruct subsampled images with information from a pixel's neighbourhood. Replicate a pixel from neighbour value, and replace with weighted average of its neighbourhood.

Convolution: procedures for noise-removal, sharpening, producing artistic effects etc.

Image Filtering

Noise: created by camera sensors, noisy image is the addition of original image and noise.

Mean (Averaging) Filter: make each pixel in the neighbour the average of all neighbours. Corresponds to a low pass filter, can “smooth-out” noise, but will blur the image. Effective in temporal image sequences. Not effective for impulse noise (occasional extreme values).

Gaussian Filter: weighted averaging filter, higher on the central region of the window, Gaussian equation discretely approximated. Used for canny edge detector, etc.

Median Filter: rank the values inside a neighbourhood, and pick the medium value/values.

Other Rank Order Statistics Filters: rank values inside a neighbourhood, apply values of specific rank/ranks, e.g. max (enhance bright areas), mean (introduce darkness). Used in

image segmentation.

Reflectance and Illumination

Reflectance model: the surface reflectance properties of an object.
Reflectance model describes what happens when light from a certain direction strike a source. Light may be: absorbed, scattered, reflected, transmitted into surface, or combinations of them.
In a reflectance model, light illuminating a surface gives rise to shading, human perceive curved surfaces through their shading patterns.
Points on the surface oriented towards the light source reflect more light than points oriented away from the light source.
Both surface reflectance properties and illumination conditions affect the representation of an object with specific camera parameters in an image.
Distance illumination: illuminated by light sources from very far away (infinite distance), direction is constant within scene, and no attenuation.
Local illumination: light source close to the scene, direction varies within scene, objects closer to the source illuminate more strongly than those further away (attenuation).
Light source attenuation: $f_{att} = 1 / d^2_L$ where d_L is the distance to light source. Physically correct but not realistic. Alternative functions like $1/d$ are used.
Point source is the simplest type of light source, represented by a 3D location (local) or 3D vector (distant). Spotlight is a point source that only emits light in a constrained cone of angles.
Ambient light: light that arrives equally from all directions.
Distributed sources: a source that emits light from an area, regions may be illuminated by part of the source only.
Environment map is a spherical approximation of lights arriving at a point from all directions. Each pixel is like a distant point source with its own direction and colour. Vertical axis is Elevation, horizontal axis is Azimuth.

The Rendering Equation

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_s \rho(x,x',x'')I(x',x'')dx''\right]$$

Intensity passing unoccluded from x' to x.	Visibility function 0 if something in the way. $1 / d^2_L$ otherwise. d^2_L is distance from x to x'.	Intensity emitted from x to x' if x itself is a light source.	^ All points on all surfaces.	How much light is reflected by x' from x'' to x.	Recursion Integral

Global illumination: with interreflection, lights reflect between surfaces before reaching the observer.
Local illumination: local models of reflectance, treat each point separately and only consider illumination coming directly from a light source.
Reflectance model examples:

Specular surface: reflected light is scattered in a narrow range of angles around the angle of reflection.
Diffuse surface: reflected light is scattered in all direction.
Translucent surface: refraction, some light penetrate the surface.

Surface Reflectance Model

$$I = f(\mathbf{n}, \mathbf{s}, \mathbf{v}, \mathcal{P})$$

n (surface **n**ormal), s (**s**ource direction), **v** (**v**iewer direction) are the three unit vectors that describe surface reflectance, while *P* describes material properties of the surface.

Simple Ambient Reflection Model (Unrealistic)
$I = L_a k_a$ where L_a is the strength of source and k_a is the ambient coefficient, $0 < k_a <= 1$, a material property. No n,s,v, so independent of normal, source and viewer directions.
This model ignores the fact that an object may occlude others, reducing ambient light reaching the surface.

