Chapter 1.0 – Sampling and Quantization

Sampling – *Digitizing the coordinate values in the image being digitized.*
Quantization – *Digitizing the amplitude values (grey levels) in the image.*

- **Figure 1.0.A** is the continuous image being digitized.
- **Figure 1.0.B** is a result of sampling and plotting the amplitude values for one line across the image (shown by the A - B line in **Figure 1.0.A**).
- **Figure 1.0.C** shows the process of quantizing the plot of amplitudes. A series of points from the continuous sample are used, from which their amplitudes are rounded to the nearest discrete grey value.
- **Figure 1.0.D** shows the digital scan line of the image.

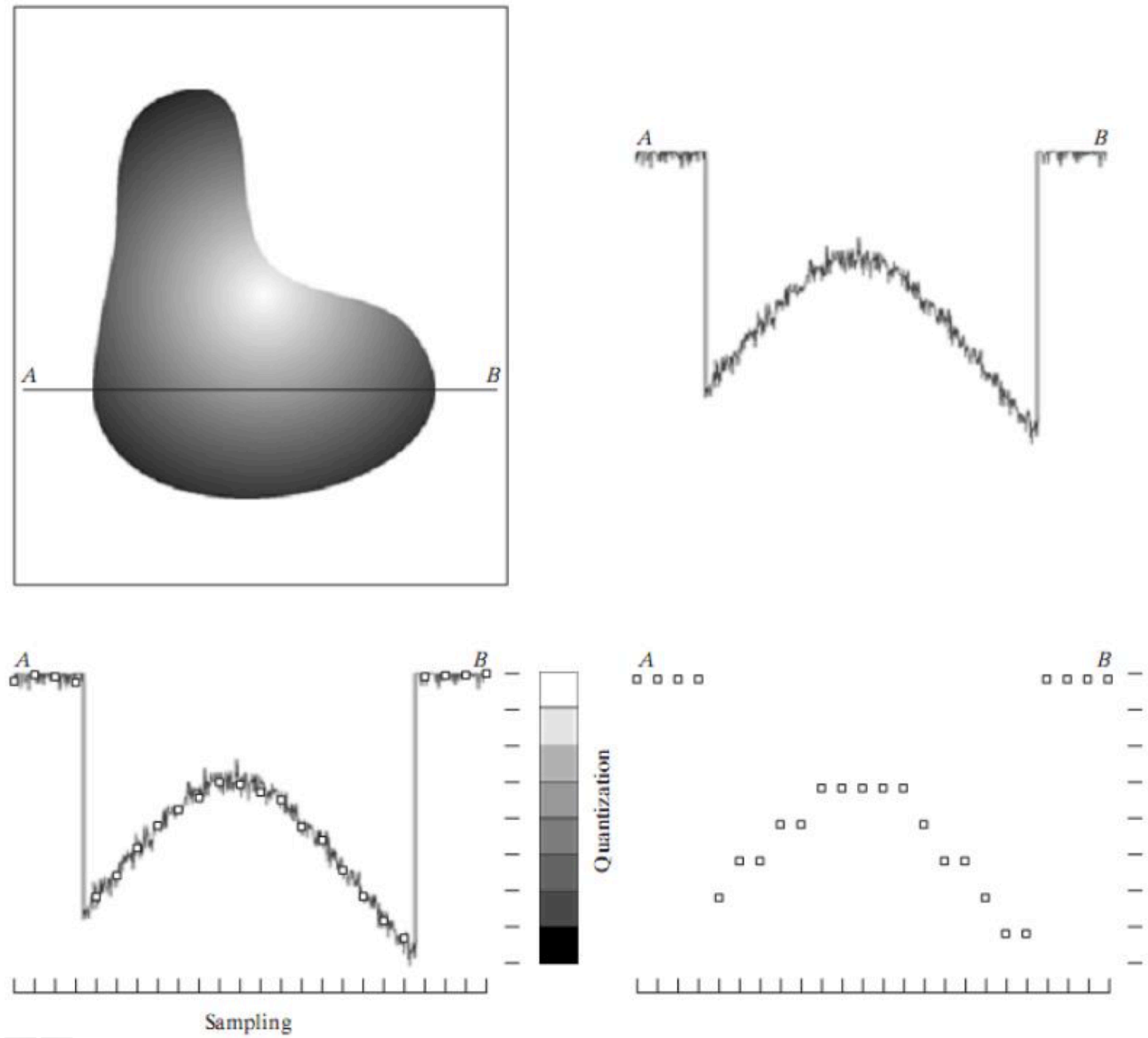


Figure 1.0 - Sampling and Quantization Process

The number of squares in **Figure 1.1** illustrate the sampling limits of the sensors used in both the horizontal and vertical directions and how a continuous image is represented in a matrix. **Figure 1.1.A** shows the continuous image projected onto the plane of an array senor. **Figure 1.1.B** shows the image after sampling and quantization.

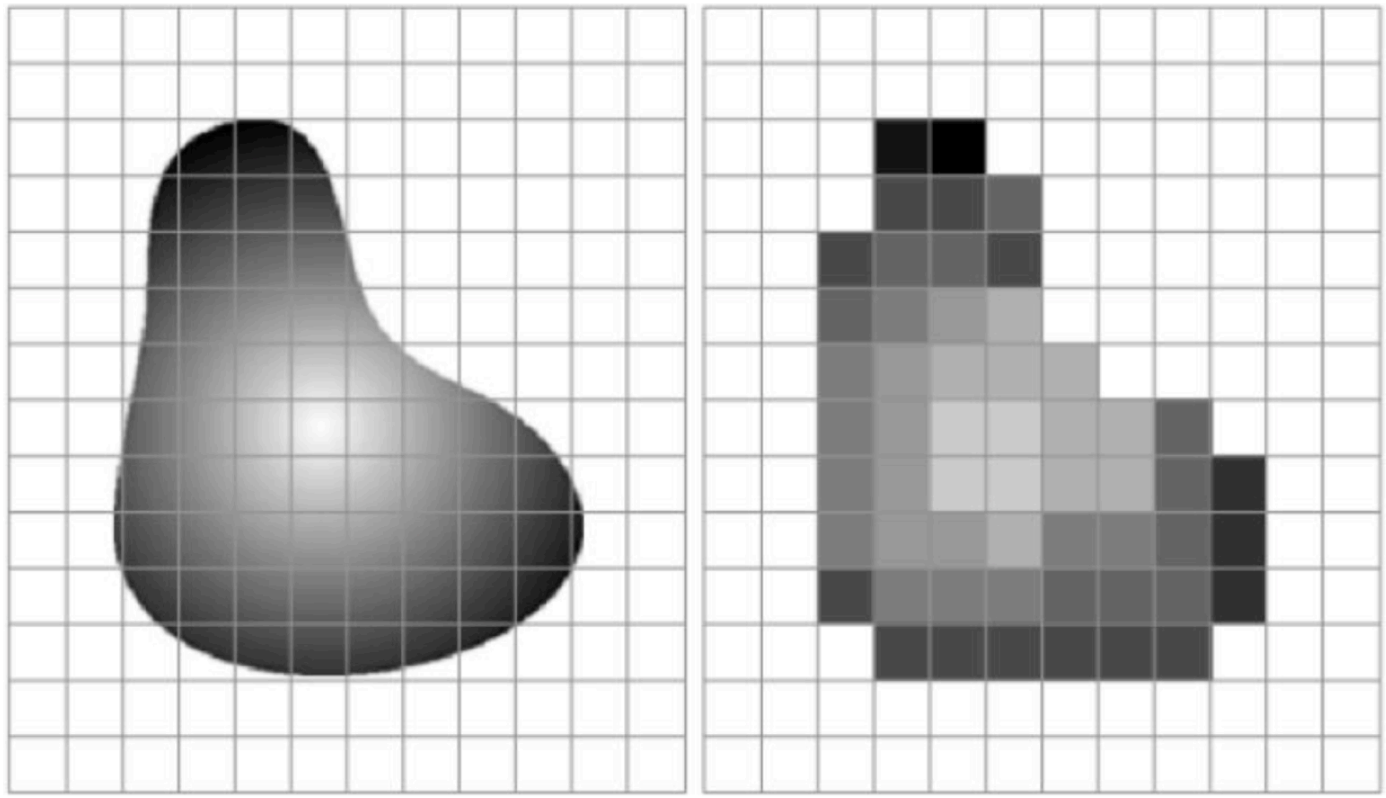


Figure 1.1 - A. Continuous Image | B. Digitized Image

Dynamic Range – *The range of values spanned by the greyscale in an image.*

Chapter 2.0 – Eye Physiology

The retina contains two classes of discrete light receptors, cones and rods. Cones are mostly located in the central part of the retina called the fovea and are highly sensitive to colour. Cones are sensitive to bright light and do not operate in dim light. Rods are distributed across the surface of the retina and are sensitive to light intensity but not colour.

- ***Cones –*** *The part of the retina sensitive to colour.*
- ***Rods –*** *The part of the retina sensitive to light intensity but not colour.*
- Humans can view an intensity range of 10^{10} achieved by sensitivity adaptation.
- Humans have a logarithmic sensitivity to light, meaning that a brightness increase of 20w to 22w is perceived to be the same as 100w to 110w (ratio of 1.1).

Chapter 3.0 – Gamma Correction

- Gamma correction is used to make images lighter or darker by adding a constant intensity to each pixel.
- All monitors have a response curve which is how various pixel intensities are displayed **Figure 3.1.A**.
- Monitor response curve formula: $I = aV^{\gamma}$
- RGB = (64, 64, 64) should be half the intensity of (128, 128, 128), but under PAL (Phase Alternating Line) is not.
- Gamma Correction Method:
  - $V = (I/a)^{1/\gamma}$
  - V, new pixel intensity
  - I, original pixel intensity
  - a, has a value of 1
  - γ, gamma value typically in the range of 0 to 10
  - A look-up table can be used to increase efficiency.
- A gamma value which is less than 1 will make the image darker (encoding gamma).
- A gamma value which is more than 1 will make the image lighter (decoding gamma).
- Gamma correction is a non-linear transformation function.

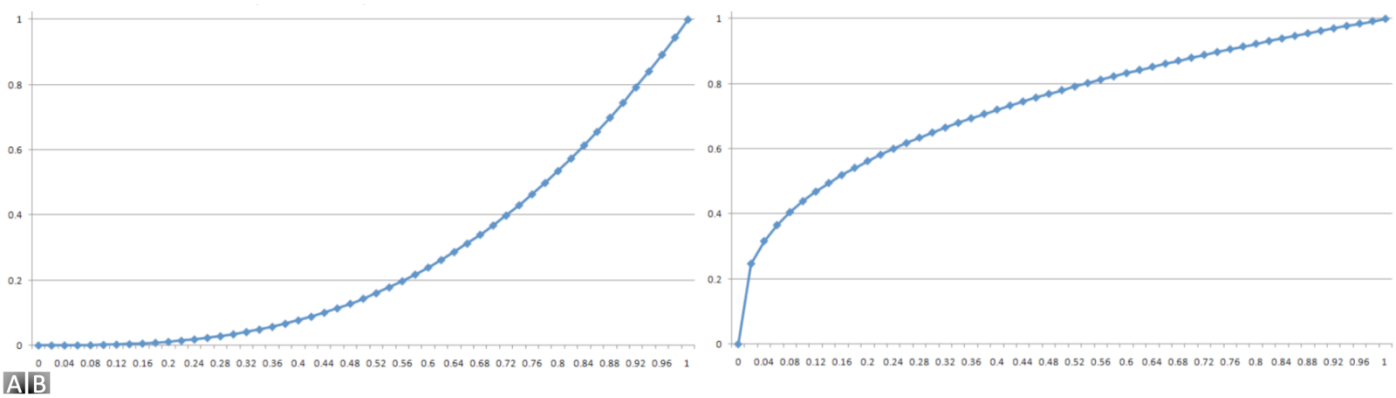


Figure 3.1 - A. Computer Response Curve | B. Gamma Correction Curve

Figure 3.1.A illustrates that computer monitors typically display lower intensities more than higher intensities. When the gamma correction curve is applied to the monitor response curve the response is linear (straight line) meaning that each pixel intensity is displayed correctly for its value.

Chapter 4.0 – Contrast Stretching

Contrast – *The difference in luminance or colour that makes an object distinguishable.*

- Contrast stretching is used to increase the dynamic range of an image by increasing contrast in some intervals of the domain and decreasing others.
- Contrast Stretching is a Piecewise Linear Transformation function (a function whose graph consists of straight-line segments).
- Levels below r1 and above r2 are compressed (solid line in **Figure 4.1**) to allow more levels between r1 and r2 (dashed line in **Figure 4.1**).
- Contrast Stretching Method:
  - $$out = \begin{cases} \frac{s1}{r1} \times in, & if\ in < r1 \\ \frac{(s2-s1)}{(r2-r1)} \times (in - r1) + s1, & if\ r1 \leq in \leq r2 \\ \frac{(255-s2)}{(255-r2)} \times (in - r2) + s2, & if\ in \geq r2 \end{cases}$$
  - out, new pixel intensity
  - in, original pixel intensity
  - $r1$, first x-axis coordinate
  - $s1$, first y-axis coordinate
  - $r2$, second x-axis coordinate
  - $s2$, second y-axis coordinate
  - A look-up table can be used to increase efficiency.

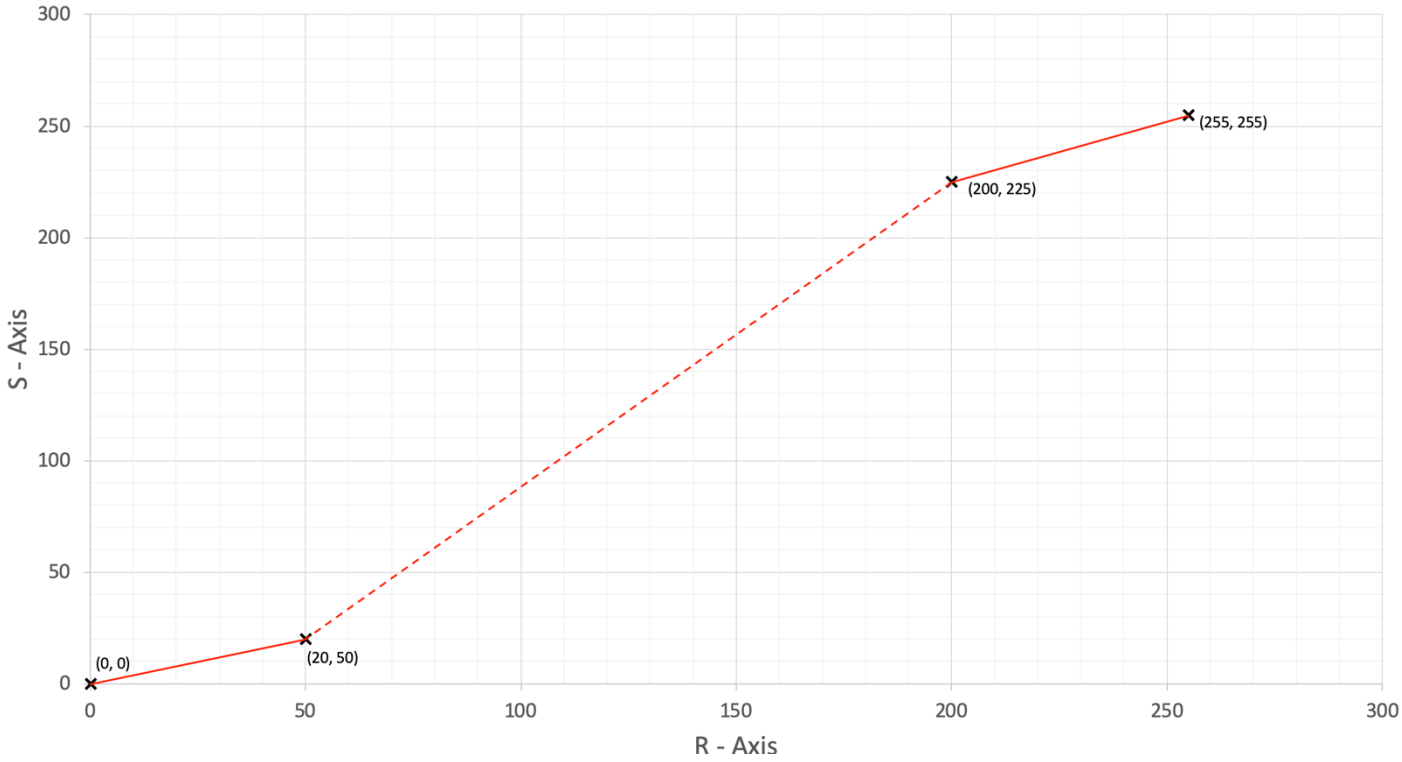


Figure 4.1 - Contrast Stretching User Inputted Coordinates Graph

Chapter 5.0 – Histogram Equalisation

- A histogram is a graph representing the number of pixels at each intensity level in an image.
- Histogram equalisation is used to improve the contrast of an image by spreading the distribution of intensities evenly over the pixels in the image.
- Histogram Equalisation Method:
  - A histogram graph is produced for the original image.
  - A cumulative distribution graph is produced for all intensities.
    - $t(i) = histogram[i] + t(i-1)$ Illustrated below:

    $t[0] = histogram[0]$
    $t[1] = histogram[1] + histogram[0] = histogram[1] + t[0]$
    $t[2] = histogram[2] + histogram[1] + histogram[0] = histogram[2] + t[1]$
  - $F(i) = \max\left\{0, \text{round}\left(\frac{g_{levels} \times t(i)}{(h \times w) - 1}\right)\right\}$
  - $F(i)$, mapping i to the new equalised value
  - g_{levels}, number of grey values
  - h, height of the image
  - w, width of the image
  - $t(i)$, cumulative distribution for intensity i
- The formulas above are using a greyscale histogram and should be used on a greyscale image.
- If the greyscale histogram is used to equalise a coloured image there will be areas of the image with vastly varying intensities.
- To perform histogram equalisation on a coloured image, the image data should be interpreted in the HSB (Hue Saturation Brightness) colour format, where the Brightness is used as the histogram. The brightness is equalised, and the Hue and Saturation remain the same.

