Université Nice SOPHIA ANTIPOLIS

Image Processing: Introduction

University of the Philippines

August 2006

Diane Lingrand

University of Nice – Sophia Antipolis, France

lingrand@polytech.unice.fr

http://www.polytech.unice.fr/~lingrand

2006

Sophia Antipolis

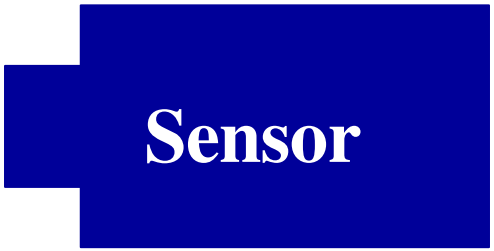
Course overview

- Basics: image format, color representation
- Quantification, Sampling and applications
- Image segmentation
- Noise and image restauration
- Image and video compression

Today

- Image acquisition
- Image format
- Data structures
- Color representation

Image formation (1)



Real world + *sensor* = *image*

Image Formation (2)

- Visual sensors :
  - photochimic (biological systems, photographic films)
  - photoelectric (CMOS, CCD)
- Other sensors
  - Medical imaging (IRM, rayons X, ...)
  - Sismic imaging

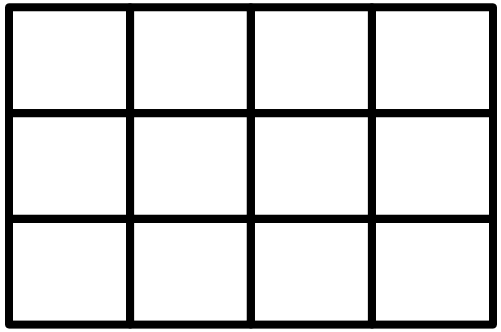
Signal

- Dimension :
  - 1D, 2D, 3D, 2D+T, 3D+T, ...
- Nature :
  - Analog
  - Digital
- How to put an image on a computer ?
  - Sampling (space)
  - Quantification (values)

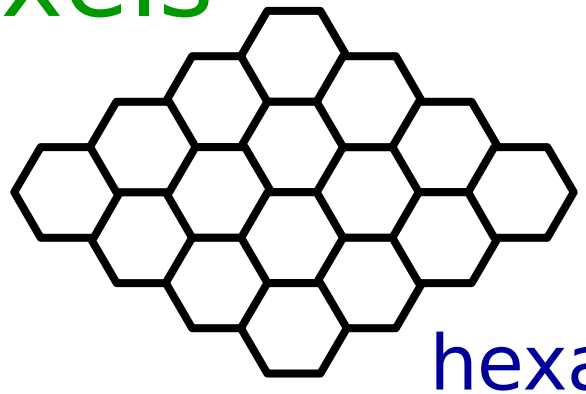
An image on computer

- An array of pixels
  - 1D array
  - 2D array
  - 3D array (video sequence or volumetric image)
  - ...
- A pixel (= picture element)
  - Its value is an intensity (scalar), a color ...
  - dimension ?

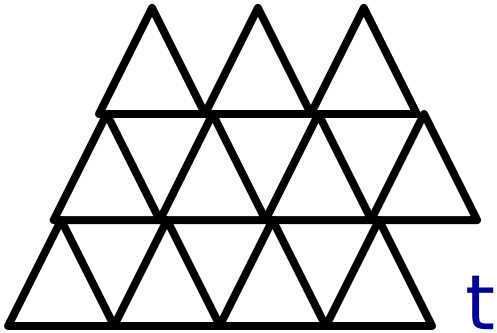
Tesselation: how to arrange the pixels



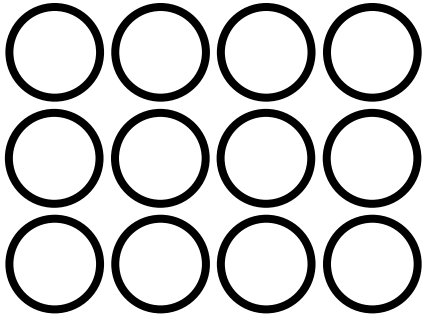
rectangular



hexagonal



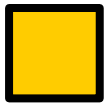
triangular



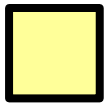
« quite real »

Neighbourhood

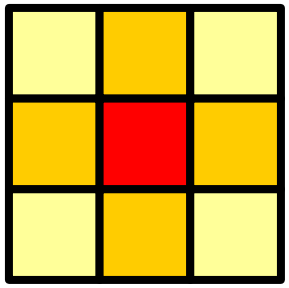
- Square or rectangular pixels :



4-connexity

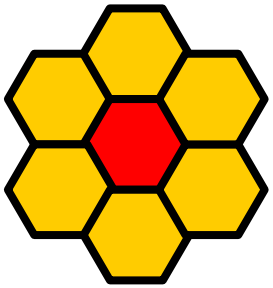


8-connexity

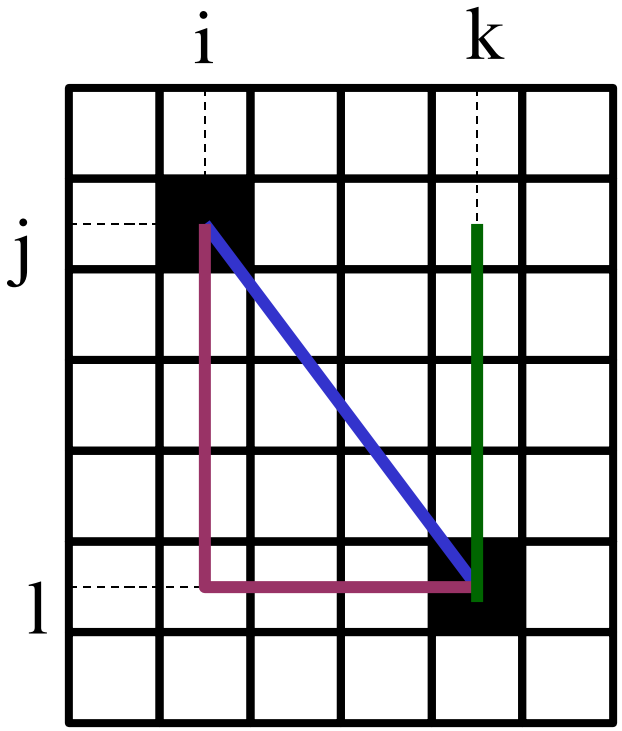


- Hexagonal pixels :

6 neighbours



Distances



- Euclidean Distance

$$\sqrt{(k - i)^2 + (l - j)^2}$$

- Block Distance (Manhattan)

$$| k - i | + | l - j |$$

- Chess Distance

$$\max(| k - i | , | l - j |)$$

2D visible images

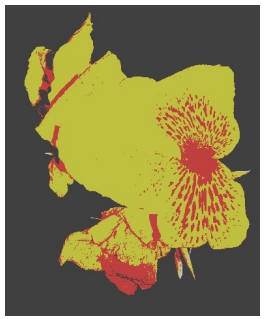
- Black and white images
  - a pixel value is black or white (binary)
- Grey images / Grey levels images
  - a pixel value is represented by a scalar value
- Color images
  - a pixel value is represented by 3 or more scalar values (RGB, …)
- Multi-spectral images, …

Quantification

- Only few grey levels or colors are considered:
  - grey levels: 256
  - number of colors: 256 * 256 * 256
- Consequences :
  - quality decreasing
  - artefacts

256*256*256
= 16777216 colors

8*8*8
= 512 colors



2*2*2 = 8 colors

Trade-off between ...

... visual quality and memory space.

The more colors ou grey levels we have, the better the visual quality but bigger the memory space used.

do not increase the quantification step if the improved quality is not visible.

Idea of values

*memory space * 2*

pixel on 8 bits	pixel on 16 bits
$2^8 = 256$ colors	2^{16} colors = 65536

*nb. of colors * 256*

Image Formats

or

How to memorize an image as a file on my computer disk.

2006

What do I need ?

- Necessary informations :
  - image width (in pixels)
  - image heigth (in pixels)
  - image value (intensity) for each pixel
- Image values :
  - are usually given line by line from top to bottom and then from left to right

A very simple image format: PBM

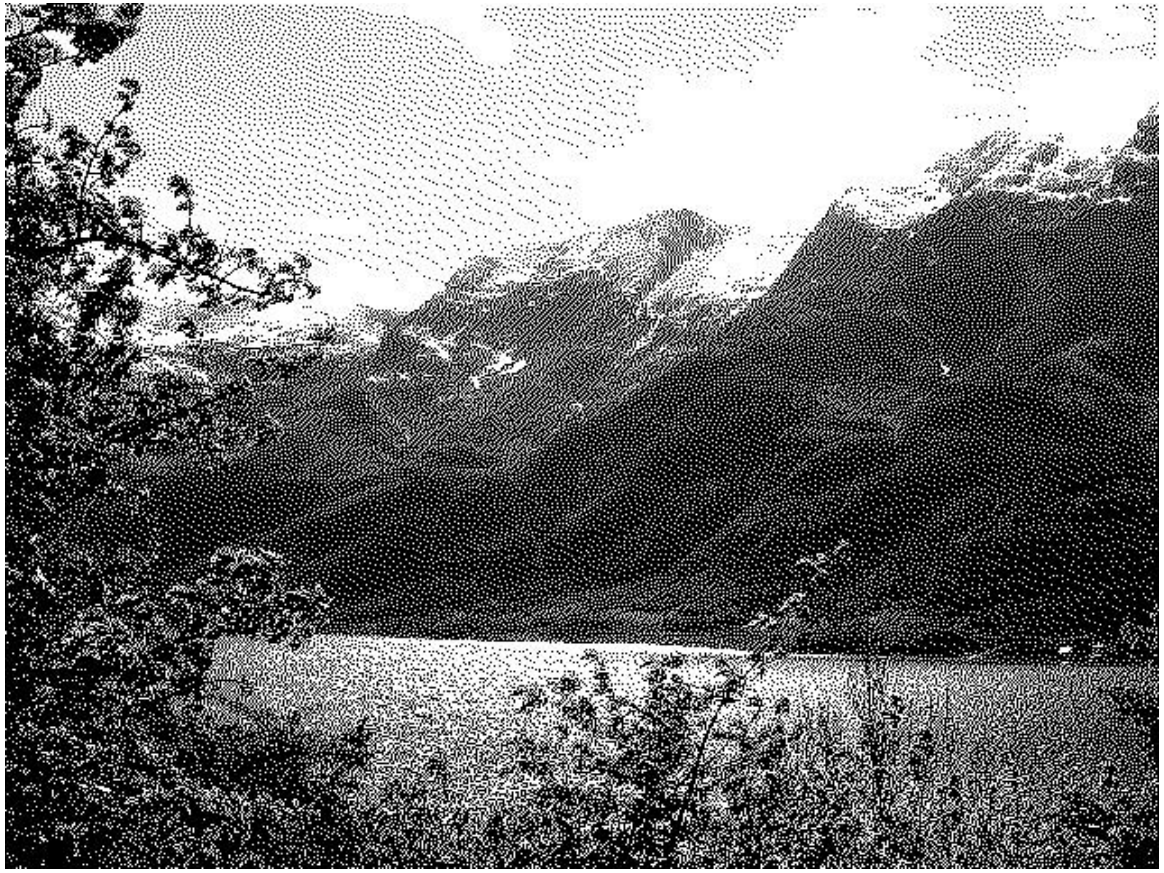
- Black and white images

```
> more ph021-ascii.pbm
P1
# CREATOR: XV Version 3.10a  Rev: 12/29/94 (PNG patch 1.2)
640 480
1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0
1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

indicates the format : PBM

width and height of images, in pixels l

The whole image in PBM



Another very simple format : PGM

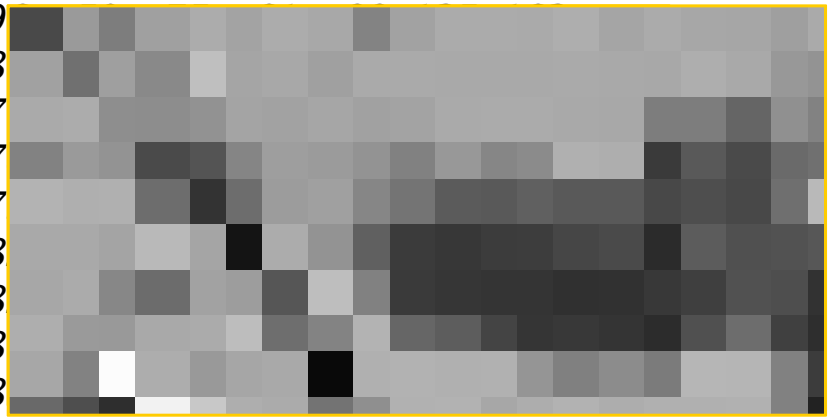
- For grey levels images