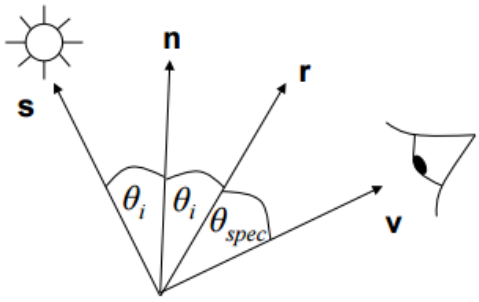
The Lambertian Reflection Model
Based on a perfectly **diffuse** reflector, accounts for changes in density due to changes in surface orientation.
Lambertian model assumes that the surface is macroscopically rough (paper, chalk), and light reflected will be scattered equally in all directions.
The amount of light reflected from an area of the surface is assumed to be completely dependent on the position of surface in relation to the direction of light source.

Lambertian intensity is simply: $I = L_d k_d \cos\theta_i$

Or in terms of unit vectors: $I = L_d k_d (\mathbf{n} \cdot \mathbf{s})$

where θ_i is the angle between surface normal **n** and light source direction **s**.

The Phong Model
Specular reflectance. Ideal for surfaces that are macroscopically smooth and appear shiny.



Imperfect specular reflectors reflect light into a range of directions around **r**, instead of just **r**. Strength of specular reflectance is determined by angle between **r** and **v**.∠**s**-**n** = ∠**n**-**r**; **r**, **s**, **n** in the

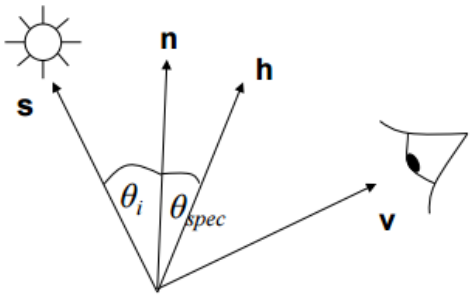
same plane. Therefore **r** = 2 (**n**·**s**) **n** – **s** can be calculated.
When θ_{spec} = 0, maximum reflectance.

$$I = L_s k_s \cos^{\eta} \theta_{spec}$$
$$I = L_s k_s (\mathbf{r} \cdot \mathbf{v})^{\eta}$$

where η is the shininess constant for surface.
This model is viewpoint dependent as **v** is in the equation.

The Blinn-Phong Model
Resolves slowness of the Phong Model due to the need of calculating **r** for every polygon face. It is however, an **approximation**.

Instead of **r**, use vector halfway between viewer and light source:



Since we assume **s** is distant, it is fixed over the image so only need to compute **h** once per image:

$$\mathbf{h} = \frac{\mathbf{s} + \mathbf{v}}{\|\mathbf{s} + \mathbf{v}\|}$$

If **n=h**, we have perfect specular direction

Phong model calculations in the exam most likely refer to this, especially if both **s** and **v** are provided in the question.

The Full Phong Model
The sum of the ambient, diffuse and specular terms, summing over all lights.

$$I = k_a L_a + \sum_{lights} \underbrace{(k_d(\mathbf{n} \cdot \mathbf{s}) L_d}_{\text{Diffuse term}} + \underbrace{k_s(\mathbf{n} \cdot \mathbf{h})^{\eta} L_s)}_{\text{Specular term}}$$

Ambient Term <----------sum over all lights---------->

We can add diffuse and specular images over different light sources together, to produce images illuminated by a combination of two illumination conditions.

Material properties in the reflectance models are wavelength-dependent. Strength of light source is also wavelength-dependent – has a colour. We provide RGB values for them. Often Ambient RGB = diffuse RGB (object colour), Specular RGB = White.

