Visual Computing Fundamentals
Image Processing
Lecture Notes

Adrian Opheim

December 4, 2017

Contents

1 Mathematical Morphology

(Lecture 09. nov 2017)

- The pixels in the image are seen as sets
- Normally, we only use binary images.
- We use normal set operations: union, subtraction, addition, ...
- We also have reflection and translation
- Structuring elements are used to probe for structures in the image.
- Does the structuring elements fit into the structure of the foreground pixels? Does it hit?
- Fit: All pixels in the structuring element fits into the background image.
- Hit: All pixels correspond to the underlying image. (If it hits, it also fits).
- Erosion. The output pixels are set to 1 (foreground) if the structuring element fits into the image foreground pixels.
- Erosion enlarges holes, breaks thin parts and shrinks objects. **Dilation**: The output pixels are set to 1 (foreground) if the (reflected) structuring elements hits image foreground pixels.
- Dilation fill in holes, thickens thin parts, and grows object.
- Can for example fix poorly scanned text.
- **Opening**: An erosion followed by a dilation.

2 Lecture 10th Nov - Morphological Operations, some more details...

- Dilation is based on **hits**. If just one underlying pixel hits the "mask", the pixel under the "mask"'s reference point is set to 1. If not, it is set to 0.
- Dilation fills in holes, thickens thin parts and grows objects. **Labelling connected components**
- Is easily done with a binary image. Note the difference between 4- and 8-connectedness.
- Hit or miss transform: A method to find the location of a shape A in an image.
- In order to extract elements from an image:

3 Eksamen desember 2015

3.1 Theme 1: Basic Concepts

1. *Which discretizations are made when an image is captured?*
   The image acquisition technique most frequently used is sensor arrays, which can be found at the sensor of digital cameras. Here, typically 4000×4000 sensors or more are packaged in an array. The sensor array produces outputs proportional to the integral of the light received at each sensor. The output we get from most image sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. These continuous data need to be converted into digital form. That is what sampling and

quantization is all about. An image may be continuous in both x, y and amplitude. To convert it into digital form, the function most be sampled in both coordinates and amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.

We normally call the continuous image function $f(s,t)$, which will be converted into a digital image by sampling and quantization. We sample $f(s,t) \mapsto f(x,y)$, containing M rows and N coloumns where (x,y) are discrete coordinates. Integer values are used for the discrete coordinates: $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. The section of the real plane spanned by the coordinates of an image is called the **spatial domain**. L, the number of intensity levels, will be an integer power of 2: $L = 2^k$. As a rule, the upper limit of L is determined by saturation, and the lower limit by noise. **Contrast** is defined as the difference in intensity between the highest and lowest intensity levels in an image. An image with a high dynamic range will have high contrast, while an image with a low dynamic range has a dull, washed-out gray look. The number, b, of bits required to store a digitized image is

$$b = M \times N \times k \tag{1}$$

For example an image with 256 possible discrete intensity values is called an 8-bit image.

2. *How is spatial resolution related to tthe point-spread-function (PSF) of an image acquisition system? (The PSF describes the response of an imaging system to a point source.)*
A more general term for the PSF is a system's impulse response, the PSF being the impulse response of a focused optical system. In functional terms it is the spatial domain version of the optical transfer function of the imaging system.
Optical transfer function of an optical system such as a camera, microscope, human eye or porjector, specifies how different spatial frequencies are handled by the system. It is used by optical engineers to describe how the optics project light from the object or scene onto a photographic film, sensor array, retina or screen.

If the optical imaging system is in focus, the OTF will be nearly linear. If not, it will have bumps. See Figure1.

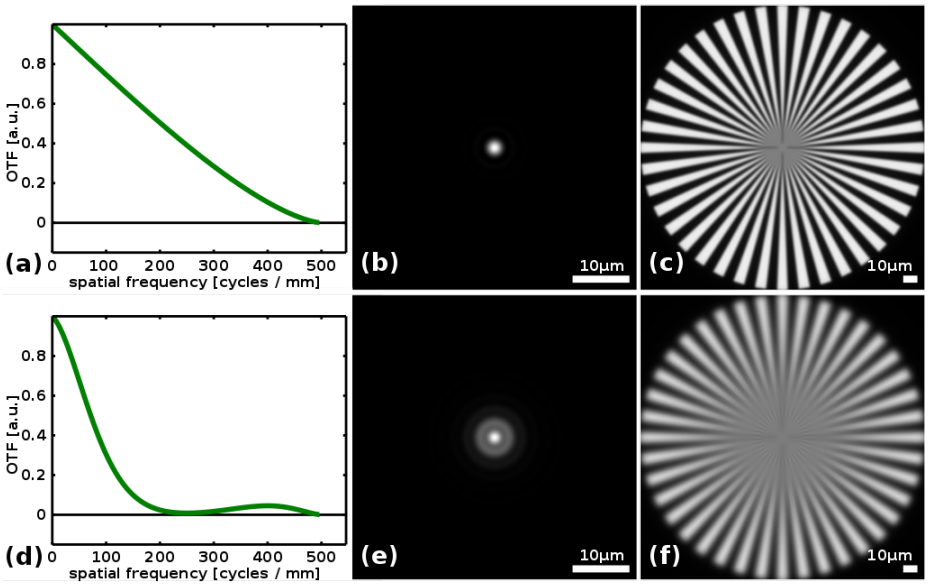


Figure 1: Illustration of the optical transfer function (OTF) and its relation to image quality. The optical transfer function of a well-focused (a), and an out-of-focus optical imaging system without aberrations (d). As the optical transfer function of these systems is real and non-negative, the optical transfer function is by definition equal to the modulation transfer function (MTF). Images of a point source and spoke target are shown in (b,e) and (c,f), respectively. Note that the scale of the point source images (b,e) is four times smaller than the spoke target images.

Given an imaging system with a 0.1mm wide PSF. What is the lowest spatial sampling frequency that can be used in order to distinguish points that are 0.1mm apart according to the sampling theorem?

The sampling theorem:
A continuous, band-limited function can be recovered completely from a set of its samples if the samples are acquired at a rate exceeding twice the highest frequency content of the function.

$$\frac{1}{\Delta T} > 2\mu_{max} \tag{2}$$

μ_{max}: the max frequency within the period.
$F(\mu)$: a continuous, periodic function with period $\frac{1}{\Delta T}$
From this, we can say that no information is lost if a continuous, band.limited function is represented by samples acquired at a rate greater than twice the highest frequency content of the function. Conversely, we can say that the *maximum* frequency that can be "captured" by sampling a signal at a rate $1/\Delta T$ is $\mu_{max} = 1/2\Delta T$

As the samples need to be acquired at a rate exceeding twice the highest frequency content of the function, we must use a spatial sampling frequency $\geq 0.2\frac{\text{mm}}{\text{s}}$ (?)

