MID TERM REPETITION

Ole-Johan Skrede

15.03.2017

INF2310 - Digital Image Processing

Department of Informatics
The Faculty of Mathematics and Natural Sciences
University of Oslo

TODAY'S LECTURE

- Optical systems
  - Rayleigh criterion
  - Object-image relation

TODAY'S LECTURE

- Optical systems
  - Rayleigh criterion
  - Object-image relation
- Neighbourhood operations and filters
  - Convolution
  - Low-pass filtering
  - High-pass filtering
  - Edge-detection filtering

TODAY'S LECTURE

- Optical systems
  - Rayleigh criterion
  - Object-image relation
- Neighbourhood operations and filters
  - Convolution
  - Low-pass filtering
  - High-pass filtering
  - Edge-detection filtering
- Color imaging
  - Color models and spaces
  - Color images

OPTICAL SYSTEMS

POINT SOURCE

- We will study the light ray trajectory from a point source.
- A point will be imaged as a ”blurred” version of itself.
- The reason for this is two unrelated phenomena: *abberation* and *diffraction*.

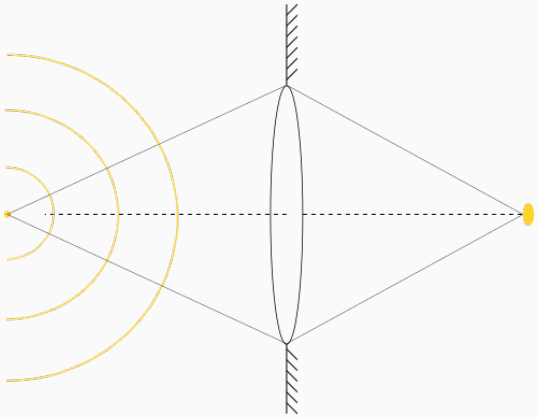
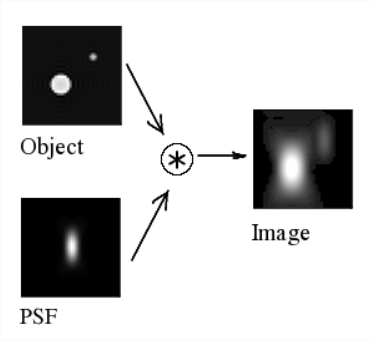


Figure 1: Light wave propagation from point source through circular aperture.

POINT SPREAD FUNCTION

- The point spread function (psf) describes the response from of an imaging system on a point source.
- The resulting image is obtained by convoluting the source with the point spread function.
- Can be used to measure the blurring of a point object, and is a measure of the quality of the imaging system.



OPTICAL ABBERATION

- Distortion in the image formed by an optical system.
- Arise because of limitations in optical components such as lenses and mirrors.
- There are many types of optical abberations, and they can to a large extent be repaired.

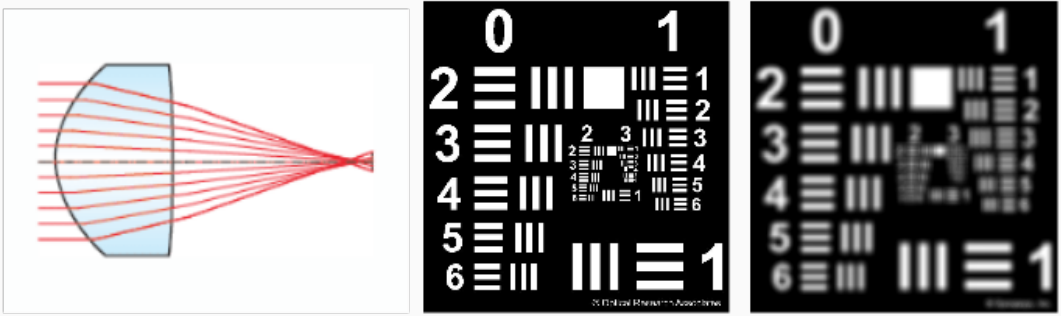


Figure 3: Spherical abberation. Rays from the edge of the lens focus points along a line in stead of at a single point. (Source: `http://www.edmundoptics.com`)

DIFFRACTION

- Caused by wavefronts of propagating waves bending in the neighborhood of obstacles.
- A fundamental limit in imaging.
- Possible to ”avoid” this limit with super-resolution imaging.

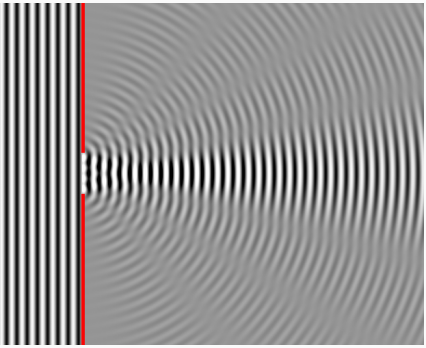


Figure 4: Illustration of diffraction pattern from a slit of width four wavelengths with an incident plane wave. (Source: By Dicklyon at English Wikipedia - Transferred from en.wikipedia to Commons by Shizhao using CommonsHelper., Public Domain, https://commons.wikimedia.org/w/index.php?curid=5699291)

DIFFRACTION LIMITED SYSTEM

- The abberation in a system can often be handled by adjusting the optics.

DIFFRACTION LIMITED SYSTEM

- The abberation in a system can often be handled by adjusting the optics.
- The diffraction is a physical limit due to the wave propagation property of light.

DIFFRACTION LIMITED SYSTEM

- The abberation in a system can often be handled by adjusting the optics.
- The diffraction is a physical limit due to the wave propagation property of light.
- A system is said to be *diffraction limited* if diffraction is the dominating the point spread function. That is, if the spatial resolution is as good as the instrument's theoretical limit.

DIFFRACTION LIMITED SYSTEM

- The abberation in a system can often be handled by adjusting the optics.
- The diffraction is a physical limit due to the wave propagation property of light.
- A system is said to be *diffraction limited* if diffraction is the dominating the point spread function. That is, if the spatial resolution is as good as the instrument's theoretical limit.
- The resulting image from a point source imaged from a diffraction limited optical instrument is termed a *Airy disk*.

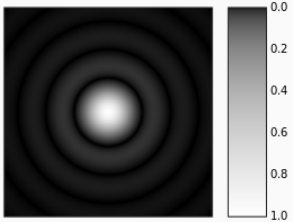


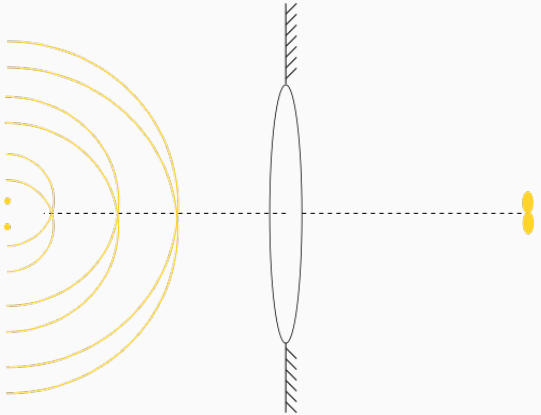
Figure 5: Airy disk

SPATIAL RESOLUTION

- Spatial (angular) resolution describes the ability of an imaging system to distinguish details in an image.

SPATIAL RESOLUTION

- Spatial (angular) resolution describes the ability of an imaging system to distinguish details in an image.
- Because of the blurring, described by the point spread function, there is a limit to the resolving ability of an image.

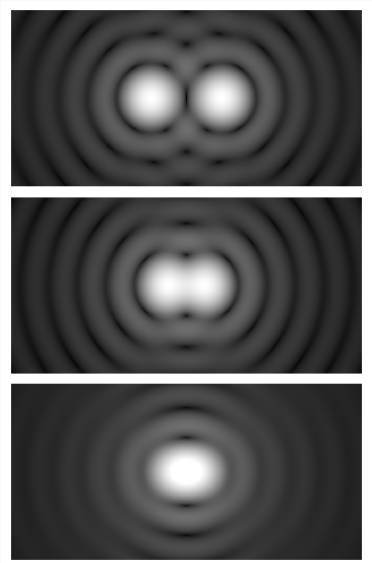


RAYLEIGH CRITERION

- The Rayleigh criterion describes the smallest distance between two light point sources that we need in order to obseve them as separate objects.

RAYLEIGH CRITERION

- The Rayleigh criterion describes the smallest distance between two light point sources that we need in order to obseve them as separate objects.
- The Rayleigh criterion: Two objects are just dissolved when the mode of the intensity function from one of the objects overlaps with the first zero of the second object's intensity function.