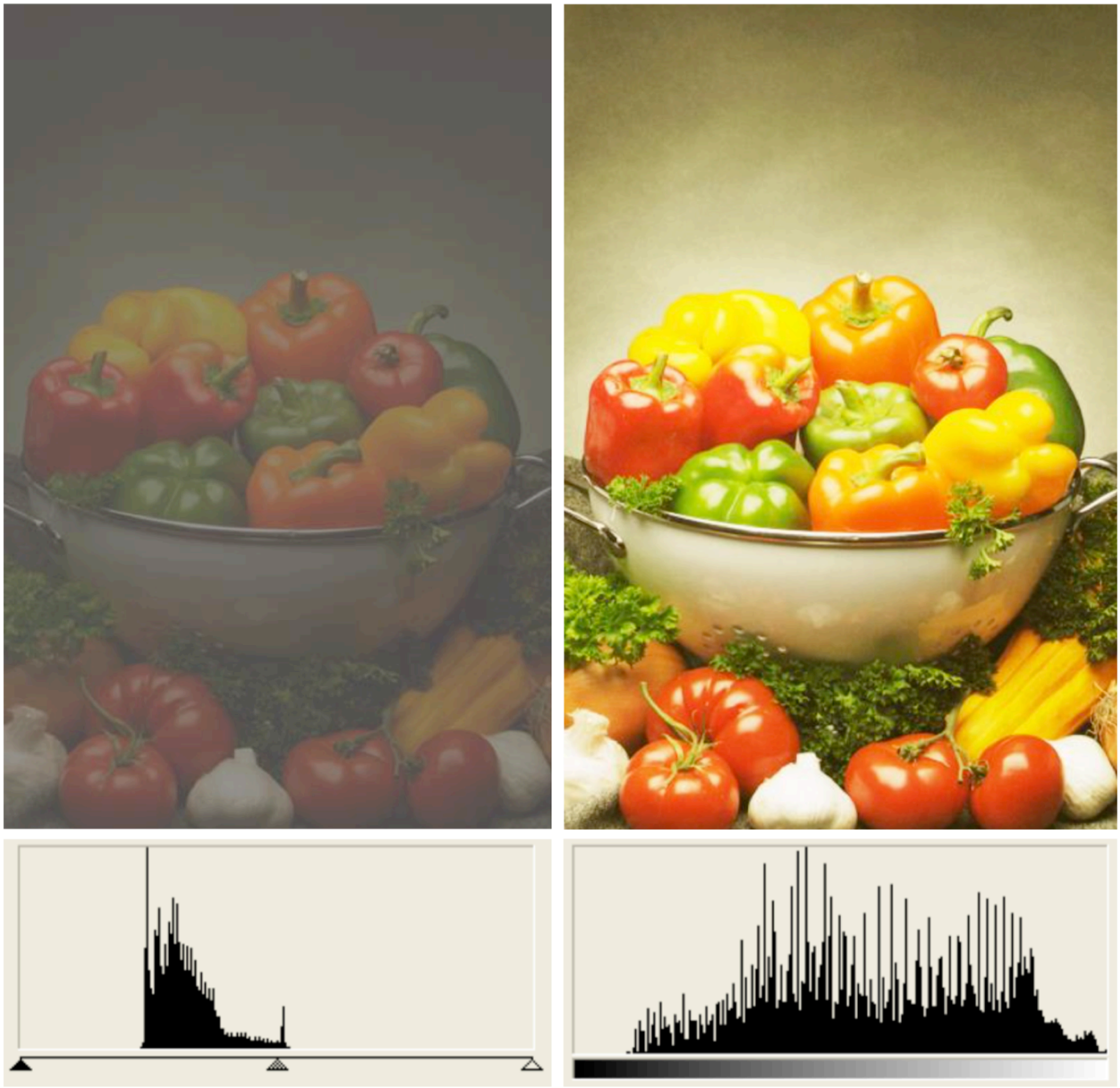


Figure 5.0 - A. Before Histogram Equalisation | B. After Histogram Equalisation

Chapter 6.0 – Cross Correlation

Spatial Linear Filtering – *Modifying an image by replacing the value at each pixel with some linear function of the values of nearby pixels.*

Image Noise – *Random variations of brightness or colour information in an image typically produced during transmission or the sensors in the digital camera.*

- Cross Correlation can be used for:
  - Noise reduction in an image typically useful after poor transmission, X-Rays, etc.
  - Edge detection, e.g., digitally extracting a brain for display.
- Cross Correlation Method:
  - For a greyscale image the image should be translated into a matrix of pixel intensities.
  - Apply a filter kernel to each pixel in the matrix which the filter can be centred over. Then calculate the sum of the products.
  - There will be a border left around the new image due to pixels which cannot have the kernel filter centred over them.
    - 1 pixel border for a 3 x 3 kernel filter.
    - 2 pixel border for a 5 x 5 kernel filter.
    - 3 pixel border for a 7 x 7 kernel filter.

$$\begin{bmatrix} 100 & 110 & 120 & 130 & 140 \\ 120 & 120 & 130 & 130 & 140 \\ 130 & 140 & 150 & 150 & 160 \\ 140 & 150 & 150 & 160 & 170 \\ 150 & 160 & 170 & 180 & 190 \end{bmatrix} \quad \begin{bmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{bmatrix} \quad \begin{bmatrix} \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \\ \square & \square & \square & \square & \square \end{bmatrix}$$

$$\begin{bmatrix} 100 & 110 & 120 & 130 & 140 \\ 120 & 120 & 130 & 130 & 140 \\ 130 & 140 & 150 & 150 & 160 \\ 140 & 150 & 150 & 160 & 170 \\ 150 & 160 & 170 & 180 & 190 \end{bmatrix} \quad \begin{bmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \square & \square & \square & 0 \\ 0 & \square & \square & \square & 0 \\ 0 & \square & \square & \square & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 100 & 110 & 120 & 130 & 140 \\ 120 & 120 & 130 & 130 & 140 \\ 130 & 140 & 150 & 150 & 160 \\ 140 & 150 & 150 & 160 & 170 \\ 150 & 160 & 170 & 180 & 190 \end{bmatrix} \quad \begin{bmatrix} 1 & 3 & 1 \\ 3 & 9 & 3 \\ 1 & 3 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3{,}080 & \square & \square & 0 \\ 0 & \square & \square & \square & 0 \\ 0 & \square & \square & \square & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(100 \times 1) + (110 \times 3) + (120 \times 1) + (120 \times 3) + (120 \times 9) + (130 \times 3) + (130 \times 1) + (140 \times 3) + (150 \times 1)$$
$$\therefore$$
$$100 + 330 + 120 + 360 + 1{,}080 + 390 + 130 + 420 + 150 = 3{,}080$$

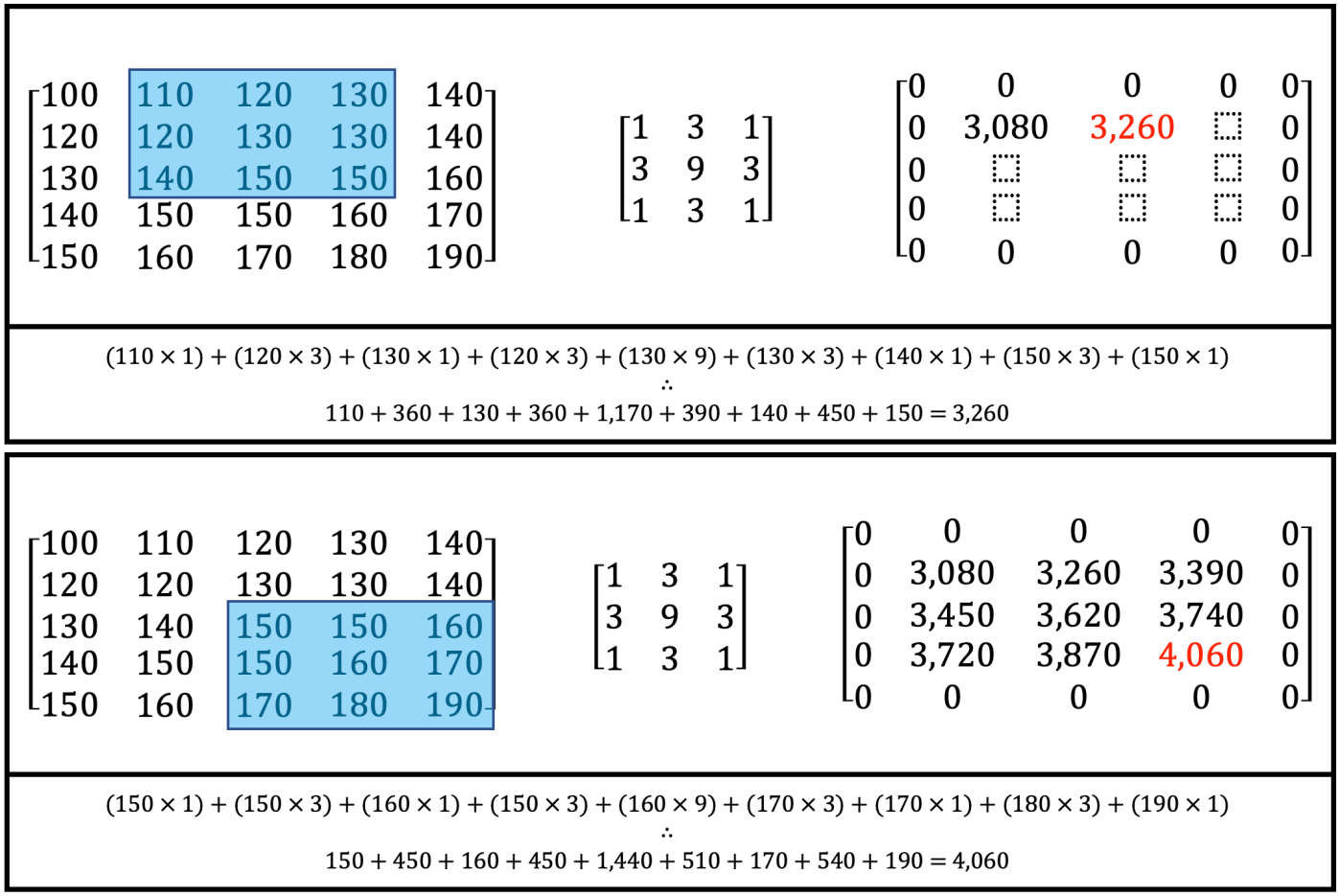


Figure 6.1 - Applying a Kernel Filter to an Image (A, B, C, D, E in order)

- **Figure 6.1.E** is the final result of applying the kernel filter. There will be a black border around the image but there are other ways of dealing with this, such as removing the border pixels from the new image, using the old pixel values, etc.
- We must now normalise the new image in order to return the values to the range of 0 to 255.
- Normalisation Equation:
  - $I' = \left(\frac{(I - min) \times 255}{\max - min}\right)$
  - I', new pixel intensity
  - I, original pixel intensity
  - min, min value out of all matrixes which have had the kernel filter applied to them.
  - max, max value out of all matrixes which have had the kernel filter applied to them.
  - If a negative value is returned by the normalisation equation, make it positive.

Figure 6.2 - Applying the Normalisation Equation to the Computed Matrix

−1	0	1	1	2	1	1	3	1	1	1	1	−1	−1	−1
−2	0	2	0	0	0	3	9	3	1	1	1	−1	8	−1
−1	0	1	−1	−2	−1	1	3	1	1	1	1	−1	−1	−1

Sobel Edge Detector X-Axis | Sobel Edge Detector Y-Axis | Gaussian Blur | Low Pass | High Pass

Figure 6.3 - Examples of 3 x 3 Filter Kernels

- The aim of *low-pass* filters is to average out the local differences in pixels in order to reduce *noise*.
- *High-pass* filters are designed to accentuate (make more prominent) small scale structures such as edges.
- *Sobel Edge Detector X-Axis* is used for detecting vertical edges.
- *Sobel Edge Detector Y-Axis* is used for detecting horizontal edges.

Thresholding – *Setting all grey levels below a certain level to zero; or above a certain level to a maximum brightness value. One use of thresholding is in optical character recognition i.e., scanning in text.*

Chapter 7.0 – Edge Detection

Edges – *An area of an image where the rate of change of intensity is high.*
Derivative – *The amount that a function is changing at a given point.*

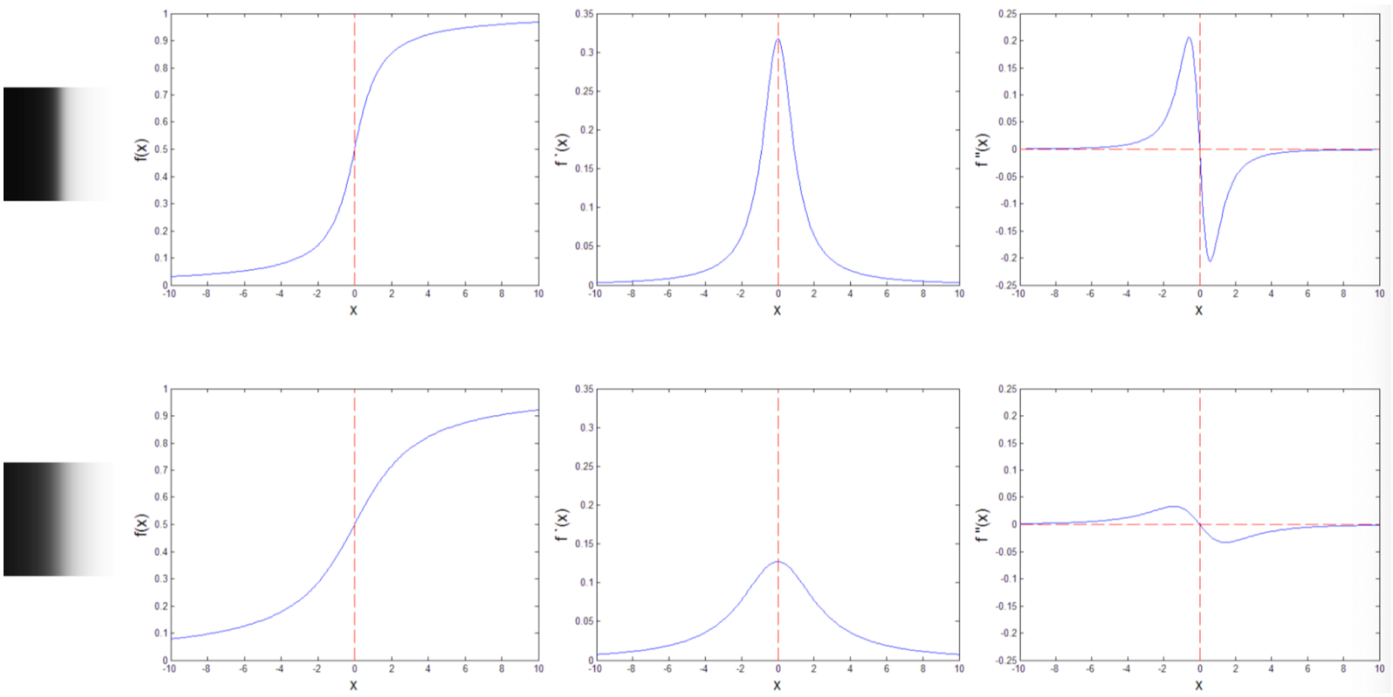


Figure 7.1 - Edge Detection Derivatives

- The point where the rate of change of intensity is highest is when the function is at its maximum value.
- The maximum of a function can be found where its derivative equals zero. Therefore, the second derivative has a maximum where the function crosses from positive to negative.
- The Prewitt operator is sensitive to noise, so it is combined with a Gaussian smoothing filter to give the Sobel operator.

$$M_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad M_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure 7.2 - A. Prewitt Operator X | B. Prewitt Operator Y | C. Sobel Operator X | D. Sobel Operator Y

Zero Crossings – *Pixels that are above zero which have a neighbour below zero (this indicates an edge).*

- Cross correlation (Prewitt / Sobel) should be used to find the 1st derivative.
- Cross correlation (2nd order operator) should be used to find the 2nd derivative, then look for zero crossings.