3.2 Theme 2: Histogram processing / Filtering in the spatial domain

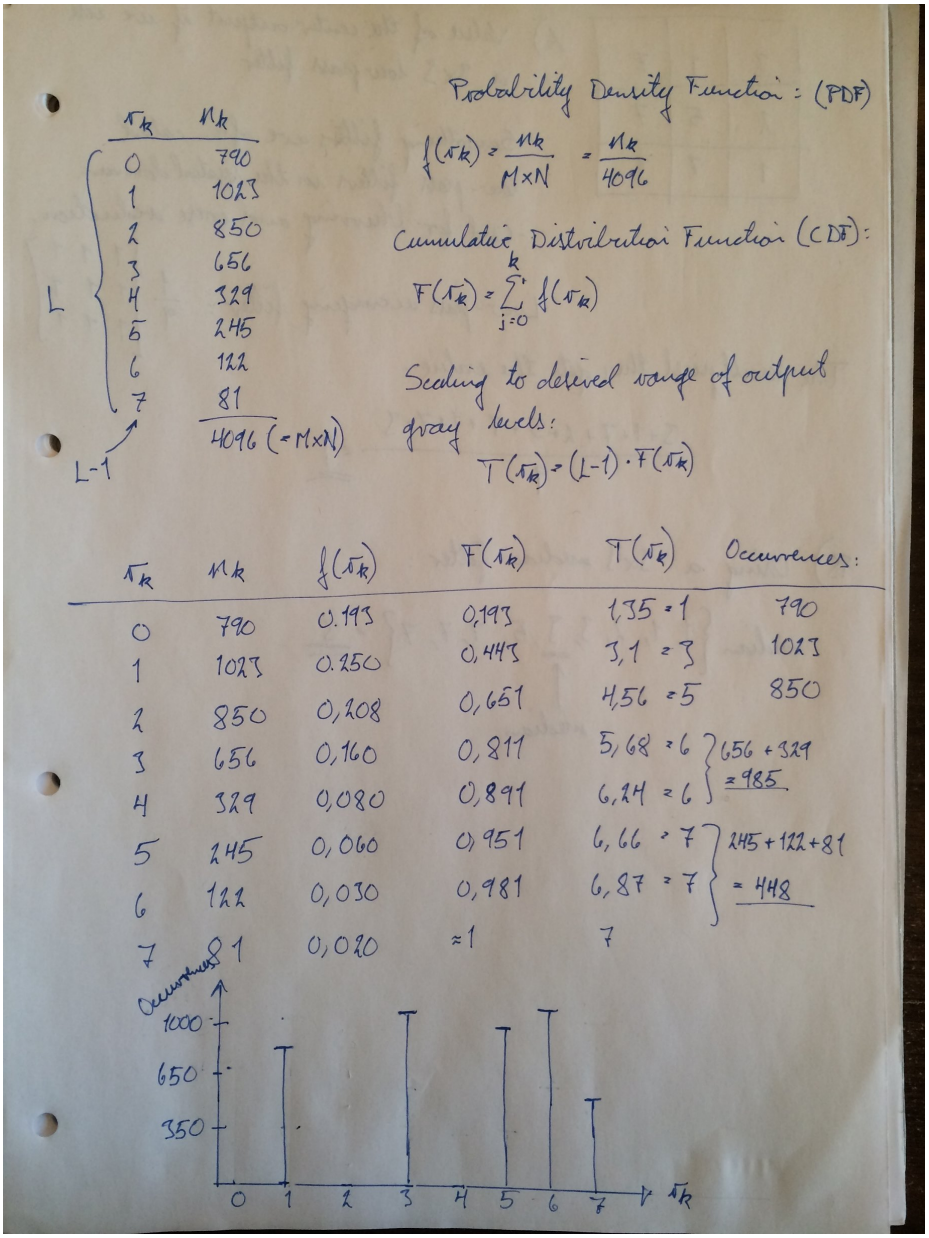


Figure 2: Example of histogram equalization

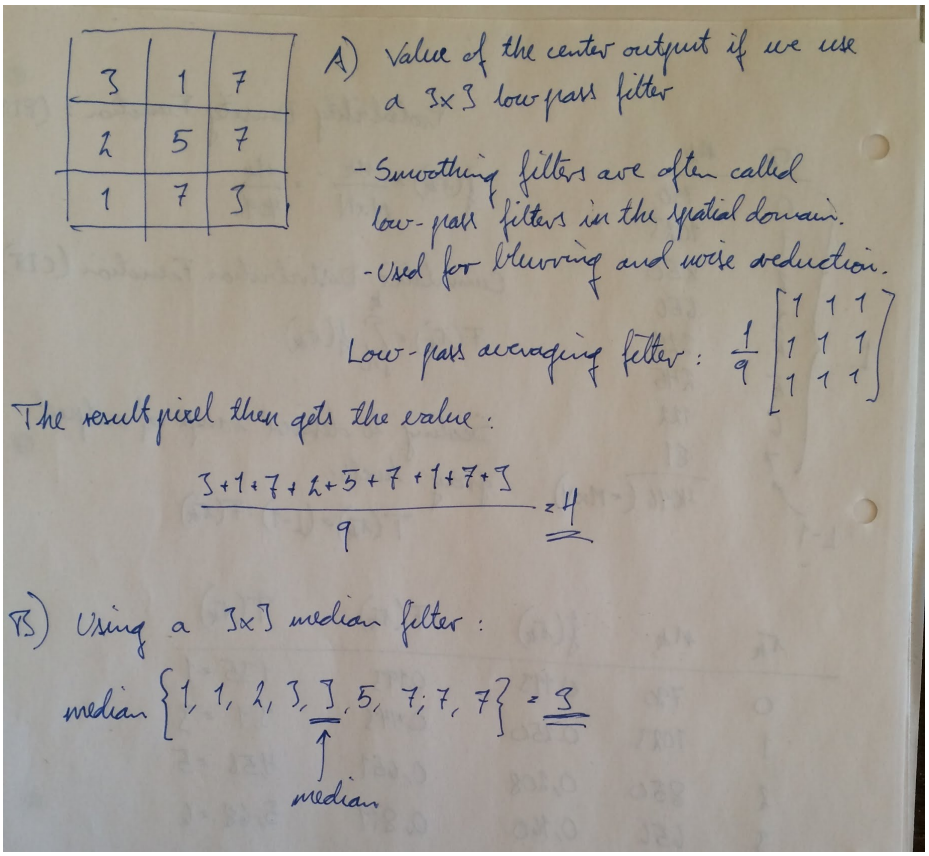


Figure 3: Filtering in the spatial domain

3.3 Theme 3: Filtering in the frequency domain

1. *Explain the convolution theorem*
   The convolution theorem states that , referring to t as the spatial domain, and μ as the frequency domain, the Fourier transform of the convolution of two functions in the spatial domain is equal to the product in the frequency domain of the Fourier transforms of the two function,

$$f(t) * h(t) \iff H(\mu)F(\mu) \tag{3}$$

   Conversely, if we have the product of the transforms, we can obtain the convolution in the spatial domain by computing the inverse Fourier transform.
   The other half of the convolution theorem states that convolution in the frequency domain is analogous to multiplication in the spatial domain,

$$f(t)h(t) \iff H(\mu) * F(\mu) \tag{4}$$

   About the transform:
   Low frequencies in the transoform are related to slowly varying intensity components in an image, such as the walls of a room or a cloudless sky in an outdoor scene. *High frequencies* are caused by sharp transitions in intensity, such as edges and noise. Therefore, a **low-pass filter** would blur an image, while a **high-pass filter** would enhance sharp detail, but cause a reduction in contrast in the image.

   Basic steps for filtering in the frequency domain:

   (i) Given an input image of size $M \times N$, obtain the padding parameters P and Q from $P \geq 2M - 1$ and $Q \geq 2N - 1$. Typically, we select $P = 2M$ and $Q = 2N$.

(ii) Form a padded image, $f_p(x, y)$ of size $P \times Q$ by appending the necessary number of zeros to $f(x, y)$.

(iii) Multiply $f_p(x, y)$ by $(-1)^{x+y}$ to center its transform.

(iv) Compute the DFT, $F(u, v)$, of the image from step 3.

(v) Generate a real, symmetric filter function, $H(u, v)$ of size $P \times Q$ with center at coordinates $(P/2, Q/2)$. Form the product $G(u, v) = H(u, v)F(u, v)$ using array multiplication; that is, $G(i, k) = H(i, k)F(i, k)$

(vi) Obtain the processed image:

$$g_p(x, y) = \left\{\text{real}\left[\mathcal{F}^{-1}\left[G(u, v)\right]\right]\right\}(-1)^{x+y} \tag{5}$$

where the real part is selected in order to ignore parasitic complex components resulting from computational inaccuracies, and the subscript p indicates that we are dealing with padded arrays.

(vii) Obtain the final processed result $g(x, y)$ by extracting the $M \times N$ region from the top, left quadrant of $g_p(x, y)$.

The most common filter types

- **Ideal lowpass filter:**

$$H(u, v) = \begin{cases} 1, & D(u, v) \leq D_0 \\ 0, & D(u, v) > D_0 \end{cases} \tag{6}$$

  - May give the "ringing" effect, which is characteristic of ideal filters.

- **Butterworth lowpass filter**:

$$H(u, v) = \frac{1}{1 + \left[D(u, v)/D_0\right]^{2n}} \tag{7}$$

  - Does not have a sharp discontinuity that gives a clear cut-off between passed and filtered frequencies.
  - Smoothes the image (removes high frequencies)
  - Has no ringing in the spatial domain for order 1. Become increasingly more ringing for higher orders. BLPFs of order 2 are a good compromise between effective lowpass filtering and acceptable ringing.

- **Gaussian Lowpass Filters (GLPF):**

$$H(u, v) = e^{-D^2(u,v)/2D_0^2} \tag{8}$$

  - No ringing in the case of GLPFs.

- **Ideal Highpass Filters (IHPF)**:

$$H(u, v) = \begin{cases} 0, & D(u, v) \leq D_0 \\ 1, & D(u, v) > D_0 \end{cases} \tag{9}$$

  - NOTE: All highpass filters can be obtained from a given lowpass filter using

$$H_{HP}(u, v) = 1 - H_{LP}(u, v) \tag{10}$$

  - Butterworth Highpass Filter and Gaussian Highpass Filter are calculated with the above formula. Gives the same ringing effects as for lowpass filters. Sharpens the images, and makes them dark (removes low frequencies).

3.4 Theme 4: Image Segmentation

1. *Explain the split and merge algorithm*
   Let R represent the entire image region and selevt a predicate Q. One approach for segmenting R is to subdivide it succesively into smaller and smaller quadrant regions so that, for any region R_i, $Q(R_i) = TRUE$. The procedure is summarized here:

   (a) Split into four disjoint quadrants any region R_i for which $Q(R_i) = FALSE$.

   (b) When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_K) = TRUE$

   (c) Stop when no further merging is possible.