FRAUENHOFER DIFFRACTION PATTERN FOR A CIRCULAR APERTURE

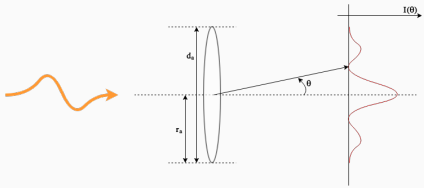


Figure 8: Diffraction pattern from circular aperture.

$$I(\theta) \propto \left[\frac{2J_1(kr_a \sin\theta)}{kr_a \sin\theta}\right]^2 \tag{1}$$

- J_1: A *first order Bessel function of the first kind*
- $k = \frac{2\pi}{\lambda}$: Wave number of the propagating light
- λ: Light wave length.
- r_a: Aperture radius.
- θ Angle to the first zero.

RAYLEIGH CRITERION EQUATION

- The intensity function in eq. (1) is zero when the Bessel function is zero.

[1] Bessel functions are solutions to a particular set of differential equations, and the mathematics is somewhat complicated. But feel free to stydy them on your own if you want to understand how the roots are found.

RAYLEIGH CRITERION EQUATION

- The intensity function in eq. (1) is zero when the Bessel function is zero.
- The first zero of a Bessel function[1] $J_1(x)$ occurs at $x \approx 3.8317$,

[1] Bessel functions are solutions to a particular set of differential equations, and the mathematics is somewhat complicated. But feel free to stydy them on your own if you want to understand how the roots are found.

RAYLEIGH CRITERION EQUATION

- The intensity function in eq. (1) is zero when the Bessel function is zero.
- The first zero of a Bessel function[1] $J_1(x)$ occurs at $x \approx 3.8317$,
- Thus

$$kr_a \sin\theta \approx 3.8317$$

or

$$\sin\theta \approx 1.22\frac{\lambda}{d_a} \tag{2}$$

which is famously known as the *Rayleigh criterion*.

[1] Bessel functions are solutions to a particular set of differential equations, and the mathematics is somewhat complicated. But feel free to stydy them on your own if you want to understand how the roots are found.

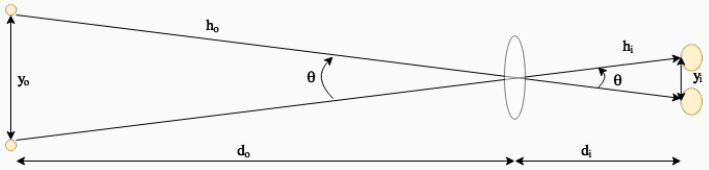


Figure 9: Schematic figure of distances.

For very small θ, $\sin\theta \approx \theta$. We can then approximate eq. (2) to

$$\theta \approx 1.22\frac{\lambda}{d_a}$$

and furthermore, $h_o \approx d_o$, yielding

$$y_o \approx 1.22\frac{\lambda}{d_a}d_o$$

This is the smallest distance we are able to dissolve in an image.

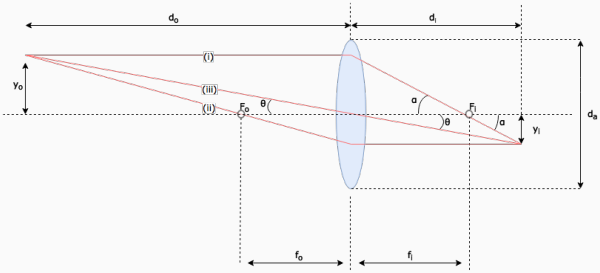


Figure 10: Idealised imaging system

(i) An incident ray which is parallel to the optic axis is refracted through the imaging focal point F_i.

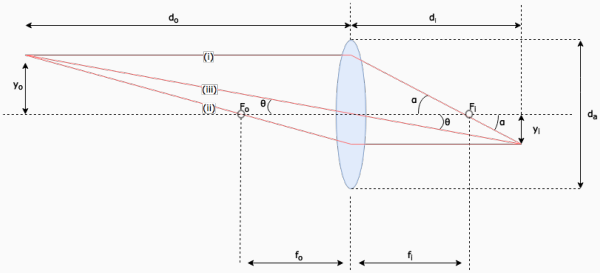


Figure 10: Idealised imaging system

(i) An incident ray which is parallel to the optic axis is refracted through the imaging focal point F_i.

(ii) An incident ray which passes through the object focal point F_o, is refracted parallel to the optic axis.

PARAXIAL RAY APPROXIMATION

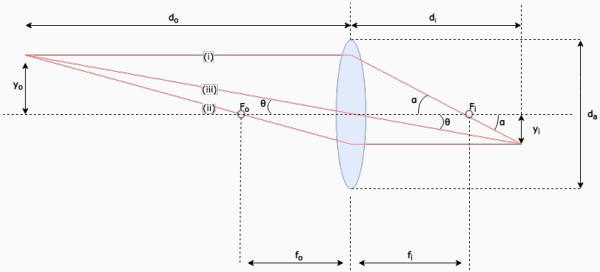
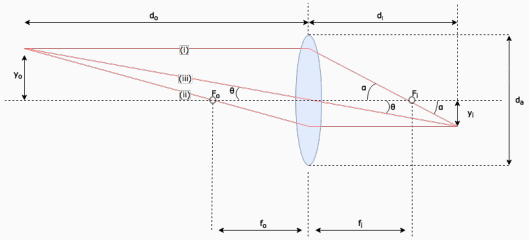


Figure 10: Idealised imaging system

(i) An incident ray which is parallel to the optic axis is refracted through the imaging focal point F_i.

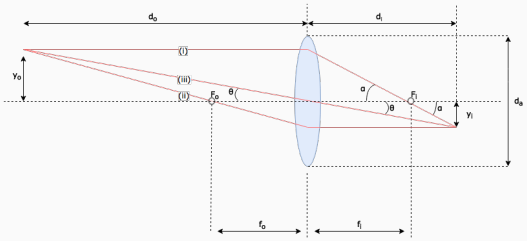
(ii) An incident ray which passes through the object focal point F_o, is refracted parallel to the optic axis.

(iii) An incident ray which passes through the optic center is refracted with the same refraction angle as incident angle (will not change direction).



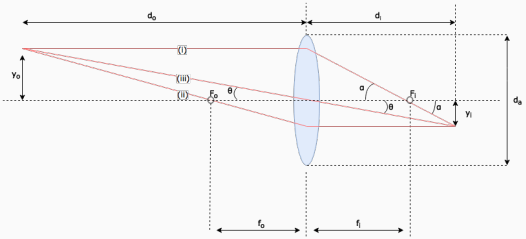
The rays following the respective patterns are illustrated in fig. 10. From (iii), we see that the angle θ is the same, and therefore

$$\frac{y_o}{d_o} = \frac{y_i}{d_i}. \tag{3}$$



Likewise, from the two equal angles α on ray (i), we get that

$$\frac{y_o}{f_i} = \frac{y_i}{d_i - f_i}. \tag{4}$$

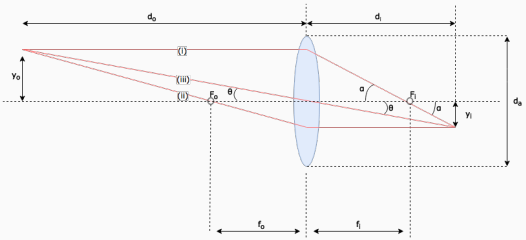


Rearranging eq. (3) yields

$$\frac{y_o}{y_i} = \frac{d_o}{d_i}$$

which, when combined with the relation in eq. (4) yelds

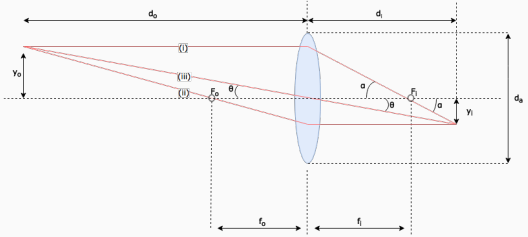
$$\frac{d_o}{d_i} = \frac{f_i}{d_i - f_i}. \tag{5}$$



Equation (5) can be rearranged to the familiar form

$$\frac{1}{f_i} = \frac{1}{d_i} + \frac{1}{d_o}. \tag{6}$$

Equation (6) is sometimes referred to as the *thin lens equation*.



Using eq. (3), we get that

$$\frac{1}{f_i} = \frac{1}{d_o}\left(\frac{y_o}{y_i} + 1\right),$$

which we can rearrange to

$$y_i = \frac{y_o f_i}{d_o - f_i}$$

This gives us (for our small, idealised imaging system) the relation between the actual size in the object plane y_o and the observed size in the imaging plane y_i.

SPATIAL FILTERING

In image analysis[1], convolution is a binary operation, taking an image f and a filter (also an image) w, and producing an image g. We use an asterisk $*$ to denote this operation

$$f * w = g.$$

[1]This is really just an ordinary *discrete convolution*, the discrete version of a *continuous convolution*.

