So far…

- Considered processing of **greyscale** images
- I[m,n] is a function defined over a 2-D discrete space

$$I: Z^2 \rightarrow Z$$

I.e. greyscale image is a **scalar valued** function

What if the image is a **vector valued** function?

$$I: Z^2 \rightarrow Z^n$$

Ex. Colour images ...... Different n for different sensors

Colour models

Colour basics

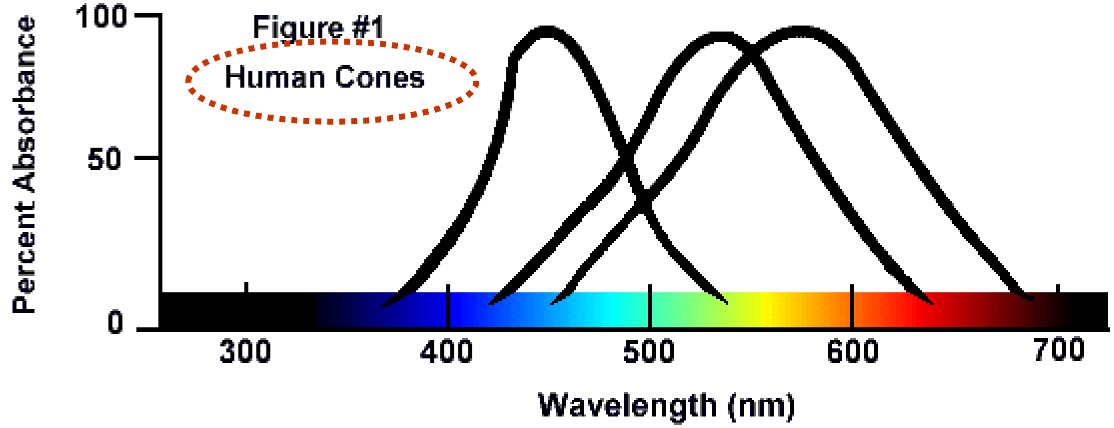
- Light ≡ electromagnetic wave (photons)
- velocity : c
- frequency : f
- wavelength : λ

$$\boldsymbol{\lambda f = c}$$

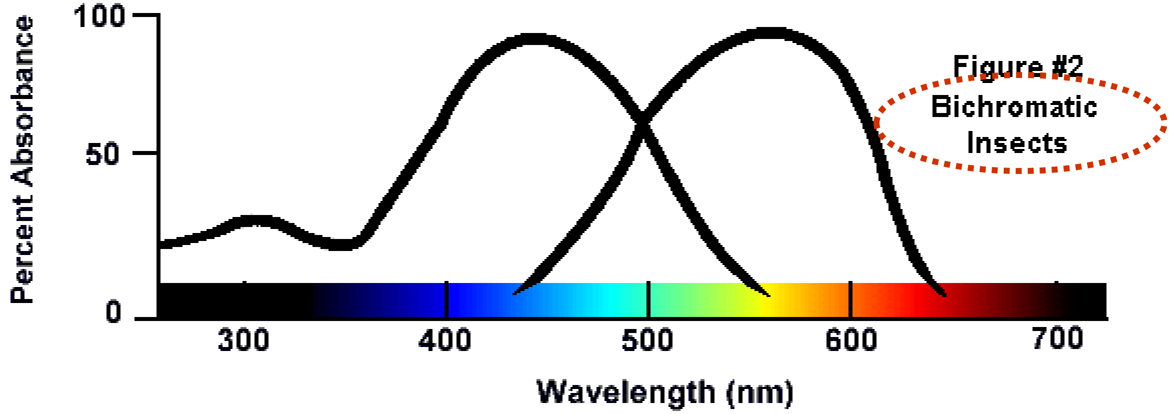
- **Colour -** response of a sensor to photons of different wavelength

- **Spectral sensitivity** of a sensor determines the range of colours it can "see"