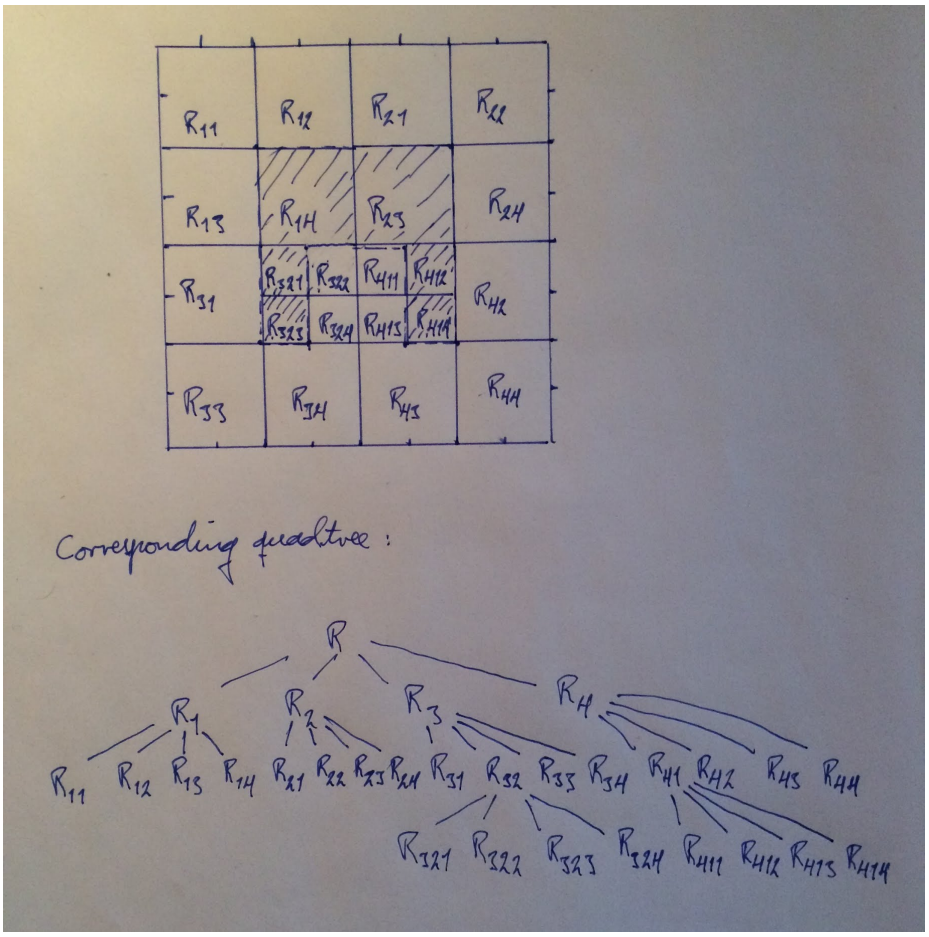


Figure 4: Resulting quadtree and splitting from the Split-and-Merge algorithm

2. *Explaing the Hough transform for finding lines in an image*

   (a) Express a point in the (x, y) spatial domain by

   $$x \cos\theta + y \sin\theta = \rho \tag{11}$$

   (b) This can then be expressed in the (ρ, θ) plane, where you get sinusoidal curves, where each curve represent the family of lines that pass through a point (x_k, y_k) to the xy-plane.

   (c) By subdividing the ρ, θ space into accumulator cells, you can count up how many curves intersect within a single cell. An intersection point corresponds to a line in the (x, y) plane, on the line $c \cos\theta_j + y \sin\theta_j = \rho_i$

3.5 Theme 5: Morphological Image Processing

1. **Erosion and dilation**:

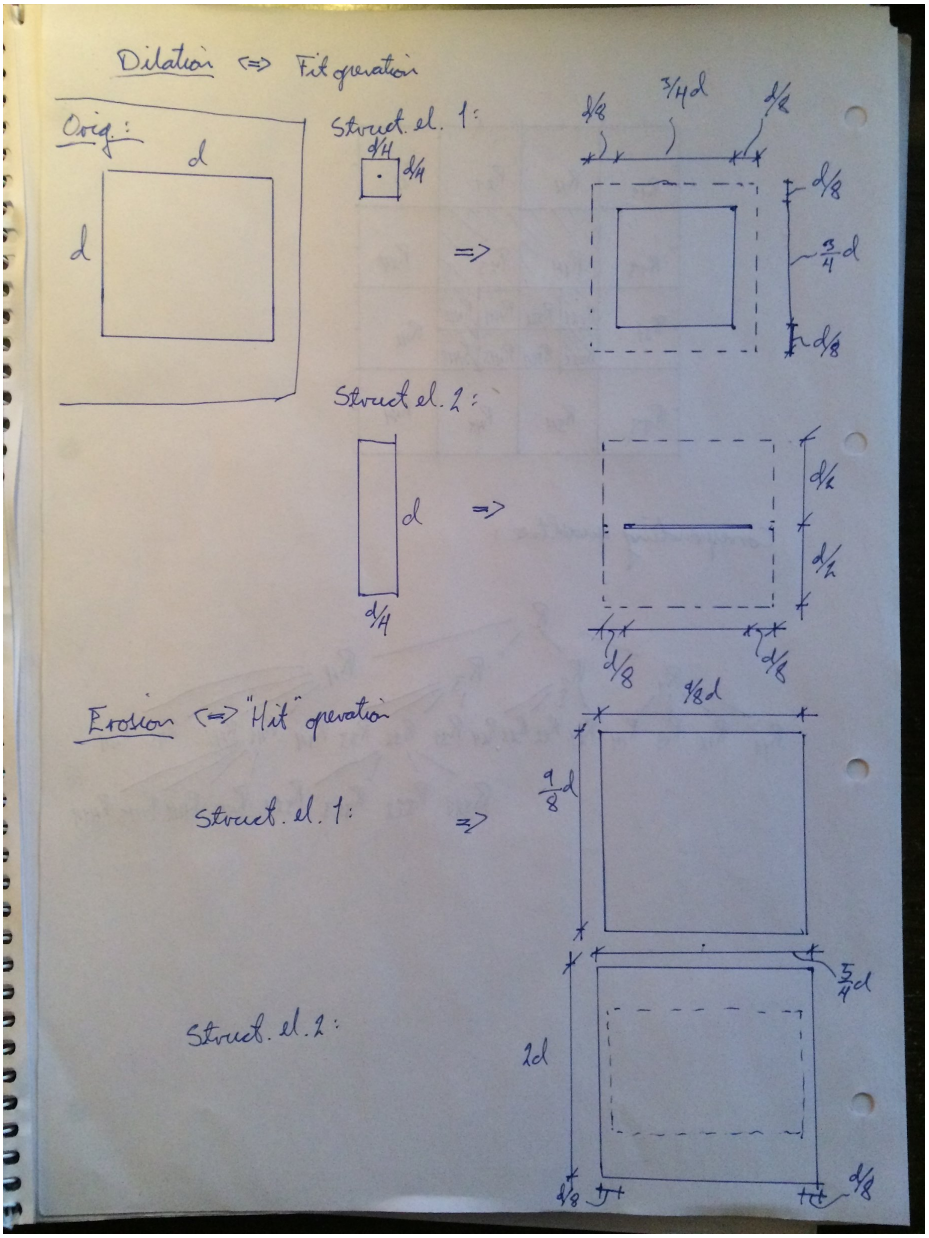


Figure 5: Result of erosion and dilation

2. *Explain how the holes can be removed as well as how the cells can be separated and counted.*

   Hole filling is done with the algorithm

   $$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3, ... \tag{12}$$

   which is the "snitt" of the previous iteraton X dilated with the structuring element B. This terminates when $X_k = X_{k-1}$

   The image can then be eroded until the cells are separated from each other. A new labelling will then be able to count the number of cells.

3.6 Theme 6: Rasterization

Suppose that you need to evaluate the function $f(x, y) = 3x + 4y^2 + 2$ over a contiguous area of the pixel grid. Show, from first principles, an efficient mathematical method for doing this. How many and what type of operations is the cost of your method?
First the equation could be expressed like

$$x = \frac{4}{3}y^2 + \frac{2}{3} \tag{13}$$

This line can be drawn using finite differences.
About rasterization

- Mathematical curves are described on two forms:
  - **Implicit form:**

$$f(x, y) = \begin{cases} < 0, & \text{implies point } (x, y) \text{ is "inside" the curve} \\ = 0, & \text{implies point } (x, y) \text{ is on the curve} \\ > 0, & \text{implies the point } (x, y) \text{ is "outside" the curve} \end{cases} \tag{14}$$

    * Line: $l(x, y) = ax + by + c = 0$
    * Circle: $c(x, y) = (x - x_c)^2 + (y - y_c)^2 - r^2 = 0$ where (x_c, y_c): the center of the circle, r: circle's radius
  - **Parametric form:**
    * Function of a parameter $t \in [0, 1]$
    * t corresponds to arc length along the curve
    * The curve is traced as t goes from 0 to 1:

$$l(t) = (x(t), y(t)) \tag{15}$$

    * Line: $l(t) = (x(t), y(t))$ where $x(t) = x_c + r\cos(2\pi t)$ and $y(t) = y_c + r\sin(2\pi t)$
- **Finite differences:**
  - Functions that define primitives need to be evaluated on the pixel grid for each pixel ⇒ wasteful
  - Forward differences:

$$\begin{aligned} \delta f_i &= f_{i+1} - f_i && \text{(first)} \\ \delta^2 f_i &= \delta f_{i+1} - \delta f_i && \text{(second)} \\ \delta^k f_i &= \delta^{k-1} f_{i+1} - \delta^{k-1} f_i && \text{(k'th order)} \end{aligned} \tag{16}$$

  - Evaluation of the **line** funciton incrementally: "From pixel (x, y) to pixel $(x + 1, y)$

$$\delta_x l(x, y) = l(x + 1, y) - l(x, y) = a \tag{17}$$

    Then compute

$$l(x, y) + \delta_x l(x, y) = l(x, y) + a \tag{18}$$

- **Bresenham's Line Algorithm**
  - Suitable for lines in the first octant
  - Meets the requirements of a good line rasterization algorithm:
    * Selection of nearest pixels to the mathematical path of the line
    * Constant line width, independent of the slope of the line

* No gaps
* High efficiency

Give two computer graphics applications where a function needs to be evaluated over a contiguous area of the pixel grid.
When expressing curves, for example Bézier curves or B-Splines.