Shape from Shading
Retrieve information (estimate each pixel's surface normal) from shading, however cannot

100% restore. A depth map can be estimated from the normals.
In the previous reflectance models, intensity is determined partly by surface normal direction. In Lambertian model, k_d is the diffuse coefficient, $0 < k_d <= 1$, representing the proportion of light diffusely reflected by the surface, the rest is absorbed by the surface.
To estimate normals, simplifications required. Assume diffuse coefficient = 1 (perfect white diffuse reflector), light source direction known, intensity = 1.
Convex-concave ambiguity: ambiguity of shading, both convex and concave surfaces could give the same shading under specific illumination conditions, causing ambiguity.
After simplification, $\theta_i = \arccos I$. Requested surface normal must lie on a cone above the surface of angle θ_i, whose opening angle is determined by image brightness. Constraints: directions of normals of adjacent pixels must be consistent, should vary slowly with no discontinuities, and convex interpretation is preferred over concave.
The Worthington and Hancock algorithm: initialise normals to on-cone position, then enforce an additional constraint (e.g. smoothing), normals will go off-cone, then rotate normals back to closest on-cone positions; repeat this process until convergence. The initial estimation of normals is lined up with local negative image gradient, in consistency with convexity assumptions.

Photometric Stereo
A more practical alternative to shape-from-shading.
Take multiple images from the same point but with varying illumination, combine partial constraints on the surface normal direction provided by these images. Direction of light sources need to be recorded, so **s** is known.
For example, two light source directions will provide two cones, two lines of interaction give two possible normal directions. If three cones given, then a normal can be determined – will not be exact, due to noise:

$$I = L_d k_d (n_x s_x + n_y s_y + n_z s_z)$$

As intensity I, diffuse coefficient k_d and light source information assumed/known, need to solve a linear equation:

$$I_1 = \bar{s}_{1x}\bar{n}_x + \bar{s}_{1y}\bar{n}_y + \bar{s}_{1z}\bar{n}_z$$

$$I_2 = \bar{s}_{2x}\bar{n}_x + \bar{s}_{2y}\bar{n}_y + \bar{s}_{2z}\bar{n}_z$$

$$I_3 = \bar{s}_{3x}\bar{n}_x + \bar{s}_{3y}\bar{n}_y + \bar{s}_{3z}\bar{n}_z$$

Subscripts are image number, bar means vector scaled by k_d or L_d, I and s known, n unknown. Three unknown variables solved by rearranging and combining equations.

For more than three images, least square method needs to be applied.

Minimise:
$$\|\mathbf{I} - \mathbf{SN}\|^2$$
I, **S** known, **N** (**n** scaled by k_d) unknown
Pseudoinverse of **S**:
$$\mathbf{N} = \mathbf{S}^{+}\mathbf{I} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T\mathbf{I}$$
Separate out **n** and k_d:
$$k_d = \|\mathbf{N}\| \qquad \mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|}$$

Weakness: Lambertian reflectance without specular reflections, require known single point light source direction, with no shadowing. It estimates normals, not full 3D surface, and require multiple images with no motion.
Strength: Simple (linear model, cones), dense shape estimate (normal for every pixel), estimates both albedo and shape, allowing relighting (rendering with different light source directions).

Shadowing
Two ways a point can be in shadow:
Self shadow: a point is facing away from the light source **s**, i.e. normal $\mathbf{n} \cdot \mathbf{s} < 0$ ($\theta_i > 90°$). Determination is fast as only local information (normal, light source direction) required. In Lambertian model, intensity $I = \max(0, \mathbf{n} \cdot \mathbf{s})$, clamping negative values to 0. Adds noise to the solution of previous reflectance models.
Cast shadow: a point blocked by other points(objects) from the light source. Need to consider all other points:
For a surface with a height map, height of pixel (x,y) is z(x,y),

$$\forall d \in \mathbb{R}^+, z(x,y) + ds_z > z(x + ds_x, y + ds_y) \Rightarrow \text{Not in cast shadow}$$

if we move, in the air, from a point on the surface in the direction of the light source, we must always be higher than the surface at that point.
This is more complex for general polygonal surfaces, need to compute equations of each line to light source, then test if these lines intersects with any other polygon in the model by computing line-plane intersection for each plane of other polygons.. Therefore, every polygon in the model needs to be computed.
Cast shadows provide cues for shape estimation.