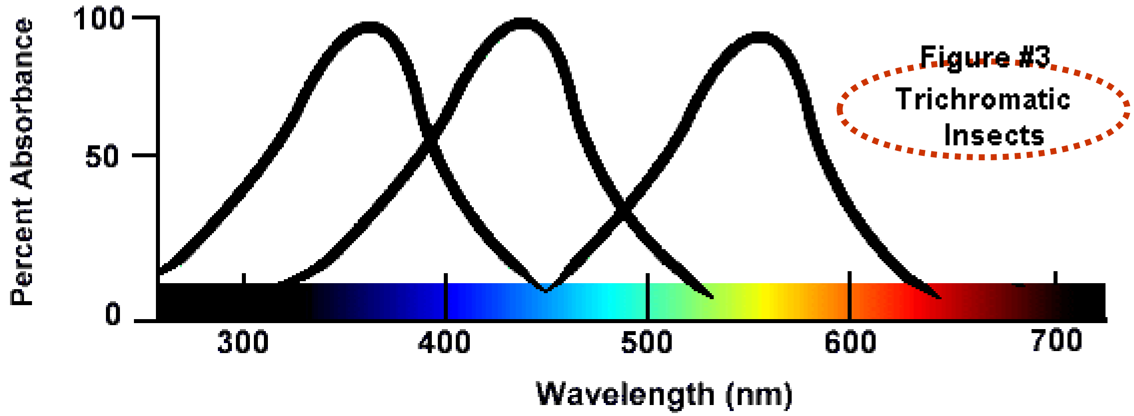
The visible world of different creatures