Find 2 errors in the following line rasterization algorithm (meant for lines in the first octant):

```
1  line (int xs, int ys, int xe, int ye, color c) {
2      float x, y, e, dx, dy;
3      e = -(dx >> 1);
4      dx = (xe - xs);
5      dy = (ye - ys);
6      (x, y) = (0, 0);     // This is wrong! Must be (x, y) = (xs, ys)
7      while (x >= xe) { // Wrong! Should be (x <= xe)
8          setpixel(x, y, c);
9          x = x + 1;
10         e = e + dy;
11         if (e >= 0) {
12             y = y + 1;
13             e = e - dx;
14         }
15     }
16 } // (This is Bresenham's line algorithm)
```

3.7 Theme 7: Transformations

We want to rotate the 2D triangle defined by its homogenous vertices $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

$$\mathbf{T} = [\mathbf{ABC}] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{19}$$

by 90 degrees about vertex $\mathbf{C}$. *Derive the transformation matrix that can achieve this, explaining your answer.*

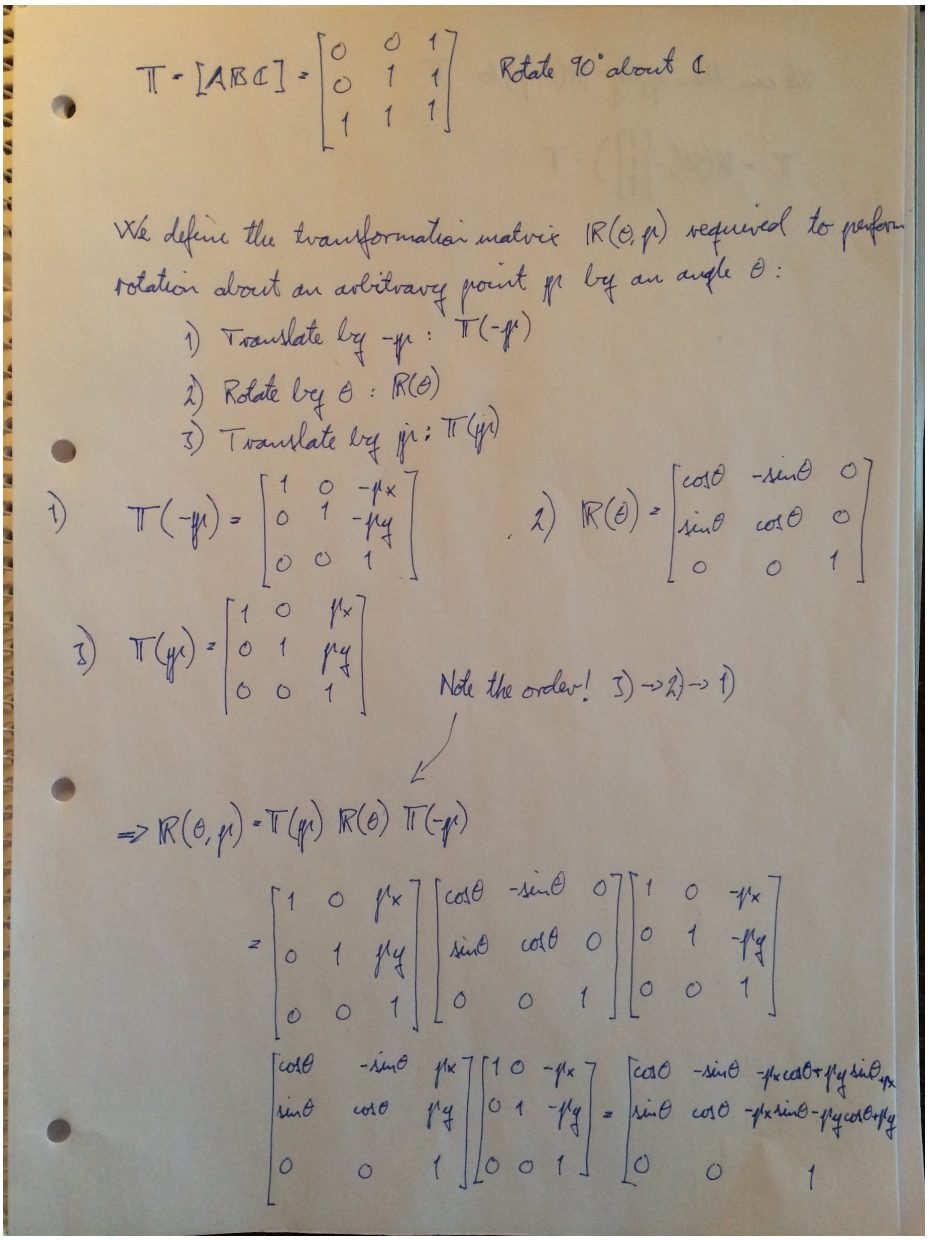


Figure 6: The derived transformation matrix

Explain why transformation matrices are post-multiplied by points (i.e. points go to the right of the transformation matrix) and why the order or composing transformation matrices is from right-to-left (this holds for the conventions made in this course).

- Matrix multiplication is not commutative: $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$, so the order of multiplying the transformation matrices is important. Transformation matrices need to be "right-multiplied" by the points when we have a coloumn representation of points.
- In order to apply the transformations in order $\mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}, \ldots \mathbf{T_n}$ we compute the composite matrix $\mathbf{T_n} \cdot \ldots \cdot \mathbf{T_2} \cdot \mathbf{T_1}$
- In order to preserve the unit value of the w coordinate, the last row of a homogenous transformation matrix is always $[0, 0, 1]$

Why can't the translation matrix be combined with other transformation matrices, unless we use homogeneous coordinates? What is the motivation behind achieving this combination of transformation matrices? Give a simple numerical example to illustrate this motivation.

- The problem with translation transformation is that translation cannot be described by a linear transformation matrix such as

$$\begin{aligned} x' &= ax + by \\ y' &= cx + dy \end{aligned} \tag{20}$$

  Because of this, translation cannot be included in a composite transform. *Homogenous coordinates* is the solution to this problem.

- As useful transformations in computer graphics and visualization rarely consist of a single basic affine transformation, transformation matrices should be combined in order to make computation more efficient.
  Example: First rotating a 2D object by 45°, $\mathbf{R}(45°)$ then isotropically scale it by a factor of 2, $\mathbf{S}(2,2)$:
  $\mathbf{S}(2,2)(\mathbf{R}(45°)\mathbf{p})$ is not efficient.
  $(\mathbf{S}(2,2)\mathbf{R}(45°))\mathbf{p}$ is much more efficient, as you only do one matrix multiplication.

3.7.1 About transformations:

- **Affine transformations:** Transformations whih preserve important geometric properties of the object being transformed. E.g.:

  Midpoint of a line segment $\mapsto$ Midpoint of the affine transformed line segment (21)

  Four basic affine transformations: translation, scaling, rotation and shearing.

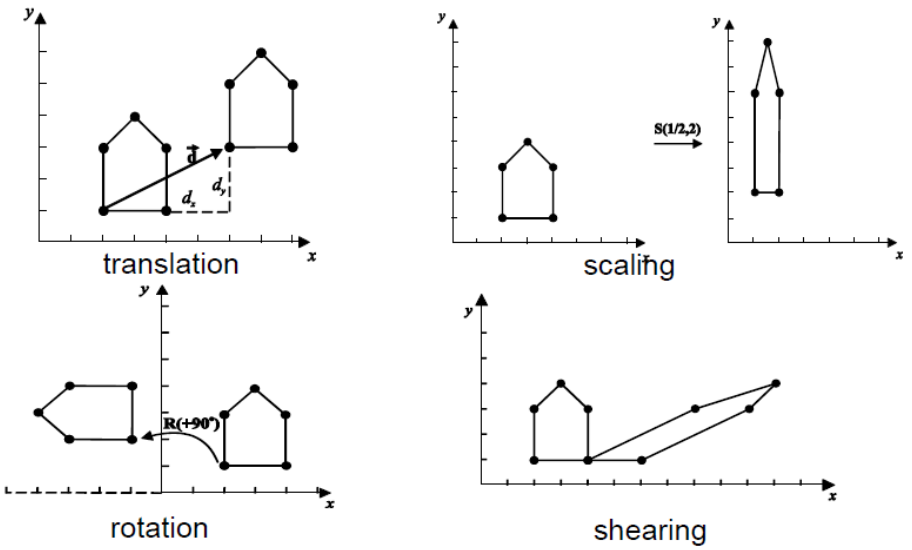


Figure 7: The four basic affine transformation

2D rotation:

– The distance from the origin does not change, only the orientation changes.

$$\mathbf{p}' = \mathbf{R}(\theta)\mathbf{p} \tag{22}$$

where

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{23}$$

Homogenous coordinates:

- Use one additional dimension that the space we want to represent
- 2D space: $\begin{bmatrix} x \\ y \\ w \end{bmatrix}$, where w is the new coordinate that corresponds to the extra dimension; $w \neq 0$
- Fixing $w = 1$ maintains our original dimensionality by taking slice $w = 1$

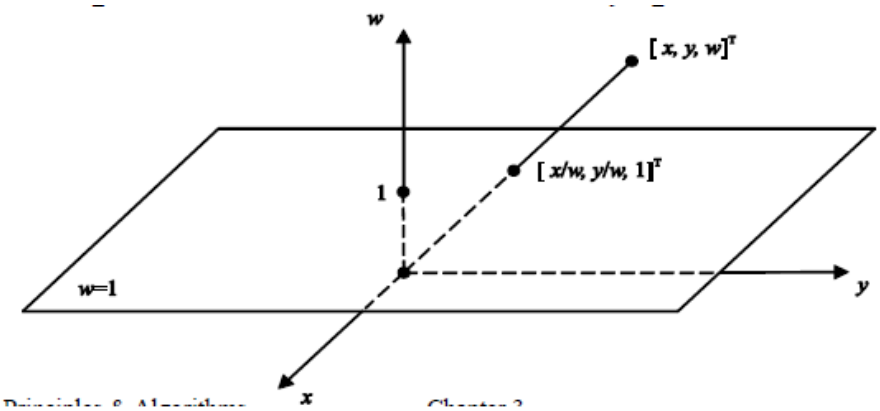


Figure 8: The (x, y, w) dimensions

- Points whose homogenous coordinates are multiples of each other are equivalent. Meaning that $[1, 2, 3]^T$ and $[2, 4, 6]^T$ represent the same point.
- The actual point that they represent is given by their unique *basic representation*, which has $w = 1$ and is obtained by dividing all coordinates by w:

$$\begin{bmatrix} x/w & y/w & w/w \end{bmatrix}^T \begin{bmatrix} x/w & y/w & 1 \end{bmatrix}^T \tag{24}$$

- The same principle is applied in 3D: $[x, y, z, 1]^T$

3.8 Theme 8: Culling and clipping

Explain what is the main purpose of culling operations in the graphics pipeline and describe 2 such operations and which coordinate system they operate in (Object, World, Eye or Screen coordinates).