In image analysis[1], convolution is a binary operation, taking an image f and a filter (also an image) w, and producing an image g. We use an asterisk $*$ to denote this operation

$$f * w = g.$$

For an element $[x, y]$, the operation is defined as

$$g[x, y] = (f * w)[x, y] \coloneqq \sum_{s=-S}^{S} \sum_{t=-T}^{T} f[x - s, y - t] w[s, t].$$

[1] This is really just an ordinary *discrete convolution*, the discrete version of a *continuous convolution*.

USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

1. Rotate the filter 180 degrees.

USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

1. Rotate the filter 180 degrees.
2. Place it on top of the image in the image's top left corner.

USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

1. Rotate the filter 180 degrees.
2. Place it on top of the image in the image's top left corner.
3. "Slide" it across each column until you hit the right boundary of the image.

USEFUL MENTAL IMAGE

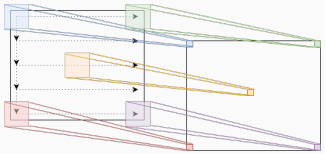
A useful way of thinking about convolution is to

1. Rotate the filter 180 degrees.
2. Place it on top of the image in the image's top left corner.
3. "Slide" it across each column until you hit the right boundary of the image.
4. Start from the left again, but now, one row below the previous.

USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

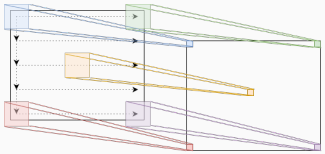
1. Rotate the filter 180 degrees.
2. Place it on top of the image in the image's top left corner.
3. "Slide" it across each column until you hit the right boundary of the image.
4. Start from the left again, but now, one row below the previous.
5. Repeat step 3. followed by 4. until you have covered the whole image.



USEFUL MENTAL IMAGE

A useful way of thinking about convolution is to

1. Rotate the filter 180 degrees.
2. Place it on top of the image in the image's top left corner.
3. "Slide" it across each column until you hit the right boundary of the image.
4. Start from the left again, but now, one row below the previous.
5. Repeat step 3. followed by 4. until you have covered the whole image.



Also, check out this nice interactive visualization:
`http://setosa.io/ev/image-kernels/`

DIFFERENT MODES

We differentiate between three different *modes* when we filter one image with another:

DIFFERENT MODES

We differentiate between three different *modes* when we filter one image with another:

Full: We get an output response at every point of overlap between the image and the filter.

DIFFERENT MODES

We differentiate between three different *modes* when we filter one image with another:

Full: We get an output response at every point of overlap between the image and the filter.

Same: The origin of the filter is always inside the image, and we *pad* the image in order to preserve the image size in the result.

DIFFERENT MODES

We differentiate between three different *modes* when we filter one image with another:

Full: We get an output response at every point of overlap between the image and the filter.

Same: The origin of the filter is always inside the image, and we *pad* the image in order to preserve the image size in the result.

Valid: We only record a response as long as there is full overlap between the image and the *whole* filter.

FULL MODE — EXAMPLE

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \overset{\text{full}}{*} \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.3 & 0.6 & 0.9 & 0.7 & 0.4 \\ 0.6 & 1.4 & 2.4 & 3. & 2.2 & 1.2 \\ 1.5 & 3.3 & 5.4 & 6.3 & 4.5 & 2.4 \\ 2.7 & 5.7 & 9. & 9.9 & 6.9 & 3.6 \\ 2.2 & 4.6 & 7.2 & 7.8 & 5.4 & 2.8 \\ 1.3 & 2.7 & 4.2 & 4.5 & 3.1 & 1.6 \end{bmatrix}$$

VALID MODE — EXAMPLE

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \underset{*}{\overset{\text{valid}}{}} \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 5.4 & 6.3 \\ 9. & 9.9 \end{bmatrix}$$

Can also pad with other constant values than 0.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \xrightarrow{\text{zero}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 0 \\ 0 & 5 & 6 & 7 & 8 & 0 \\ 0 & 9 & 10 & 11 & 12 & 0 \\ 0 & 13 & 14 & 15 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \overset{\text{same}}{*} \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 1.4 & 2.4 & 3. & 2.2 \\ 3.3 & 5.4 & 6.3 & 4.5 \\ 5.7 & 9. & 9.9 & 6.9 \\ 4.6 & 7.2 & 7.8 & 5.4 \end{bmatrix}$$

SEPARABLE FILTERS

- A 2D filter w, is said to be *separable* if the filtration can be done with two sequential 1D filtrations, w_V and w_H.

SEPARABLE FILTERS

- A 2D filter w, is said to be *separable* if the filtration can be done with two sequential 1D filtrations, w_V and w_H.
- In other words, if $w = w_V * w_H$

- A 2D filter w, is said to be *separable* if the filtration can be done with two sequential 1D filtrations, w_V and w_H.
- In other words, if $w = w_V * w_H$
- An example is a 5×5 mean filter

$$\frac{1}{5}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} * \frac{1}{5}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

SEPARABLE FILTERS ARE FASTER

- From the associativity of convolution, with f as an image and $w = w_V * w_H$, we get

$$f * w = f * (w_V * w_H) = (f * w_V) * w_H$$

SEPARABLE FILTERS ARE FASTER

- From the associativity of convolution, with f as an image and $w = w_V * w_H$, we get

$$f * w = f * (w_V * w_H) = (f * w_V) * w_H$$

- As can be seen, we can now do two convolutions with a 1D filter, in stead of one convolution with a 2D filter, that is

$$g = f * w, \quad \text{2D}$$

vs.

$$h = f * w_V \quad \text{1D}$$

$$g = h * w_H \quad \text{1D}$$

HOW MUCH FASTER? COMPLEXITY

For an $M \times N$ image and a square filter kernel with sidelengths L, a (naive) $2D$ convolution implementation would have complexity of

$$\mathcal{O}(MNL^2),$$

HOW MUCH FASTER? COMPLEXITY

For an $M \times N$ image and a square filter kernel with sidelengths L, a (naive) $2D$ convolution implementation would have complexity of

$$\mathcal{O}(MNL^2),$$

while a (naive) $1D$ convolution implementation would have complexity of

$$\mathcal{O}(MNL).$$

HOW MUCH FASTER? COMPLEXITY

For an $M \times N$ image and a square filter kernel with sidelengths L, a (naive) $2D$ convolution implementation would have complexity of

$$\mathcal{O}(MNL^2),$$

while a (naive) $1D$ convolution implementation would have complexity of

$$\mathcal{O}(MNL).$$

So the speed up should be linear in L.

Low-pass filters

Lets through low frequencies, stops high frequencies. ”Blurs” the image.

SOME COMMON FILTER CATEGORIES

Low-pass filters

Lets through low frequencies, stops high frequencies. ”Blurs” the image.

High-pass filters

Lets through high frequencies, stops low frequencies. ”Sharpens” the image.

SOME COMMON FILTER CATEGORIES

Low-pass filters
: Lets through low frequencies, stops high frequencies. ”Blurs” the image.

High-pass filters
: Lets through high frequencies, stops low frequencies. ”Sharpens” the image.

Band-pass filters
: Lets through frequencies in a certain range.

SOME COMMON FILTER CATEGORIES

Low-pass filters
: Lets through low frequencies, stops high frequencies. ”Blurs” the image.

High-pass filters
: Lets through high frequencies, stops low frequencies. ”Sharpens” the image.

Band-pass filters
: Lets through frequencies in a certain range.

Feauture detection filters
: Used for detection of features in an image. Features such as edges, corners and texture.

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).
- Stops or damps high frequencies (noise, details and sharp edges in an image).

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).
- Stops or damps high frequencies (noise, details and sharp edges in an image).
- We will learn more about frequencies in the lectures about Fourier transforms.

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).
- Stops or damps high frequencies (noise, details and sharp edges in an image).
- We will learn more about frequencies in the lectures about Fourier transforms.
- The resulting image is a ”smeared”, ”smoothed” or ”blurred” version of the original image.

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).
- Stops or damps high frequencies (noise, details and sharp edges in an image).
- We will learn more about frequencies in the lectures about Fourier transforms.
- The resulting image is a ”smeared”, ”smoothed” or ”blurred” version of the original image.
- Typical applications: Remove noise, locate larger objects in an image.

LOW-PASS FILTER — OVERVIEW

- Lets through low frequencies (large trends and slow variations in an image).
- Stops or damps high frequencies (noise, details and sharp edges in an image).
- We will learn more about frequencies in the lectures about Fourier transforms.
- The resulting image is a ”smeared”, ”smoothed” or ”blurred” version of the original image.
- Typical applications: Remove noise, locate larger objects in an image.
- Challenging to preserve edges.