Humans are uv-blind

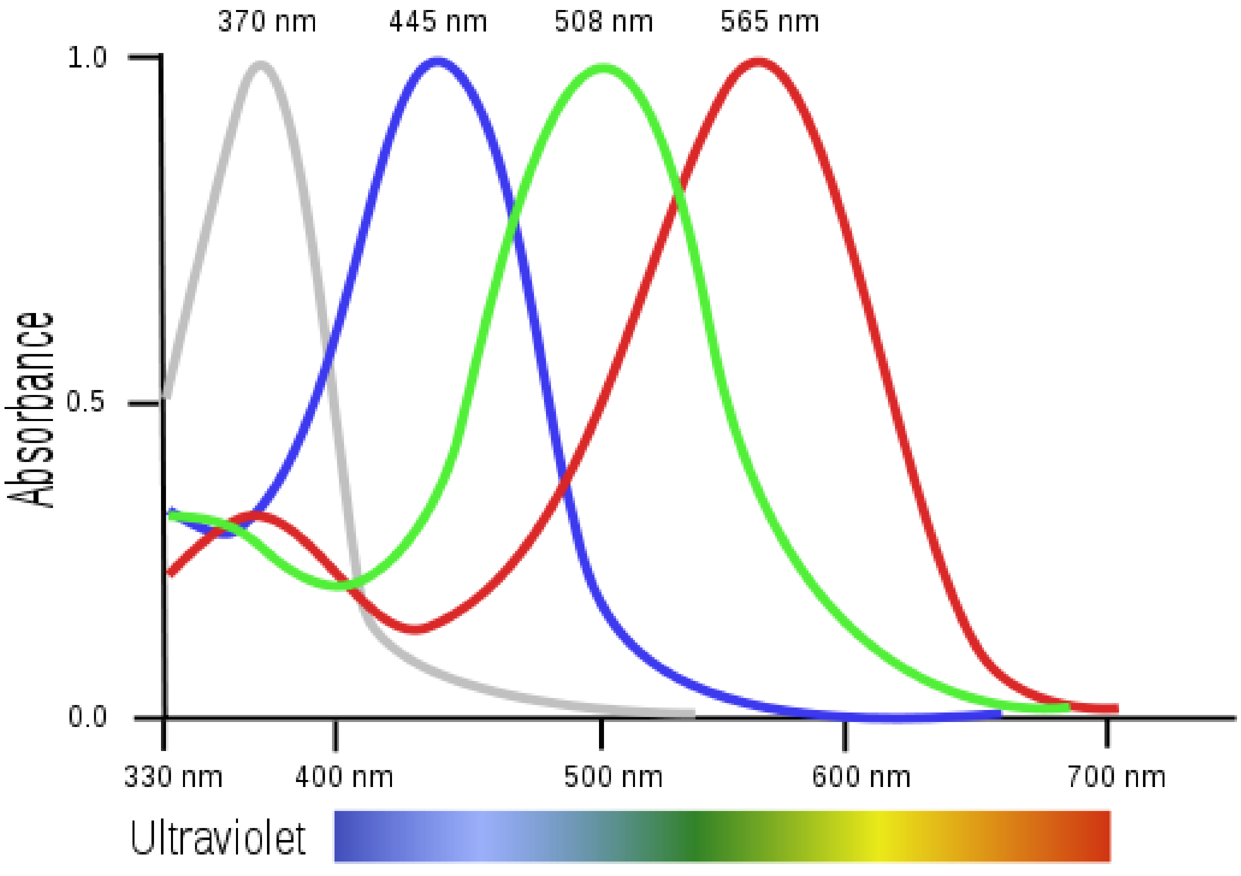


Many insects don’t see red well!
n=2



honeybees, bumblebees and butterflies have widest and *true-colour* vision
n=3

Birds - Tetrachromatic vision



Pigeons have <u>pentachromatic</u> vision

The balsam flower as seen by

humans

bees

butterfly

The world we perceive is different!