```
> more ph021-ascii.pgm
P2
# CREATOR: XV Version 3.10a  Rev: 12/29/94 (PNG patch 1.2)
640 480
255
 73 154 125 160 172 163 172 168 131 162 171 171 170 174 165 171 167
166 159 170 164 170 157  93  54  67  86  9
 54  85  82  70  66  72  45  56  56  54  8
169 175 182 174 175 172 172 176 175 173 17
104  99  91 102  95 102 117 103 106 190 17
176 176 175 177 179 179 178 180 182 180 17
180 178 178 179 179 179 181 181 183 185 18
183 184 183 185 182 184 186 186 184 185 18
185 184 184 186 183 184 184 187 190 183 18
188 187 188 187 185 191 189 185 186 188 18
```

indicates the format: PGM

width and height

max grey level



Previous image in PGM format

Another very simple format : PPM

- For RGB color images

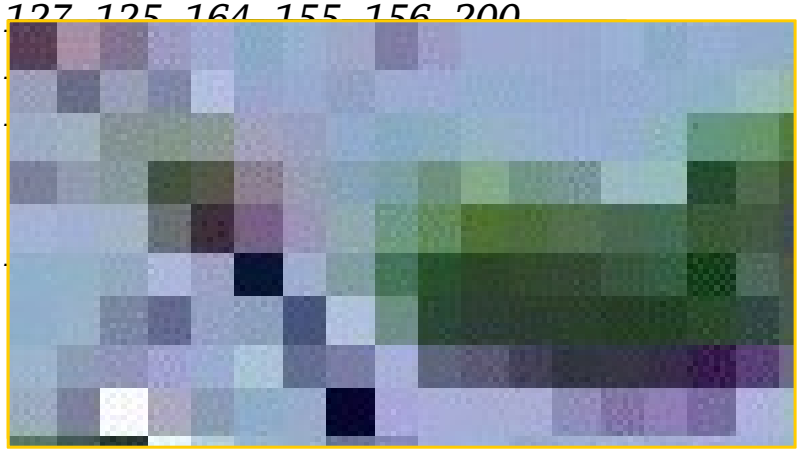
```
> more ph021-ascii.ppm
P3
# CREATOR: XV Version 3.10a  Rev: 12/29/94 (PNG patch 1.2)
640 480
255
89  62  79 167 143 165 130 117 145 155 155 191 160 169 210
137 170 203 151 176 206 158 167 198 127 125 164 155 156 200
156 169 211 155 170 211 153 170 213
150 174 210 146 171 202 140 175 195
144 175 177 148 183 177 140 175 142
 62  79  45  82  97  64  92 108  71
 64  79   0  87 109  26 129 157  82
 79 107  30  71 102  43  60  88  37
 44  55  21  53  67  31  51  69  27
```

indicates the format: PPM

width and height

each color component between 0 and 255

(r,g,b)



The same image but ... in color !

Size of files

- In our example (640x480):

PBM : 620 KB

PGM : 1,2 MB

PPM : 3,6 MB

Binary formats

- The headings are still in ASCII, the data is saved in binary.
- We need to know the pixel value representation (int, char, ….).
  - example of PGM : unsigned char (8 bits)
- Code in C / C++

```
size_t fread(void *data, size_t size, size_t nobj, FILE *stream)
size_t fwrite(const void *data, size_t size, size_t nobj, FILE
   *stream)
```

Be careful of little and big endians !

- Little endian :
  - light bytes will be stored to lowest addressel
  - ex: Intel processor
- Big endian :
  - heavy level bytes will be saved to the lowest adresses
  - ex: Motorola processor (Mac)
- Example : Byte3 Byte2 Byte1 Byte0 will be saved as :

Little endian	Big endian
@ + 0 : Byte0	@ + 0 : Byte3
@ + 1 : Byte1	@ + 1 : Byte2
@ + 2 : Byte2	@ + 2 : Byte1
@ + 3 : Byte3	@ + 3 : Byte0

Endians and Images

- Two choices are possible :
  - format depends on the computer that wrote the file: we need to store this information in the file heading
  - format does not depend on the computer that wrote the file: little or big endian is chosed for the file format
- Examples :
  - Big endian : Adobe Photoshop, JPEG, MacPaint
  - Little endian : BMP, GIF, QTM (QuickTime)
  - Both : TIFF, XWD (X Window Dump)

Back to our example

P1

PBM ascii : 620 KB

P4

PBM raw : 40 KB

** 16*

P2

PGM ascii : 1,2 MB

P5

PGM raw : 310 KB

**4*

P3

PPM ascii : 3,6 MB

P6

PPM raw : 900 KB

**4*

ASCII / binary : no confusion !

Compression

- Some consecutives pixels have same value : counting identical pixels:
  - [red, red, red] => [(red, 3)]
- Eliminate rare colors or similar colors
- Some pixel's values are rare, some others are frequent: using less memory for frequent pixels
- Same reasoning on image transforms.

Color Model

- Direct : pixel value corresponds to a grey level or to a (R;G;B) triplet value :

  *it is the pen's **color***

- Indirect: pixel value corresponds to a color index in a color table :

  *it is the pen's **index***

Other image formats

- without compression : PBM, PGM, PPM, TIFF
- with compression:
  - lossless :
    - TIFF (PZW)
    - GIF (8 bits, propriétaire)
    - PNG (licence GNU www.visualtek.com/PNG)
  - lossly : JPEG
- vectorial drawing : PostScript, SVG
- … back to our example :
  - JPEG format = 88 KB

Data structure

Data structure

- We want a structure that enables us :
  - to access to pixel values regarding the coordinates (i,j)
  - to browse an image from the first to the last pixel
  - to access to pixel (i,j) neighbours

Image frame

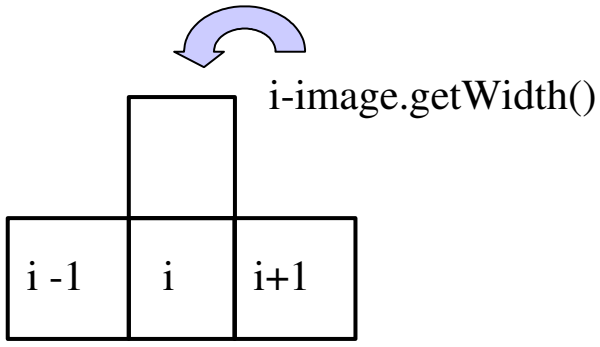
- Usually :

A 2D image seen as a 1D array

- allow to have only one loop for the entire image browsing :

```
for (int i = 0 ; i < im.getWidth()*im.getHeight(); i++)
   { ...}
```

- not so easy to access to neighbours:



A 2D image seen as a 2D array

- Allow to easily access to neighbours

  data[i+1][j-1] ....

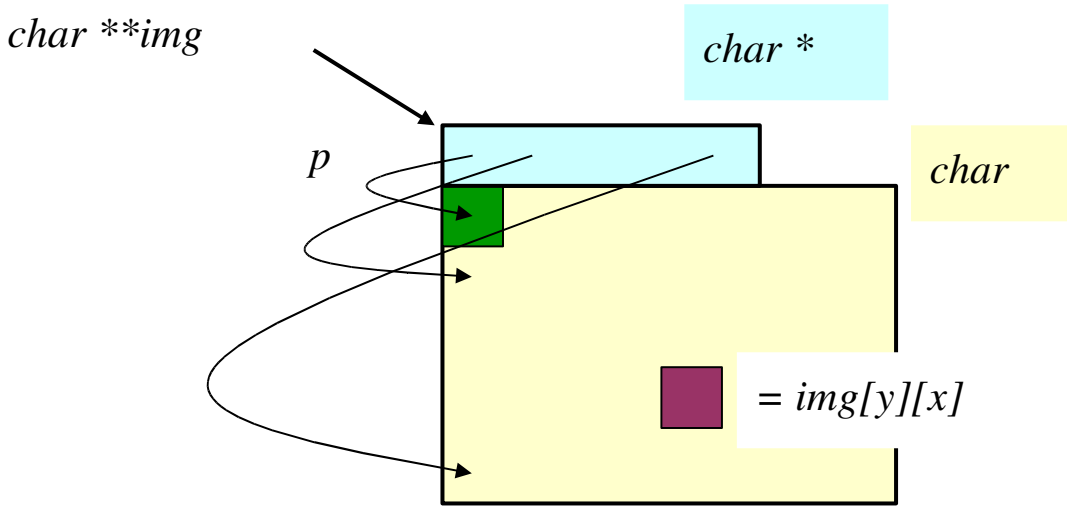
- Need two imbricated loops for the entire image browsing:

```
for( int i = 0 ; i < image.getWidth() ; i++) {
   for ( int j = 0 ; j < image.getHeight(); j++) {
                ....
}}
```

Several “tricks”

- Introducing an auxiliary counter variable :

  int ij = i + j*image.getWidth();
- Adding pointers to rows :



pointer on data :

*char *p = &img[0][0];*

Example in C

```
typedef struct {
    int width;
    int height;
    unsigned char
      *data;
    } ImageChar ;
```

```
typedef struct {
    int width;
    int height;
    unsigned char
      **img;
    unsigned char
      *data;
    } ImageChar ;
```

Example in C++ : template class

```
template <class Type> class Image {
   public:
        Image( const Type&);
   private:
        Type *data;
        int width, height;
...
}
```

File reading in C++

- Problem of templates instanciation: we do not know the type of pixels before file reading.
- Solution : polymorphism !
  - a mother class `BaseImage` with the template class as daughter
  - while reading, we create a `BaseImage` qui est that is in fact a `Image <short> i` or a `Image <double> i` ...

In Java

```
public class Image {
    private int width;
    private int height;
    private int [] data;

    public Image(int w, int h) {
                    height = h;
                    width = w;
                    data = new int [h*w];
    }

    public int getPixel(int i, int j) { .... }
    public int getPixel(int ij) { .... }
    public int setPixel(int i, int j, int value)
    { .... }
....
}
```

1 pixel = 32 bits

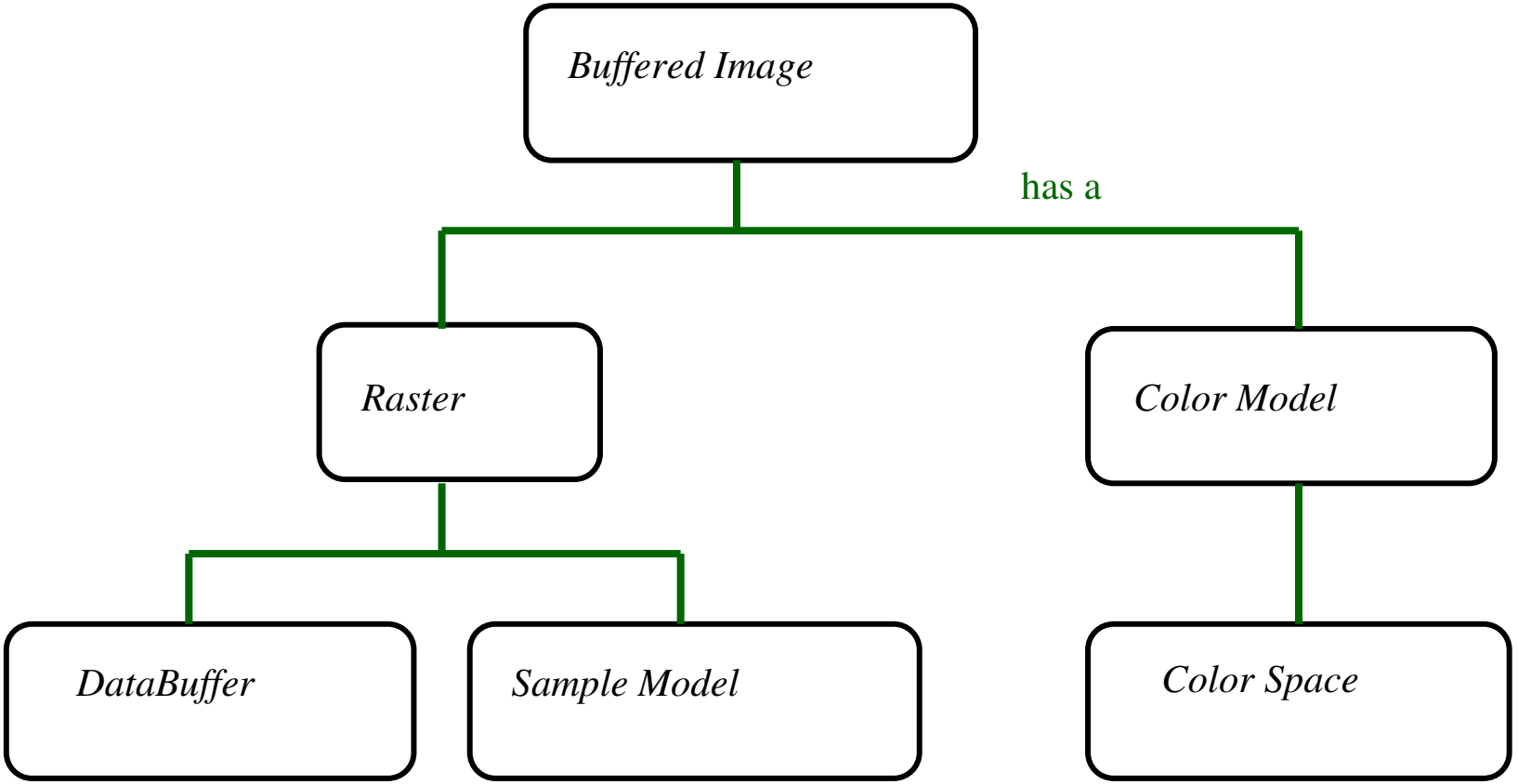
*= 4 * 8 bits*

Alpha (8 bits)	red (8 bits)	green (8 bits)	blue (8 bits)

Images in Java

- *Push Model* ou *Producer/Consumer* (jdk 1.1.*, 1.2.*)
  - Image
  - ImageProducer
  - ImageConsumer
  - ImageObserver
- *Immediate Mode* ou *Image Buffer Model* (new in Java 2D)
  - BufferedImage, Raster
  - BufferedImageOp, RasterOp
- *Pull Model* (JAI = Java Advanced Imaging)
  - RenderableImage, RenderableImageOp, ...

BufferedImage



Java 2D

- *Imaging Interfaces*
- *Image Data Class*
  - *BufferedImage, LookupTable, DataBuffer, DataBufferByte, DataBufferInt, ..., Kernel, Raster, ...*
- *Image Operation Classes*
  - *AffineTransformOp, BandCombineOp, BufferedImageFilter, ColorCongreenOp, ConvolveOp, LookupOp, RescaleOp*
- *Sample Model Classes*
  - *BandedSampleModel, ComponentSampleModel, ...*
- *Color Model Classes*
  - *ColorModel, ComponentColorModel, DirectColorModel, IndexColorModel, PackedColorModel*
- *Exception Classes*
  - *ImagingOpException, RasterFormatException*

Input - Output

- Javax.imageio.ImageIO (since j2sdk 1.4.0)
  - Simplification of reading and writing
  - Formats : gif, jpeg, png, xbm
  - Possibility of adding other formats
- Image file reading