- Culling algorithms remove primitives that are not relevant to the rendering of a specific frame because:
  - They are not outside the field of view → frustum culling
    Frustum culling
    - Removes primitives that are not in the field of view
    - Implemented by 3D clipping algorithms.
  - They are occluded by other objects → occlusion culling
  - They are occluded by front-facing primitives of the same object → back-face culling

The occlusion problem:

– Determination of the visible object in every part of the image.

– For correct rendering we must solve the occlusion problem, which is what we use Hidden Surface Elimination (HSE) algorithms for. HSE algorithms are classified according to their working space: the object space, or the image space.

  * General form of object space HSE:

```
1 for each primitive {
2     find visible part // (compare against all other
          primitives)
3     render visible part
4 }
```

  * General form of image space HSE:

```
1 for each pixel {
2     find closest primitive
3     render pixel with color of closest primitive
4 }
```

Back-face culling
The visible polygons of an object are those that lie on the hemisphere facing the viewer. Using this, back faces can be detected by computing the angle formed by a polygon's normal vector $\mathbf{n}$ (pointing outwards from the opaque solid) and the view vector $\mathbf{v}$.
Frustum culling
The viewing transformation defines the field of view of the observer. This field is restricted by a minimum and maximum depth value, defining a 3D solid called the view volume or *view frustum.* We want to eliminate all primitives outside the view frustum. This is done after transformation from Eye Coordinate System (ECS) to Canonical Screen Space (CSS), but before division by w for perspective projection. Frustum culling is performed in 3D space using 3D clipping.
In orthographic or parallel projection, we use the $\mathbf{M}^{ortho}_{ECS\rightarrow CSS}$ matrix which maps the clipping planes to -1 and 1 so that:

$$\begin{aligned} x_{min} = y_{min} = z_{min} = -1 \\ x_{max} = y_{max} = z_{max} = 1 \end{aligned} \tag{25}$$

In orthographic or parallel projection, the projection $\mathbf{M}^{persp}_{ECS\rightarrow CSS}$ matrix maps the clipping planes to $-w$ and w so that:

$$\begin{aligned} x_{min} = y_{min} = z_{min} = -w \\ x_{max} = y_{max} = z_{max} = w \end{aligned} \tag{26}$$

What is the relationshsip between the frustum culling operation of the graphics pipeline and clipping algorithms?
Frustum culling must be performed in 3D space using 3D clipping.

3.8.1 Culling and clipping

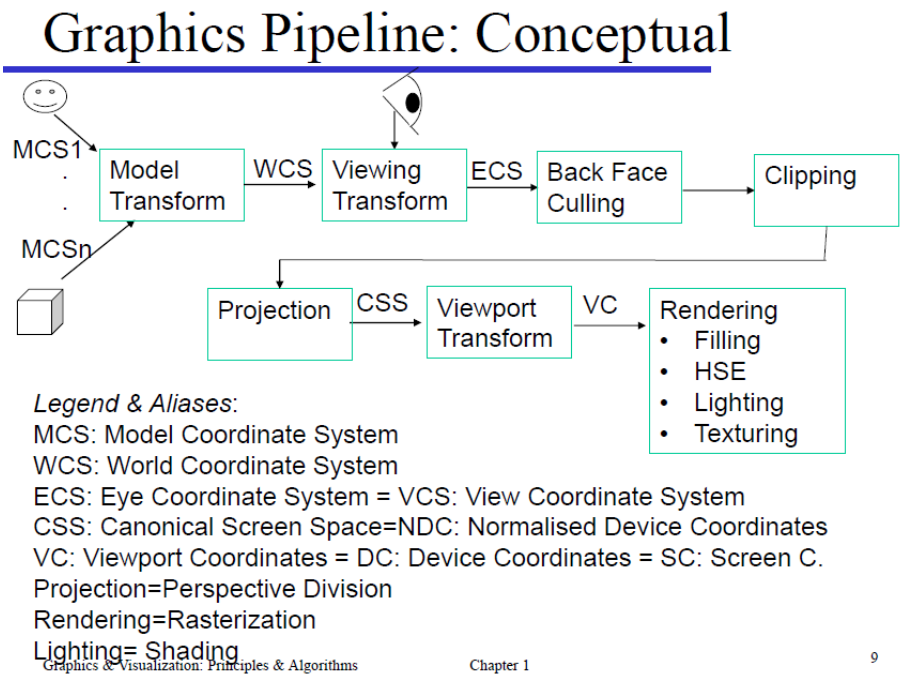


Figure 9: Overview of the graphics pipeline

3.9 Theme 9: Color

How does the human visual system perceive differences in intensity?

- The human eye perceives *intensity ratios* rather than absolute intensity values. Therefore, we should opt for a logarithmic distribution of intensity values.
- Let γ be the ratio between successive intensity values. If $\gamma < 1.01$, the human eye cannot distinguish between successive intensity values.
- Letting Φ_0 be the minimum intensity value, we can derive the relation

$$\Phi_{n-1} = \gamma^{n-1}\Phi_0 = 1 \tag{27}$$

Then, we can compute γ as

$$\gamma = \left(\frac{1}{\Phi_0}\right)^{\frac{1}{n-1}} \tag{28}$$

Using this, we could set $\gamma = 1.01$ and solve for n:

$$n = \log_{1.01}\left(\frac{1}{\Phi_0}\right) + 1 \tag{29}$$

Typical monitors have $\Phi_0 \approx 1/300$, which gives $n \approx 500$.
d bits are avaible to store the intensity component $\iff n = 2^d \iff d = \log_2(n) \approx 8.97$. This is why 8-bit representations of images normally are used.

How is this property used in High Dynamic Range (HDR) images?

- Motivation: "How do we record images in a potentially immortal format?
- Human eye has a tremendous dynamic range capabilities (10 000 : 1), while conventional displays typical dynamic range is (90 : 1).
- HDR images are produced by combining multiple images of a scene taken at different brightness levels.
- It is possible to record HDR images by increasing the bits per pixel, e.g. 32 bits per color for a total of 96 bits per pixel.
- The Just Noticeable Difference (JND) is the smallest intensity difference detectable by the human eye at a given intensity level.
- LogLuv, which is a way of saving HDR images, is composed of 32 bits:
  - 32 bits per pixel (bpp)
  - 15 bits for the intensity value
  - 1 bit for the intensity sign (negative intensity is allowed)
  - 16 bits for chromaticity.
  - To convert between real intensity value L and its (integer) stored value L_e:

$$\begin{aligned} L_e &= \lfloor c_1(\log_2 L + c_2) \rfloor \\ L &= 2^{\left[\frac{L_e}{c_1 - c_2}\right]} \end{aligned} \tag{30}$$

What is the difference between the CMY and the CMYK colour model? What are the motivations behind CMYK?

About color models:

- **Device-independent:**
- The coordinates of a color will represent a unique color. Useful for conversion between device-dependent color models.
- Example: The CIE XYZ model
- **Device-dependent**:
- The same color coordinates may produce a slightly different visible color value on different display devices.
- Examples: RGB, CMY models.
- Some models follow a device's philosophy of producing arbitrary colors from the primary colors:
  - Additive model: adds the contributions of primaries (monitor)
  - Subtractive model: Resembles the working of a painter/printer. Color mixing is achieved through a subtractive process.

3.9.1 The RGB color model

- Additive color model with basic colors Red, Green and Blue (RGB). Chosen because human vision is based on RGB color-sensitive cells. An arbitrary color **F** is expressed as:

$$\mathbf{F} = r \cdot \mathbf{R} + g \cdot \mathbf{G} + b \cdot \mathbf{B} \tag{31}$$

- On computer diplays, colors are created using an additive method. The additive color mixing start with black (no light). Ends with white (the sum of all basi colors). As more color is added, the result is lighter and tends to white.

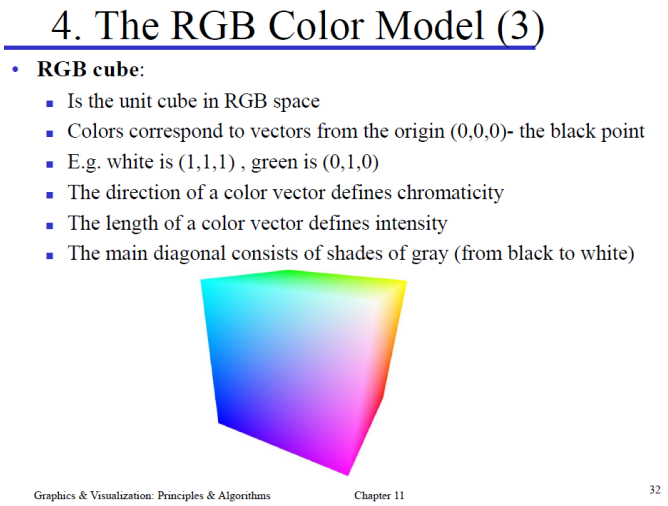


Figure 10: The RGB cube

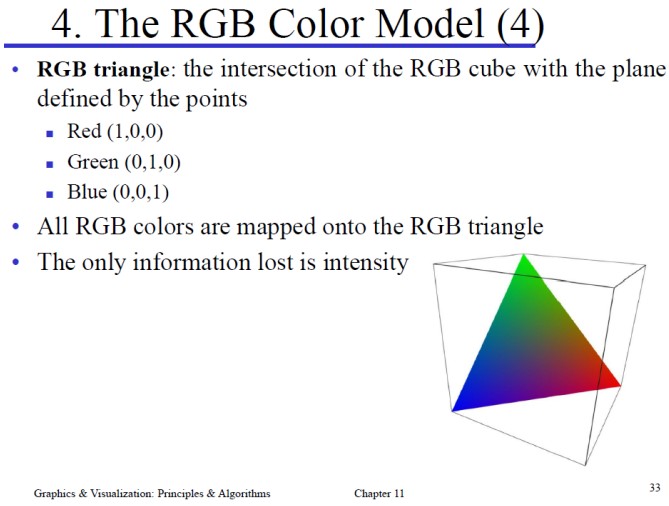


Figure 11: The RGB triangle

- Chromaticity is splitted into:
  Hue:
  - Equals the dominiant wavelength
  - Gives a color its identity
  - All hues are found on the perimeter of the RGB triangle

Saturation:

- Equals the amount of white that is present in a color
- Maximum at the center of the triangle
- Minimum at its perimeter

- In order to properly display a color on display, the RGB color is often converted to the device-independent CIE XYZ model.
- RGB models typically use 8 pits per color channel → 24 bits per pixel (bpp). As the computer words are 32 bits, the remaining 8 bits represent the **alpha value.** , a
- a represents the "area" in which the energy of the color is held.