MEAN VALUE FILTER

- Computes the mean value in the neighbourhood.

MEAN VALUE FILTER

- Computes the mean value in the neighbourhood.
  - All weights (values in the filter) are equal.

MEAN VALUE FILTER

- Computes the mean value in the neighbourhood.
  - All weights (values in the filter) are equal.
  - The sum of the weights is equal to 1.

MEAN VALUE FILTER

- Computes the mean value in the neighbourhood.
  - All weights (values in the filter) are equal.
  - The sum of the weights is equal to 1.
- The neighbourhood size determines the ”blurring magnitude”.

MEAN VALUE FILTER

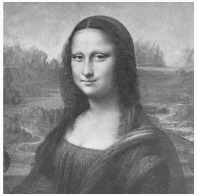
- Computes the mean value in the neighbourhood.
  - All weights (values in the filter) are equal.
  - The sum of the weights is equal to 1.
- The neighbourhood size determines the ”blurring magnitude”.
  - Large filter: Loss of details, more blurring.

MEAN VALUE FILTER

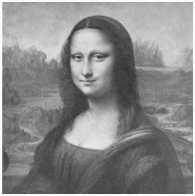
- Computes the mean value in the neighbourhood.
  - All weights (values in the filter) are equal.
  - The sum of the weights is equal to 1.
- The neighbourhood size determines the ”blurring magnitude”.
  - Large filter: Loss of details, more blurring.
  - Small filter: Preservation of details, less blurring.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad \frac{1}{49}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

MEAN VALUE FILTER — EXAMPLE



(a) Original



(b) 3×3

(c) 9×9

(d) 25×25

Figure 11: Gray level Mona Lisa image filtered with a mean value filter of different sizes.

- For integer valus x, y, let

$$w[x, y] = A \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

where A is usually such that $\sum_x \sum_y w[x, y] = 1$.

GAUSSIAN FILTER

- For integer valus x, y, let

$$w[x, y] = A \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

  where A is usually such that $\sum_x \sum_y w[x, y] = 1$.
- This is a *non-uniform low-pass filter*.

- For integer valus x, y, let

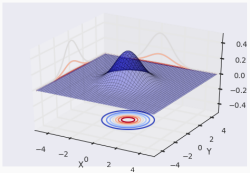
$$w[x, y] = A \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

  where A is usually such that $\sum_x \sum_y w[x, y] = 1$.
- This is a *non-uniform low-pass filter*.
- The parameter σ is the standard deviation and controls the amount of smoothing.
  - Small σ: Less smoothing
  - Large σ: More smoothing

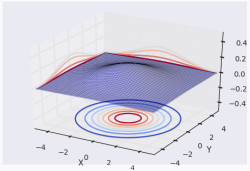
- For integer valus x, y, let

$$w[x, y] = A \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\}$$

  where A is usually such that $\sum_x \sum_y w[x, y] = 1$.
- This is a *non-uniform low-pass filter*.
- The parameter σ is the standard deviation and controls the amount of smoothing.
  - Small σ: Less smoothing
  - Large σ: More smoothing
- A Gaussian filter smooths less than a uniform filter of the same size.



(a) $\sigma = 1$



(b) $\sigma = 2$

Figure 12: Continuous bivariate Gaussian with different σ. Implementation can be found at `https://ojskrede.github.io/inf2310/lectures/week_06/`

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").
- Not unusual with non-rectangular neighbourhood, e.g. plus-shaped neighbourhoods.

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").
- Not unusual with non-rectangular neighbourhood, e.g. plus-shaped neighbourhoods.
- Some challenges:
  - Thin lines can dissappear

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").
- Not unusual with non-rectangular neighbourhood, e.g. plus-shaped neighbourhoods.
- Some challenges:
  - Thin lines can dissappear
  - Corners can be rounded

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").
- Not unusual with non-rectangular neighbourhood, e.g. plus-shaped neighbourhoods.
- Some challenges:
  - Thin lines can dissappear
  - Corners can be rounded
  - Objects can be shrinked

MEDIAN FILTERING

- A rank filter where we choose the middle value in the sorted list of values in the neighbourhood of (x, y).
- One of the most frequently used edge-preserving filters.
- Especially well-suited for reducing *impulse-noise* (aka. "salt-and-pepper-noise").
- Not unusual with non-rectangular neighbourhood, e.g. plus-shaped neighbourhoods.
- Some challenges:
  - Thin lines can dissappear
  - Corners can be rounded
  - Objects can be shrinked
- The size and shape of the neighbourhood is important

HIGH-PASS FILTERING — OVERVIEW

- Filters that let through high frequencies, and damps or block low frequencies.

HIGH-PASS FILTERING — OVERVIEW

- Filters that let through high frequencies, and damps or block low frequencies.
- Effect:
  - Damps slow variations, e.g. background
  - Emphasizes sharp edges, lines, and details

HIGH-PASS FILTERING — OVERVIEW

- Filters that let through high frequencies, and damps or block low frequencies.
- Effect:
  - Damps slow variations, e.g. background
  - Emphasizes sharp edges, lines, and details
- Is typically used to ”improve” the sharpness, and to detect edges.

HIGH-PASS FILTERING — OVERVIEW

- Filters that let through high frequencies, and damps or block low frequencies.
- Effect:
  - Damps slow variations, e.g. background
  - Emphasizes sharp edges, lines, and details
- Is typically used to ”improve” the sharpness, and to detect edges.
- What happens with noise?

HIGH-PASS FILTERING — OVERVIEW

- Filters that let through high frequencies, and damps or block low frequencies.
- Effect:
  - Damps slow variations, e.g. background
  - Emphasizes sharp edges, lines, and details
- Is typically used to ”improve” the sharpness, and to detect edges.
- What happens with noise?
- Noise is amplified.

- The convolution mask we have investigated can be rewritten as "high-pass = k * (original - low-pass)", where $k = 9$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = 9 \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

- The convolution mask we have investigated can be rewritten as "high-pass = k * (original - low-pass)", where $k = 9$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = 9 \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

- With this, our image improvement strategy can be viewed as:
  1. Low pass filtering
  2. Subtract the low-pass filtered image from the original
  3. Add k times the difference from point 2. to the original image.
- The described method is called *highboost-filtering*.

UNSHARP MASKING AND HIGHBOOST-FILTERING

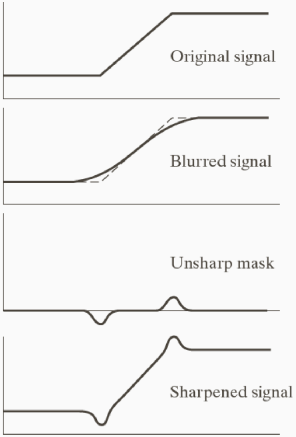


Figure 13: Schematic illustration of image sharpening on a 1D ramp.

UNSHARP MASKING AND HIGHBOOST-FILTERING

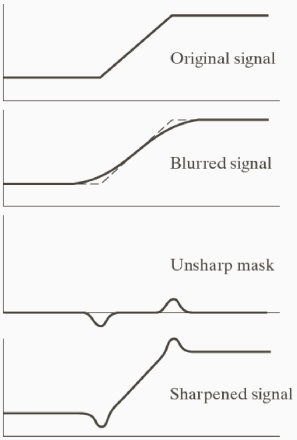


Figure 13: Schematic illustration of image sharpening on a 1D ramp.

- Given an image.
- Blur the image with a low-pass filter.

UNSHARP MASKING AND HIGHBOOST-FILTERING

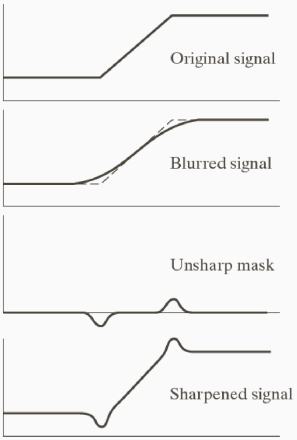


Figure 13: Schematic illustration of image sharpening on a 1D ramp.

- Given an image.
- Blur the image with a low-pass filter.
- Compute the difference $original - blurred$

UNSHARP MASKING AND HIGHBOOST-FILTERING

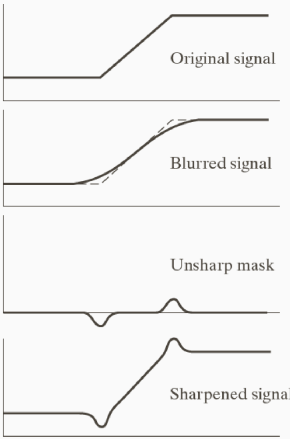


Figure 13: Schematic illustration of image sharpening on a 1D ramp.