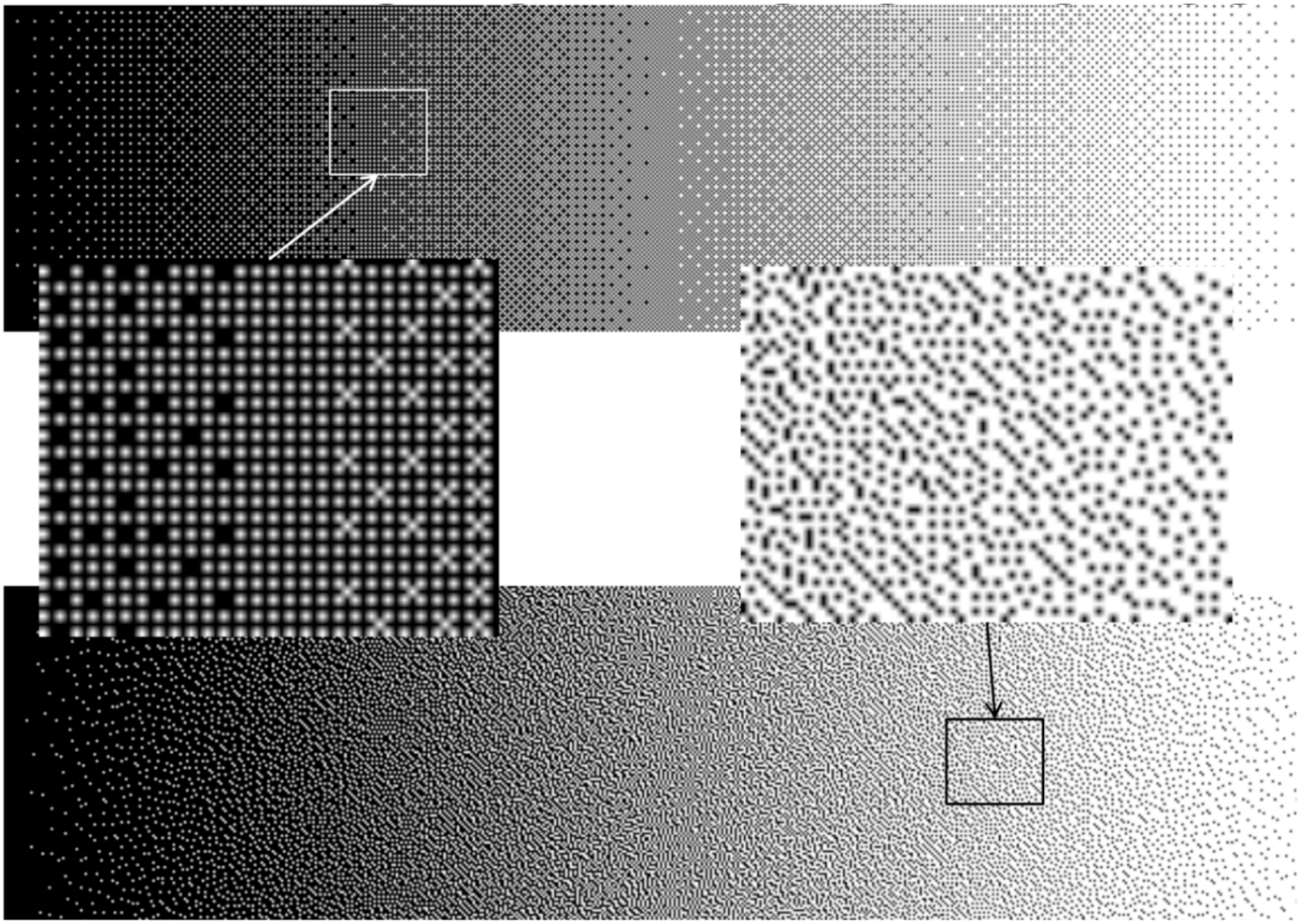
Chapter 8.0 – Dithering

Dithering – *The process of intentionally applying a form of noise, used to randomize quantization error, preventing large scale patterns such as colour banding in images.*

Colour Banding – *A problem of inaccurate colour representation in an image which makes the colours in a gradient look grouped together.*

Figure 8.1 - A. Severe Colour Banding | B. Normal Gradient

8.1 Dithering Methods



A
B

Figure 8.1.1 - A. Pattern Dithering | B. Floyd-Steinberg Dithering

8.2 Calculating Error

100	100	120	140
110	110	130	150
120	150	170	200
140	170	200	250

0	0	0	255
0	0	255	255
0	255	255	255
255	255	255	255

Figure 8.2.1 - A. Original Grey Image | B. Threshold Image

- **Figure 8.2.1.A:**
  - $Total\ Intensity = 100 + 100 + 120 + \cdots 250 = 2360$
  - $Average = \frac{2360}{16} = 147.5$
- **Figure 8.2.1.B:**
  - $Total\ Intensity = 0 + 0 + 0 + 255 + \cdots + 255 = 2550$
  - $Average = \frac{2550}{16} = 159.375$
- $Error\ (Total) = 2550 - 2360 = 190$
- $Error\ (Average) = 159.375 - 147.5 = 11.875$

8.3 Error Diffusion Dithering

The error from one pixel is added onto the next pixel's original intensity ready for thresholding and then error calculation; from which the new error value will then be passed on again.

Figure 8.3.1 – Error Diffusion Dithering Calculation Example; Threshold = 128

- **Figure 8.3.1:**
  - $Total\ Intensity = 0 + 255 + 0 + 255 + \cdots + 255 = 2295$
  - $Average = \frac{2295}{16} = 143.4375$
- $Error\ (Total) = 2295 - 2360 = -65$
- $Error\ (Average) = 143.4375 - 147.5 = -4.0625$

8.4 Floyd-Steinberg Error Diffusion Dithering

Floyd-Steinberg Error Diffusion Dithering – *Instead of passing 100% of the error to the next pixel, distribute it to the 4 un-displayed neighbours of the current pixel.*

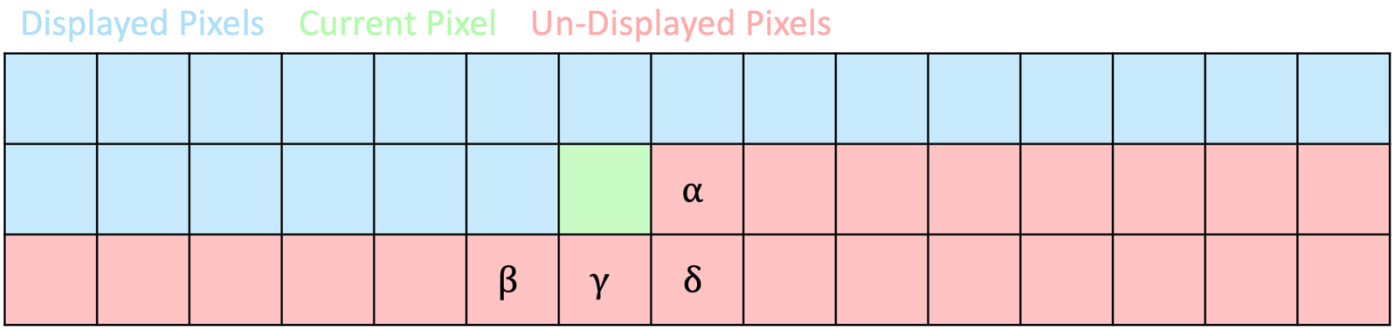


Figure 8.4.1 - Floyd-Steinberg Error Diffusion Dithering Pixel Diagram

- $\alpha = \frac{7}{16}$
- $\beta = \frac{3}{16}$
- $\gamma = \frac{5}{16}$
- $\delta = \frac{1}{16}$

Threshold Value	0	255	0	255
New Pixel Value	100	143.75	71.33	171.21
Calculating New Pixel Value	100 – 0	100 + 43.75	120 – 48.67	140 + 31.21
$\alpha = 7/16$	43.75	-48.67	31.21	~~-36.66~~
$\beta = 3/16$	~~18.75~~	-20.86	13.37	-15.71
$\gamma = 5/16$	31.25	-34.77	22.29	-26.18
$\delta = 1/16$	6.25	-6.95	4.46	~~-5.24~~

Threshold Value	0	255	0	255
New Pixel Value	120.39	147.52	82.61	164.42
Calculating New Pixel Value	110 + 31.25 – 20.86	110 + 6.25 – 34.77 + 13.37 + 52.67	130 – 6.95 + 22.29 – 15.71 – 47.02	150 + 4.46 – 26.18 + 36.14
$\alpha = 7/16$	52.67	-47.02	36.14	~~-39.63~~
$\beta = 3/16$	~~22.57~~	-20.15	15.49	-16.98
$\gamma = 5/16$	37.62	-33.59	25.82	-28.31
$\delta = 1/16$	7.52	-6.72	5.16	~~-5.66~~

Threshold Value	255	0	255	255
New Pixel Value	137.47	88	210.62	157.44
Calculating New Pixel Value	120 + 37.62 – 20.15	150 + 7.52 – 33.59 + 15.49 – 51.42	170 – 6.72 + 25.82 – 16.98 + 38.5	200 + 5.16 – 28.31 – 19.41
$\alpha = 7/16$	-51.42	38.5	-19.41	~~-42.68~~
$\beta = 3/16$	~~-22.04~~	16.5	-8.32	-18.29
$\gamma = 5/16$	-36.73	27.5	-13.87	-30.49
$\delta = 1/16$	-7.35	5.5	-2.77	-6.09

Threshold Value	0	255	255	255
New Pixel Value	119.77	234.23	164.26	177.04
Calculating New Pixel Value	140 – 36.73 + 16.5	170 – 7.35 + 27.5 – 8.32 + 52.40	200 + 5.5 – 13.87 – 18.29 – 9.08	250 – 2.77 – 30.49 – 39.70
$\alpha = 7/16$	52.40	-9.08	-39.70	~~-34.11~~
$\beta = 3/16$	~~22.46~~	~~-3.89~~	~~-17.01~~	~~-14.62~~
$\gamma = 5/16$	~~37.43~~	~~-6.49~~	~~-28.36~~	~~-24.36~~
$\delta = 1/16$	~~7.49~~	~~-1.30~~	~~-5.67~~	~~-4.87~~

Figure 8.4.2 – Floyd-Steinberg Error Diffusion Dithering Calculation Threshold = 128

- **Figure 8.4.2:**
  - $Total\ Intensity = 0 + 255 + 0 + 255 + \cdots + 255 = 2550$
  - $Average = \frac{2550}{16} = 159.375$
- $Error\ (Total) = 2550 - 2360 = 190$
- $Error\ (Average) = 159.375 - 147.5 = 11.875$

8.5 Halftoning

Halftoning – *A process that simulates shades of grey (gradients) by using varying sizes / patterns of black and white pixels.*

- Black and white images produced using differently sized black circles.
- The diameter of a circle is proportional to the darkness of the region.
- $n \times n$ pixels can represent $n^2 + 1$ intensity levels.
- The patterns must form a growth sequence so that any pixel on for intensity level j is also on for all levels k > j.
- One advantage of halftoning is that the dynamic range is reduced which means that fewer colours are used but the impression of a continuous tone is still achieved.

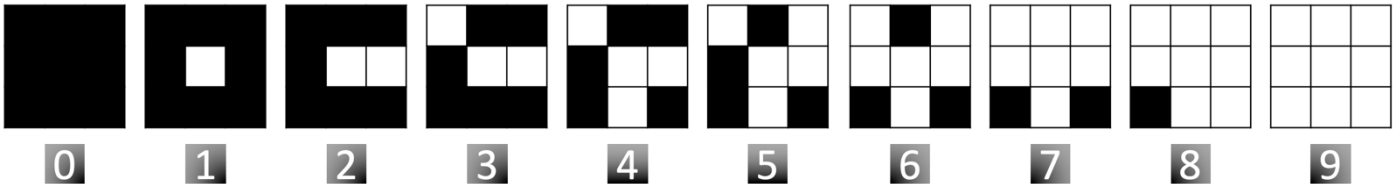


Figure 8.5.1 – Halftoning Patterns; 3x3 Displays 10 Intensity Levels Distributed from Black to White

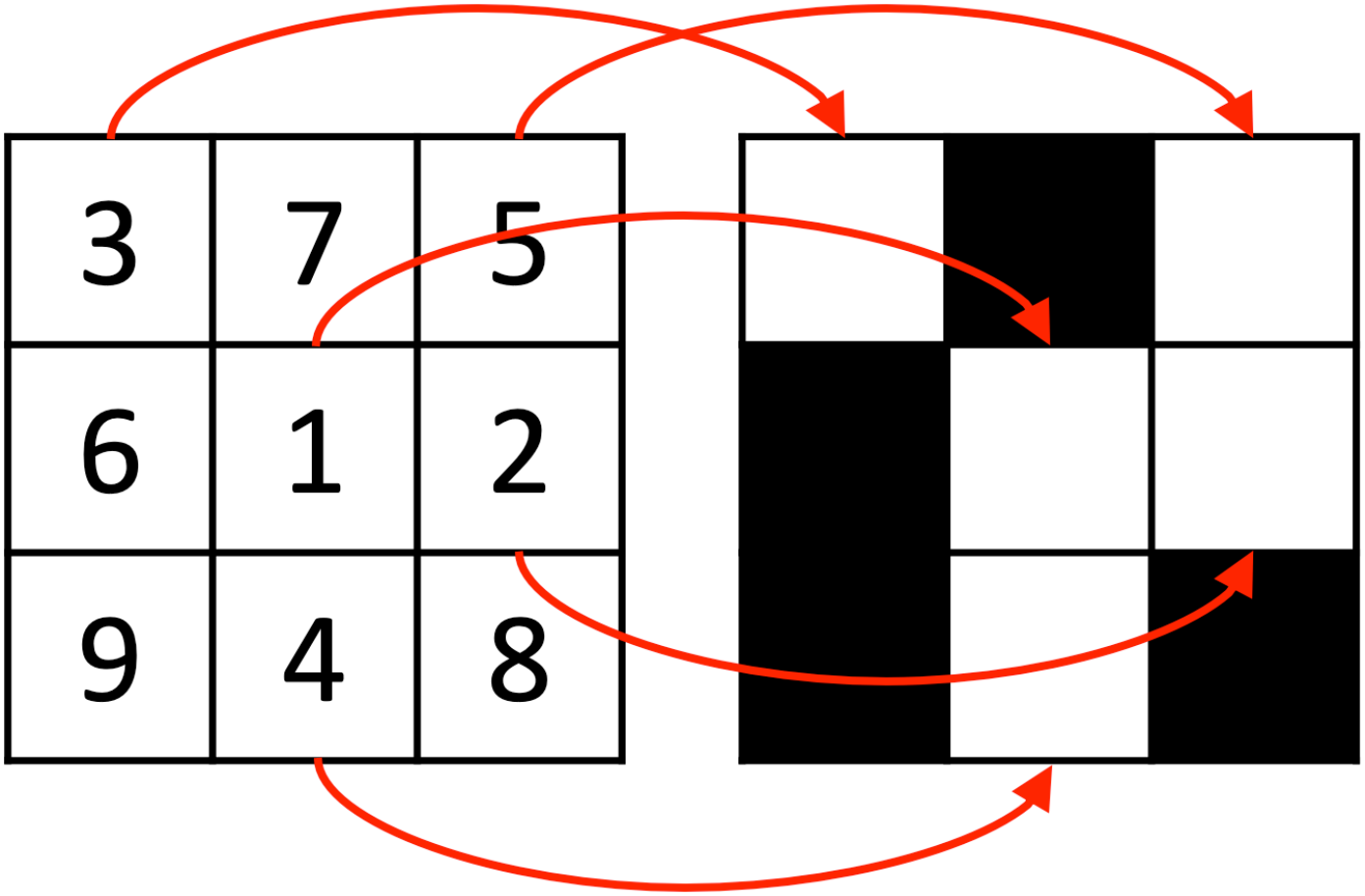


Figure 8.5.2 - Intensity Level Five Number Filter Matched with Pattern Filter

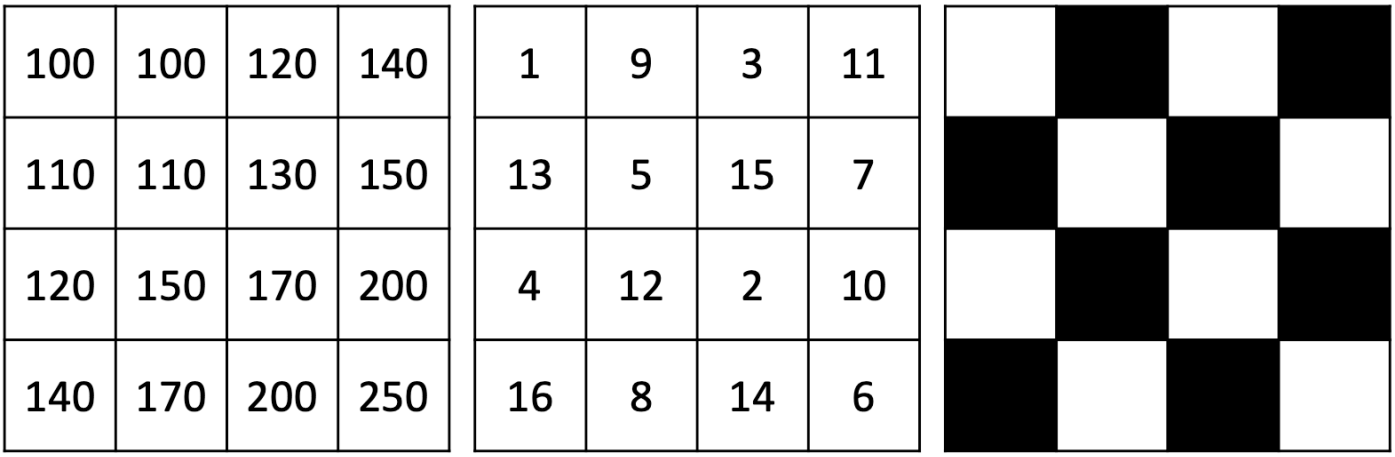
- With intensity level five, all pixels up to and including 5 are set to white, and all pixels above 5 are set to black.
- Given original intensity $I \in [0, 1]$ and an $n \times n$ halftone template, intensity I is represented by pattern p, where $p = \min(floor(I \times (n^2 + 1)), n^2)$

8.6 Pattern Dithering

- Process:
  - Calculate total scaled intensity.
  - Find the pattern template to use.
  - Compare with 4×4 template.
  - Set pixel to white if p >= template number.
- Example:
  - $Total\ Intensity = 100 + 100 + 120 + 140 + 110 + 110 + 130 + 150 + 120 + 150 + 170 + 200 + 140 + 170 + 200 + 250 = 2360$
  - $Average\ Intensity = \frac{Total\ Intensity}{Number\ of\ Pixels} \therefore \frac{2360}{16} = 147.5$
  - $Total\ Scaled\ Intensity = \frac{Average\ Intensity}{255} \therefore \frac{147.5}{255} = 0.578$
  - $4 \times 4\ Pattern\ p = \min(floor(I \times (n^2 + 1)), n^2) \therefore \min(floor(0.578 \times (4^2 + 1)), 4^2) = 9$

100	100	120	140
110	110	130	150
120	150	170	200
140	170	200	250

1	9	3	11
13	5	15	7
4	12	2	10
16	8	14	6



A B C

Figure 8.6.1 - 4×4 Pattern Dithering Example | A. Original Image | B. Template | C. Final Image

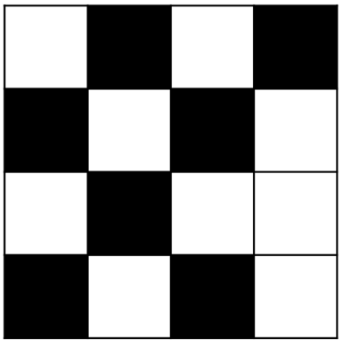
8.7 Ordered Dithering

- Process:
  - Get the pattern number for each pixel.
  - Compare the pattern number with the template for each pixel. If the pattern number is greater than the template number, set the pixel to white.
- Example:
  - $Top\ Left\ Pixel = 100$
    - $Scaled\ Intensity = \frac{Intensity}{255} \therefore \frac{100}{255} = 0.39$
    - $4 \times 4\ Pattern\ p = \min\left(floor\left(0.39 \times (4^2 + 1)\right), 4^2\right) = 6$
  - $Top\ Right\ Pixel = 140$
    - $Scaled\ Intensity = \frac{Intensity}{255} \therefore \frac{140}{255} = 0.55$
    - $4 \times 4\ Pattern\ p = \min\left(floor\left(0.55 \times (4^2 + 1)\right), 4^2\right) = 9$
  - (Repeat for all pixels in the image)

100	100	120	140
110	110	130	150
120	150	170	200
140	170	200	250

1	9	3	11
13	5	15	7
4	12	2	10
16	8	14	6

6	6	8	9
7	7	8	10
8	10	11	13
9	11	13	16



A B C D

Figure 8.7.1 - Ordered Dithering Example | A. Original Image | B. Template | C. Patterns | D. Final Image

Chapter 9.0 – Colour

9.1 The RGB (Red Green Blue) Colour Model

Pixel Depth – *The number of bits used to represent each pixel in RGB space.*

- Model based on a Cartesian Coordinate System.
- Three corners of the cube have red, green and blue.
- Three other corners of the cube have cyan, magenta and yellow.
- Black is at the origin.
- White is at the corner furthest away from the origin.
- The greyscale (points of equal RGB values) extends from black to white along the line joining these two points.
- All values of RGB are assumed to be in the range of 0 to 1.
- The total number of colours in a 24-bit RGB image is $2^{8^3} = 16{,}777{,}216$.