Flat Shading
Passing geometric, light source and surface property information to a reflectance model to illuminate objects.
Flat shading computes surface normal for each polygon, normal (constant over the polygon) is used to compute a single colour for the polygon, and fill the polygon with this colour.
Also called **constant shading**, good for planar surfaces, bad for curved surfaces.

Interpolation Shading
Lateral inhibition: human eyes perceives edge (change in intensity) strongly, causes **Mach banding**.
Need to smoothly vary the colour of polygons along the object surface. Therefore, it is necessary to compute a surface normal for each vertex:
If the surface is described by a parametric function (e.g. sphere), need to derive an analytical expression for the surface normal from that function.
If the surface is a polygonal mesh, compute one normal per polygon and combine them to find vertex normals.
Face normals: cross product two edges ($\mathbf{e}_1$, $\mathbf{e}_2$) of a flat polygon is the surface normal **n** (but not of unit length). To compute unit vector: $\mathbf{n} = (\mathbf{e}_1 \times \mathbf{e}_2) / |\mathbf{e}_1 \times \mathbf{e}_2|$ (|u| is the square root of the sum of each component squared).
Vertex normals: combination of the surface normals of the facets adjacent to the vertex. In a simple case, average is taken for the four facets surrounding a vector. Most often a weighted average is used, as larger facets contribute more to the vertex normal. Get

weighting for free if we do not normalise the cross products for face normals, and just average the unnormalised normals.
After normals are computed, two methods available to smooth shading:
Gouraud: calculate a colour for each vertex, interpolate **colours** along the polygon edges, then interpolate colours of scan lines passing through these edges.
For points on edges: $I_u = I_a \times D_{bu} / D_{ab} + I_b \times D_{au} / D_{ab}$, where a and b are two vertices forming the edge, u is a point on the edge, D is distance, I is density,
For points on the scan lines: $I_p = I_u \times D_{vp} / D_{uv} + I_v \times D_{up} / D_{uv}$, where u and v are two points the scan line is intersecting edges, p is a point of the scan line.
Drawback: assuming smooth intensity change within a polygon, lower resolution could miss large intensity changes over a small distance.
Phong (not the reflectance model!): interpolate vertex normals along the polygon edges, then interpolate normals along the scan lines passing through these edges. Then use normals to compute colours for each pixel.
At the distance of each pixel, a normal is interpolated on the polygon edges and scan lines, and calculate the colour intensity for each pixel.
Can correctly detect specular highlights in small areas, reduce Mach banding, but expensive to compute.
Drawbacks of interpolation shading: polygons still flat, straight lines may persist; shading carried out after transformation, may cause non-linear transformations that produce odd effects if perspective transformations are applied. Can be reduced by using smaller polygons (higher resolution), but Mach banding may still be available due to drastic edge changes.

Drawing Lines
Pixel: defined in a rectangle axis system, each pixel is defined as a coordinate in the system, either of its centre or bottom-left corner.
Line: defined by equation $y = mx + c$, m is slope and c is y-axis intersection. To draw a line, vary x and compute y at each x location.
Rasterization: converting drawing of the line to pixels in image, also called scan conversion.
Digital Differential Analyzer (DDA) Algorithm: compute gradient $m = (y_1-y_0)/(x_1-x_0)$, consider a point (x,y) on the line, then consider (x+1,y+m) when drawing to right, and vice versa. Round each y value to closest y values in the pixel coordinate system. Drawbacks: floats and rounding operations expensive.
Bresenham's Algorithm: only need to round the start and end points of a line segment. First round the start point to a pixel, then according to the gradient and drawing direction, the next pixel can only lie in one of two pixels (north-east or east for positive gradient and drawing to the right, etc), converting the problem to a decision problem.
$y = mx + c \rightarrow mx - y + c = 0 \rightarrow (\Delta y/\Delta x)x - y + c = 0$
$\rightarrow F(x,y) = x\Delta y - y\Delta x + c\,\Delta x = x\Delta y - y\Delta x + B = 0$
Therefore, $F(x,y) < 0$, $mx_1+c<y_1 \rightarrow$ point above the line, closer to $(X_0 + 1, y)$;
$F(x,y) > 0$, $mx_1+c>y_1 \rightarrow$ point below the line, closer to $(X_0 + 1, y_0 + 1)$;
To make a decision, for the next pixel to (x_p,y_p), test $F(x_p+1,y_p+(y\text{-height}/2))$ – mid point between this and next graph point, if smaller or equal to 0, draw the next pixel with the same y value; otherwise, draw the next pixel with y displacement of 1 by the direction defined by gradient. The next test is dependent on the previous test, if no displacement was chosen, the next test is $F(x_p+1, y_p + \Delta y)$; otherwise the next test is $F(x_p+1, y_p + (\Delta y-$