- Given an image.
- Blur the image with a low-pass filter.
- Compute the difference $original - blurred$
- $sharpened = original + k(original - blurred)$

UNSHARP MASKING AND HIGHBOOST-FILTERING

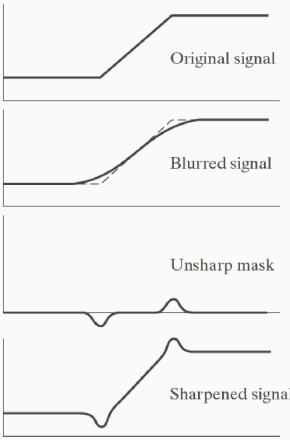
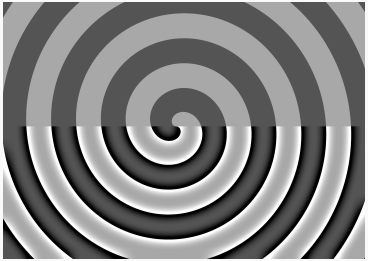


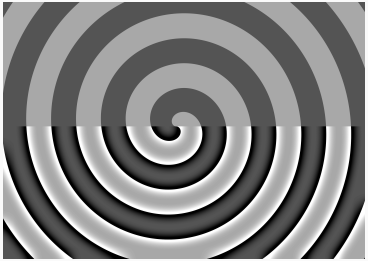
Figure 13: Schematic illustration of image sharpening on a 1D ramp.

- Given an image.
- Blur the image with a low-pass filter.
- Compute the difference $original - blurred$
- $sharpened = original + k(original - blurred)$
- k is a positive constant.
  - $k = 1$: Unsharp masking
  - $k > 1$: Highboost-filtering

EXAMPLES



EXAMPLES



EDGE-DETECTION — OVERVIEW

- There exists much image information at the edges of objects in an image.

EDGE-DETECTION — OVERVIEW

- There exists much image information at the edges of objects in an image.
- Biological visual systems are based on edge-detection.

EDGE-DETECTION — OVERVIEW

- There exists much image information at the edges of objects in an image.
- Biological visual systems are based on edge-detection.
- Such systems often works in parallel and/or sequential:
  - Local areas is treated independently
  - Local results can be dependent on earlier results

- The gradient points in the direction where the function ascends the most. We call this the *gradient angle*

$$\theta_g = \arctan\left(\frac{g_y}{g_x}\right)$$

SUMMARY — GRADIENT ANGLE AND GRADIENT MAGNITUDE

- The gradient points in the direction where the function ascends the most. We call this the *gradient angle*

$$\theta_g = \arctan\left(\frac{g_y}{g_x}\right)$$

- The ascent magnitude at this point is called the *gradient magnitude* and is

$$\begin{aligned} |\nabla f| &= \frac{\partial f}{\partial r}(\theta_g) \\ &= \sqrt{g_x^2 + g_y^2}. \end{aligned}$$

- Where

$$g_x = \frac{\partial f}{\partial x}, \text{ and } g_y = \frac{\partial f}{\partial y}$$

are the components of the gradient of f, ∇f.

DIGITAL GRADIENT APPROXIMATIONS

- We will approximate the gradient at some pixel (i, j) by the difference between pixels in the neighbourhood.

DIGITAL GRADIENT APPROXIMATIONS

- We will approximate the gradient at some pixel (i, j) by the difference between pixels in the neighbourhood.
- We can do this by using two convolution filters (h_x, and h_y) that approximates the two components of the gradient operator.

- We will approximate the gradient at some pixel (i, j) by the difference between pixels in the neighbourhood.
- We can do this by using two convolution filters (h_x, and h_y) that approximates the two components of the gradient operator.
- The convolution result $g_x[i, j]$ and $g_y[i, j]$ will then approximate the gradient at (i, j) in the vertical and horizontal direction, respectively.

- We will approximate the gradient at some pixel (i, j) by the difference between pixels in the neighbourhood.
- We can do this by using two convolution filters (h_x, and h_y) that approximates the two components of the gradient operator.
- The convolution result $g_x[i, j]$ and $g_y[i, j]$ will then approximate the gradient at (i, j) in the vertical and horizontal direction, respectively.
- In the next slides, we will introduce several different gradient operators. Note that we write convolution filters, while G&W use correlation filters, and they are therefore rotated 180 degrees compared to ours.

Increasing noise robustness: 2D gradient operators

- If we first apply a smoothing filter to reduce the noise.

Increasing noise robustness: 2D gradient operators

- If we first apply a smoothing filter to reduce the noise.
- Then apply the desired gradient filter.

Increasing noise robustness: 2D gradient operators

- If we first apply a smoothing filter to reduce the noise.
- Then apply the desired gradient filter.
- We can utilize the separability of filters, and create new more noise-robust filters.

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix},$$

PREWITT OPERATOR

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix},$$

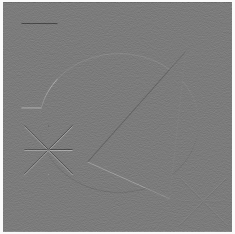
$$h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}.$$

$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

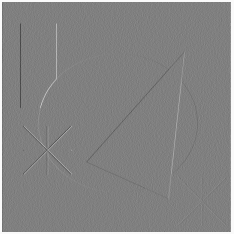
$$h_x = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

$$h_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}.$$

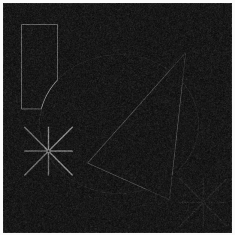
3x3 Sobel gradient operators



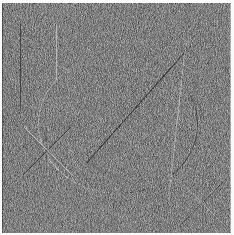
(a) g_x



(b) g_y

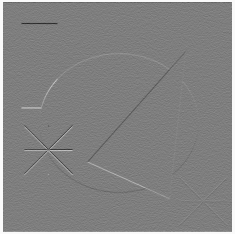


(c) $|\nabla f|$

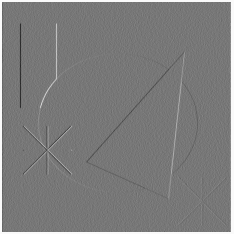


(d) θ

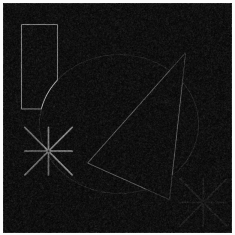
5x5 Sobel gradient operators



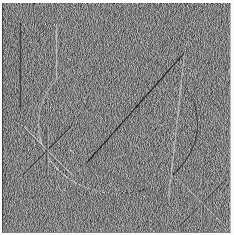
(a) g_x



(b) g_y



(c) $|\nabla f|$



(d) θ

- For a continuous bivariate function f, the *Laplace operator* is given by

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

THE LAPLACE OPERATOR

- For a continuous bivariate function f, the *Laplace operator* is given by

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Function curvature information:
  - It is positive for decreasing gradient,
  - negative for increasing gradient,
  - zero at critical points in the gradient.

THE LAPLACE OPERATOR

- For a continuous bivariate function f, the *Laplace operator* is given by

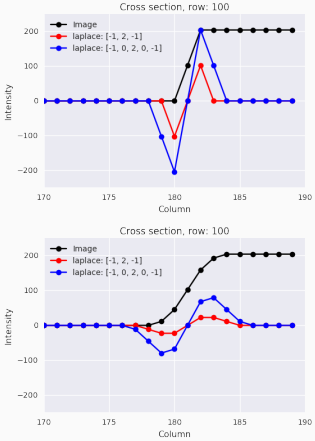
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Function curvature information:
  - It is positive for decreasing gradient,
  - negative for increasing gradient,
  - zero at critical points in the gradient.
- Has two extrema for each edge, which is why we used it for image improvement earlier.

- For a continuous bivariate function f, the *Laplace operator* is given by

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Function curvature information:
  - It is positive for decreasing gradient,
  - negative for increasing gradient,
  - zero at critical points in the gradient.
- Has two extrema for each edge, which is why we used it for image improvement earlier.
- If we use the location of the zero crossing, we can get a thin, well defined location of the edge.