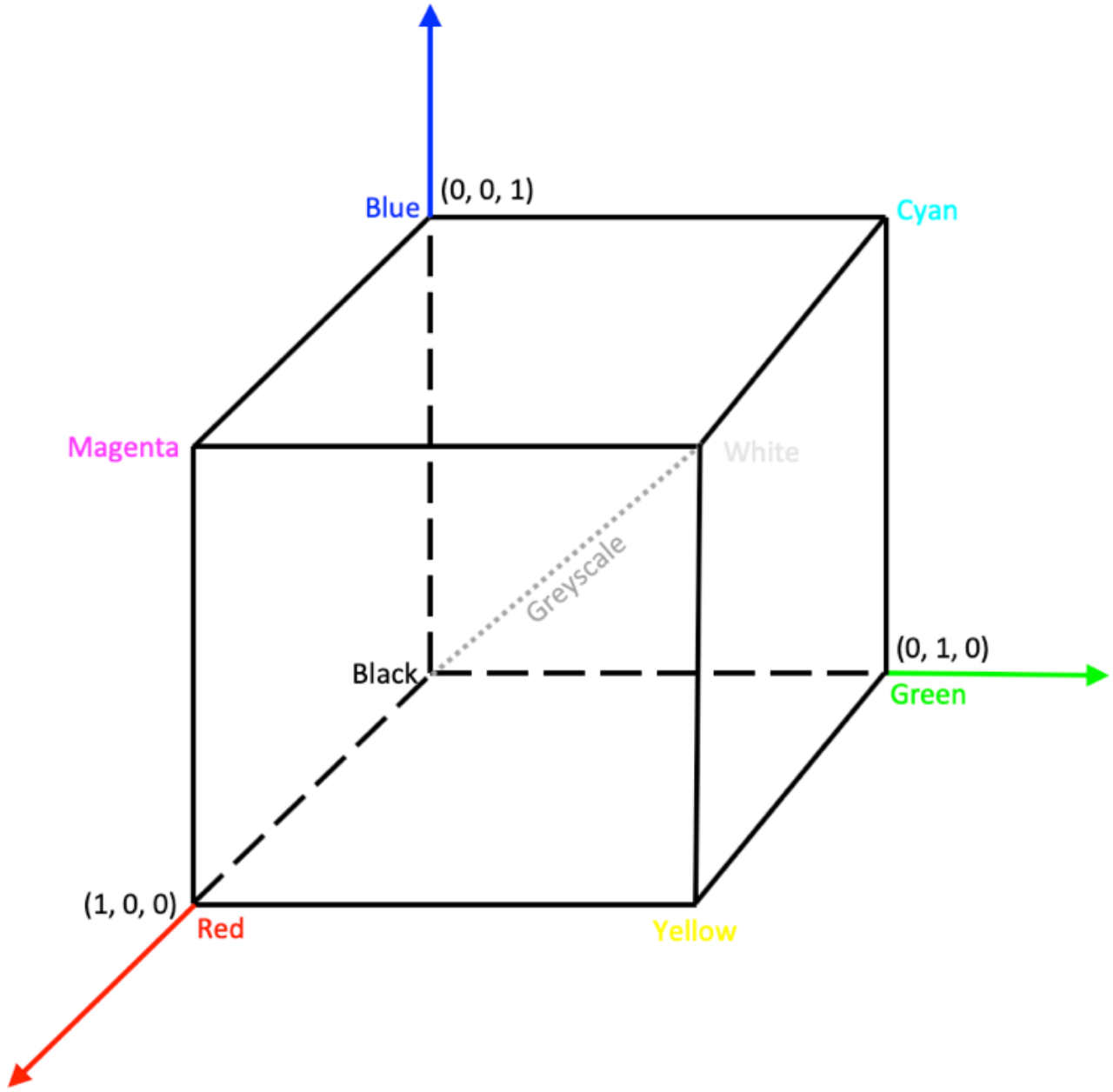


Figure 9.1.1 - RGB Colour Model Diagram; Cartesian Cube

9.2 The HSV (Hue Saturation Value) or HSI (Hue Saturation Intensity) or HSB (Hue Saturation Brightness) Colour Model

- One advantage of this colour model is that it is well suited to making image processing algorithms that are based on colour descriptions that are natural and intuitive to humans.

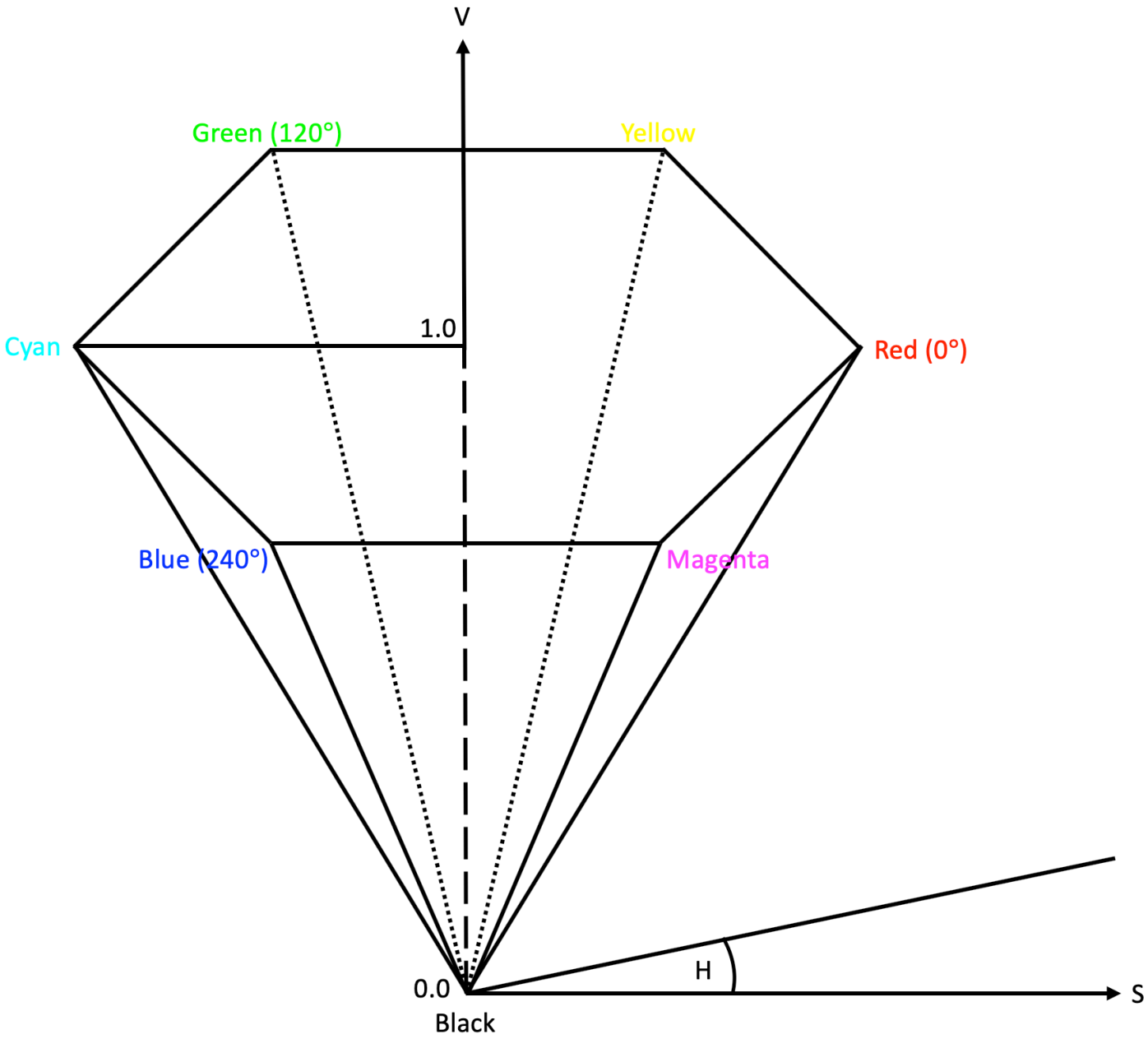


Figure 9.2.1 - HSV Colour Model Diagram

Chapter 10.0 – Noise Reduction Non-Linear Filtering

- Take the median filter size (e.g. 3×3).
- Create a list of the pixels within the filter and sort it.
- The median value will be the pixel in the 5^{th} position.
- Use the median value as a new value for the centre pixel of the filter.
- Advantages of Non-Linear Filtering:
  - Better edge preservation.
  - Good at removing dots and lines.
  - Not so affected by outliners.
- Sharp corners are smoothed.

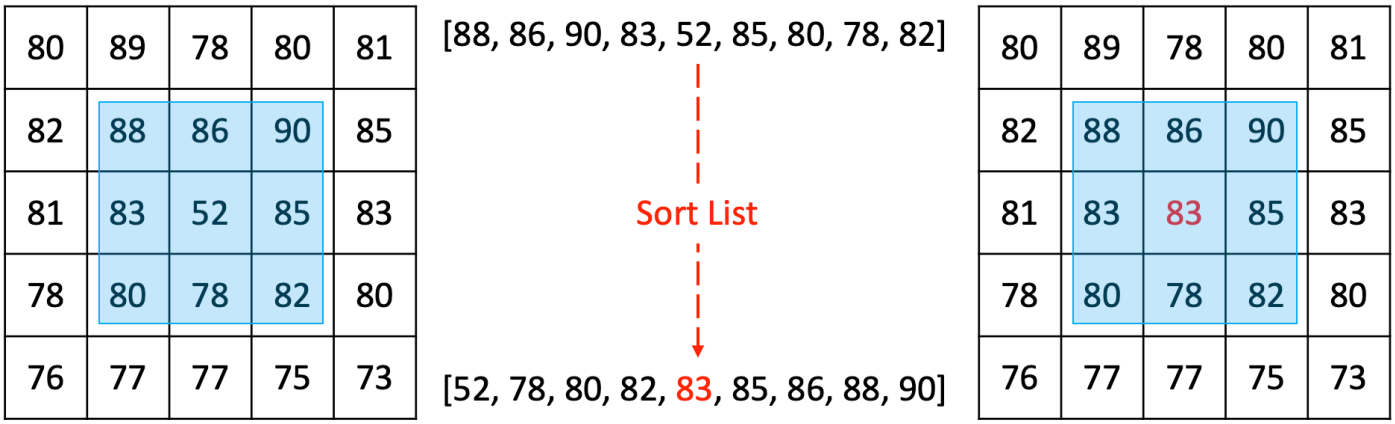


Figure 10.1.1 - Theoretical Example of Non-Linear Filtering

Figure 10.1.2 - Practical Example of Non-Linear Filtering

Chapter 11.0 – Object Representation

Graphics Processing Unit (GPU) – *A chip dedicated to the processing of vertices for the purpose of display.*

Open Graphics Language (OpenGL) – *A language for programming graphics applications. The language maps to GPU hardware and is OS independent.*

Direct3D – *Microsoft's graphical programming language. The language maps to GPU hardware and is OS dependent.*

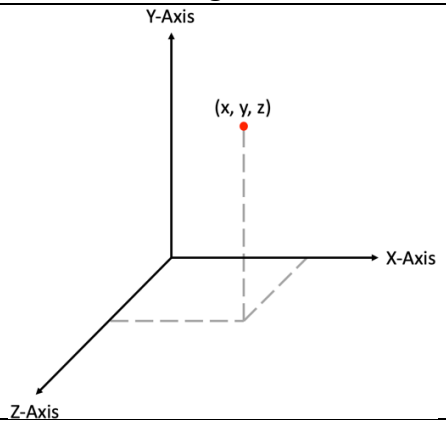
Radiosity – *A method of rendering based on detailed analysis of light reflections off defuse surfaces.*

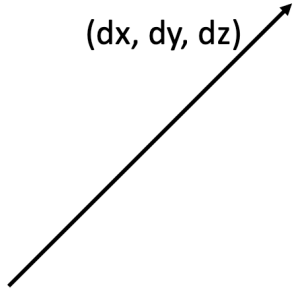
Triangular Meshes – *A type of polygon mesh which comprises of a set of triangles (typically in three dimensions) that are connected by their common edges or corners.*

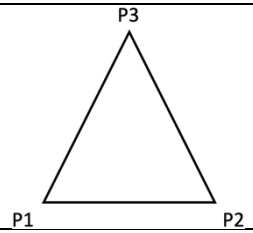
- Storing objects via an equation provides extremely compact storage but not all objects can be represented using this method.
- Advantages of Triangular Meshes:
    - Ray / triangle intersection for ray-tracing is easy.
    - Can project triangles in GPU hardware at millions per second; useful for games.
    - Easy to manipulate and subdivide (e.g. for level of detail models or radiosity).

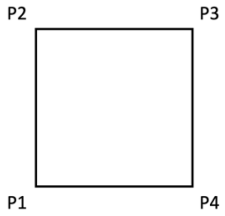
11.1 Modelling: 3D Primitives

Diagram	Code	Details

	```public class Point {
	    private float x;
	    private float y;
	    private float z;
	}```
		• Point
		• 3D Location in space
		• Represented by coordinates

	```public class Vector {
	    private float dx;
	    private float dy;
	    private float dz;
	
	    public float Magnitude() {
	        return Math.sqrt((dx*dx)+(dy*dy)+(dz*dz));
	    }
	}```
		• Vector
		• 3D direction and magnitude
		• Has no position

	```public class Triangle {
	    private Point P1;
	    private Point P2;
	    private Point P3;
	}```
		• Triangle
		• This explicit representation leads to duplicated points

	```public class Quad {
	    private Point P1;
	    private Point P2;
	    private Point P3;
	    private Point P4;
	}```
		• Quad
		• This explicit representation leads to duplicate points

11.2 Explicit Representation of a Cube

- A cube has 6 faces / quads.
- Only two quads should be needed to represent the whole cube (8 vertices). But due to the 6 quads used, there are 24 vertices.
- A solution to the duplicate points issue is to use pointers to the various vertices wherever they are needed.
    - Vertices / Points:
        - 0 = (0, 0, 0)
        - 1 = (0, 0, 1)
        - 2 = (0, 1, 0)
        - 3 = (0, 1, 1)
        - 4 = (1, 0, 0)
        - 5 = (1, 0, 1)
        - 6 = (1, 1, 0)
        - 7 = (1, 1, 1)
    - Quads Representation:
        - 0 1 5 4
        - 4 6 2 0
        - 6 7 3 2
        - 7 5 1 3
        - 4 5 7 6
        - 1 0 2 3
- Advantages:
    - 3D transformations use just 8 vertices.
    - Rounding errors are not a problem.
- Disadvantages:
    - Draw 24 edges (rather than 12).
    - Extra memory.
    - Extra processing during modelling.