```
File f = new File("toto.gif");
BufferedImage bi;
try { bi = ImageIO.read(f);} catch(IOException e) {...}
```

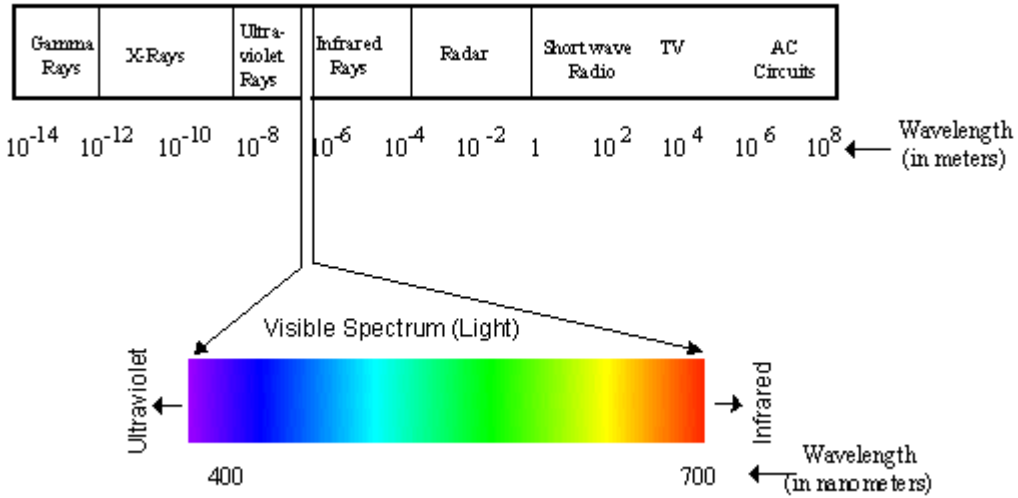
- Image file writing