2D LAPLACE OPERATOR

- Use the 1D Laplace approximation in both directions

$$
\begin{aligned}
-\nabla^2 f &= -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \\
&\approx -(f[x+1,y] - 2f[x,y] + f[x-1,y]) - (f[x,y+1] - 2f[x,y] + f[x,y-1]) \\
&= -f[x+1,y] - f[x,y+1] + 4f[x,y] + f[x-1,y] + f[x,y+1]
\end{aligned}
$$

2D LAPLACE OPERATOR

- Use the 1D Laplace approximation in both directions

$$
\begin{aligned}
-\nabla^2 f &= -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \\
&\approx -(f[x+1,y] - 2f[x,y] + f[x-1,y]) - (f[x,y+1] - 2f[x,y] + f[x,y-1]) \\
&= -f[x+1,y] - f[x,y+1] + 4f[x,y] + f[x-1,y] + f[x,y+1]
\end{aligned}
$$

- This can be computed by convolving the image with the following filter

$$
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 4 & -1 \\
0 & -1 & 0
\end{bmatrix}
$$

FULL 3X3 LAPLACE OPERATOR

- If we, in addition, use the 1D Laplace approximation on the diagonal directions, we get the following filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

FULL 3X3 LAPLACE OPERATOR

- If we, in addition, use the 1D Laplace approximation on the diagonal directions, we get the following filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- This convolution filter is the same as we used earlier in this slides for point detection and image sharpness enhancement.

COLOR IMAGING

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.
- Therefore, the color of an object is determined by
  - The light hitting the object.
  - How much light is reflected and absorbed.

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.
- Therefore, the color of an object is determined by
  - The light hitting the object.
  - How much light is reflected and absorbed.
- Or, in other words
  - The spectral distribution of the incident light.
  - The spectral distribution of the reflected light.

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.
- Therefore, the color of an object is determined by
  - The light hitting the object.
  - How much light is reflected and absorbed.
- Or, in other words
  - The spectral distribution of the incident light.
  - The spectral distribution of the reflected light.
- The reflection properties are determined by
  - Chemical pigments
  - Physical surface structures

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.
- Therefore, the color of an object is determined by
  - The light hitting the object.
  - How much light is reflected and absorbed.
- Or, in other words
  - The spectral distribution of the incident light.
  - The spectral distribution of the reflected light.
- The reflection properties are determined by
  - Chemical pigments
  - Physical surface structures
- Together, this determines what wavelengths are reflected, absorbed or transmitted.

THE COLOR OF AN OBJECT

- The color we percieve in an object is determined by the nature of the light reflected by the object.
- Therefore, the color of an object is determined by
  - The light hitting the object.
  - How much light is reflected and absorbed.
- Or, in other words
  - The spectral distribution of the incident light.
  - The spectral distribution of the reflected light.
- The reflection properties are determined by
  - Chemical pigments
  - Physical surface structures
- Together, this determines what wavelengths are reflected, absorbed or transmitted.
- How we actually percieve color is complicated. Two complementary theories: *Trichromacy* and *Color opponency*.

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.

SOME WORDS AND THEIR MEANINGS

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.
- Saturation: How saturated the color is with white light, from a pure color (no white) to fully saturated (white).

SOME WORDS AND THEIR MEANINGS

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.
- Saturation: How saturated the color is with white light, from a pure color (no white) to fully saturated (white).
- Intensity: How bright the color is, some notion of darkness.

SOME WORDS AND THEIR MEANINGS

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.
- Saturation: How saturated the color is with white light, from a pure color (no white) to fully saturated (white).
- Intensity: How bright the color is, some notion of darkness.
- Hue and saturation together determines the color, and we call these two *chromaticity*.

- The chromaticity determines both the dominating wavelength and the saturation of the color.

SOME WORDS AND THEIR MEANINGS

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.
- Saturation: How saturated the color is with white light, from a pure color (no white) to fully saturated (white).
- Intensity: How bright the color is, some notion of darkness.
- Hue and saturation together determines the color, and we call these two *chromaticity*.
- The chromaticity determines both the dominating wavelength and the saturation of the color.
- Chromaticity and brightness together fully describes a color.

All colors can be fully described by their *hue*, *saturation* and some form of *intensity, or brightness value*.

- Hue: The dominant wavelength of the color.
- Saturation: How saturated the color is with white light, from a pure color (no white) to fully saturated (white).
- Intensity: How bright the color is, some notion of darkness.
- Hue and saturation together determines the color, and we call these two *chromaticity*.

- The chromaticity determines both the dominating wavelength and the saturation of the color.
- Chromaticity and brightness together fully describes a color.
- For instance, different graylevels have the same chromaticity, but different intensity.

Figure 17: Chromaticity modeled as a unit polar coordinate chart (ρ, θ) where fully saturated colors are at the boundary $(\rho = 1)$, and the angle θ represent the wavelength, and thereby the hue.

ADDITIVE COLOR MIXING

- Mixing light of two or more colors

ADDITIVE COLOR MIXING

- Mixing light of two or more colors
- Additive primaries: Red, Green, Blue

ADDITIVE COLOR MIXING

- Mixing light of two or more colors
- Additive primaries: Red, Green, Blue
- Additive secondaries: Yellow, Cyan, Magenta

ADDITIVE COLOR MIXING

- Mixing light of two or more colors
- Additive primaries: Red, Green, Blue
- Additive secondaries: Yellow, Cyan, Magenta
- Utilized in TV and Computer monitors

ADDITIVE COLOR MIXING

- Mixing light of two or more colors
- Additive primaries: Red, Green, Blue
- Additive secondaries: Yellow, Cyan, Magenta
- Utilized in TV and Computer monitors
- Difference between a mix of red and green (yellow) and yellow light (wavelength at about 580 nm), but we detect no difference.

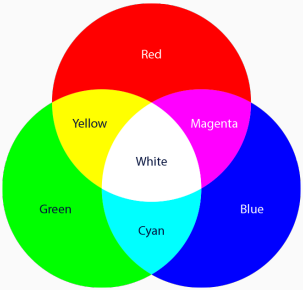


Figure 18: Additive mixing

SUBTRACTIVE COLOR MIXING

- Mixing filters or pigments that absorbs and reflect different colors.

SUBTRACTIVE COLOR MIXING

- Mixing filters or pigments that absorbs and reflect different colors.
- If you begin with white light, the result color you see is a result of a subtractive color process where different wavelengths have been absorbed.

SUBTRACTIVE COLOR MIXING

- Mixing filters or pigments that absorbs and reflect different colors.
- If you begin with white light, the result color you see is a result of a subtractive color process where different wavelengths have been absorbed.
- Subtractive primaries: Yellow, Cyan, Magenta

SUBTRACTIVE COLOR MIXING

- Mixing filters or pigments that absorbs and reflect different colors.
- If you begin with white light, the result color you see is a result of a subtractive color process where different wavelengths have been absorbed.
- Subtractive primaries: Yellow, Cyan, Magenta
- Subtractive secondaries: Red, Green, Blue

SUBTRACTIVE COLOR MIXING

- Mixing filters or pigments that absorbs and reflect different colors.
- If you begin with white light, the result color you see is a result of a subtractive color process where different wavelengths have been absorbed.
- Subtractive primaries: Yellow, Cyan, Magenta
- Subtractive secondaries: Red, Green, Blue
- Mixing of paint is subtractive (although, you probably used a RYB color model in kindergarden).

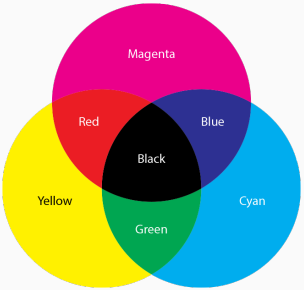


Figure 19: Subtractive mixing

RGB COLOR MODEL

- An additive color model where the primaries red, green and blue, are added together to form a wide variety of colors.

[1]By Gona.eu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6223756
[2]By SharkD - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3375025

RGB COLOR MODEL

- An additive color model where the primaries red, green and blue, are added together to form a wide variety of colors.
- Device dependent, meaning that the color is dependent on the device that detect or output the color.

[1]By Gona.eu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6223756
[2]By SharkD - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3375025

RGB COLOR MODEL

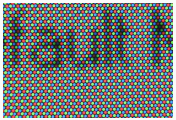
- An additive color model where the primaries red, green and blue, are added together to form a wide variety of colors.
- Device dependent, meaning that the color is dependent on the device that detect or output the color.
- Typical input devices are video cameras, digital cameras and image scanners.

[1]By Gona.eu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6223756
[2]By SharkD - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3375025

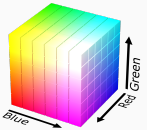
RGB COLOR MODEL

- An additive color model where the primaries red, green and blue, are added together to form a wide variety of colors.
- Device dependent, meaning that the color is dependent on the device that detect or output the color.
- Typical input devices are video cameras, digital cameras and image scanners.
- Typical output devices are TV, computer and mobile displays.

- Organized in a unit cube with coordinates (r, g, b), where $(0, 0, 0)$ is black and $(1, 1, 1)$ is white.
- A typical 24 bit display has 256 values for each channel.