http://landsat.gsfc.nasa.gov/education/compositor/em.html

Visual world of machines

Wide-spectrum imaging is possible by using different types of *special sensors*

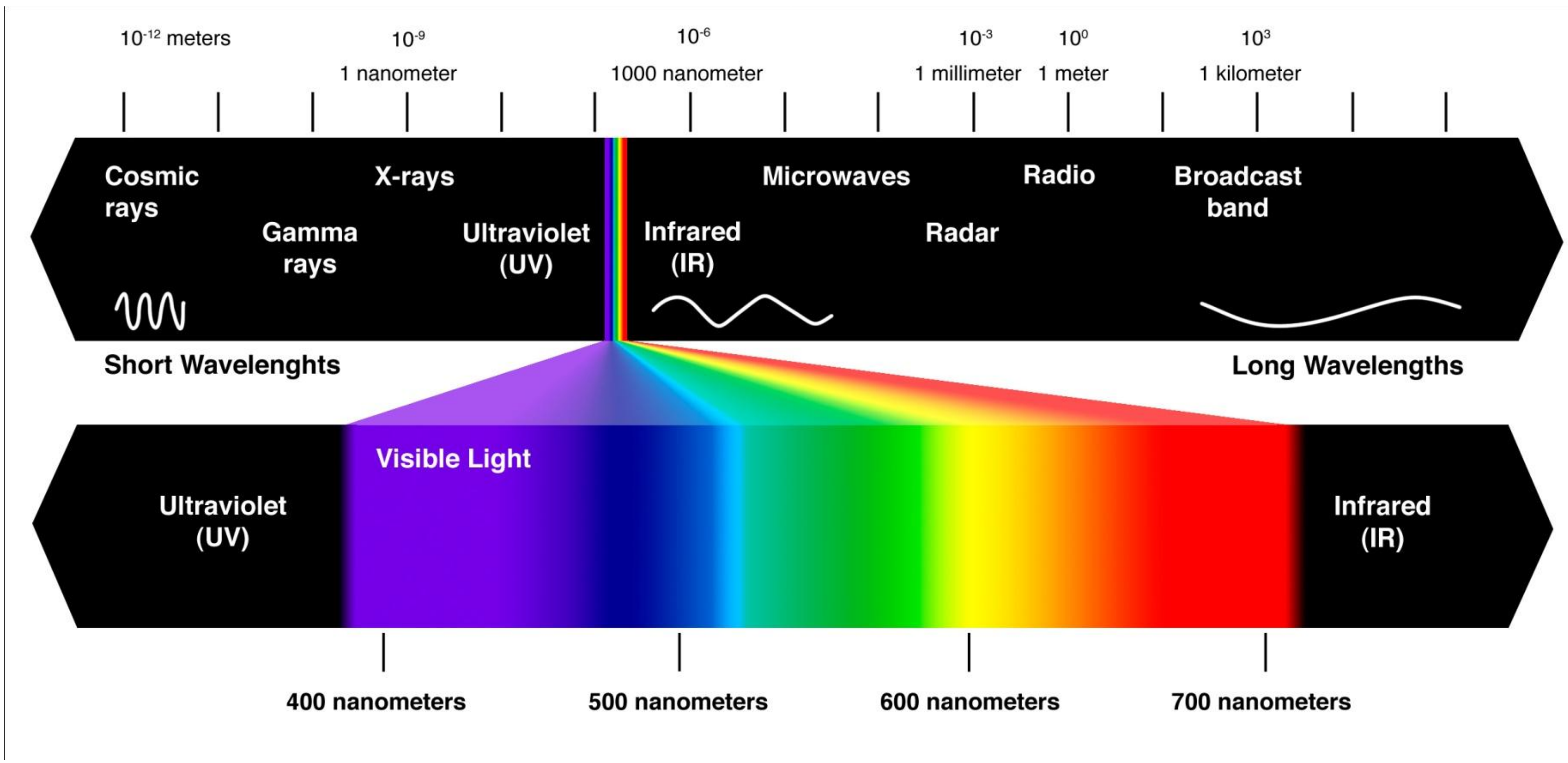
Ultraviolet range: 100 - 380 nm

Visible range: 380 - 780 nm

Mid-infrared: 1400 - 3300 nm

Near-infra red: 780-1400 nm

Far-infrared: 3 -10 μm



Representing colour

- Spectral colour space (continuous)
- By sampling this space (using sensors) we get the sampled space c → $(c_1,c_2,..c_n)$, where $c_i = c(\lambda_i)$ the spectral colour distribution

- A colour image $I(x,y) = \{ c_i(x,y)\}$
- I.e. *I* is a vector-valued function $\vec{I}(\vec{r}); \quad \vec{r} = [x, y]$
- A colour space can be represented using different colour models/spaces
  - color space is usually 3-dimensional or $i = 1,2,3$

Colour models

- RGB (red, green, blue)
  - Used in image acquisition and display
- CMYK (cyan, magenta, yellow and black)
  - Used in printing
- HSI (hue, saturation and intensity)
  - Used in image manipulation
- YIQ / NTSC
  - Used in TV broadcasting
- YC_bC_r
  - Used in digital video

RGB model

- Additive colour model

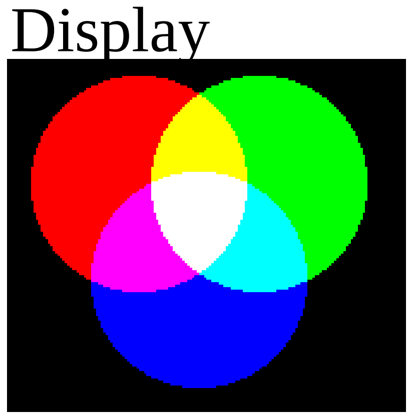
$$c(x,y) = \alpha_1 R + \alpha_2 G + \alpha_3 B$$

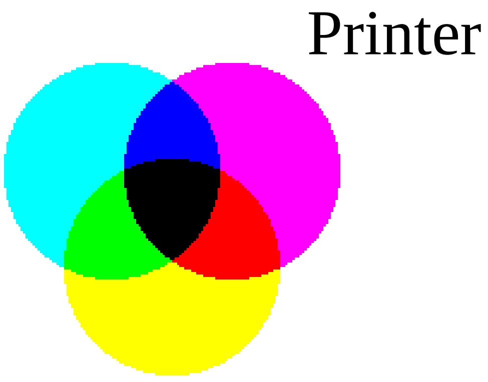
- Primary colours
  - Red (700 nm)
  - Green (546 nm)
  - Blue (435 nm)
- Non-uniform space in the perceptual sense