Δx)).

Edge Detection

Convert an image into a set of curves representing significant image transitions (edges). Edges can be caused by object boundaries, discontinuities in depth, illumination and shading, shadow and texture variations, as well as noise.

Edge detection:

Take the gradient of image values , take derivatives to observe variations.

Gradient: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Vertical edge (change horizontally): $$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

Horizontal edge (change vertically): $$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

Combine both: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Edge orientation: $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Edge strength: $$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

To construct a gradient image, take vertical and horizontal gradient, apply thresholding, and convolute with a gradient operator (Sobel, Roberts, Prewitt, Isotropic, etc).

Laplacian operator: edge detection operator looking for edges at zero crossings with second derivatives of the image gradient. Mask used. Used for detection of edge location and peaks.

Results of edge detection: edges usually fragmented, affected by noise and illumination, shadows; and multiple misplaced edges may be detected for the same transition. Can be improved by image filtering.

Canny edge detector:

Criteria for good edge detection: good detection of edge points (reduce noise ratio), good localisation of detected edges, unique response from each transition.

Stages: blur the image with a Gaussian filter to remove noise and small features; apply four filters to detect horizon, vertical and two diagonal edges in the blurred image; calculate image gradient; trace edge continuity with two thresholds:

1. apply a higher threshold for segmenting clearly-separate edge segments
2. connect the edge segments in regions where the gradient exceeds a second, lower threshold (they are likely continuous).

Global Illumination

Model all illumination arriving at a point, including light which has been reflected or refracted by other objects.

Global interactions (photon mapping): start with the light source, follow every possible ray-of-light as it travels through the environment, stop when light hits observing point, travels

out of the environment or energy below the admissible limit due to absorption in objects. Very expensive to compute, cannot be used in real time but used for offline rendering.
Types of Interaction: diffuse-diffuse(radiosity), specular-defuse, diffuse-specular, specular-specular(ray-tracing).

Whitted (Reverse) Ray Tracing
Trace light rays in the reverse direction of propagation, from the observer(eye) to the light source, through the scene. Calculate recursively by spawning reflection rays and refraction rays for every hit point.
Use specular model in global illumination, along with a local model. Considers diffuse-specular interactions but not diffuse-diffuse interactions (would be very expensive).
At each hit point, spawn reflection and refraction rays, forming a recursion tree. Terminate when spawned rays miss all objects or limits reached. {-- **recursion** --}

$$I = k_a L_a + \sum_{i=1}^{m} f_{\mathrm{att}_i} L_i \left[(k_d(\mathbf{n} \cdot \mathbf{s}_i) + k_s(\mathbf{n} \cdot \mathbf{h}_i)^{\eta}) \right] + k_r I_r + k_t I_t$$

Reflected light
Ambient term
Diffuse term
Specular term
Transmitted light
Standard Blinn-Phong model
(+ attenuation/shadow term)

where transmitted light is refracted light.
Disadvantages of ray tracing: shadow rays usually not refracted, false shadows due to numerical precision, overly simplistic ambient light model, very expensive computation.