3.9.2 The CMY(K) model

- Is a subtractive color model. Used during painting or printing, when colors are mixed. The mixing starts with white, and as one adds color, the result gets darked and tends to black.
- The CMY model is the complement of RGB. Its basic colors are cyan, magenta and yellow.
- A color **F** is expressed as a linear combination of the basic colors:

$$\mathbf{F} = c \cdot \mathbf{C} + m \cdot \mathbf{M} + y \cdot \mathbf{Y} \tag{32}$$

where c, m, y are the color coordinates of **F**.

The **CMY cube**:

- Is the unit cube in CMY space
- White appears at (0, 0, 0)
- Black is at (1, 1, 1)
- Other colors are in opposite vertices of those on the RGB cube

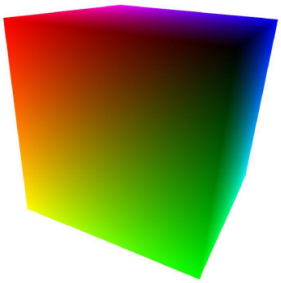


Figure 12: The CMY cube

The CMYK color model

- Is a derivative of CMY that includes black.
- Black is used to offset the color composition process by the minimum components of a color **F**. **Conversion from CMY to CMYK**:

$$\begin{aligned} b &= min(c, m, y) \\ c' &= \frac{c - b}{1 - b} \\ m' &= \frac{m - b}{1 - b} \\ y' &= \frac{y - b}{1 - b} \end{aligned} \tag{33}$$

c', m', y', b are then the components of CMYK.

- Most printers include black ink in addition to cyan, magenta and yellow. This is to economize the use of black, as you are able to represent the CMY color using the CMYK model with less CMY and more K (black).

4 Exam 2016 (digital exam)

4.1 Q1: Human Eye

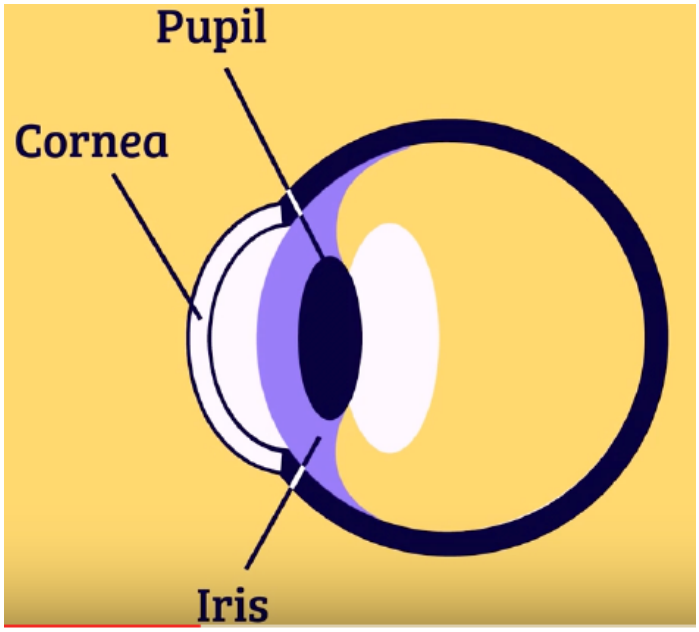


Figure 13: The human eye

- The retina contains both rods and cones. 120 million rods, 7 million cones. Rods are sensitive to light. Cones detect color.

4.2 Q2: Image formation

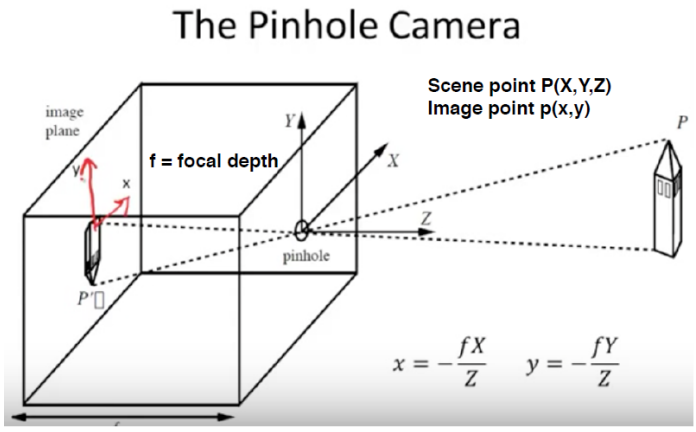
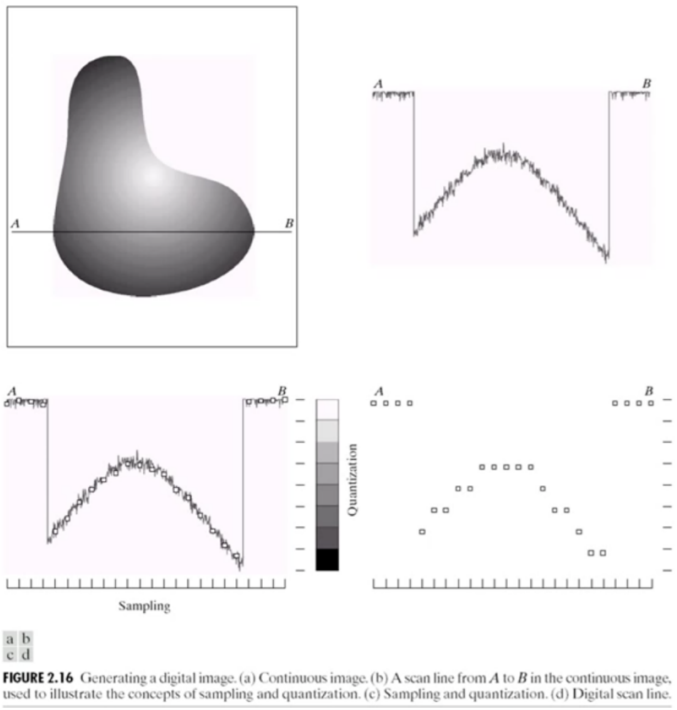


Figure 14: The pinhole camera model

4.3 Q3: Digital Images

- For creating a digital image, the two processes involved are sampling and quantization



a b
c d

FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

Figure 15: Image sampling and quantization

- **Spatial resolution:** A measure of the smallest discernible detail in an image.
- **Image resolution**: Can be expressed in multiple ways. Dots per unit distance (dpi) is most commonly used.
- **Image size:** Normally referring to the number of pixels an image holds. A digital image with M rows and N coloumns holds MN pixels.
- 3 bpp gives $2^3 = 8$ gray levels. 8 bpp gives $2^8 = 256$ gray values.
- Storing a 8 bit per channel RGB image containing 724 pixels:

$$\Rightarrow 24\frac{\text{bits}}{\text{pixel}} \cdot (1024 \cdot 1024)\text{pixels} = 25.17 \times 10^6\text{bits} = 3.15 \times 10^6\text{bytes} = 3.15\text{MB} \qquad \begin{matrix}\text{8 bit per channel} \cdot 3 \text{ channels}\end{matrix} \tag{34}$$

4.4 Q4: Intensity Transformations

a) *What do we mean by an intensity transformation and how does it look like mathematically?*

When using a kernel of size (1×1), g depends only on the value of a single point (x, y). Then, $T[f(x, y)]$ becomes an *intensity transformation function* of the form $s = T(r)$. Here, s and r are denoting, respectively, the intensity of g and f at any point (x, y).

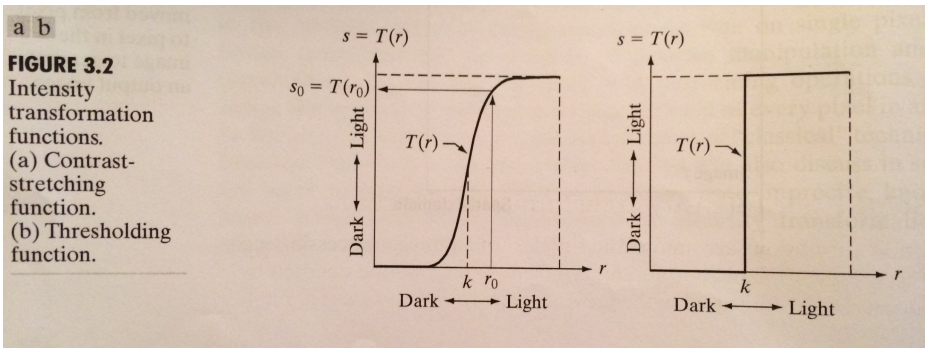


Figure 16: Intensity transformations

b) *Explain b1) the logarithmic and b2) the gamma transformations by providing the formulas, illustrating the transformations graphically (including the "inverse" transformations) and briefly say what happens to an image if the transformations are applied.*

4.4.1 Image negative

The negative of an image with intensity levels in the range $[0, L-1]$ is obtained by the negative transformation

$$s = L - 1 - r \tag{35}$$

This produces the equivalent of a photographic negative.