```
File f = new File("toto.jpg");
try { ImageIO.write(bi, "jpg", f);} catch(IOException e)
   {...}
```

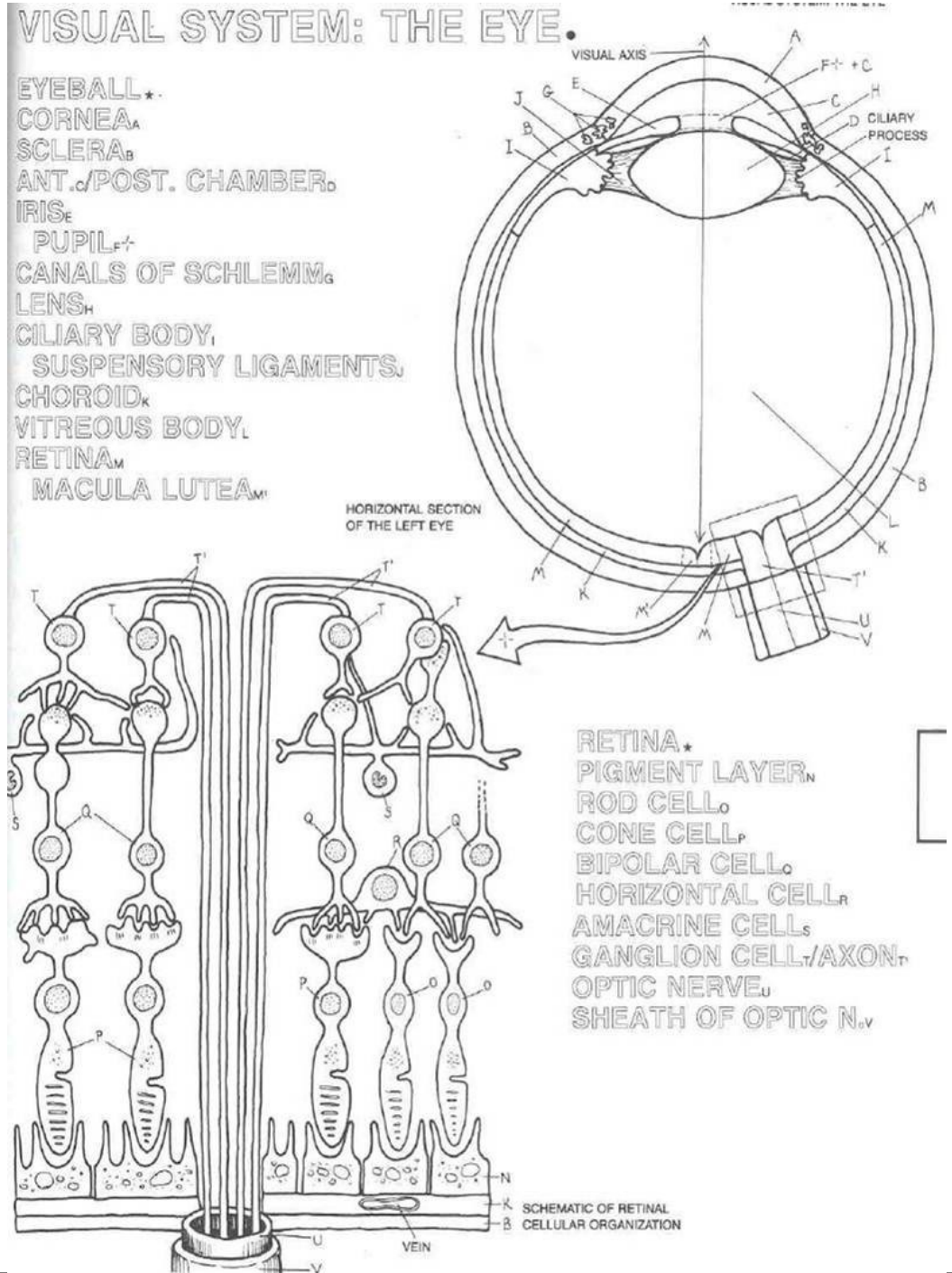
A very famous image : LENA (http://www.lenna.org/)

Let's talk about color

What is color ?



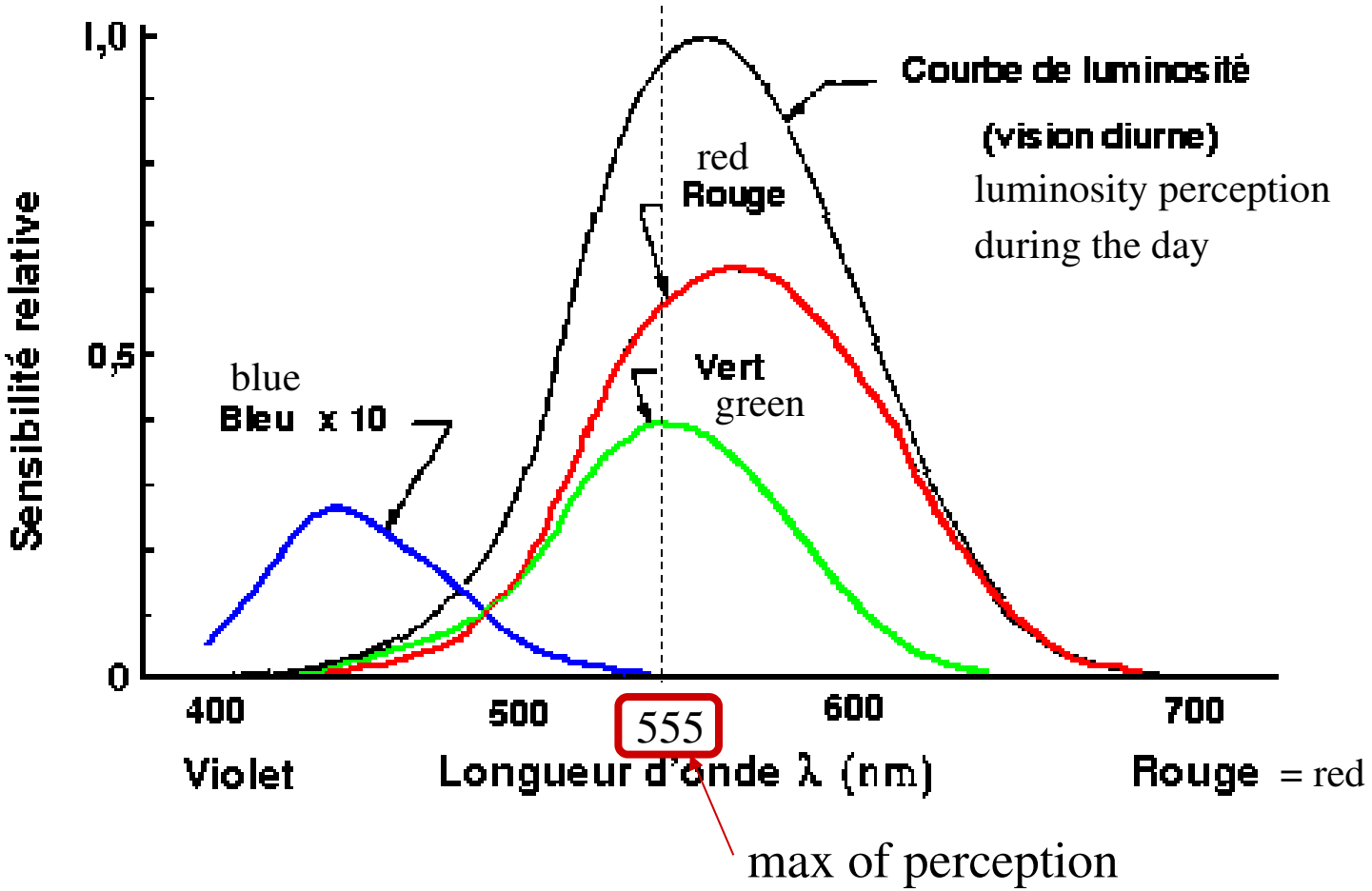
- physical process
  - physical interactions of electromagnetic waves with physical objects
- psycho-physiological process
  - our visual system's interpretation of these penonomenons (eye + brain)



Color perception

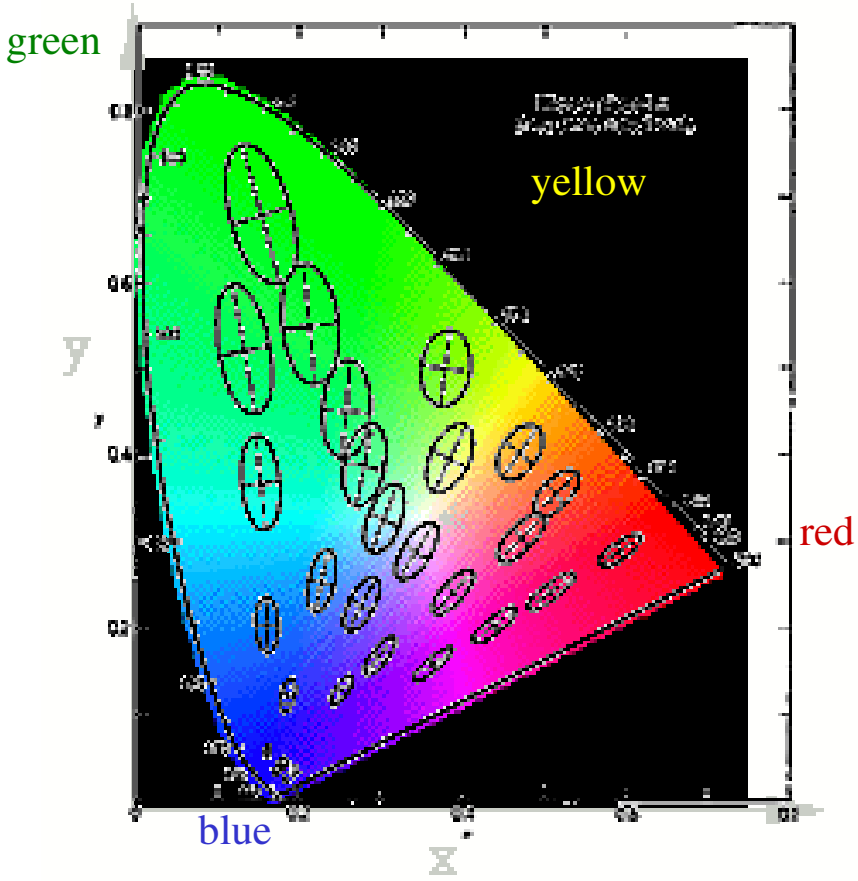
- in the retina: photoreceptors
  - rod cells
    - 1 pigment
  - cone cells
    - 3 pigments:
      - red
      - green
      - blue

Eye's sensibility

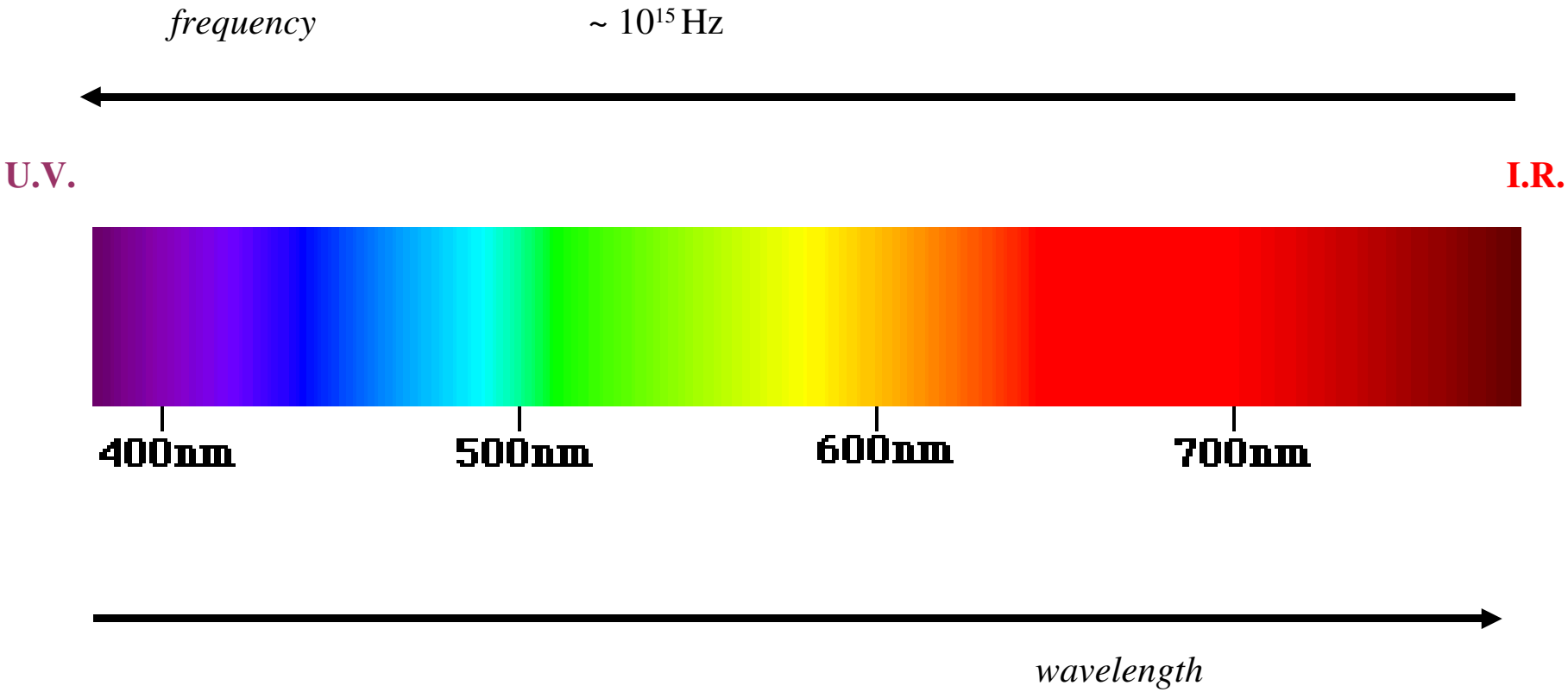


Eye's performances

- The eye distinguishes:
  - 350000 colors,
  - 150 hues
  - 3000 colored lights of same intensity



Light spectrum



Definitions

- hue : name of the color, dominant wavelength
  - non additive
- saturation : degree of dilution in white
- luminosity : intensity of achromatic light
  - measurable and additive
  - unit: $1cd.m^{-2} = 10$ nits

Small history of color photography

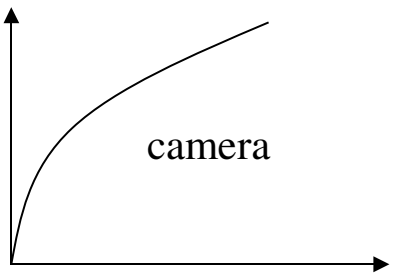
- 1848 : 1st photography of the visible spectrum by Becquerel