Image-based Illumination
A less computationally expensive alternative to ray-tracing.
Each pixel (x,y) in an environment map *I*(x,y) corresponds to a light source with direction **s**(x,y) and colour I(x,y). To render with an environment map, sum over light sources like in Phong Model.

Convert (x, y) coordinates into angles:

$$\theta(x) = \pi \left(\frac{2x}{x_{\max}} - 1 \right)$$

$$\phi(y) = \frac{y\pi}{y_{\max}}$$

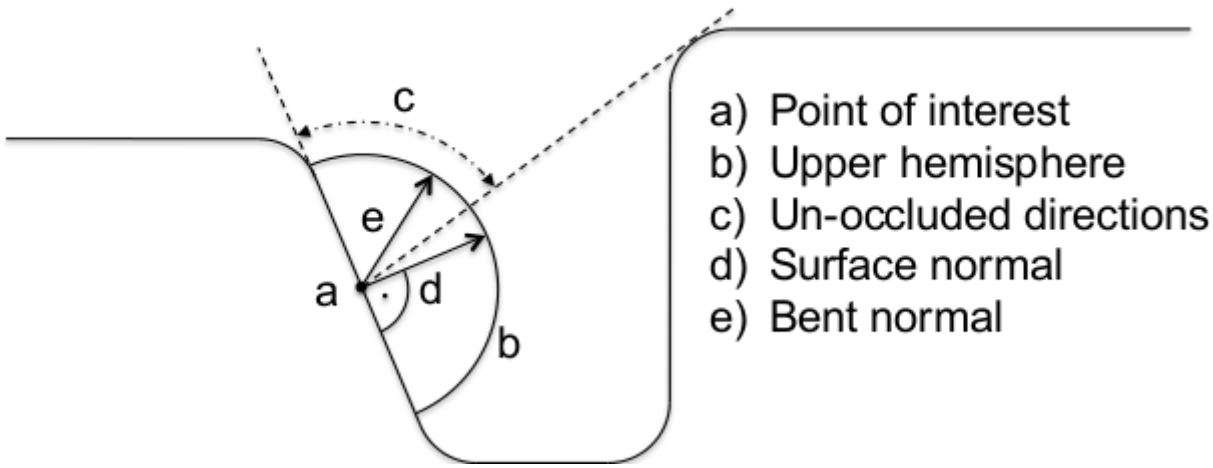
Convert angles into a direction:

$$\mathbf{s}(x, y) = \begin{bmatrix} \sin(\phi(y)) \sin(\theta(x)) \\ \cos(\phi(y)) \\ \sin(\phi(y)) \cos(\theta(x)) \end{bmatrix}$$

Environment/Reflection Mapping
Computationally cheap approximation for appearance of specular (mirrorly) objects under

environment illumination.
Reflect viewing direction about surface normal, give specular lighting direction.
s = 2(**n**·**v**)**n** - **v** , use this vector to lookup colour in environment map, copy to pixels. Environment map is stored as texture on reference object, such as sphere or cube.
Possible in real time.

Occlusions
A point on an object may not see all of the environment, for example, it may be shadowed from certain sources. Self-shadowing can exclude half of the surrounding area.
Ambient occlusion: pre-compute the occluded proportion of the hemisphere above the surface, centred at the object, and apply environment lighting.
Ambient occlusion = 1: fully unoccluded, use all sources in the whole hemisphere.
Ambient occlusion = 0: fully occluded, no light reaches that point.
Bent normal: the average unoccluded direction. Describes the dominant direction from which light arrives , can be unrepresentative when point blocked in the middle from the light source, but receive similar level of illumination on the two sides of the block.



To use ambient occlusion with environment lighting, first render surface ignoring occlusions, then down-weight resulting intensities by ambient occlusion amount, and use bent normal instead of surface normal.