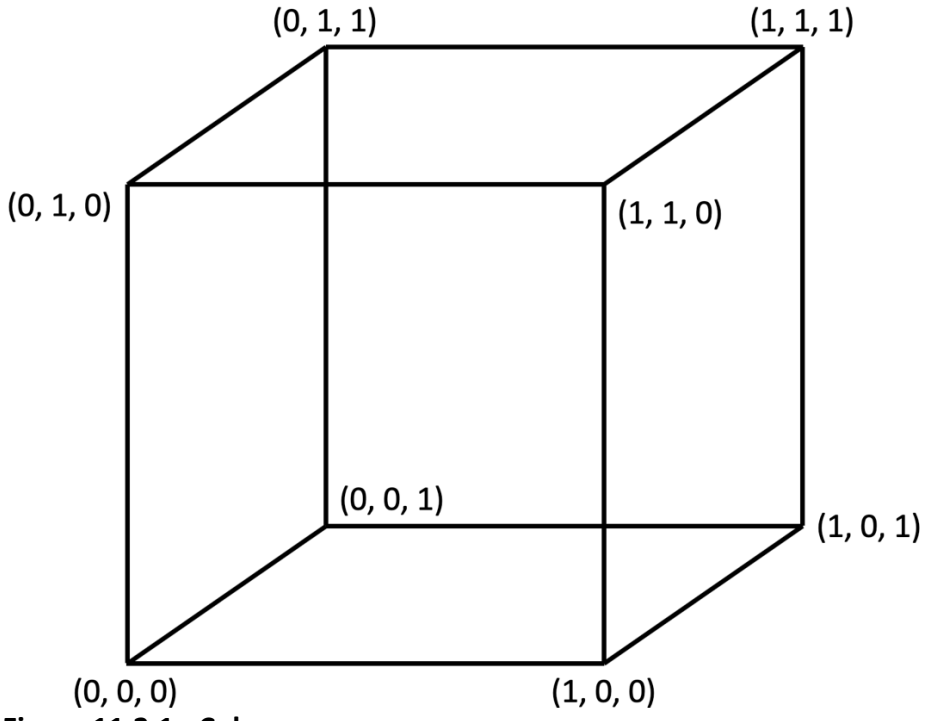


Figure 11.2.1 - Cube

11.3 Triangles Method One

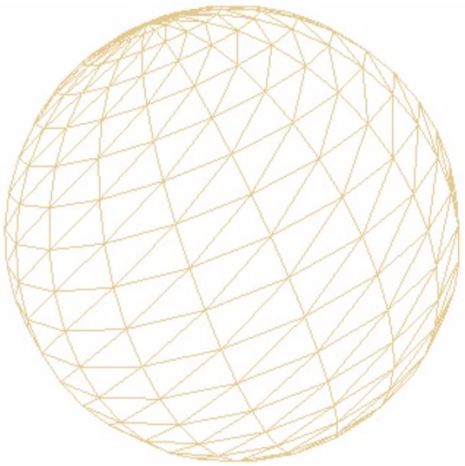


Figure 11.3.1 - Sphere

- The sphere in **Figure 11.3.1** consists of 382 vertices and 760 triangles.
- Each vertex is 3 floats (3×4 bytes = 12)
- Explicit representation:
  - 760 triangles × 3 vertices each × 12 bytes per vertex = 27,360 bytes
- Pointers to vertex list:
  - Each triangle is a list of 3 pointers pointer to the relevant vertices which make up the triangle (3 × 4 bytes = 12)
  - 382 × 12 (vertex memory) + 760 (pointers) × 12 = 13,704 bytes

11.4 Triangular Strips Method Two

- Compact (n triangles represented using n + 2 vertices).
- Transmission to GPU is quicker.
- Very efficient when drawing.
- Can be hard to create triangle strips from arbitrary geometry.
- **Figure 11.4.1** Drawing order: V0, V1, V2 then V2, V1, V3 then V2, V3, V4.

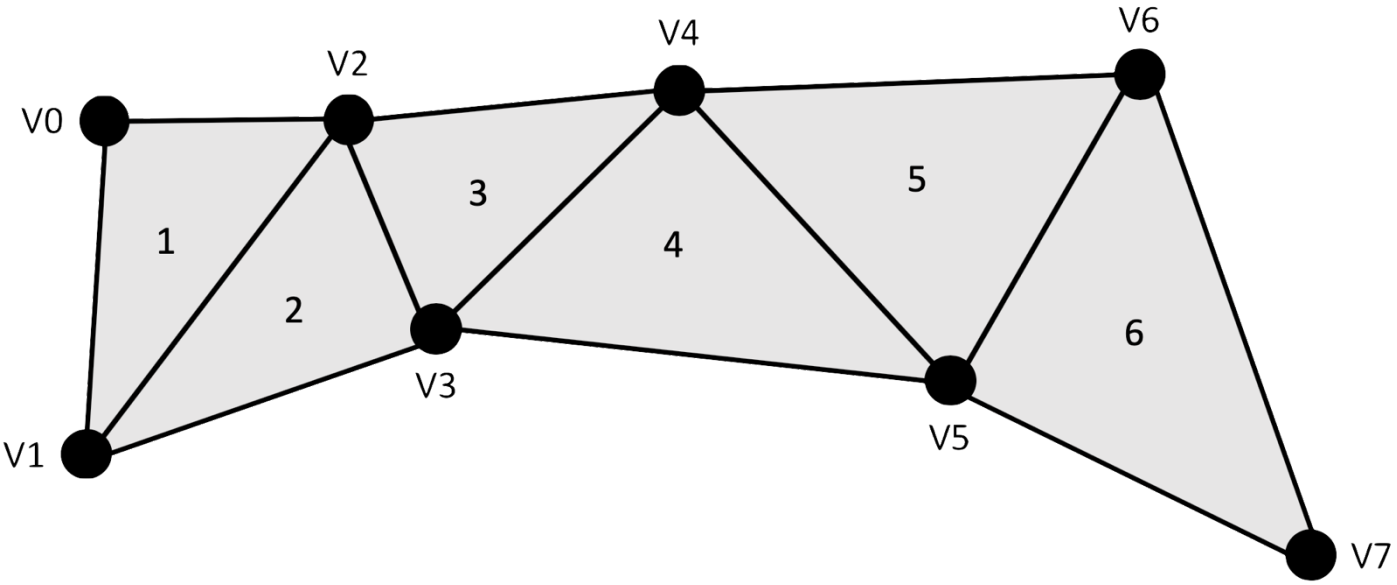


Figure 11.4.1 - Triangular Strip

11.5 Direct3D Drawing Primitives

Primitive	Description	Data Set
D3D_POINTLIST	A list of isolated points.	n vertices
D3D_LINELIST	A list of isolated lines (each pair of points are the ends of a line).	2n vertices = n lines
D3D_LINESTRIP	The vertices make a continuous line.	n + 1 vertices = n lines
D3D_TRIANGLELIST	Each group of 3 points define an isolated triangle.	3n vertices = n triangles
D3D_TRIANGLESTRIP	(Illustrated in previous section).	n + 2 vertices = n triangles
VERTEX BUFFERS	Direct3D allows points to a vertex list.	

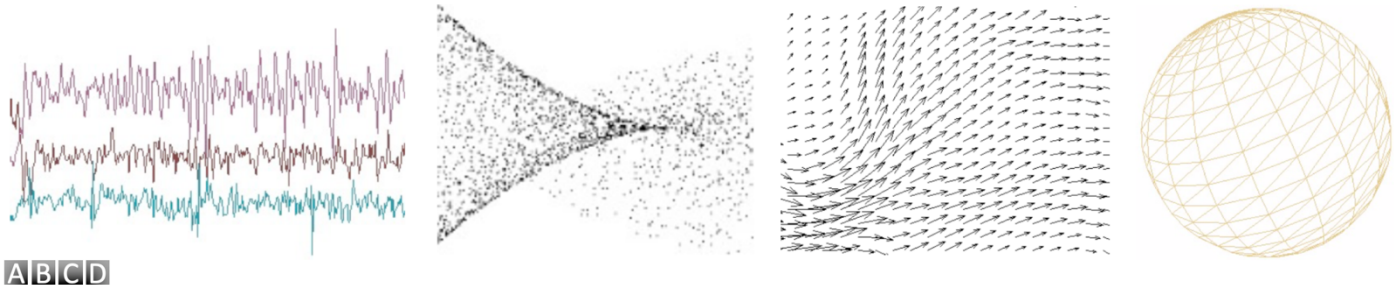


Figure 11.5.1 – Various Data Structures that can be represented in Direct3D

- **Figure 11.5.1.A** – D3D_LINESTRIP
- **Figure 11.5.1.B** – D3D_POINTLIST
- **Figure 11.5.1.C** – D3D_LINELIST
- **Figure 11.5.1.D** – D3D_TRIANGLESTRIP

Chapter 12.0 Rendering

12.1 Rasterization

Rasterization – *The task of taking an image described in vector graphics format and converting it into a raster image.*

Raster Image – *A series of pixels, dots or lines that when they come together on a display, they recreate the image.*

Camera Model

- A camera model is needed for both Rasterization and Ray Tracing.
- ***View Plane Normal –*** *A vector which can be calculated from the position the camera looks at minus the view reference point of the camera.*

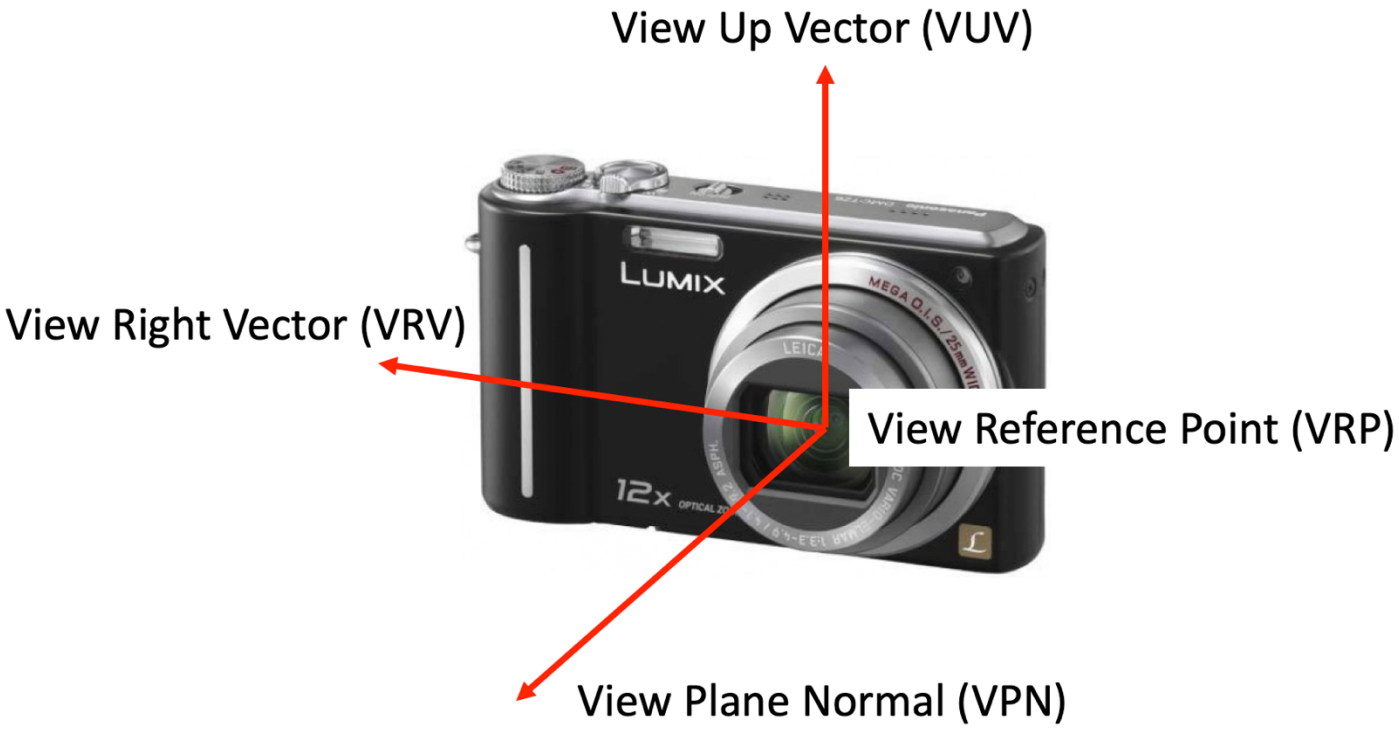


Figure 12.1.1 - Camera Model Illustration

Rasterization Process:

1. ***Transformation –*** *Matrices for scaling, translation and rotation are applied to each vertex within the object.*
2. Each vertex (x, y) gives the screen coordinates, and z gives the depth.
3. ***Clipping –*** *Removes any part of the scene not visible within the image.*
4. ***Scan Conversion –*** *Colouring pixels according to the object's colour and lighting model.*

13.1 General Concepts

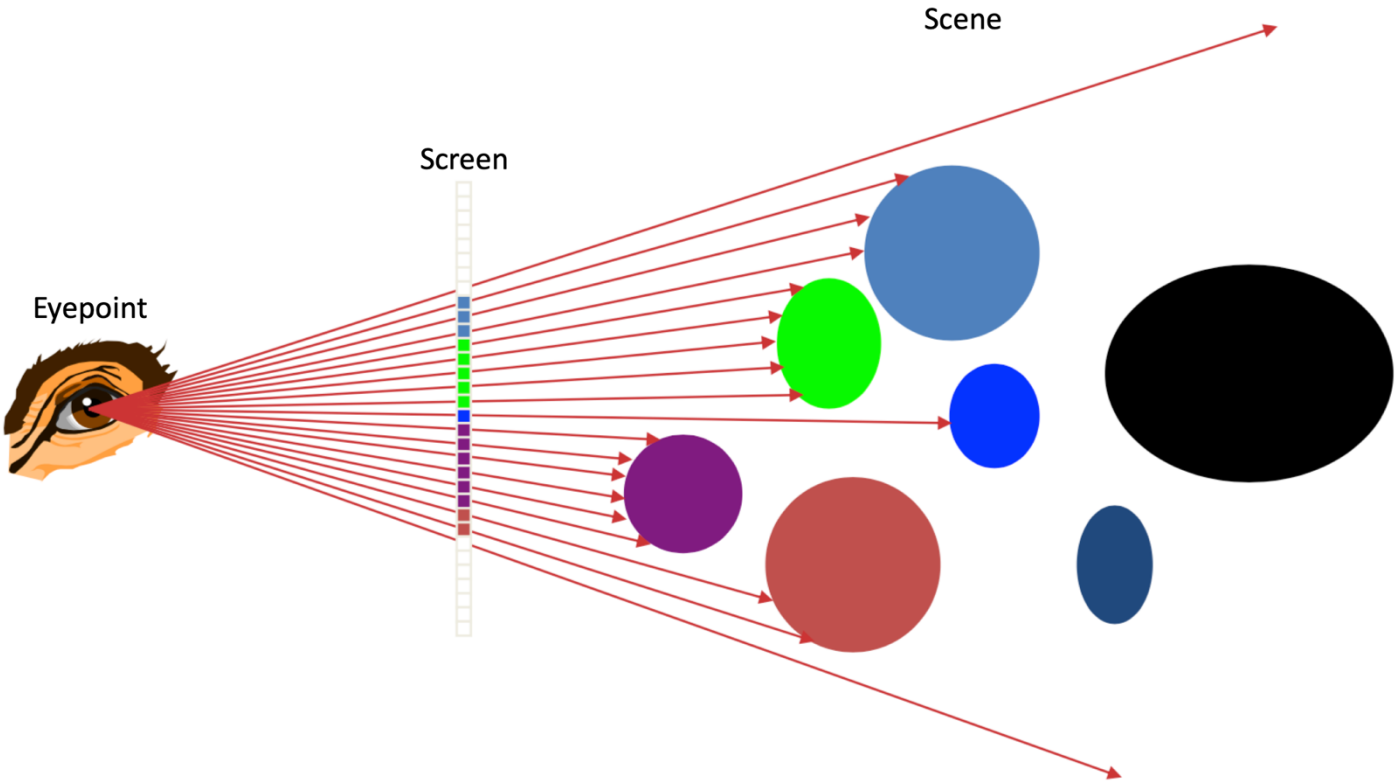


Figure 13.1.1 - Perspective Projection

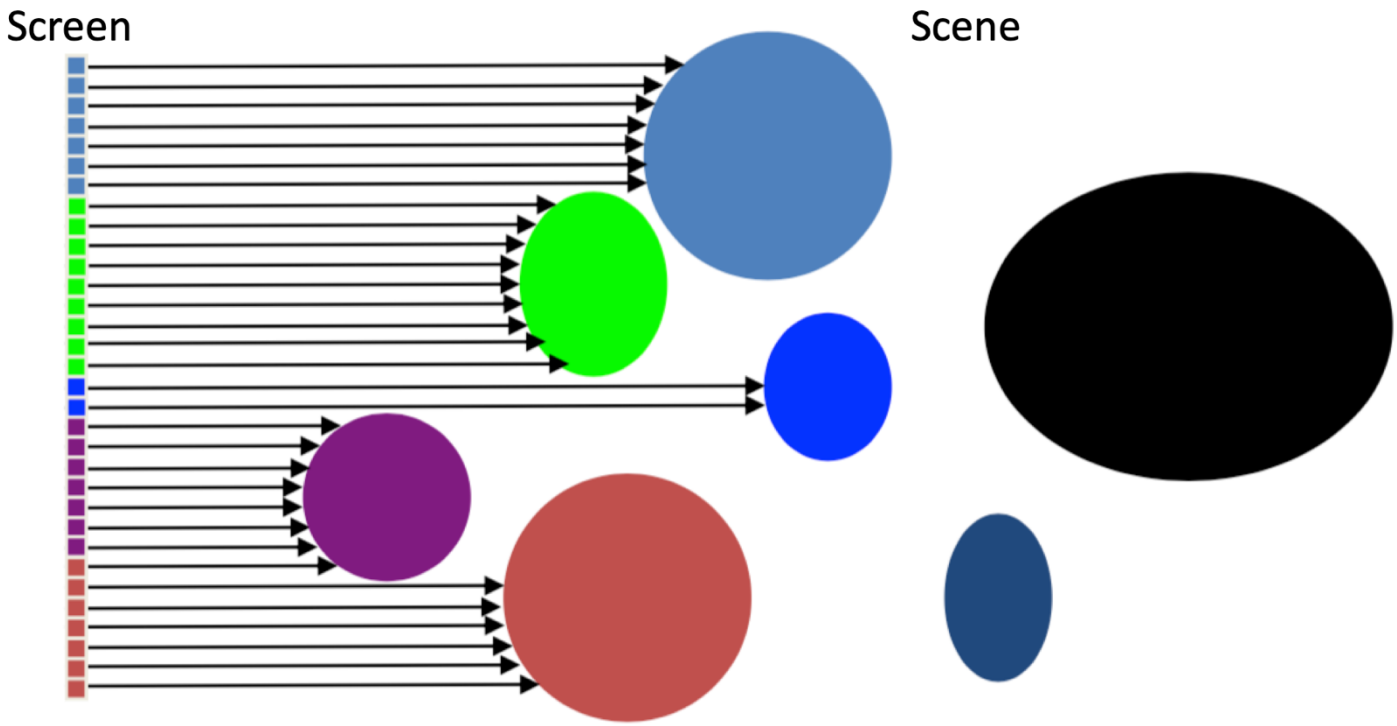


Figure 13.1.2 - Orthographic Projection

13.2 Ray / Sphere Intersection

- Intersection between a 3D line and a sphere.
- Ray $p = o + dt$
- Sphere $r^2 = (p - c)^2$
- Substitute the ray equation p into sphere equation:
  - $r^2 = ((o + dt) - c)^2$
- Expand the new equation:
  - $r^2 = (o - c)^2 + (dt)^2 + 2dt(o - c)$
- Rearrange the new equation:
  - $0 = d^2t^2 + 2d(o - c)t + (o - c)^2 - r^2$
- To solve the equation above we use:
  - $t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Where $a = d^2$, $b = 2d(o - c)$, $c = (o - c)^2 - r^2$
- The line starts at the eyepoint o, and goes in a direction d through each pixel.