(a) CRT monitor [1]



(b) RGB color cube [2]

[1] By Gona.eu - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6223756
[2] By SharkD - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3375025

CMYK COLOR MODEL

- **C**yan, **M**agenta, **Y**ellow, **K**ey (black)

[1] By BenRG and cmglee - http://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=32158329

CMYK COLOR MODEL

- **C**yan, **M**agenta, **Y**ellow, **K**ey (black)
- CMYK (in $[0,1]$) to RGB (in $[0,1]$):

$$
\begin{aligned}
R &= (1-C)(1-K) \\
G &= (1-M)(1-K) \\
B &= (1-Y)(1-K)
\end{aligned}
$$

- RGB (in $[0,1]$) to CMYK (in $[0,1]$):

$$
\begin{aligned}
K &= 1 - \max\{R,G,B\} \\
C &= (1-R-K)/(1-K) \\
M &= (1-G-K)(1-K) \\
Y &= (1-B-K)(1-K)
\end{aligned}
$$

[1] By BenRG and cmglee - http://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=32158329

CMYK COLOR MODEL

- **C**yan, **M**agenta, **Y**ellow, **K**ey (black)
- CMYK (in $[0,1]$) to RGB (in $[0,1]$):

$$
\begin{aligned}
R &= (1-C)(1-K) \\
G &= (1-M)(1-K) \\
B &= (1-Y)(1-K)
\end{aligned}
$$

- RGB (in $[0,1]$) to CMYK (in $[0,1]$):

$$
\begin{aligned}
K &= 1 - \max\{R,G,B\} \\
C &= (1-R-K)/(1-K) \\
M &= (1-G-K)(1-K) \\
Y &= (1-B-K)(1-K)
\end{aligned}
$$

- The CMYK-model is subtractive

[1] By BenRG and cmglee - http://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=32158329

CMYK COLOR MODEL

- **C**yan, **M**agenta, **Y**ellow, **K**ey (black)
- CMYK (in $[0, 1]$) to RGB (in $[0, 1]$):

$$R = (1 - C)(1 - K)$$
$$G = (1 - M)(1 - K)$$
$$B = (1 - Y)(1 - K)$$

- RGB (in $[0, 1]$) to CMYK (in $[0, 1]$):

$$K = 1 - \max\{R, G, B\}$$
$$C = (1 - R - K)/(1 - K)$$
$$M = (1 - G - K)(1 - K)$$
$$Y = (1 - B - K)(1 - K)$$

- The CMYK-model is subtractive
- RGB is common in display monitors, CMYK is common in printers.

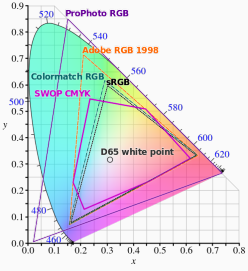


Figure 21: [1] Collection of gamuts

[1] By BenRG and cmglee - http://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=32158329

HUE, SATURATION, INTENSITY (HSI)

- Describe colors by their hue, saturation and intensity.

HUE, SATURATION, INTENSITY (HSI)

- Describe colors by their hue, saturation and intensity.
- This can be veiwed in a double cone (fig on the right).
  - Hue: An angle from red (H = 0).
  - Saturation: Distance from center axis.
  - Intensity: Vertical axis.

HUE, SATURATION, INTENSITY (HSI)

- Describe colors by their hue, saturation and intensity.
- This can be veiwed in a double cone (fig on the right).
  - Hue: An angle from red (H = 0).
  - Saturation: Distance from center axis.
  - Intensity: Vertical axis.
- RGB is useful for color generation. HSI more useful for color description and color image processing.

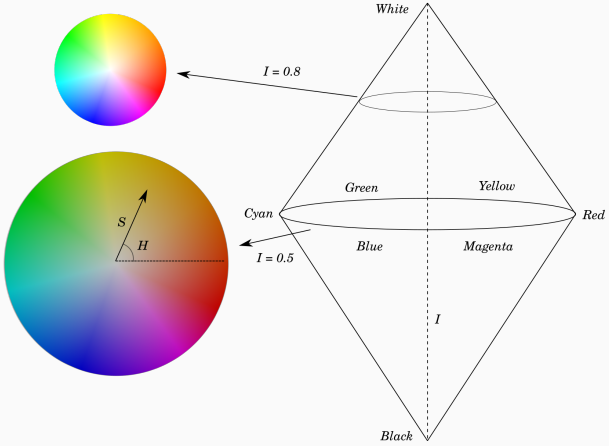


Figure 22: HSI double cone

True color uses all colors in the color space.

True color uses all colors in the color space.

- Used in applications that contain many colors with subtle differences. E.g. digital photography or photorealistic rendering.
- Two main ways to organize true color: Component ordering and packed ordering.

ORGANIZATION OF COLOR IMAGES

True color uses all colors in the color space.

- Used in applications that contain many colors with subtle differences. E.g. digital photography or photorealistic rendering.
- Two main ways to organize true color: Component ordering and packed ordering.

Indexed color uses only a subset of colors.

ORGANIZATION OF COLOR IMAGES

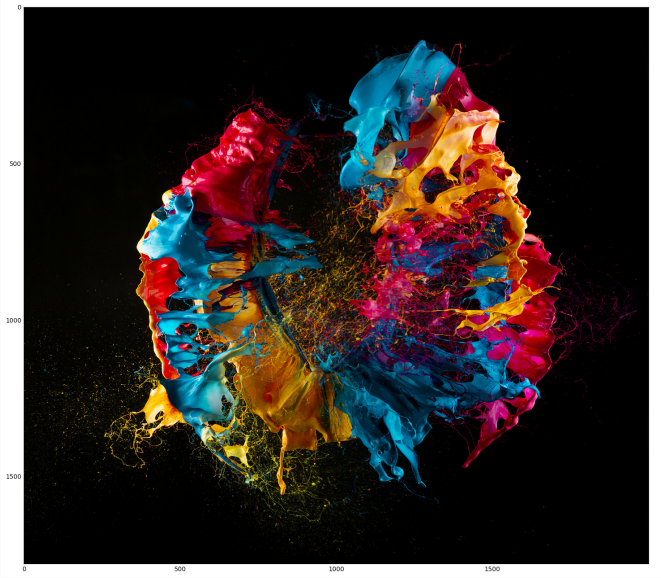
True color uses all colors in the color space.

- Used in applications that contain many colors with subtle differences. E.g. digital photography or photorealistic rendering.
- Two main ways to organize true color: Component ordering and packed ordering.

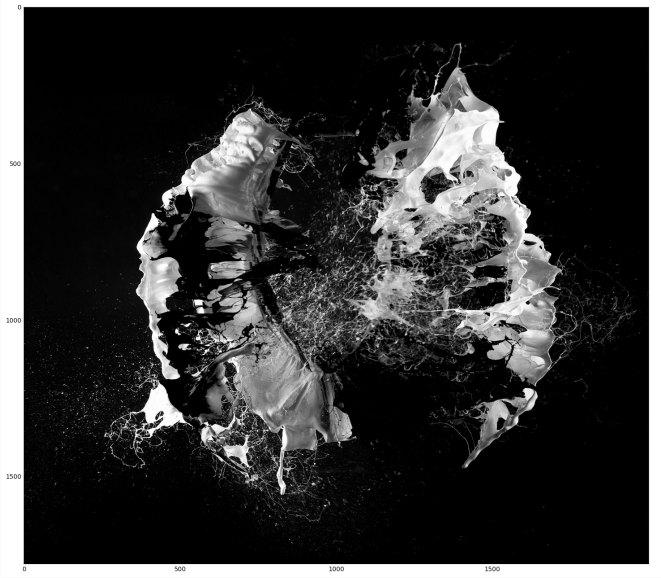
Indexed color uses only a subset of colors.

- Application dependence on which subset to use.
- Reduces memory and computation cost.
- Used when subtle color differences is not vital.