- 1870 : black chamber de Ducos du Hauron
- 1894 : interferentiel process de Gabriel Lippman
- 1907 : autochrome process from the Lumières brothers
- 1930 : Kodachrome system
- 1932 : Color film

Color acquisition

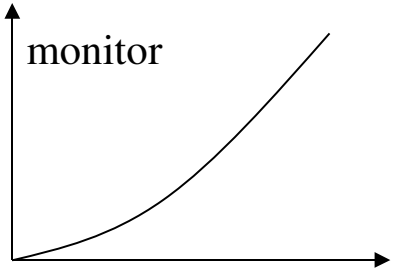
- 1st step :
  - Filter to stop infra-red wavelenghts
- Then :
  - colored optical filters
    - filters + monochrome camera
  - mono-CCD color camera
    - overload of filters masks
  - tri-CCD color camera
    - prism duplicates the image in 3

Gamma correction

- The intensity measured by cameras depends on a concave logarithm of real intensity. Colors appear less satured and chromatic coordinates are erroneous.



$$I = A.D^{\gamma}$$



Examples of gamma correction

1

1.7

2

0.7

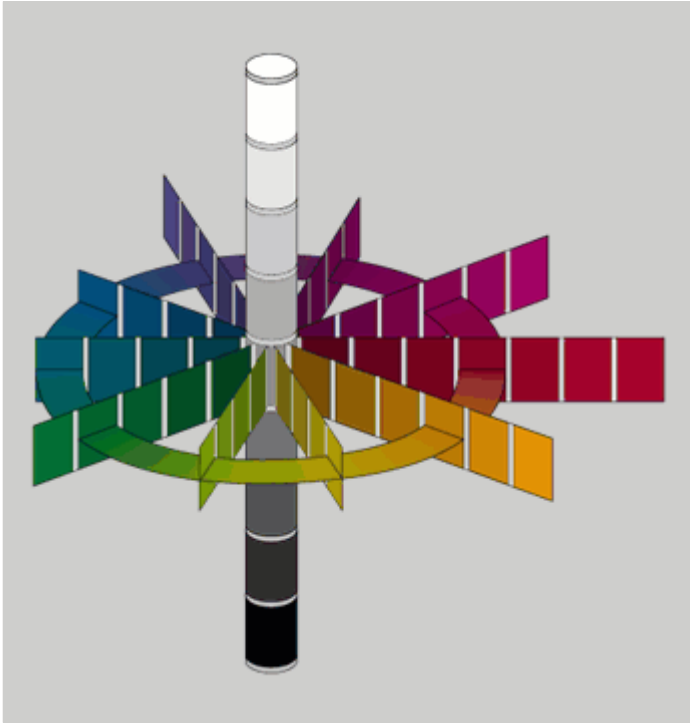
1

1.5

Color representation

Color representation

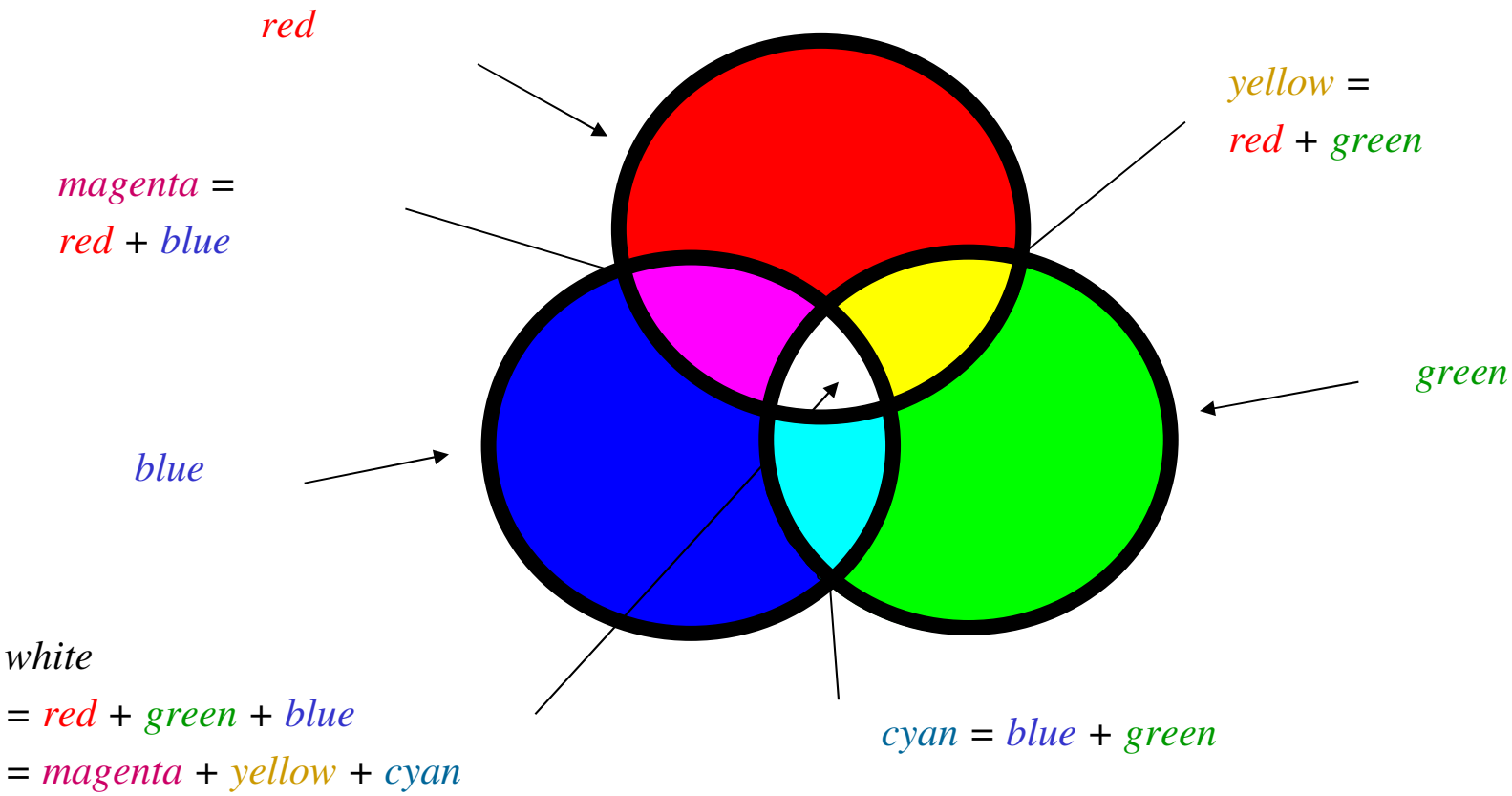
- Color Atlas :
  - Munsell, 1929
    - ordered by tonality, clarity and saturation
    - 10 tonalitiess, 10 levels of clarity
  - NCS (*Natural System of Colors*)
- 3-D space representations :
  - based on our perception
  - related to physical entities



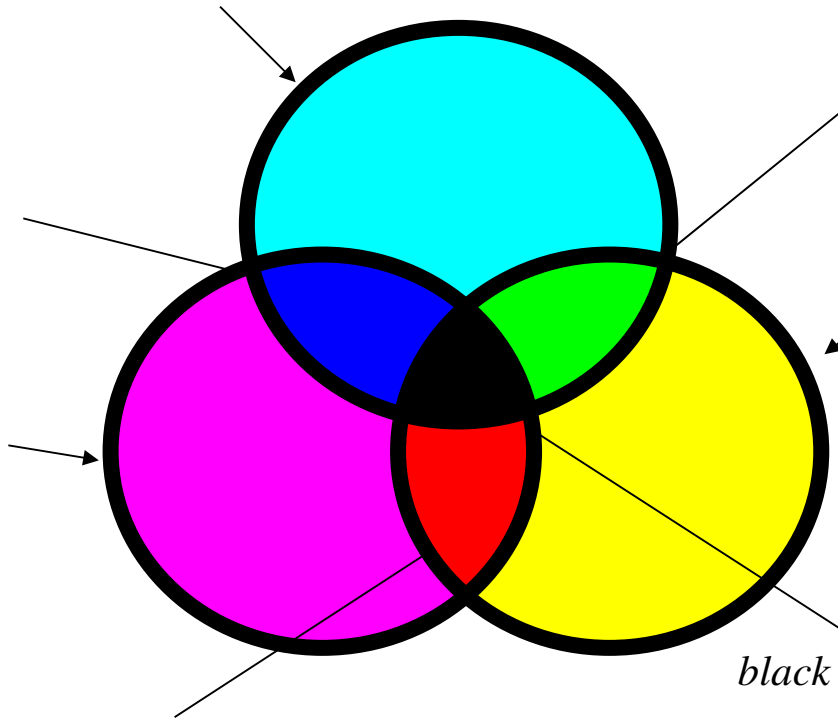
3D space

- primary colors (RBG)
  - red (700 nm), green (546 nm), blue (435.8 nm)