Surface Normal – *A surface normal at a point is a vector from that point perpendicular to the tangent plane at that point.*

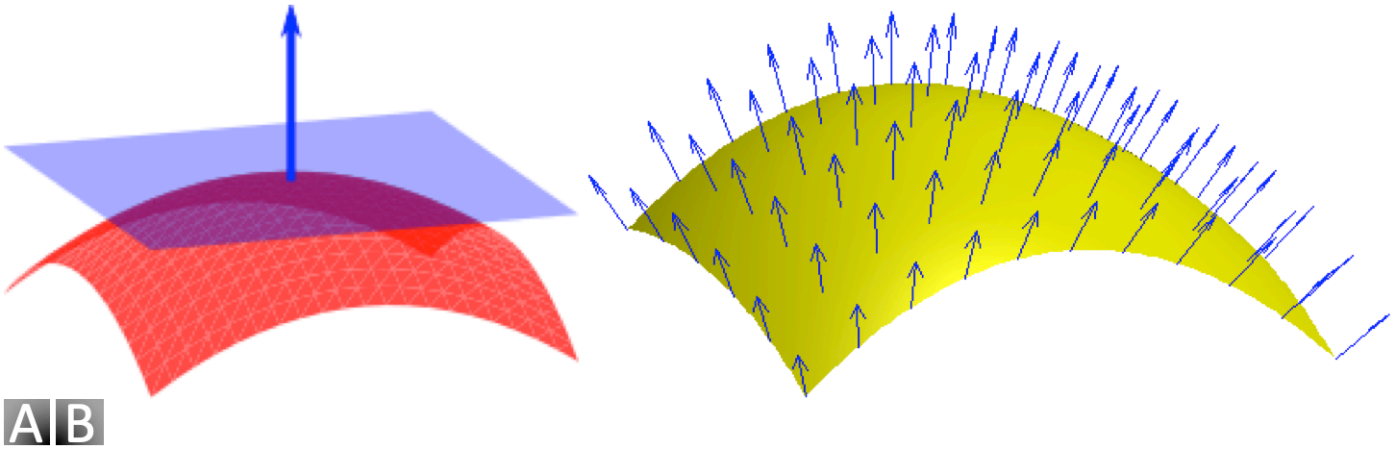
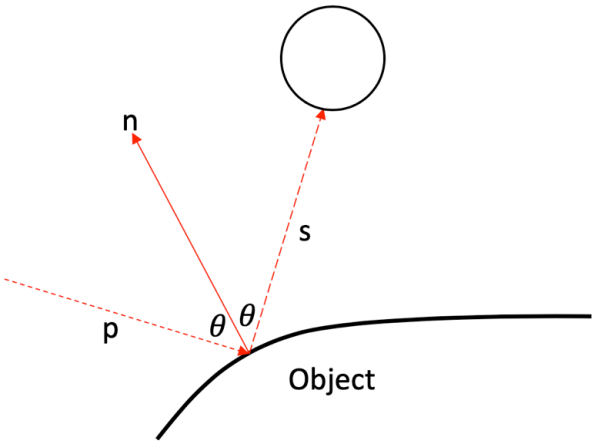


Figure 13.2.1 - Surface Normal Illustration



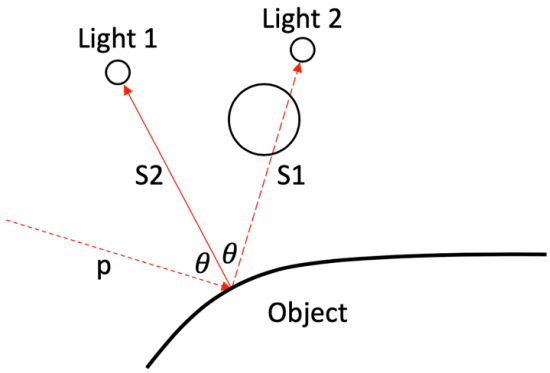
n = Surface Normal

θ = Angle of Incidence (Physics of Reflection)

p = Primary Ray (From Pixel)

s = Secondary Ray (From Object)

Figure 13.2.2 - Reflections Illustration



θ = Angle of Refraction (Physics of Light Moving from one Medium to Another)

p = Primary Ray

S1 = First Secondary Ray

S2 = Second Secondary Ray

Figure 13.2.3 - Shadows Illustration

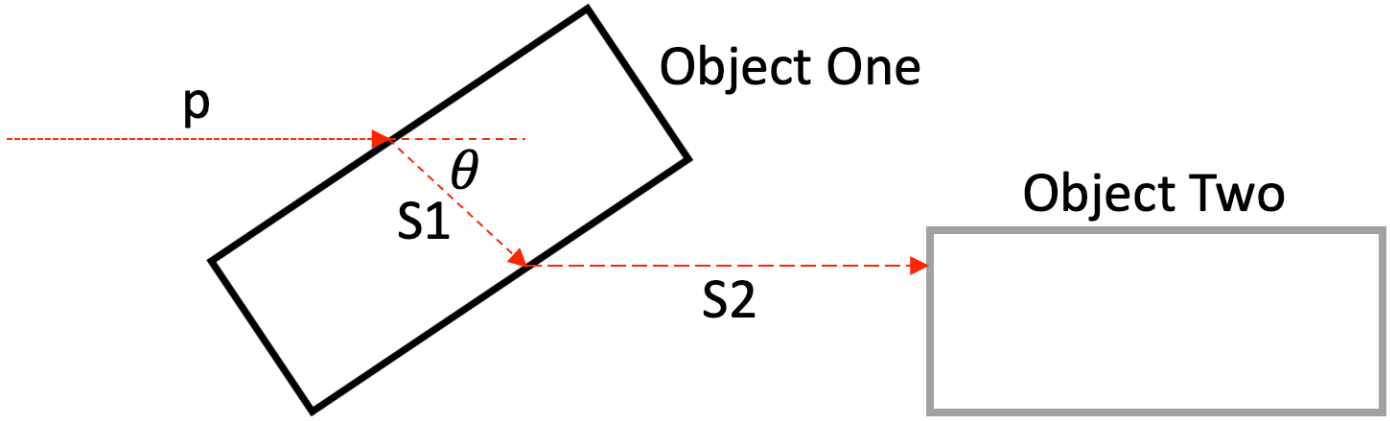


Figure 13.2.4 - Snell's Law Illustration (Refraction)

13.3 Recursive Ray Tracing

Recursive Ray Tracing – *The process of continuing or initiating new rays at object intersection points where each ray is treated like a primary ray, i.e. can spawn shadow rays, reflected rays and refracted rays.*

Recursive Cut-Off – *The number of recursive levels that each ray is allowed to go down before being terminated; preventing infinite recursion e.g. a ray bouncing between two mirrors.*

Figure 13.3.1 – Recursive Depth Example

- Advantages of Ray Tracing:
  - Ray tracing is the most accurate rendering model so far.
  - Closely models the real physics of light; such as frequency, speed of light in different mediums (refraction), intersurface absorption, colour, reflection, etc.
- Disadvantage of ray Tracing:
  - The large computational expense involved.

13.4 Bounding Volumes

Bounding Volume (BV) – *A bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set. Bounding volumes are used to improve efficiency of geometrical operations by using simple volumes to contain more complex objects.*

Hierarchical Bounding Volumes – *A tree structure on a set of geometric objects. All geometric objects are wrapped in bounding volumes that form leaf nodes of the tree.*

- Advantages of *Hierarchical Bounding Volumes*
  - Fast ray tracing times.
  - User can decide on tight fitting objects.
  - User can decide logical splitting of the scene.
  - Due to the tree like structure of this method, we can eliminate up to half of the search space at each stage by just one direction decision, thus enabling us to get down to the item we require quickly.
- Disadvantages of *Hierarchical Bounding Volumes*
  - User interaction is required.
  - Could take a lot more time that is saved.
  - Not all scene modellers will understand *Bounding Volumes*.

Octrees – *A tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three-dimensional space by recursively subdividing it into eight octants.*

Process of Raytracing with an Octree Figure 13.4.1.C:

- The ray is tested with the top bounding box, and if it is found to intersect, it will be tested with each of its eight children.
  - If not, we have saved testing with the complex model (empty boxes).
- If it intersects with any children, the closest one is checked by checking the ray against each of its eight children and so on recursively.
- The recursive search ends with leaf nodes.
- Each leaf notes are either empty (in which case we return, and the recursion ensures the next box is checked) or has a pointer to a list of objects with which the ray is checked.

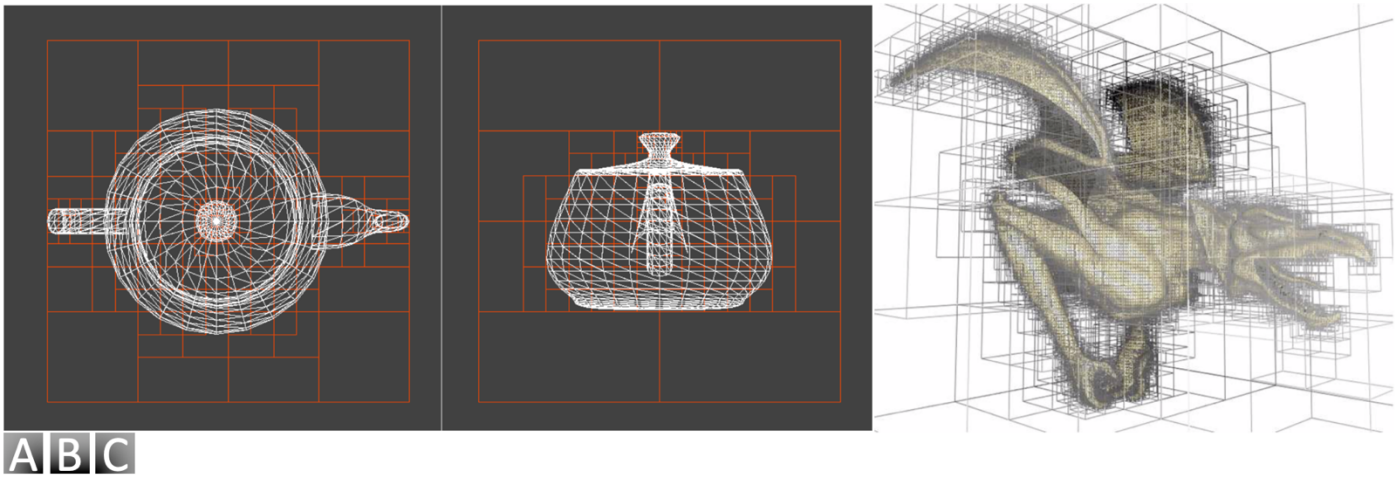


Figure 13.4.1 - Octree Examples

K-Dimensional Trees (KD-Trees) – *A tree data structure where data in each node is a K-Dimensional point in space. Each section is subdivided into further sections, but the cut is not necessarily made in the centre of the current section but split the longer dimension near to the data median.*

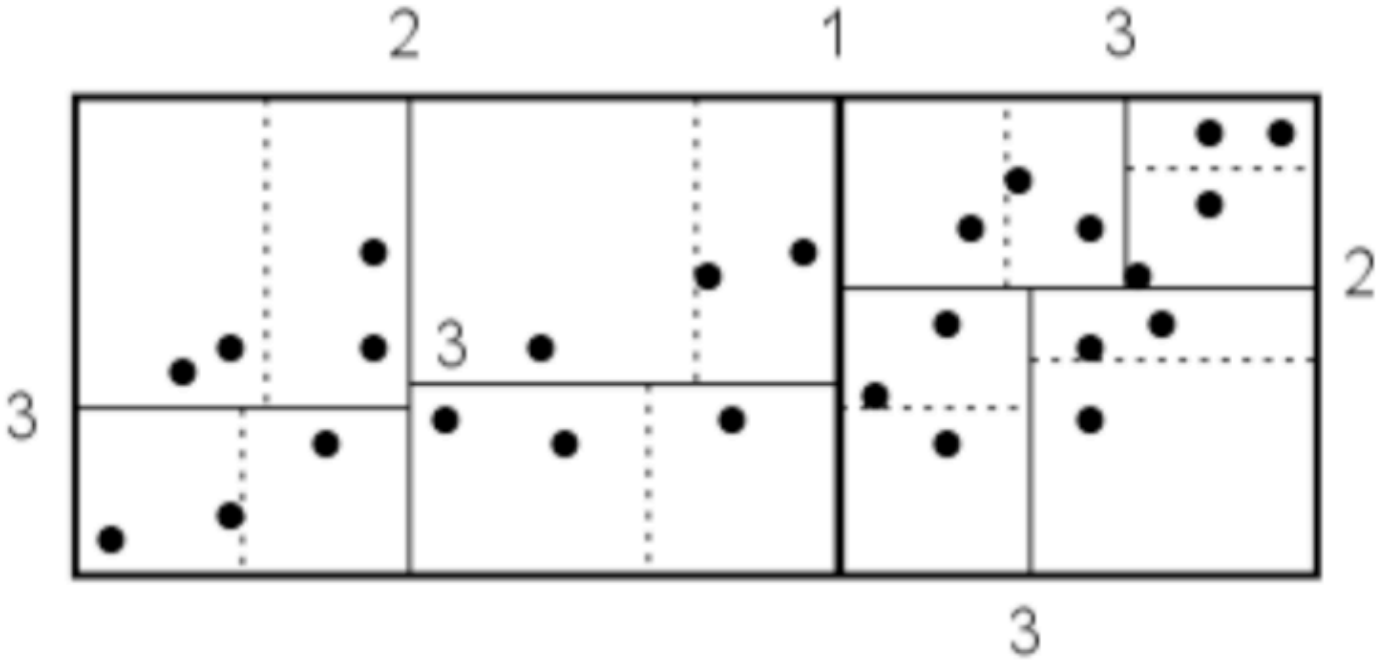


Figure 13.4.2 - K-Dimensional Tree

Binary Space Partitioning Tree (BSP Tree) – *A tree data structure used to partition a three-dimensional space where the space is recursively subdivided into further planes which can be any shape.*