4.4.2 Log Transformations

The general form of the log transformatin is

$$s = c \log(1 + r) \tag{36}$$

where c is a constant and it is assumed that $r \geq 0$. We use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values.

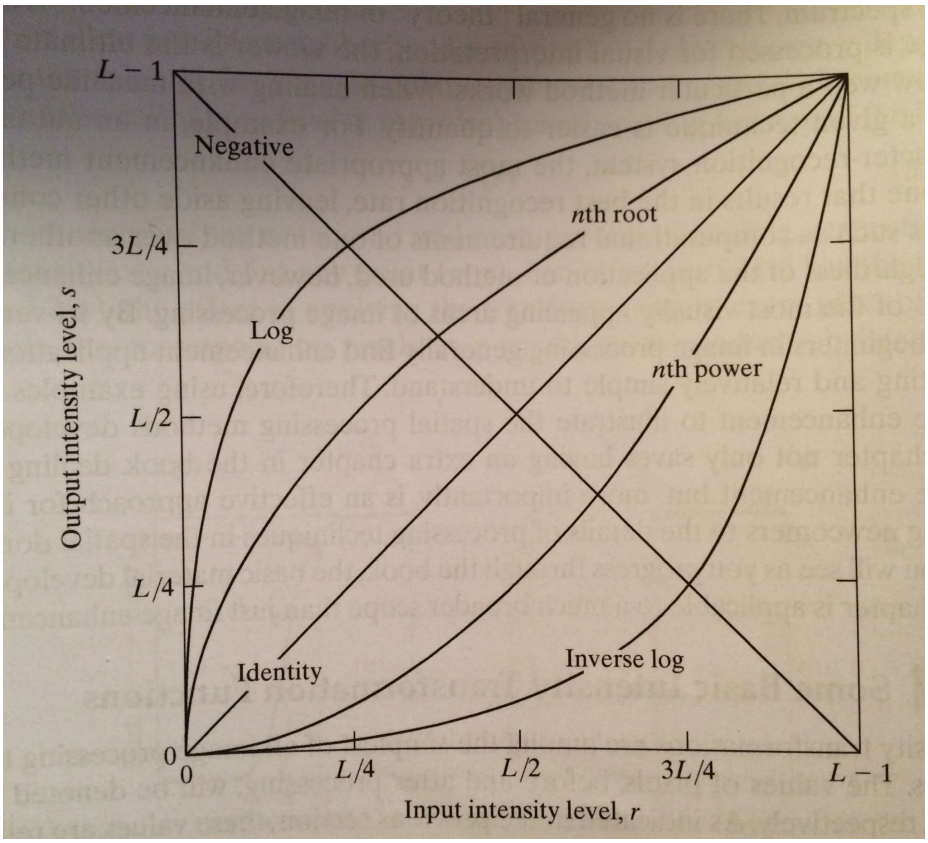


Figure 17: Some basic intensity transformations

The log transformation is typically used in the Fourier spectrum, where there is a large variation of pixel values. This can be in a range from 0 to 10^6, or even higher.

4.4.3 Power-law (Gamma) Transformations

These transformations are on the basic form

$$s = cr^{\gamma} \tag{37}$$

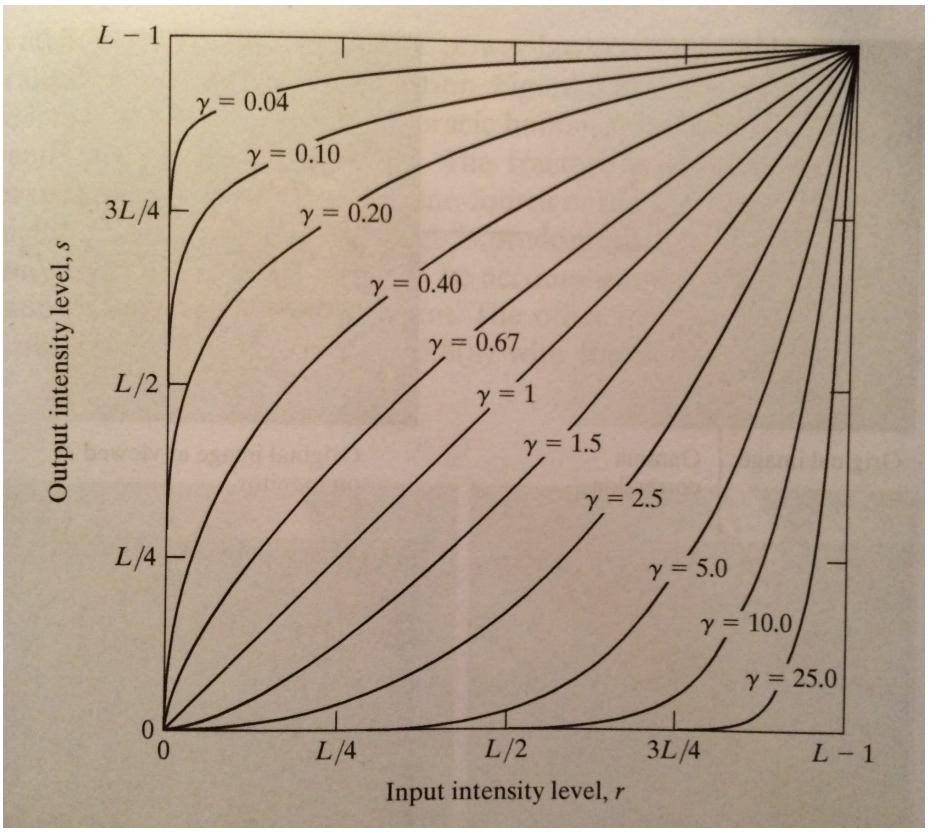


Figure 18: The gamma transformation with variying γ values

Gamma correction is important if displaying an image accurately on a computer screen is of concern. Here, the image is often washed out or is too dark. Gamma correction will adjust this.

c) *Looking at the figure above, c1) what happens to an image when low values are decreased and high values are increased as illustrated in the graph to the left. c2) Or the other way around, i.e. increasing low and decreasing high values, as illustrated to the right. c3) Also, what happens if we decrease/increase to much in the two cases.*

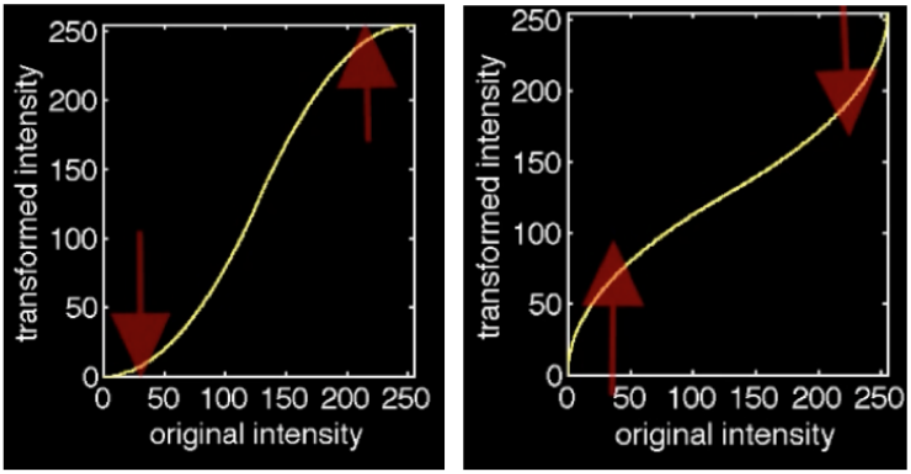


Figure 19: Images to 4c). Example of intensity stretching.

The left image at Figure 19 is an example of intensity stretching. This will increase contrast in

the image. Doing the opposite will result in reducing contrast in the image. Doing to much of the left image in Figure 19 gives a binary image. Doing to much in the right image gives a grey image (only grey).

4.5 Q5: Histogram Methods

a) *Briefly describe the four images that corresponds to the four histograms above (1(left)-4(right)*

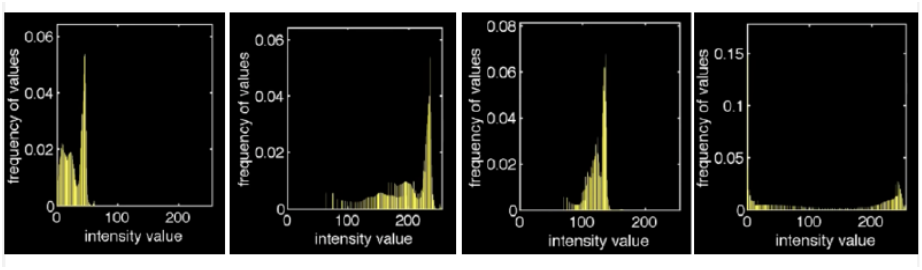


Figure 20: Images to q5

Image 1: Dark image, where all pixels are in the range "black to grey". Low contrast.
Image 2: Light image, with many white areas.
Image 3: Low-contrast image. Gray.
Image 4: High-contrast image.