  - used by projection
- secondary colors (YCM)
  - yellow, cyan, magenta
  - used by printers

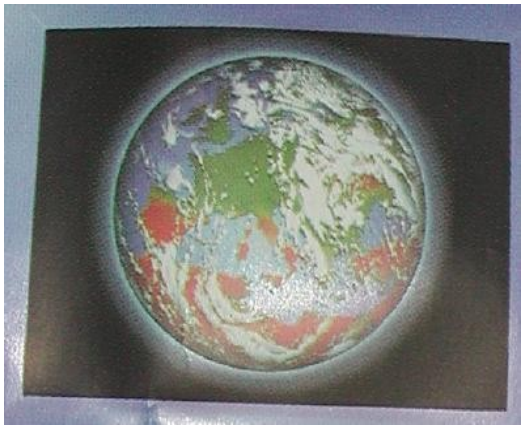
additives colors

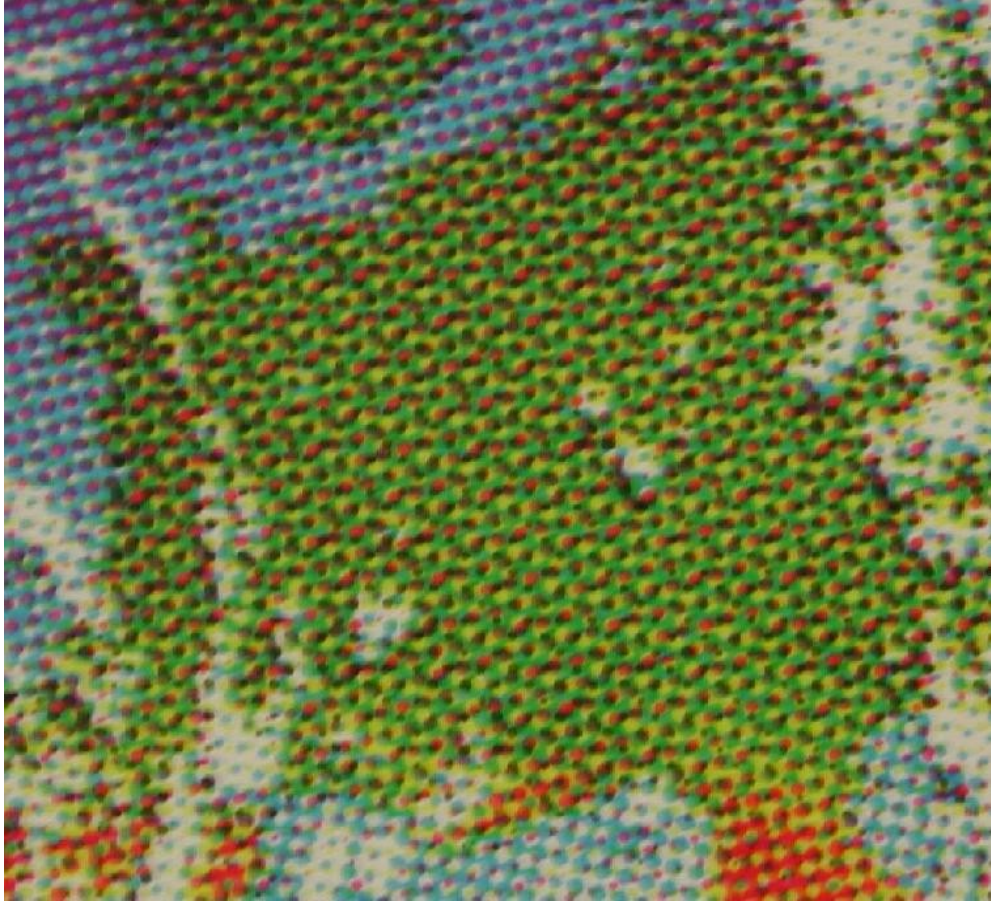


substractives colors

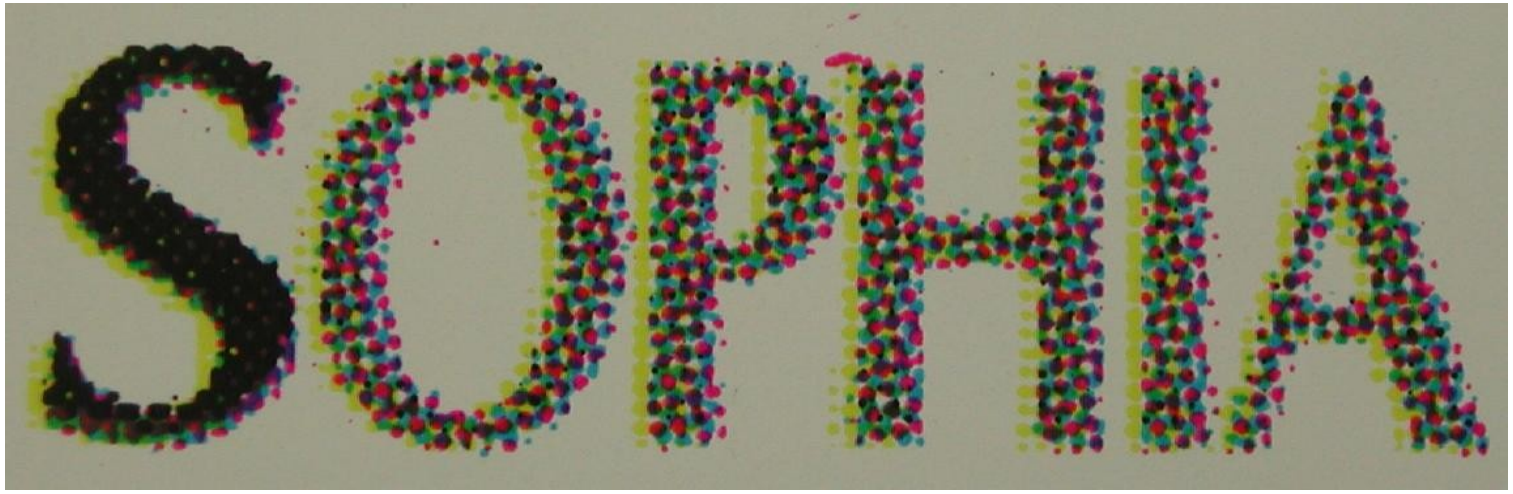


Printing





UNIVERSITÉ
NICE
SOPHIA
ANTIPOLIS



Addition - substraction

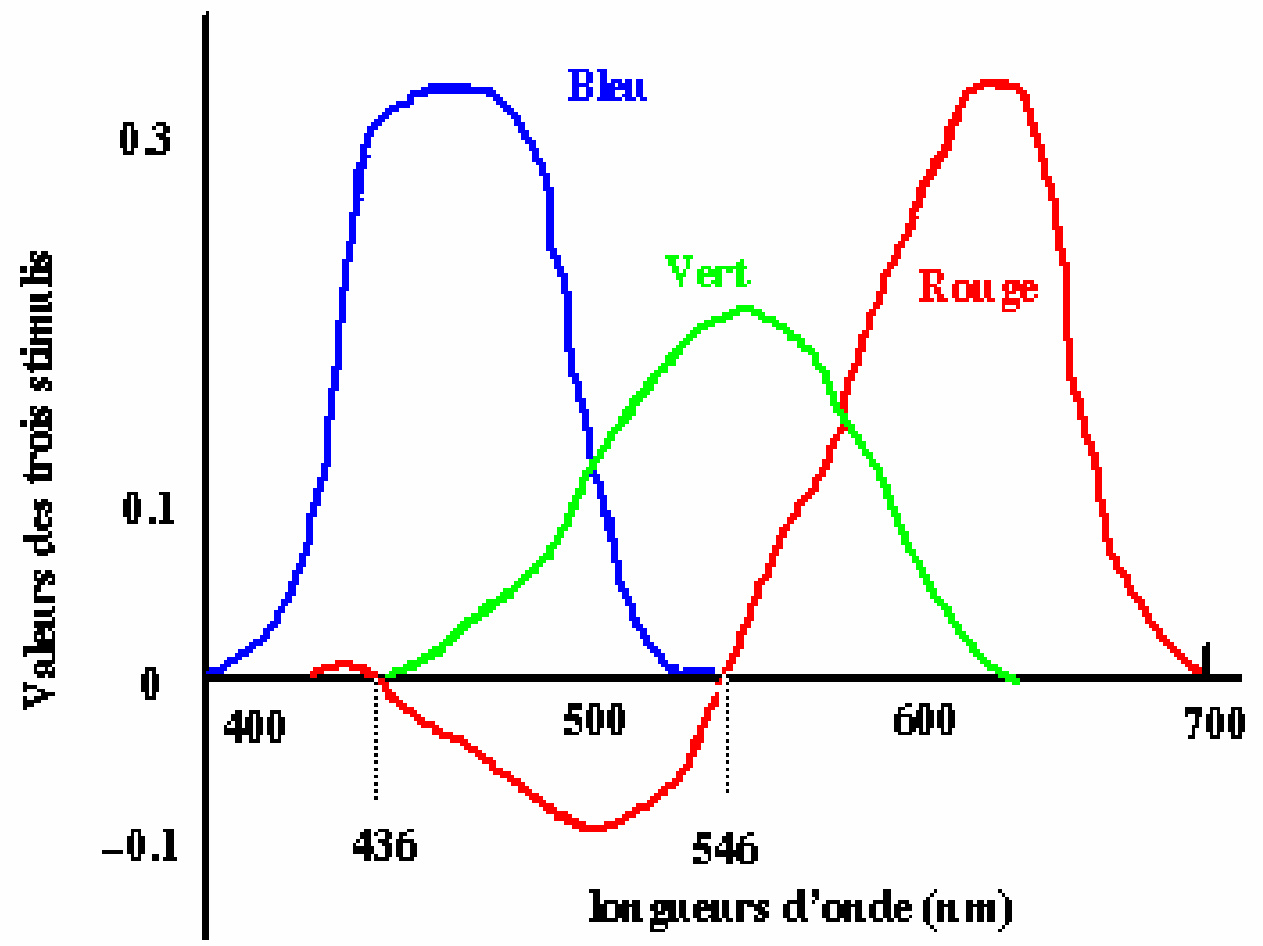
- color **C** = $\underline{r}$**R** + $\underline{g}$**G** + $\underline{b}$**B**
- Coordinates or chromaticity values :

$\underline{r} = r / (r + g + b)$

$\underline{g} = g / (r + g + b)$

$\underline{b} = b / (r + g + b)$

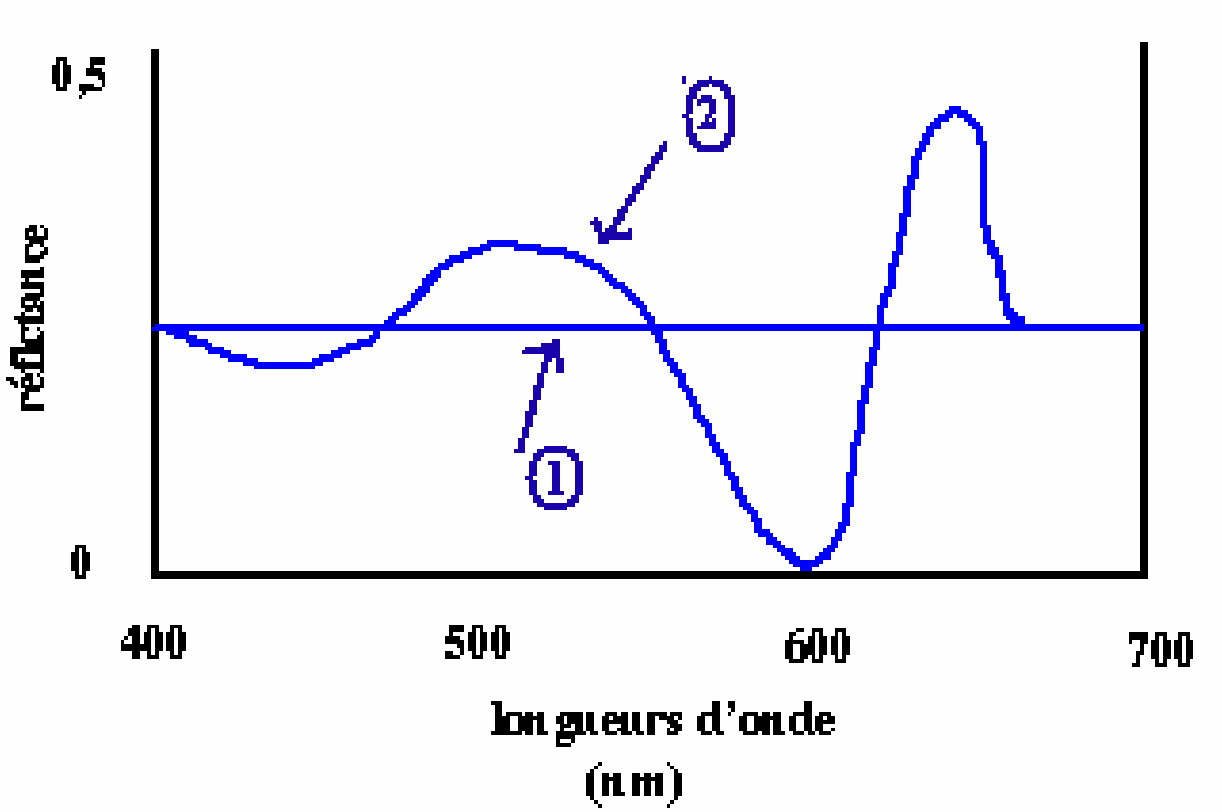
RGB color model



wavelength

metamerism: different spectrum, same color

luminosity

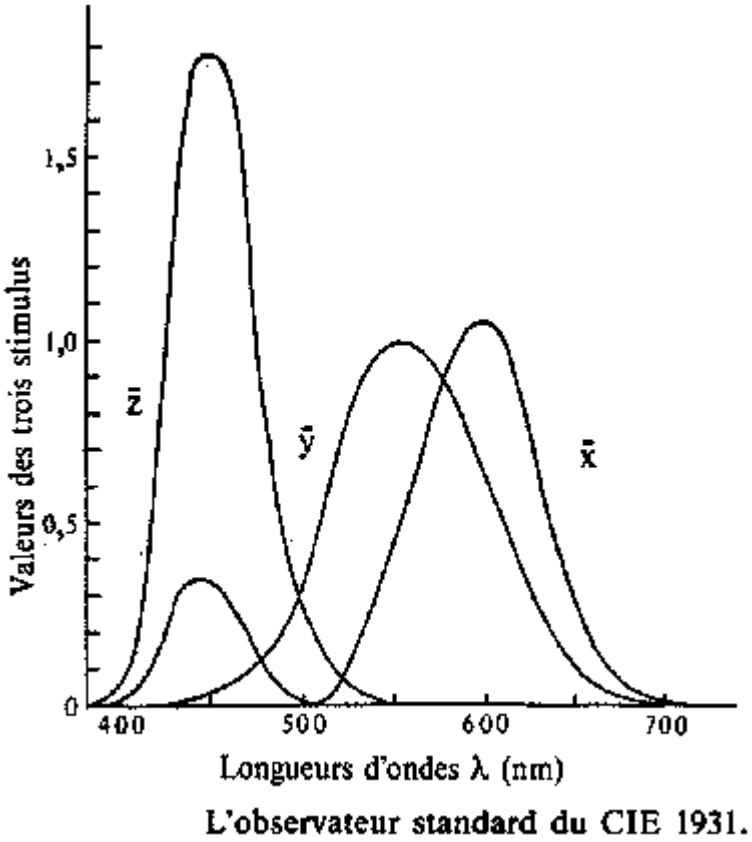


wavelength

-> color blindness

Chromaticity diagram CIE (1935)

- Primary colors :
  - Theoretical colors : X Y et Z
  - Y luminance
  - Equal proportions gives white
- No more negative values
- CIEXYZ model



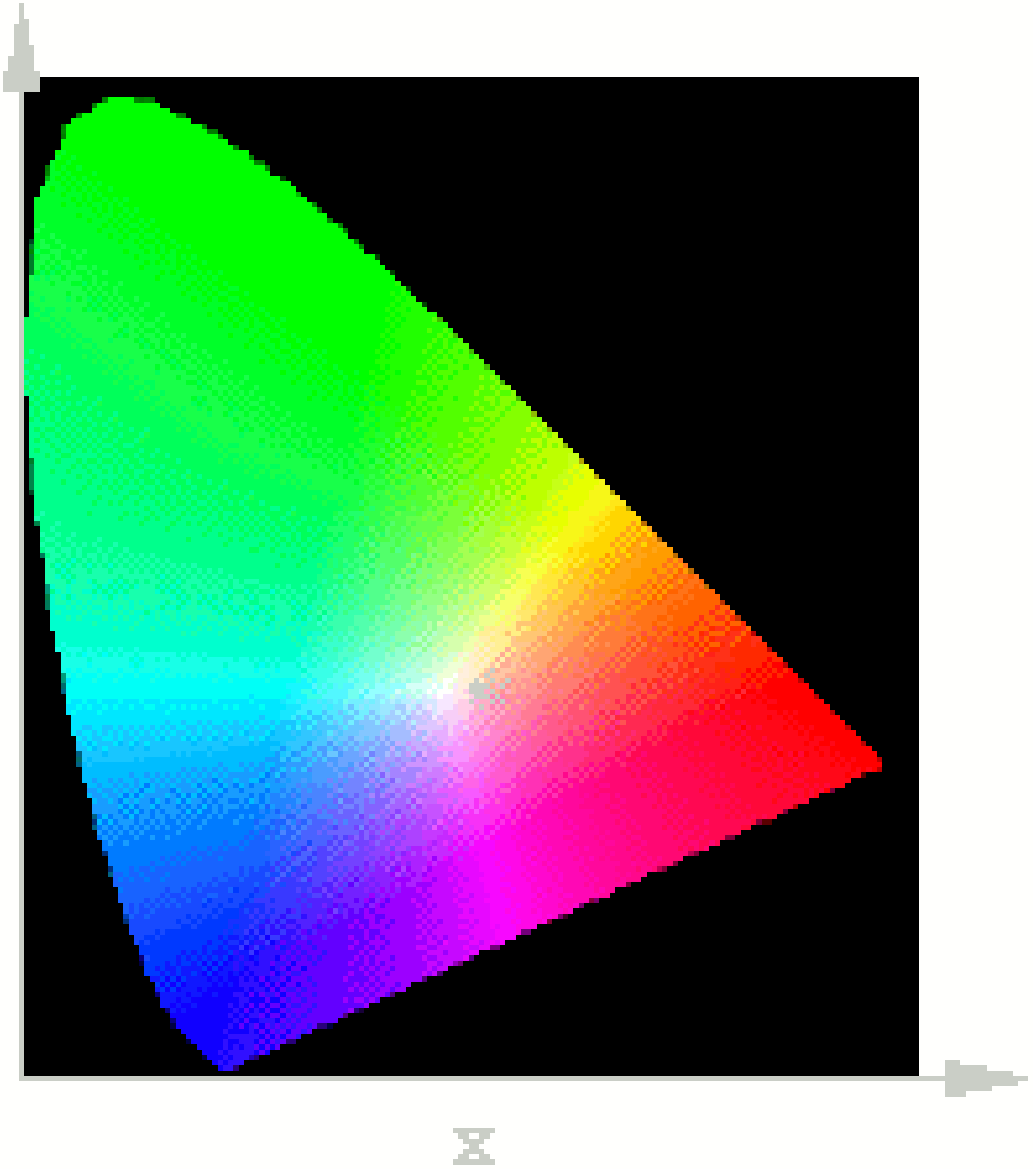
L'observateur standard du CIE 1931.

CIE XYZ

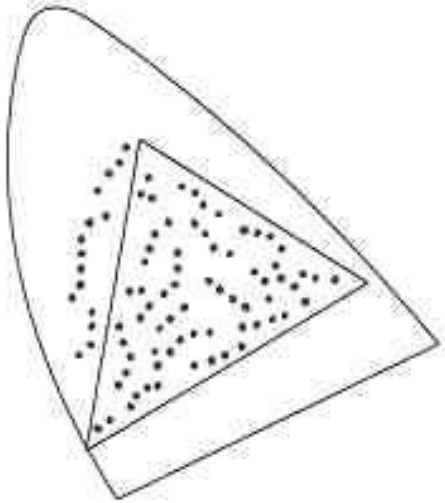
$$X = 0.489989\ R + 0.310008\ G + 0.2\ B$$
$$Y = 0.26533\ R + 0.81249\ G + 0.01\ B$$
$$Z = 0.0\ R + 0.01\ G + 0.99\ B$$