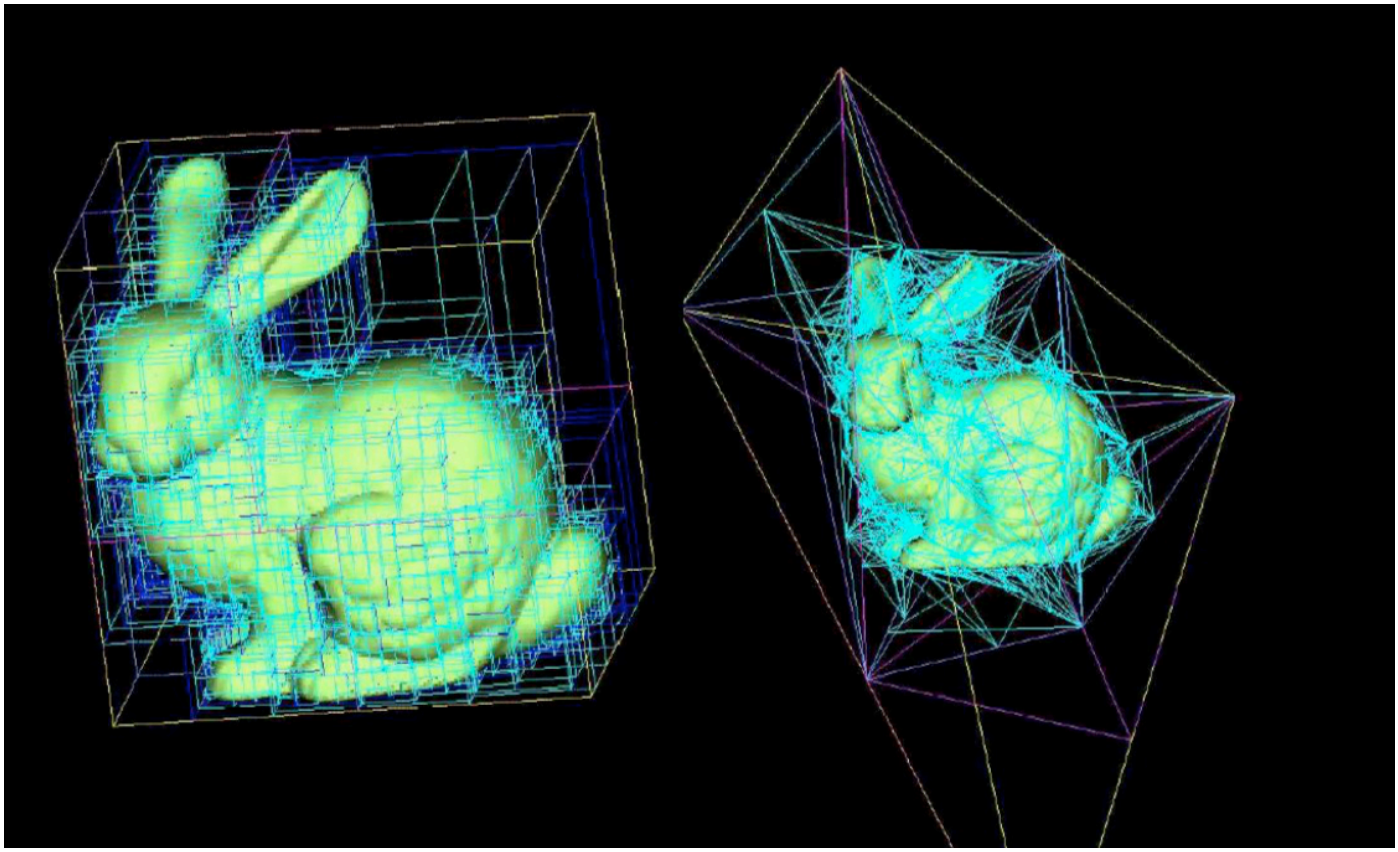


Figure 13.4.3 - Binary Space Partitioning Tree for a Rabbit

Example 13.4.1 – Ray Tracing a Chess Set

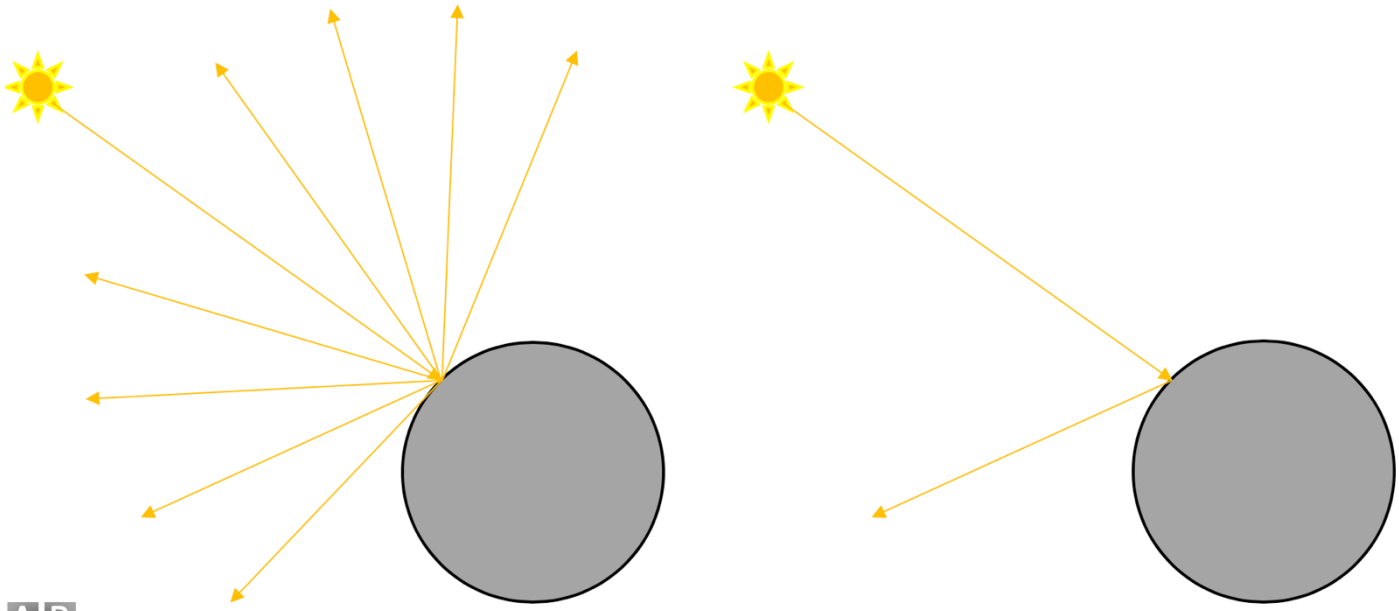
- No reflections, shadows or transparency
  - Chess Board:
    - 32 pieces with 100 triangles each
    - 20 extra triangles for the board, plane, sky, etc
    - Total = 3220 triangles
  - Image has 1000×1000 = 1 million pixels
  - The algorithm solves the ray with each triangle for each pixel resulting in $1000 \times 1000 \times 3220 = 3.22$ billion intersection calculations.
- Using bounding volumes; a cylinder around each chess piece
  - 32 cylinders = 52 intersections when tracing it against the general scene geometry.
  - If the closest hit point is a cylinder, then interest ray with 100 triangles.
  - Many rays will require just these 152 intersections.
  - Some will miss the chess piece although they hit the cylinder.
  - Assume 60% of rays miss cylinders, 30% hit a triangle in the first cylinder and 10% hit a triangle in the second cylinder.
    - $0.6 \times 1000 \times 1000 \times 52 = 31{,}200{,}000$
    - $0.3 \times 1000 \times 1000 \times 152 = 45{,}600{,}000$
    - $0.1 \times 1000 \times 1000 \times 252 = 25{,}200{,}000$
    - Total = 102 million (about 30 times faster)
- Hierarchical bounding volumes
  - Hierarchical Model
    - 1 box around the whole of the chess board and pieces (6 intersection tests).
    - 3 other planes for the sky and two ground textures (3 intersection tests).
    - Inside the chess box, place 2 boxes, one over the white pieces and one over the black (6 intersection tests)
    - Inside the previous boxes, have 32 cylinders (1 intersection tests).
    - Inside each cylinder have 3 cylinders (3 intersection tests).
    - Inside each of those have 30-40 triangles (30-40 intersection tests).
    - At the first level of our hierarchy, 9 intersection tests are used to see if we have hit the sky, ground or are in the region of the board.
    - If we hit the board region, another 6 intersection tests to see if we have hit a white or black chess piece.
    - If we have hit white, another 16 intersection tests to see if we have hit a chess piece.
    - Another 3 intersection tests determine which part.
    - A maximum of 40 intersection tests to determine the triangle.
    - The total number of intersection tests for a hit piece is 74.
  - Number of intersections
    - Assume 60% of rays miss cylinders, 30% hit a triangle in the first cylinder and 10% hit a triangle in the second cylinder.
      - $0.6 \times 1000 \times 1000 \times 9 = 5{,}400{,}000$
      - $0.3 \times 1000 \times 1000 \times 74 = 22{,}200{,}000$
      - $0.1 \times 1000 \times 1000 \times 117 = 11{,}700{,}000$
      - Total = 39.3 million (about 100 times faster)

14.0 – Lighting and Shading

14.1 Surface Lighting

Diffuse Reflection Figure 14.1.1.A – *The property of reflected light being scattered in all directions after intersecting with a rough surface.*

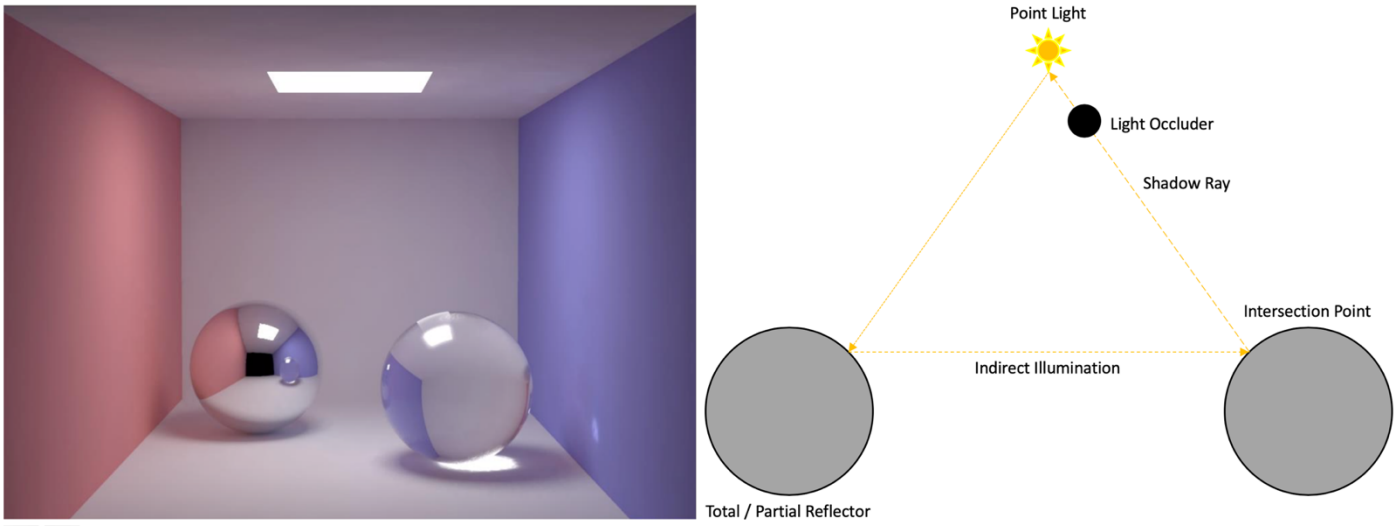
Spectral Reflection Figure 14.1.1.B – *The process of reflected light being concentrated into a highlight, or bright spot after intersecting with a shiny surface.*



A B

Figure 14.1.1 - A. Diffuse Reflection | B. Spectral Reflection

14.2 Indirect Illumination



A B

Figure 14.2 - Indirect Illumination; A. Example | B. Diagram

14.3 Ambient Reflection

Ambient Reflection – *Natural light from a scene which is reflected off the objects within the scene.*

- Pixel Intensity, $I = I_a K_a$
- I_a = Ambient Light Intensity
- K_a Ambient Reflection Coefficient of the Surface (how much light it reflects; between 0 and 1)

14.4 Light Sources

- To create a realistic rendering, we need to calculate the light incident upon objects in a scene.
- A point (x, y, z) can be the location of a point of light source.
- A vector (dx, dy, dz) can be the directional light source.

Specular Reflection – *Light that is perfectly reflected (like a mirror).*

- Light per unit area is higher when at 90°, compared to when at 45°.
- Light energy hitting the ground is $\frac{I}{A} = \frac{I \cos\theta}{w}$, **Figure 41.4.1**.

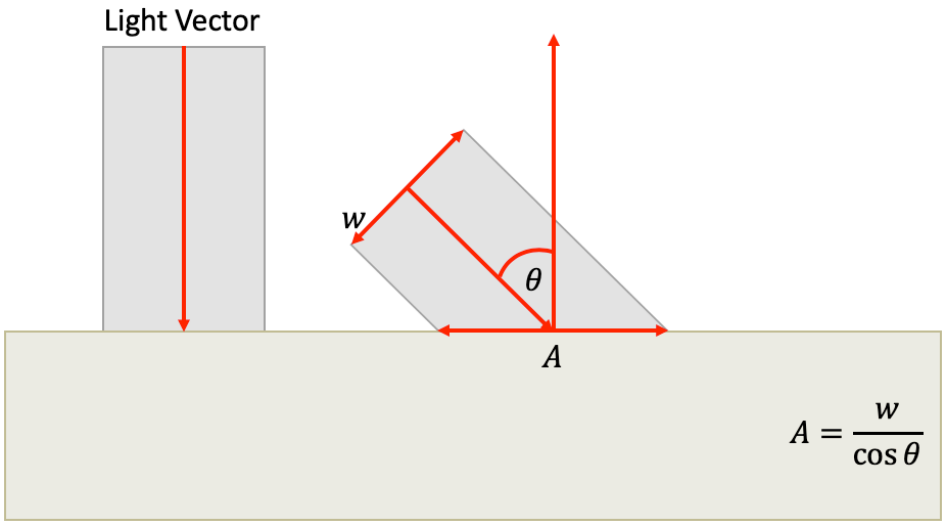


Figure 14.4.1 - Light Energy Hitting the Ground

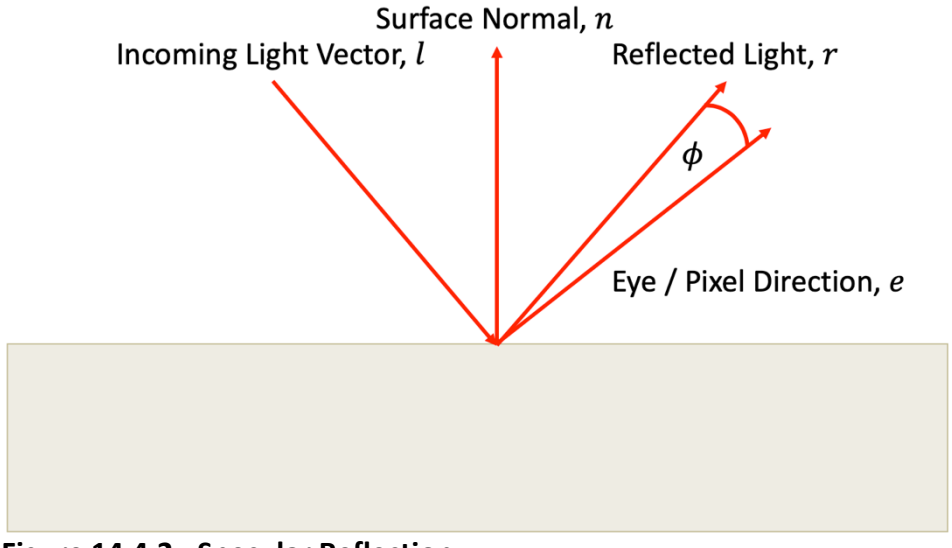


Figure 14.4.2 - Specular Reflection

14.5 Lambertian / Diffuse Reflection

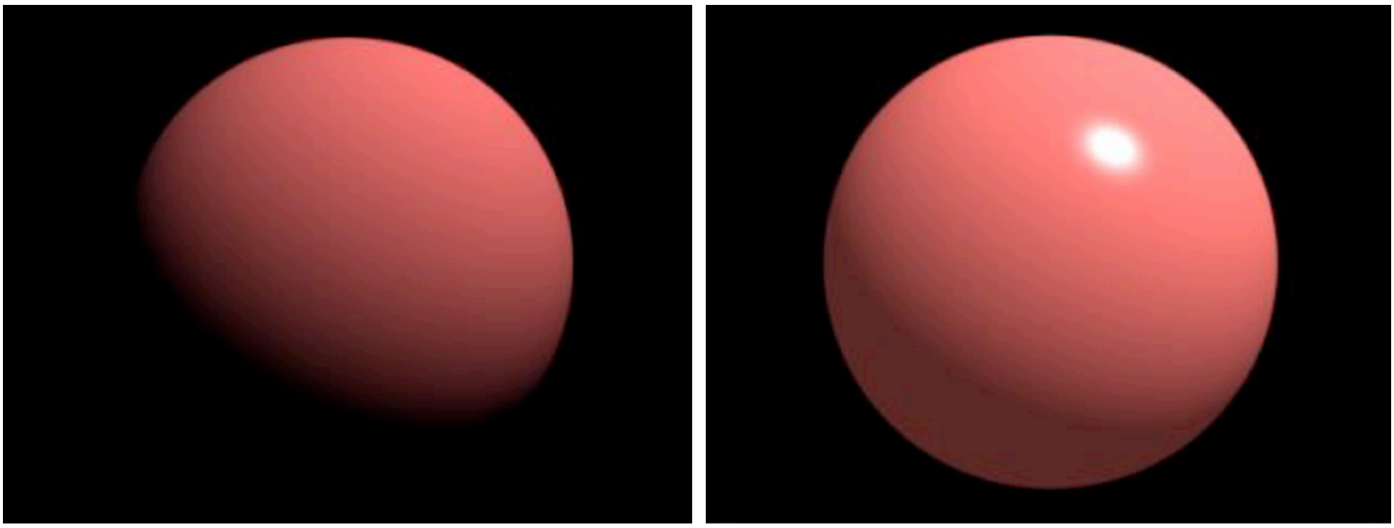
- Incoming light at an intersection point, $I_{Light} \cos\theta$
- Outgoing light at an intersection point, $I_{Out} = I_{Light} \cos\theta$, if $\cos\theta < 0$ use 0
- $\cos\theta$ is the angle between the surface normal n and the direction to the light l.
    - Both n and l are vectors.
- If we assume the vectors are normalised, $\cos\theta$ is the dot product of the two vectors:
    - $I_{out} = I_{Light} n.l$
- A vector can be normalised by finding its length, $l = \sqrt{dx^2 + dy^2 + dz^2}$ and dividing each component by the length to get a new vector, $\left(\frac{dx}{l}, \frac{dy}{l}, \frac{dz}{l}\right)$

Lambertian Reflection in Colour (Figure 14.4.2):

- Assign each object a diffuse reflection coefficient K_d in each colour channel $(K_{d,r}, K_{d,g}, K_{d,b})$ where each is a value between 0 (absorb all light) and 1 (reflect all light).
- The colour of a pixel I_{out} is:
    - $I_{out,r} = K_{d,r} I_{Light,r} n.l$
    - $I_{out,g} = K_{d,g} I_{Light,g} n.l$
    - $I_{out,b} = K_{d,b} I_{Light,b} n.l$

Phong Reflection (Figure 14.4.2):

- Phong's equation for specular reflection:
    - $I_{out,r} = K_{s,r} I_{Light,r} (r.e)^n$
    - $I_{out,g} = K_{s,g} I_{Light,g} (r.e)^n$
    - $I_{out,b} = K_{s,b} I_{Light,b} (r.e)^n$
- $\cos\phi$ is calculate using the dot product of r and e
- K_s is the specular reflection coefficient for the surface.



A B

Figure 14.5.1 - A. Lambertian Reflection in Colour | B. Phong Reflection

14.6 Simple Reflection Model

$$I_{out,c} = K_{a,c}I_{Light,c} + K_{d,c}I_{Light,c}n.l + K_{s,c}I_{Light,c}(r.e)^n$$

- c takes the values of colour channels: r, g and b.
- The equation adds together Ambient, Diffuse and Specular terms.

15.0 – Aliasing and Sampling

15.1 Aliasing and Anti-Aliasing

Aliasing – *The process by which smooth curves become jagged because of the resolution of the image is not high enough. This can be caused by taking samples at too low a frequency.*

Anti-Aliasing – *Techniques used for minimizing the distortion artifacts known as aliasing when representing a high-resolution image at a lower resolution.*

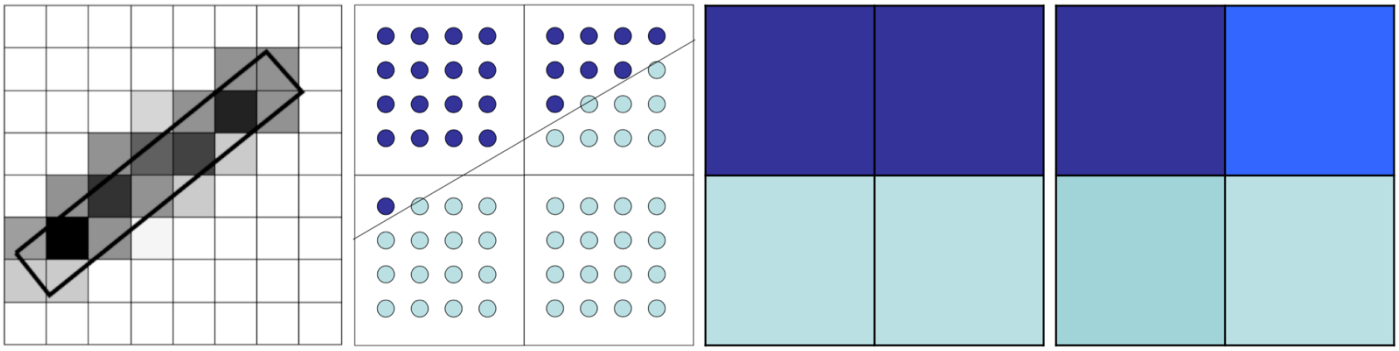


Figure 15.1.1 - A. Unweighted Area Sampling | B. Super-Sampling | C. Four Pixels without Super-Sampling | D. Four Pixels with Super-Sampling

15.2 Unweighted Area Sampling

- **Figure 15.1.A**
- Treat the line as a 1-pixel thick box measuring the area and setting the colour appropriately.