b) *Cumulative histogram for each of the four cases:*

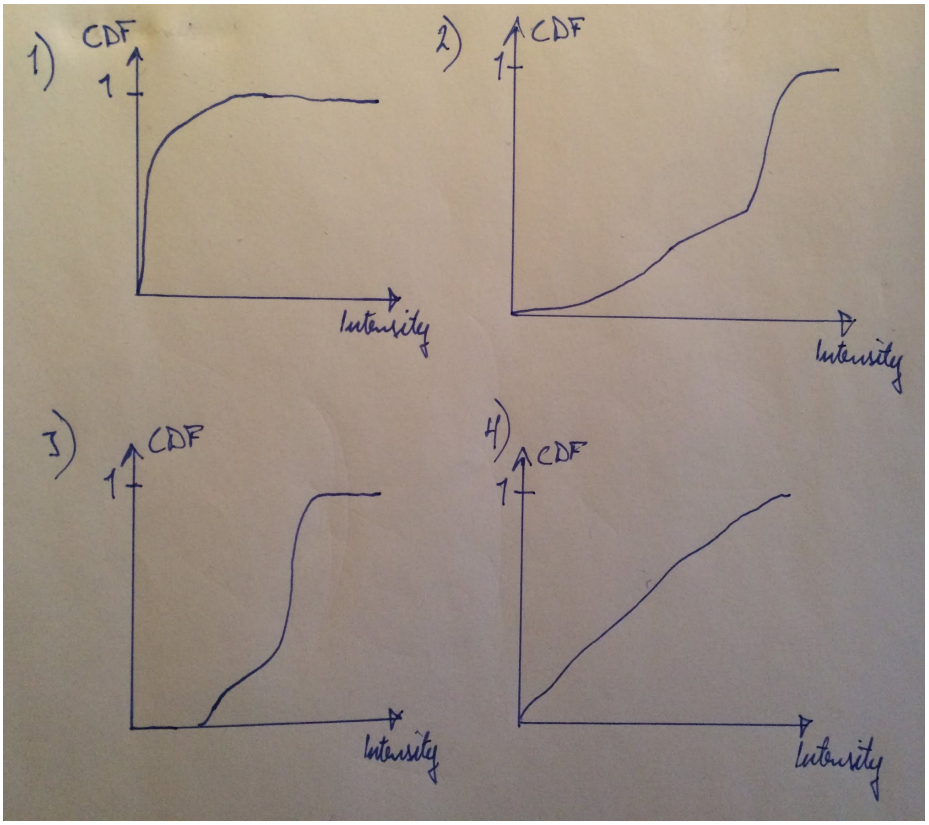


Figure 21: Cumulative histogram

c) *Shortly describe the trnasformations needed to histogram equalize each case, i.e. what do we need to do with the original intensities in the image.*

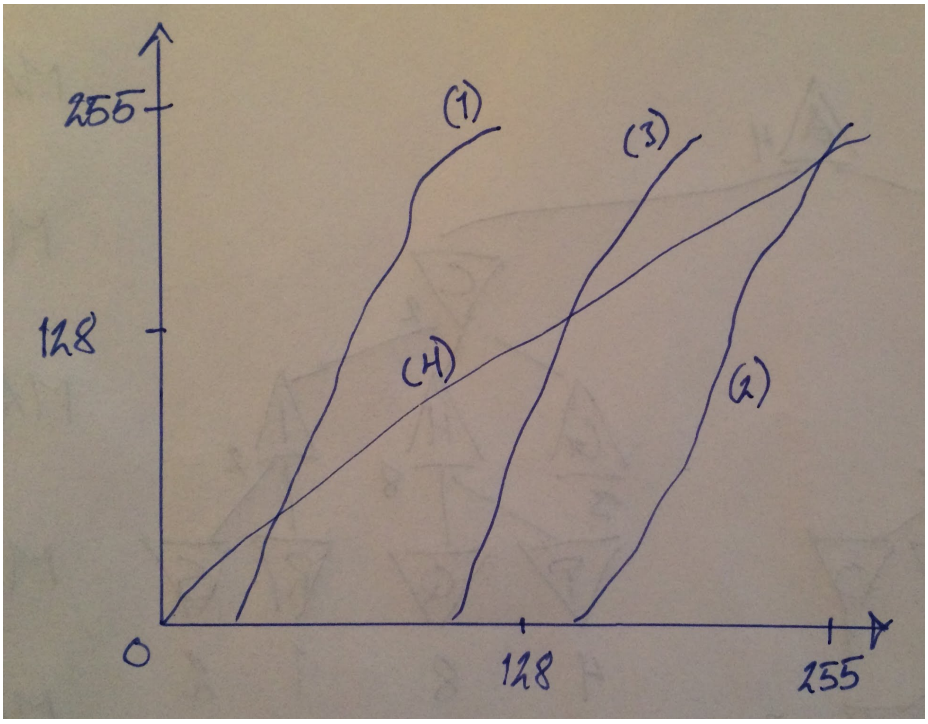


Figure 22: Approximate histogram equalization functions needed for the images in Figure 20.

4.6 Q6: Correlation and Convolution

5 Exam December 2014

5.1 Theme 3: Morphological Image Processing

a) *Mathematical morphology is based on set theory. Name the 6 set operations seen in the figure below.*

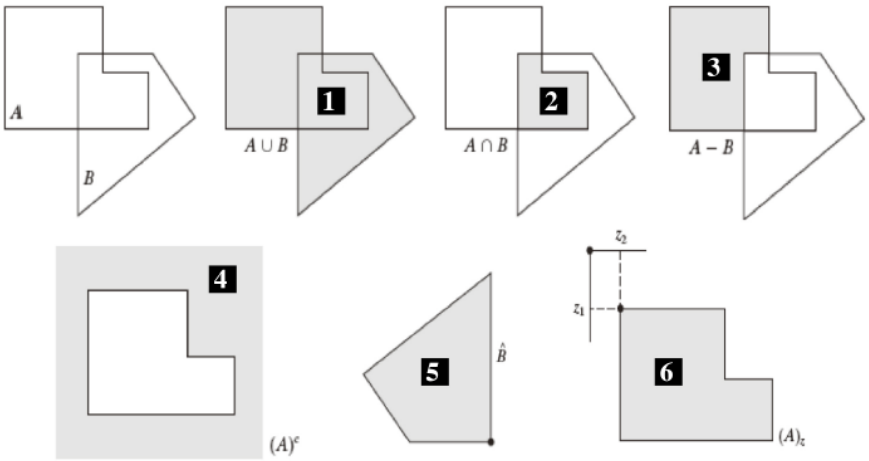


Figure 23: Set theory in morphological operations

- 1: Union
- 2: "Snitt"
- 3: Subtraction
- 4: Complement
- 5: Reflection
- 6: Translation

b) *Give the mathematical definitions for dilation and erosion (using symbols from set theory and morphology). Explain the two methods using the "hit" and "fit" operations.*
Erosion: Erosion is denoted $A \ominus B$:

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{38}$$

This indicates that the erosion of A by B is the set of all points z such that B, translated by z, is contained in A. Here, set B is seen as the structuring element. Erosion is equivalent to the "fit" operations. A pixel is not suppressed if the whole structuring element fits in the original image.

Dilation: The dilation of A by B is denoted

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \varnothing \tag{39}$$

The dilation of A by B then is the set of all displacements z such that $\hat{B}$ and A overlap by at least one element. This is equivalent to the "hit" operation.

c) *Give the mathematical definitions for opening and closing*

Opening: Generally smooths the contour of an object, breaks narrow isthmuses and eliminates protrusions.

$$A \circ B = (A \ominus B) \oplus B \tag{40}$$

Thus, it is the erosion of A by B, followed by a dilation of the result by B.

Closing: Tends to smoothes sections of contours of an object, fuses narrow breaks and long thin gulfs, eliminates small holes and fills gaps in the contour.

$$A \bullet B = (A \oplus B) \ominus B \tag{41}$$

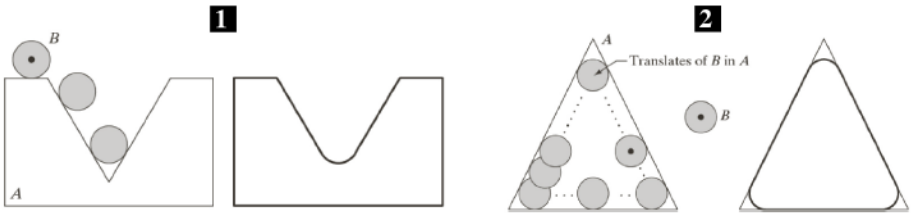


Figure 24: Image 1 is the closing of A. Image 2 is the opening of A

d) *Iterative procedure for extracting connected components:*

$$X_k = (X_{k-1} \oplus B) \cap A \quad k = 1, 2, 3, \ldots \tag{42}$$

5.2 Theme 4: Image Segmentation

1. Approximating first and second derivatives

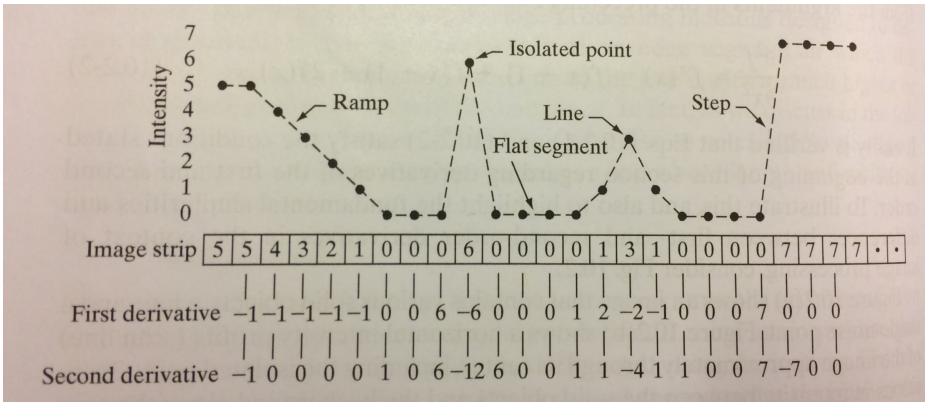


Figure 25: From the book.

This is from the book. They have used

$$\begin{aligned} f'(x) &= f(x+1) - f(x) \\ f''(x) &= f(x+1) + f(x-1) - 2f(x) \end{aligned} \tag{43}$$