$$R = 2.3647\ X - 0.89658\ Y - 0.468083\ Z$$
$$G = -0.515155\ X + 1.426409\ Y - 0.088746\ Z$$
$$B = -0.005203\ X - 0.014407\ Y + 1.0092\ Z$$

Chromaticity diagram



colors



All perceivable colours
(the human visual system)

Colours reproducible
on a monitor

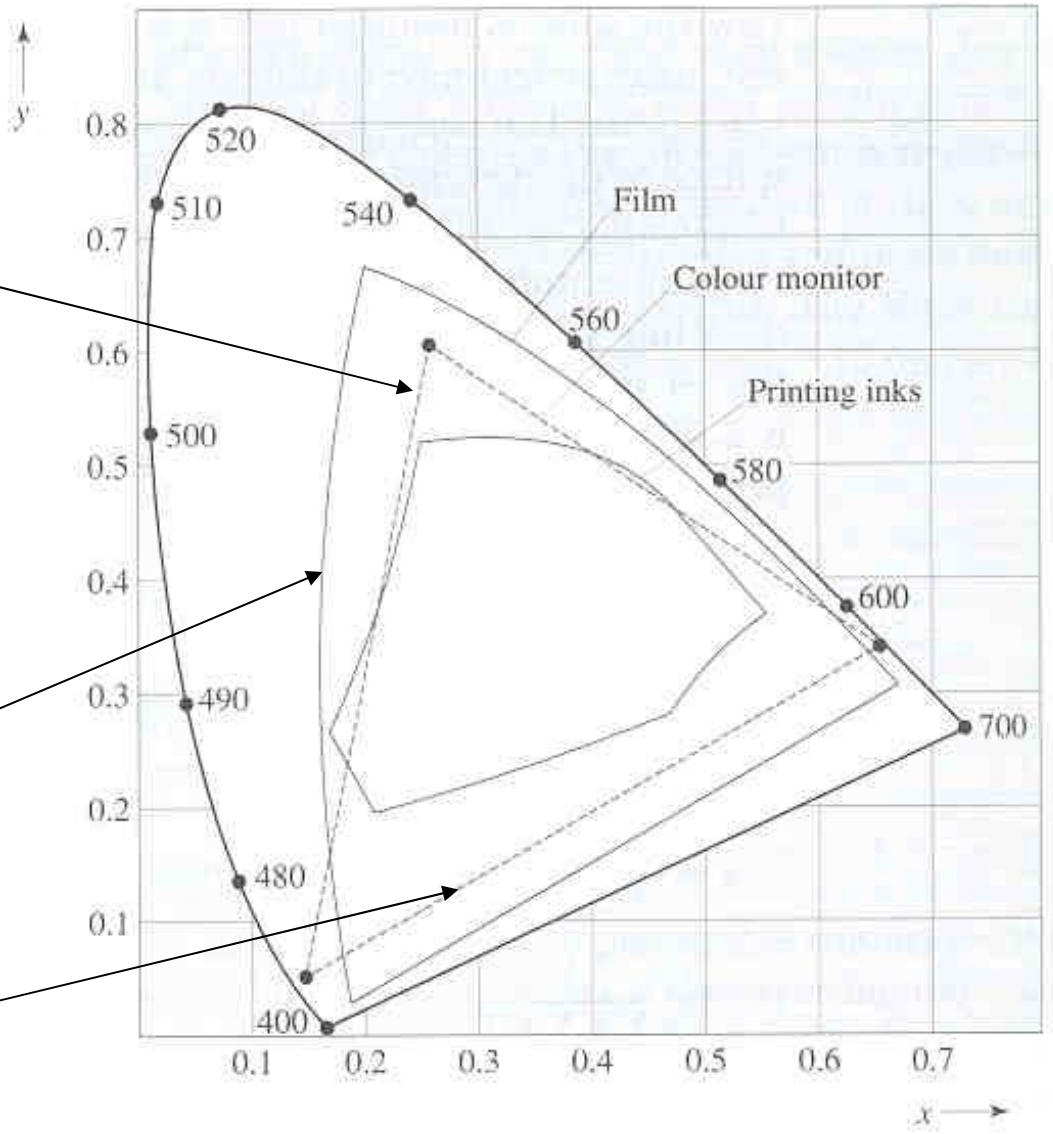
Colours calculated by a program
(normally we would require these
to fall within the monitor gamut)

colors (2)

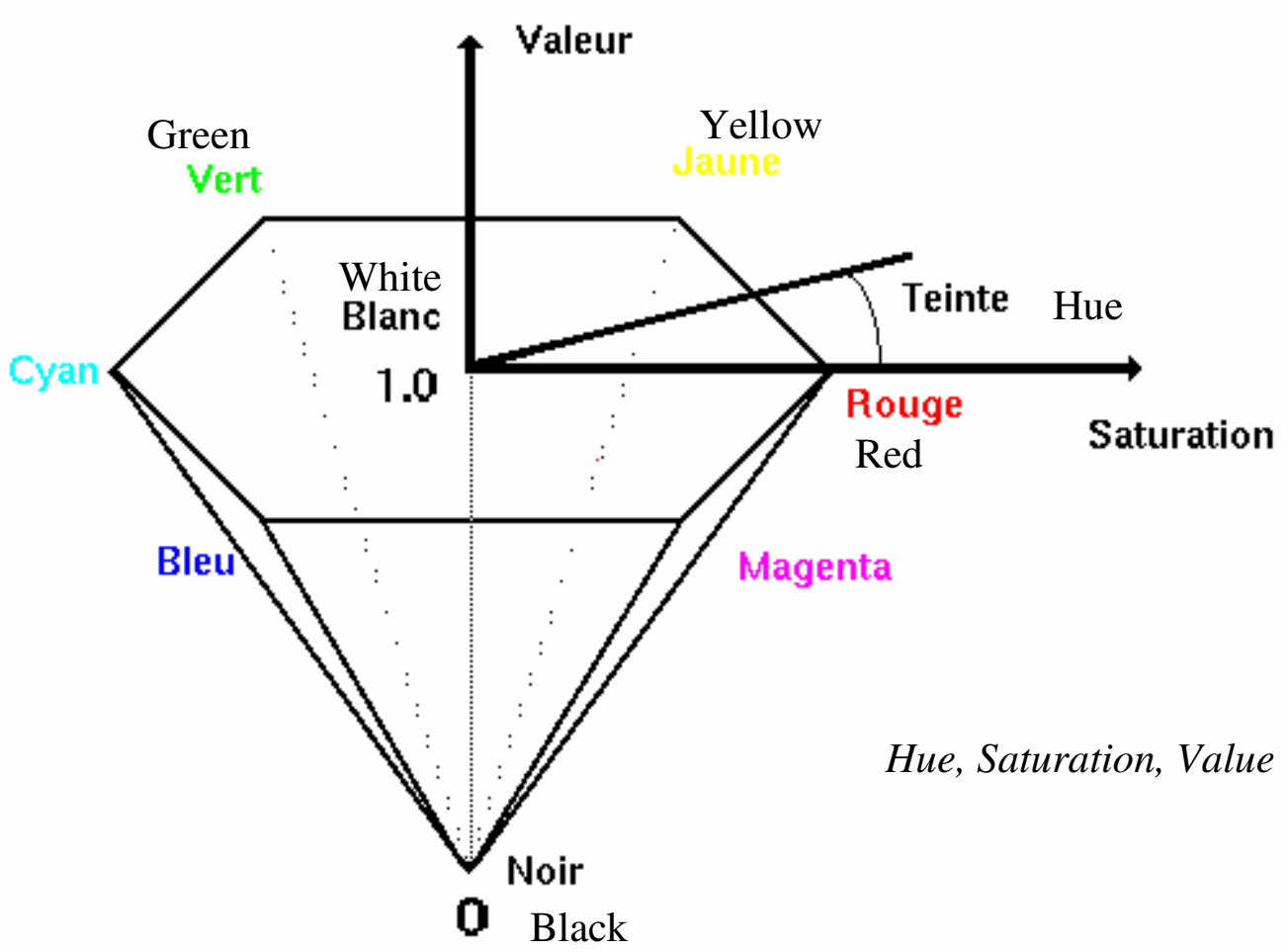
Colour monitor (sRGB)

Film

Printing inks



Smith's HSV Pyramid



Hue, Saturation, Value

Other models

Luminance : $Y = 0.299\ R + 0.587\ G + 0.114\ B$

- Betacam ($Y\ p_b\ p_r$)

  $p_b = 0.5\ (B - Y) / (1 - 0.114)$ $\quad$ $p_r = 0.5\ (R - Y)/ (1 - 0.299)$
- Digital Video System ($Y\ C_b\ C_r$)

  $C_b = 128 + 112\ p_b$ $\quad$ $C_r = 128 + 112\ p_r$
- YUV

  $U = 0.493\ (\ B - Y\ )$ $\quad$ $V = 0.877\ (\ R - Y)$
- NTSC YIQ

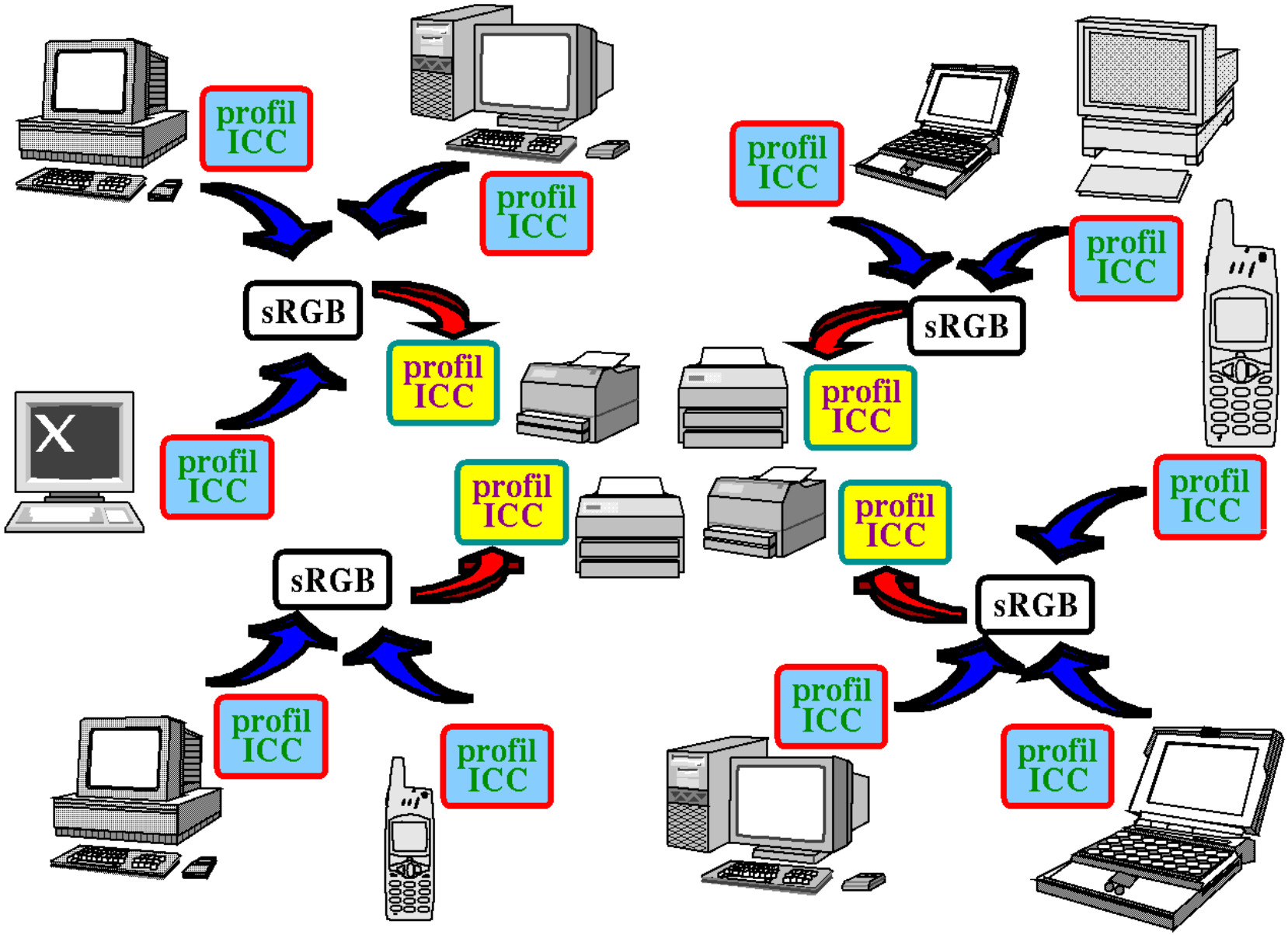
  $I = 0.6\ R - 0.28\ G - 0.32\ B$ $\quad$ $Q = 0.21\ R - 0.52\ G + 0.31\ B$

sRGB format

- in order to normalize the representation of \ RGB / YCMK for all printers, monitors, ...
- RGB + gamma de 2.2
- exists in Java2D

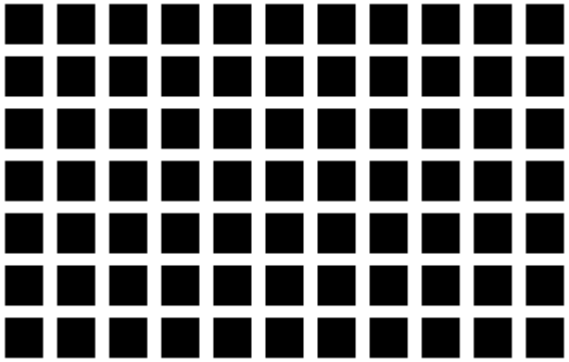
sRGB monitor

without transformation !