- Advantages of Unweighted Area Sampling:
  - Good for drawing primitives e.g. lines, circles, etc.
- Disadvantages of Unweighted Area Sampling:
  - The primitive must intersect the pixel in order to have an effect.

15.3 Super-Sampling

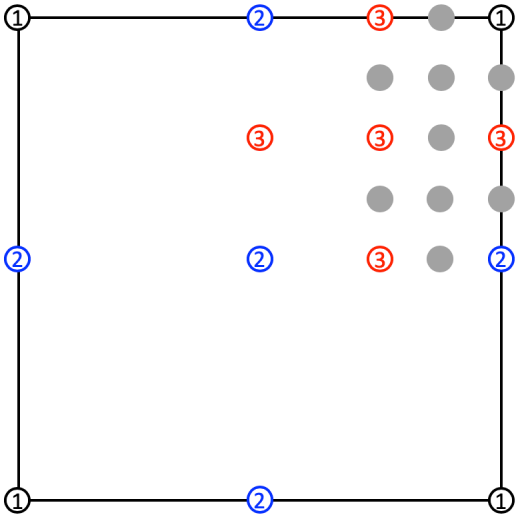
- **Figure 15.1.1.B/C/D**
- 4 pixels, each sampled by $4 \times 4 = 16$ sub-pixels.
- Each pixel becomes the average of 16 the sample's colours.

- Advantages of Super-Sampling:
  - Good for ray tracing.
  - Better images are produced.
- Disadvantages of Super-Sampling:
  - Takes much longer to compute.

15.4 Adaptive Super-Sampling

Adaptive Super-Sampling Figure 15.1.4 – *A method of Anti-Aliasing which sets the colour of the current pixel to the average of the 4 corner pixels if certain conditions are met.*

- Decide a threshold value, t, e.g. 20.
- Sample the scene at each pixel corner.
- Obtain 4 colours; c1, c2, c3, c4.
- If $(\max(c1, c2, c3, c4) - \min(c1, c2, c3, c4)) \leq t$ then pixel colour = $average(c1, c2, c3, c4)$



- Pixels Labelled 1 were the first 4 samples.
- If max - min > t, then take additional samples.
- These are labelled as 2 in the picture.
- We now recurse on the 4 new sub-pixels.
- Recurse each one until max – min ≤ t, or until a limit is reached.
- Final pixel is set as the average of all samples.

Figure 15.1.4 - Adaptive Super-Sampling

- Advantages of Adaptive Super-Sampling:
  - Good for ray tracing.
- Disadvantages of Adaptive Super-Sampling:
  - (Unknown).

15.5 Sample Positions

Sample Positions – *The points in an image which are sampled when representing a high-resolution image at a lower resolution.*

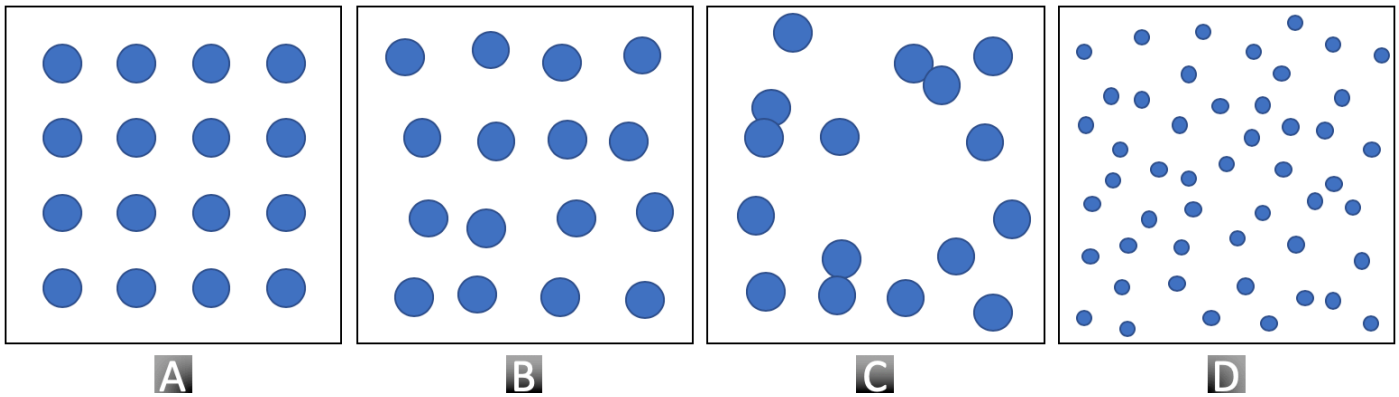


Figure 15.5.1 - Sample Positions | A. Grid | B. Jitter | C. Random | D. Poisson-Disk

- ***Grid –*** *Samples are taken in a grid like fashion.*
  - Advantages of *Grid*:
    - Easy and fast to compute.
  - Disadvantages of *Grid*:
    - Can lead to aliasing.

- ***Jitter –*** *Using the grid, each sample is moved a random but small amount, so that only one sample still occurs in each grid cell.*
  - Advantages of *Jitter*:
    - Sample pattern is fast to compute.
  - Disadvantages of *Jitter*:
    - (Unknown).

- ***Random –*** *Samples are taken in a random fashion.*
  - Advantages of *Random*:
    - Easy and fast to compute.
  - Disadvantages of *Random*:
    - Can lead to some areas being under sampled or oversampled.

- ***Poisson Disk –*** *The initial sample is randomly placed, but during an iterative process, further samples are randomly placed, but are immediately tested against all existing samples, and are removed if they are closer than a certain distance to an existing sample.*
  - Advantages of *Poisson Disk*:
    - Produces an excellent sampling pattern.
  - Disadvantages of *Poisson Disk*:
    - Computationally expensive (higher complexity than all previous methods).

16.0 – Maximum Intensity Projection

16.1 Volume Data

- 3D array of data. Each volume element (voxel) is a value representing some measurement or calculation.
- ***Magnetic Resonance Imaging (MRI) Data –*** *Strong magnetic fields align magnetization of hydrogen atoms, and then measures radio waves they emit.*
- ***Computed Tomography (CT) Data –*** *Interior images of the body are calculated using many X-Rays of the body.*
- CT scan 1800+ slices, each 512×512 voxels. Each voxel is 2 bytes (integer) $\sim$ 1GB.
- Body then photographed at 1800+ 1mm intervals (each photograph represents a slice through the body).
- Maximum Intensity Projection (MIP) has the advantage that it is close to a doctor's understanding of an X-ray.
- Its disadvantage is that the depth of the structure is hard to distinguish.

17.0 – Resizing Images

17.1 General Concepts

Interpolation – *A method of constructing new data points within the range of a discrete set of known data points.*

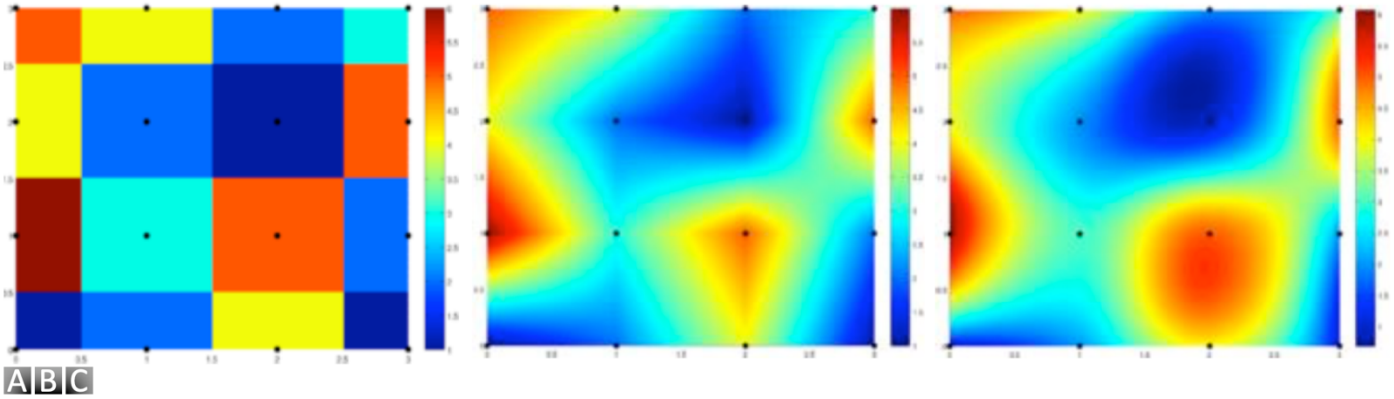


Figure 17.1.1 – A. Nearest Neighbour 2D | B. Bilinear | C. Bicubic

17.2 Nearest Neighbour

Nearest Neighbour – *Assigns the value of the nearest pixel to the pixel in the output visualization.*

- Method:
  - To resize image Ia from size (Xa,Ya) to image Ib with size (Xb,Yb) we need to find a value for every colour component of every pixel in image Ib.
  - In other words, we need to find Ib[y][x][c] for all x,y,c where x in [0,Xb-1], y in [0,Yb-1], c in [0,1,2]=[R,G,B] (or [B, G, R] depending on which way around the bytes are stored.
- Advantages:
  - Easy to code.
  - Fast to compute (only look up one old pixel for each new pixel).
- Disadvantages:
  - Poor quality because each pixel is effectively made bigger.
  - The resulting image may contain jagged edges.

17.3 Linear Interpolation

Linear Interpolation – *Surveys the two closest pixels, drawing a line between them and designating a value along that line as the output pixel value.*

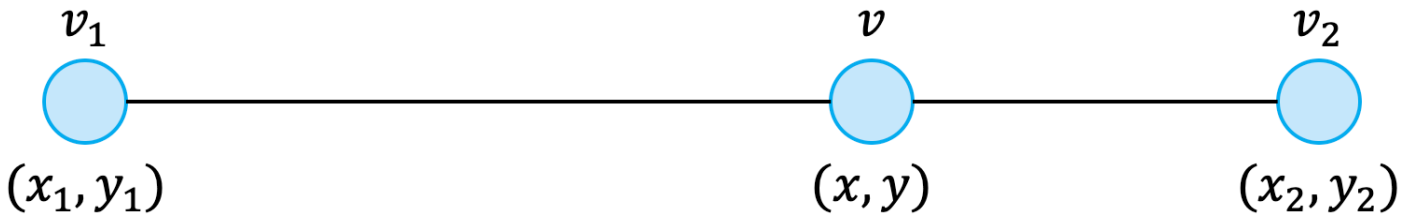


Figure 17.3.1 - Linear Interpolation Points

- Equation to find the value (colour) of a given position on the x-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(x-x_1)}{(x_2-x_1)}$
- Equation to find the value (colour) of a given position on the y-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(y-y_1)}{(y_2-y_1)}$
- Equation to find the position of a given value (colour) on the x-axis **Figure 17.3.1**:
  - $v = x + (x_2 - x_1)\frac{(v-v_1)}{(v_2-v_1)}$
- Equation to find the position of a given value (colour) on the y-axis **Figure 17.3.1**:
  - $v = y_1 + (y_2 - y_1)\frac{(v-v_1)}{(v_2-v_1)}$
- Equation explanation:
  - v, new pixel value (colour)
  - v_1, first pixel value
  - v_2, second pixel value
  - x, new pixel x-axis position
  - y, new pixel y-axis position

17.4 Bilinear Interpolation

Bilinear Interpolation – *Surveys the four closest pixels and creates a weighted average based on the nearness and brightness of the surveyed pixels and assigns that value to the pixel in the output image.*

- This function is smooth between points, but the derivative is not smooth at the boundaries.
- Performs linear interpolation first in one direction and then again in the other direction.

17.6 Bicubic Interpolation

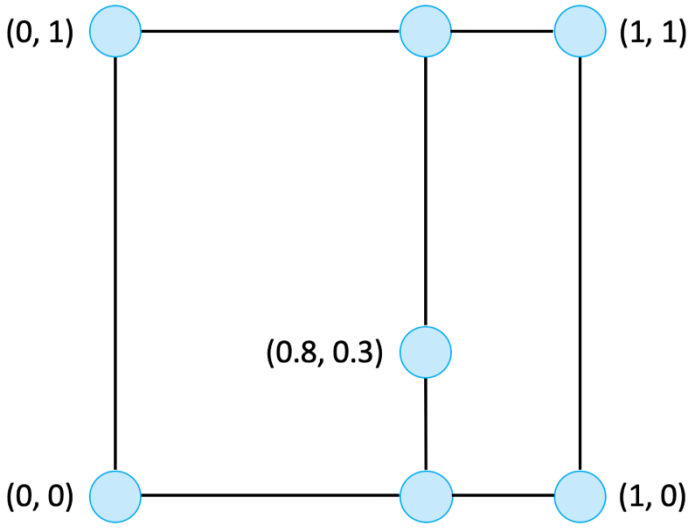
Bicubic Interpolation – *An extension of cubic interpolation for interpolating data points on a two-dimensional regular grid.*

- This function is smooth between points and the derivative is smooth at the boundaries.

18.0 – Questions and Answers

Question Reference 18.1
Exam Question May 2007

Point	Intensity
(0, 0)	0
(1, 0)	120
(0, 1)	80
(1, 1)	140



Describe linear and bilinear interpolation (giving equations where appropriate) and demonstrate their use to calculate the intensity at positions (0.8, 0) and (0.8, 0.3) in the above square.

Linear interpolation is a method of interpolation which surveys the two closest pixels, drawing a line between them and designating a value along that line as the output pixel value. Bilinear interpolation is another method of interpolation which surveys the four closest pixels and creates a weighted average based on the nearness and brightness of the surveyed pixels and assigns that value to the pixel in the output image.

- Equation to find the value (colour) of a given position on the x-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(x-x_1)}{(x_2-x_1)}$
- Equation to find the value (colour) of a given position on the y-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(y-y_1)}{(y_2-y_1)}$
- Equation to find the position of a given value (colour) on the x-axis **Figure 17.3.1**:
  - $v = x + (x_2 - x_1)\frac{(v-v_1)}{(v_2-v_1)}$
- Equation to find the position of a given value (colour) on the y-axis **Figure 17.3.1**:
  - $v = y_1 + (y_2 - y_1)\frac{(v-v_1)}{(v_2-v_1)}$

Calculating intensity for point (0.8, 0):

$$v = v_1 + (v_2 - v_1)\frac{(x - x_1)}{(x_2 - x_1)}$$
$$\therefore v = 0 + (120 - 0)\frac{(0.8 - 0)}{(1 - 0)}$$
$$\therefore v = 96$$

Calculating intensity for point (0.8, 1):

$$v = v_1 + (v_2 - v_1)\frac{(x - x_1)}{(x_2 - x_1)}$$
$$\therefore v = 80 + (140 - 80)\frac{(0.8 - 0)}{(1 - 0)}$$
$$\therefore v = 128$$

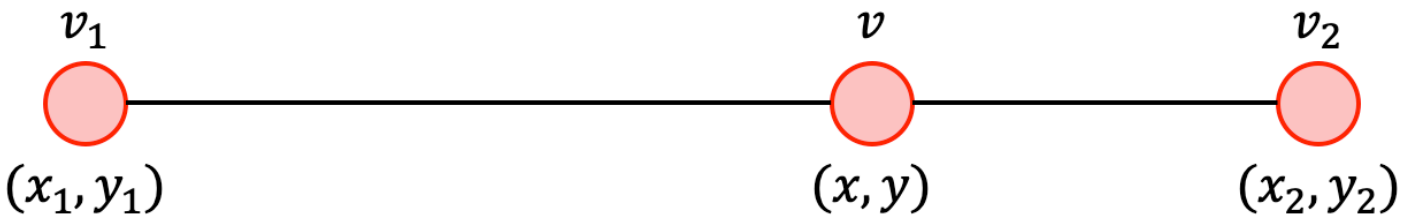
Calculating intensity for point (0.8, 0.3):

$$v = v_1 + (v_2 - v_1)\frac{(y - y_1)}{(y_2 - y_1)}$$
$$\therefore v = 128 + (96 - 128)\frac{(0.3 - 1)}{(0 - 1)}$$
$$\therefore v = 105.6$$

19.0 – Additional Notes

- ***Look-up Table –*** *A table from which all values are computed at the start so that the table can be used to retrieve values rather than performing the necessary calculation every time.*
- Look-up tables are typically used to improve speeds when modifying images.

20.0 – Equations to Learn



Linear Interpolation Equations

- Equation to find the value (colour) of a given position on the x-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(x-x_1)}{(x_2-x_1)}$
- Equation to find the value (colour) of a given position on the y-axis **Figure 17.3.1**:
  - $v = v_1 + (v_2 - v_1)\frac{(y-y_1)}{(y_2-y_1)}$
- Equation to find the position of a given value (colour) on the x-axis **Figure 17.3.1**:
  - $v = x + (x_2 - x_1)\frac{(v-v_1)}{(v_2-v_1)}$
- Equation to find the position of a given value (colour) on the y-axis **Figure 17.3.1**:
  - $v = y_1 + (y_2 - y_1)\frac{(v-v_1)}{(v_2-v_1)}$
- When performing bilinear interpolation, use the linear interpolation equations to calculate each point one at a time, and use the answers to reach further points, for example in **Question Reference 18.1**.