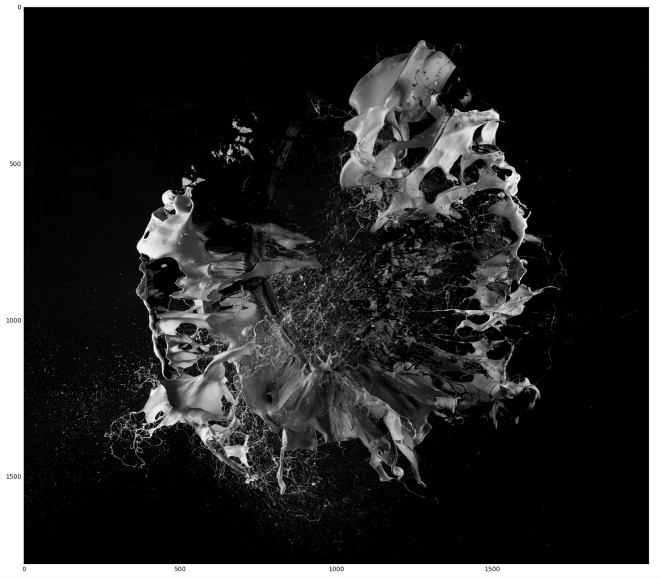
EXAMPLE



RED CHANNEL

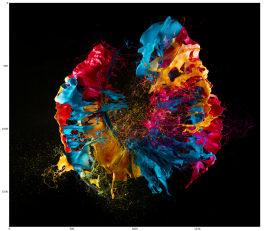


GREEN CHANNEL

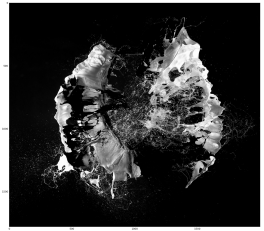


BLUE CHANNEL

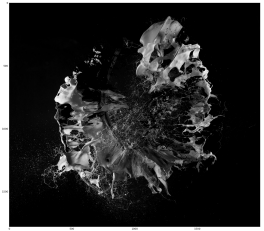
FULL EXAMPLE SUMMARY



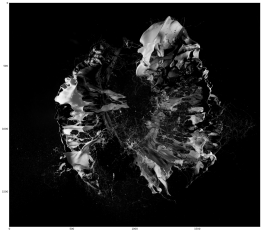
(a) All



(b) Red



(c) Green



(d) Blue

INDEXED COLOR

- An image contains indices of a look up table (LUT) of colors (palette) in stead of color intensity values.

[1]Wilhelm Burger, Mark J. Burge, *Principles of Digital Image Processing: Fundamental Techniques*, 2010

INDEXED COLOR

- An image contains indices of a look up table (LUT) of colors (palette) in stead of color intensity values.
- Permits only a limited number of colors.

[1]Wilhelm Burger, Mark J. Burge, *Principles of Digital Image Processing: Fundamental Techniques*, 2010

INDEXED COLOR

- An image contains indices of a look up table (LUT) of colors (palette) in stead of color intensity values.
- Permits only a limited number of colors.
- Often are these images stored in indexed GIF or PNG formats.

[1]Wilhelm Burger, Mark J. Burge, *Principles of Digital Image Processing: Fundamental Techniques*, 2010

INDEXED COLOR

- An image contains indices of a look up table (LUT) of colors (palette) in stead of color intensity values.
- Permits only a limited number of colors.
- Often are these images stored in indexed GIF or PNG formats.
- Use color quantization to transform optimally from a true color image to indexed color image.

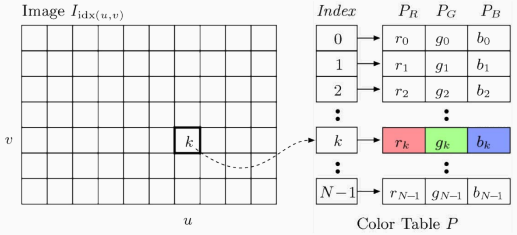


Figure 24: Indexed color ordering[1].

[1]Wilhelm Burger, Mark J. Burge, *Principles of Digital Image Processing: Fundamental Techniques*, 2010

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.
- A region with many nuances of a color would not loog good.

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.
- A region with many nuances of a color would not loog good.
- It is certain that all 512 colors exist in the image.

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.
- A region with many nuances of a color would not loog good.
- It is certain that all 512 colors exist in the image.
- Alternatively, one could use 8 bits and a LUT.

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.
- A region with many nuances of a color would not loog good.
- It is certain that all 512 colors exist in the image.
- Alternatively, one could use 8 bits and a LUT.
- Each row in the table represent a 24 bit RGB color.

COLOR IMAGES AND LOOK UP TABLES

- As we have seen, in true color, an RGB pixel can be stored in 24 bits (8 for each color).
- To reduce this size, we could e.g. assign 3 bits to each color.
- This would result in only 512 different possible colors, and a total of 9 bits ber pixel.
- A region with many nuances of a color would not loog good.
- It is certain that all 512 colors exist in the image.
- Alternatively, one could use 8 bits and a LUT.
- Each row in the table represent a 24 bit RGB color.
- The table consist of the 256 colors that best represent the image.

PSEUDO COLORS

- Pseudo-color images can be graylevel images where each graylevel is assigned an RGB value according to some LUT.

PSEUDO COLORS

- Pseudo-color images can be graylevel images where each graylevel is assigned an RGB value according to some LUT.
- Is often used to emphasize small graylevel differences (e.g. in medical imaging).

PSEUDO COLORS

- Pseudo-color images can be graylevel images where each graylevel is assigned an RGB value according to some LUT.
- Is often used to emphasize small graylevel differences (e.g. in medical imaging).
- Also often used in graphical display of data.

PSEUDO COLORS

- Pseudo-color images can be graylevel images where each graylevel is assigned an RGB value according to some LUT.
- Is often used to emphasize small graylevel differences (e.g. in medical imaging).
- Also often used in graphical display of data.
- If the color-LUT is mapped back to graylevels, the intensity should(!) be correct.

PSEUDO COLORS

- Pseudo-color images can be graylevel images where each graylevel is assigned an RGB value according to some LUT.
- Is often used to emphasize small graylevel differences (e.g. in medical imaging).
- Also often used in graphical display of data.
- If the color-LUT is mapped back to graylevels, the intensity should(!) be correct.
- Colormaps in plotting libraries are often pseudo-color LUT.

HOW TO CHOOSE THE CORRECT COLORMAP

- Often, we want the colormap to be *perceptually uniform*: Equal steps in the data are percieved as equal steps in the color space.

[1] See a more detailed wrap-up here: `https://bids.github.io/colormap/`

HOW TO CHOOSE THE CORRECT COLORMAP

- Often, we want the colormap to be *perceptually uniform*: Equal steps in the data are percieved as equal steps in the color space.
- The human brain percieves changes in lightness as changes in data better than changes in hue.

[1] See a more detailed wrap-up here: `https://bids.github.io/colormap/`

HOW TO CHOOSE THE CORRECT COLORMAP

- Often, we want the colormap to be *perceptually uniform*: Equal steps in the data are percieved as equal steps in the color space.
- The human brain percieves changes in lightness as changes in data better than changes in hue.
- Up until recently, Matlab (until 2014) and Python's Matplotlib (until 2.0 release) used `jet` as the default colormap for data display[1].

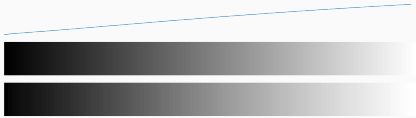
[1] See a more detailed wrap-up here: `https://bids.github.io/colormap/`

- Often, we want the colormap to be *perceptually uniform*: Equal steps in the data are percieved as equal steps in the color space.
- The human brain percieves changes in lightness as changes in data better than changes in hue.
- Up until recently, Matlab (until 2014) and Python's Matplotlib (until 2.0 release) used `jet` as the default colormap for data display[1].
- `jet` is a rainbow colormap that is not perceptually uniform.

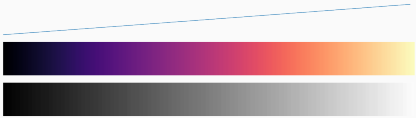
Figure 25: A gradient of `jet`, and the lightness component of the `CAM02-UCS` colorspace.

[1] See a more detailed wrap-up here: `https://bids.github.io/colormap/`

COMPARISON OF SOME COLORMAPS

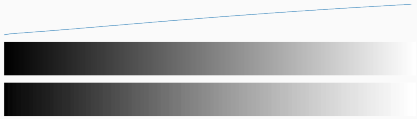


(a) `gray` (sequential)

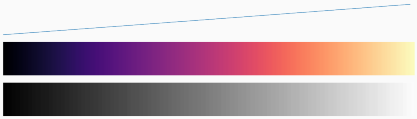


(b) `magma` (sequential perceptual uniform)

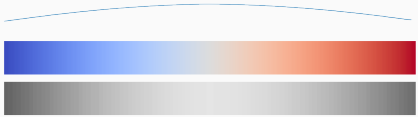
COMPARISON OF SOME COLORMAPS



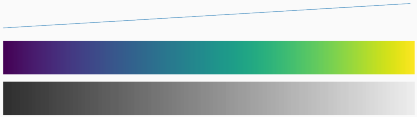
(a) `gray` (sequential)



(b) `magma` (sequential perceptual uniform)



(c) `coolwarm` (diverging)



(d) `viridis` (sequential perceptual uniform) (new matplotlib default)

Questions?