profil ICC
profil ICC
sRGB
profil ICC
profil ICC
profil ICC
profil ICC
sRGB
profil ICC
profil ICC
profil ICC
profil ICC
sRGB
profil ICC
profil ICC
profil ICC
profil ICC

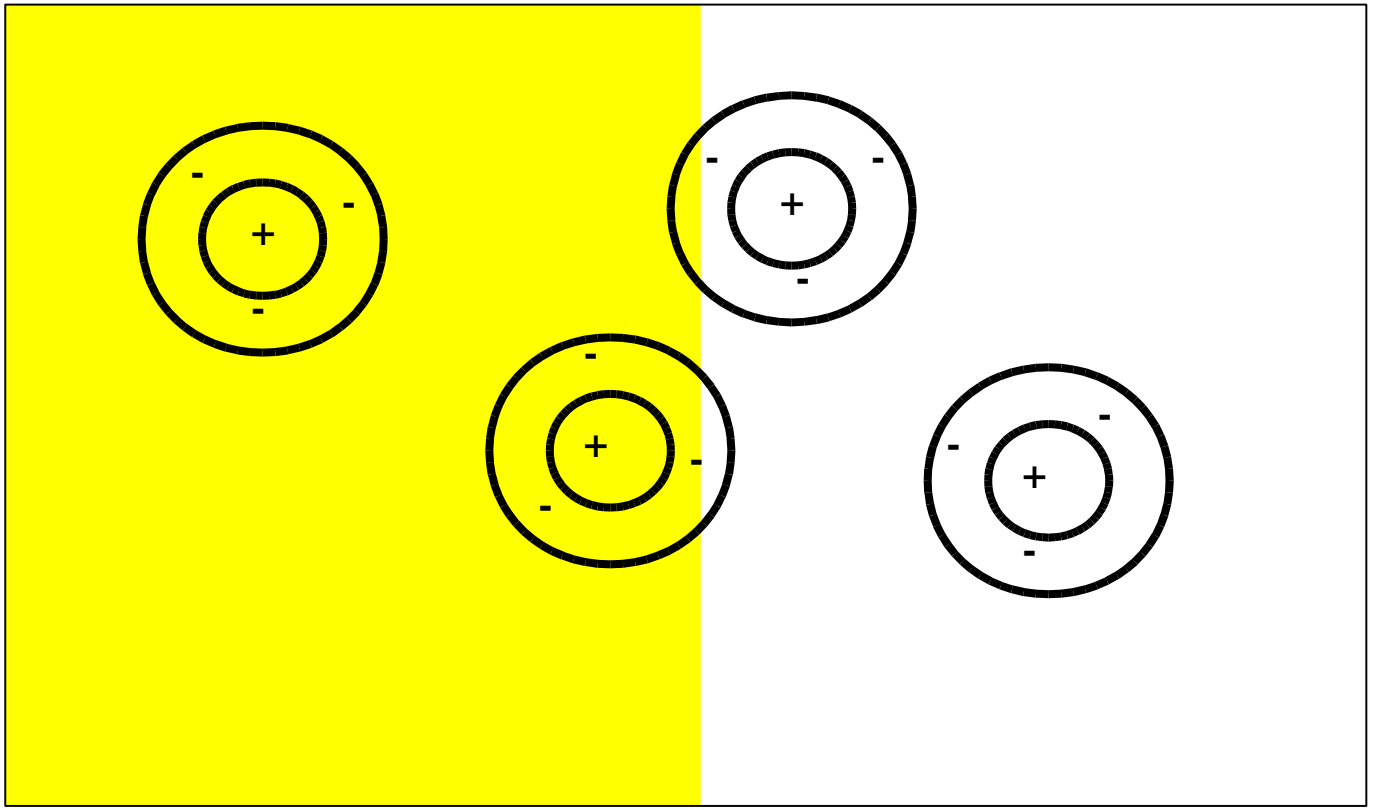
2006

Illusions in color perception



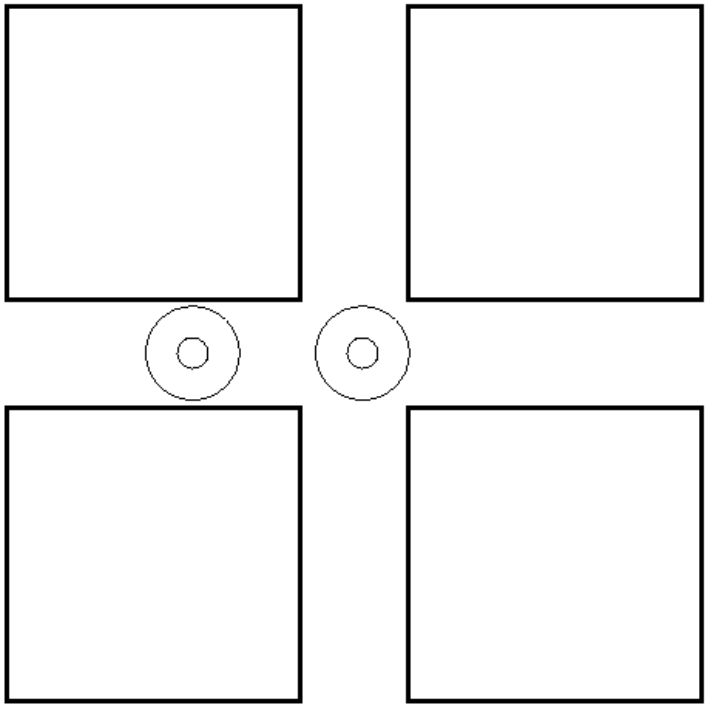
the Hermann Grid Illusion

Cells « on-center »

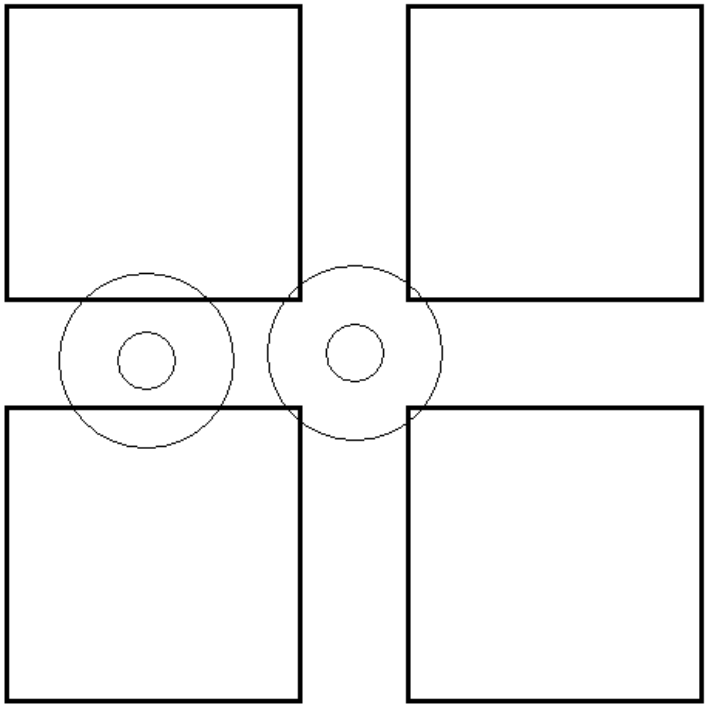


Hermann Grid Illusion

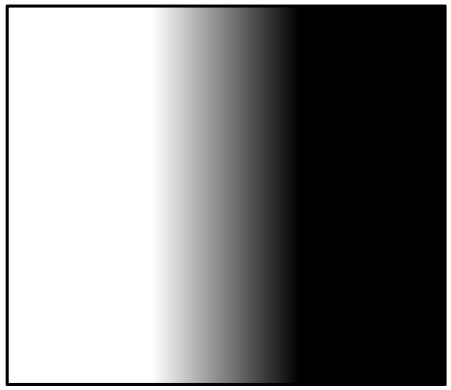
- Near the fovea

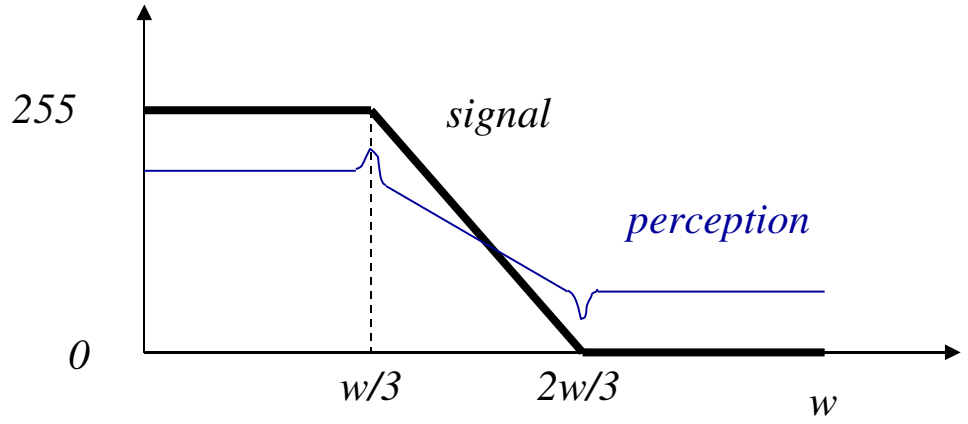


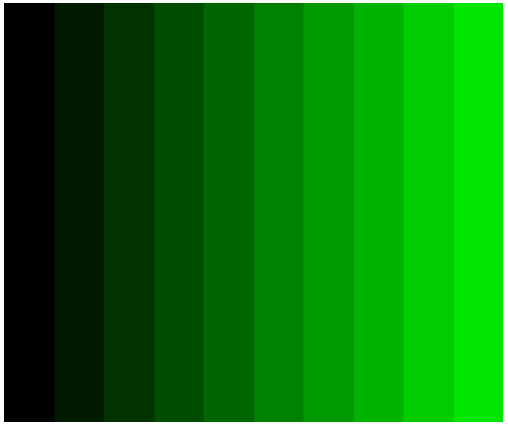
- Peripheric area

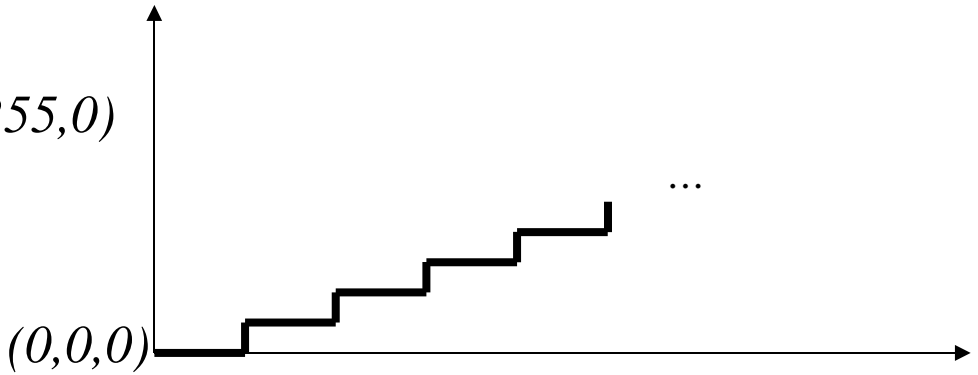


Mach Bandes

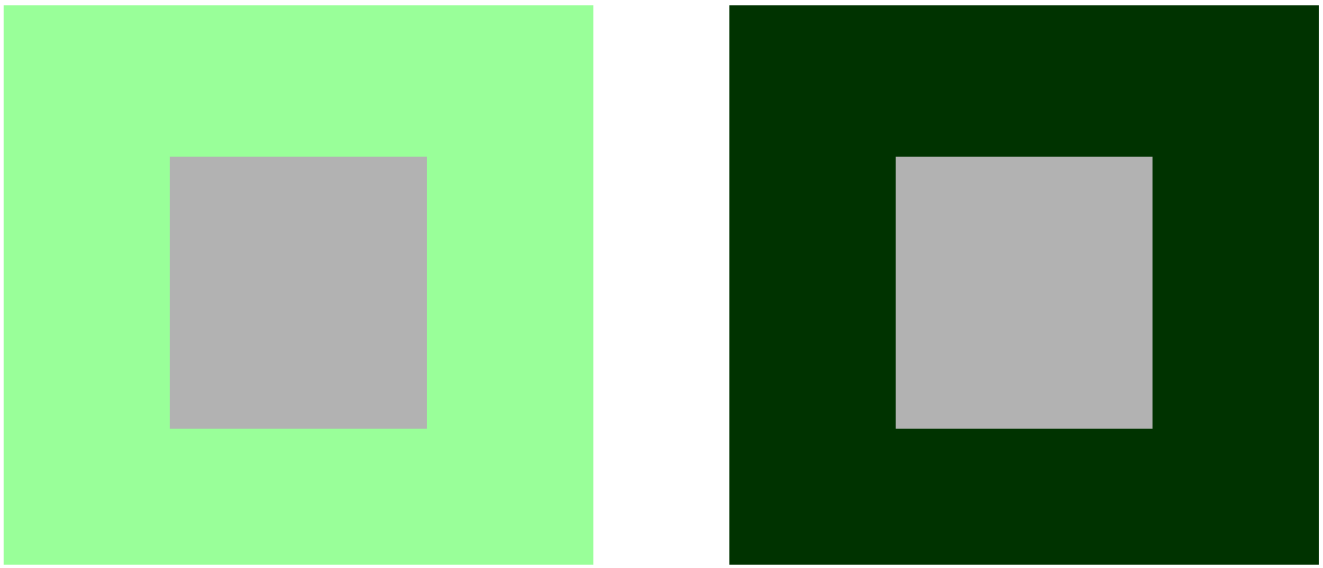




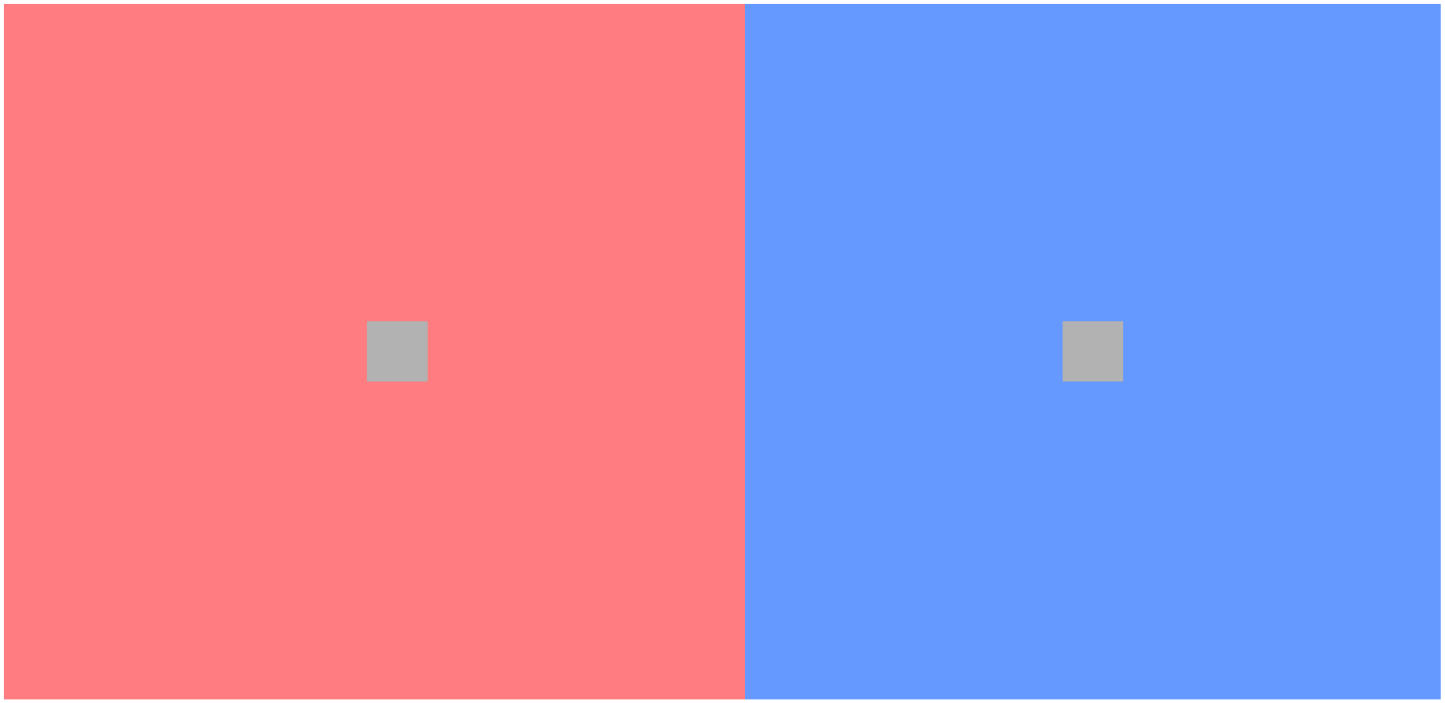




Illusions (suite)



Illusions (suite)



Bibliography

- “Digital Image Processing" par W. K. Pratt, John Wiley & Sons, inc., Third Edition, 2001
- "Digital Image Processing" par Gonzalez et Woods, Prentice Hall, Second Edition, 2002
- http://www.dai.ed.ac.uk/HIPR2/
- Books available on the web http://homepages.inf.ed.ac.uk/rbf/CVonline/books.ht
- Computer Vision Online http://www.dai.ed.ac.uk/CVonline/transf.htm
- Vetterli's talks (wavelets, ..) http://lcavwww.epfl.ch/~vetterli/talks/index.html

Illusions (suite)
Color persistence