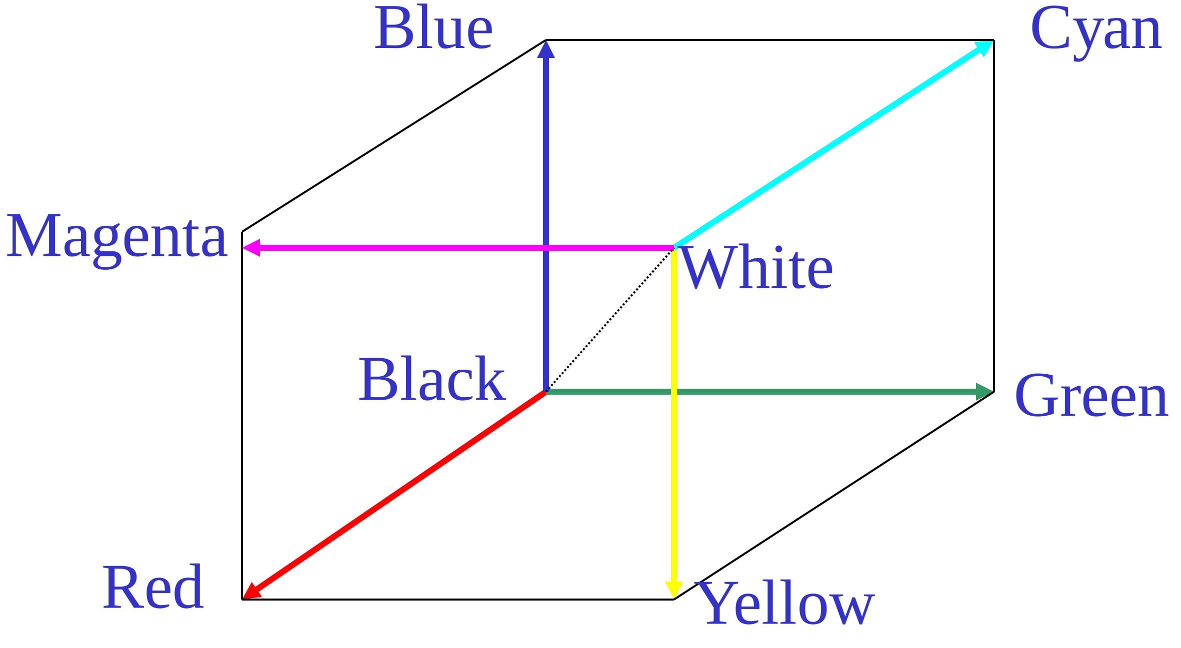
CMY model

- Subtractive colour model
- Secondary colours
  - Cyan, Magenta and Yellow
  - Plus Black (to get pure black in printing)
- Non-uniform space

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$


Display

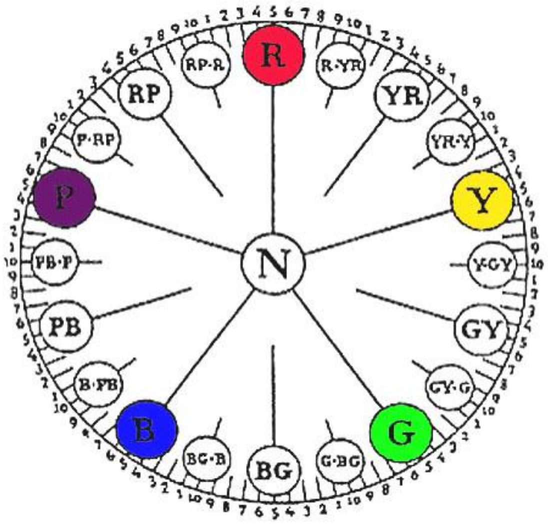

Printer


Blue
Cyan
Magenta
White
Black
Green
Red
Yellow

Colour cube

HSI Model

- Hue saturation and Intensity model
  - A close relative is **HSV** (for value)
- Perceptually uniform (meaningful) model
- Chromatic
  - **H**ue (400 – 700 nm)
    - Commonly understood as colour, ex: blue vs. green
  - **S**aturation
    - Spectral purity of colour ex: light blue vs.
  - Achromatic
    - **I**ntensity or grey **V**alue



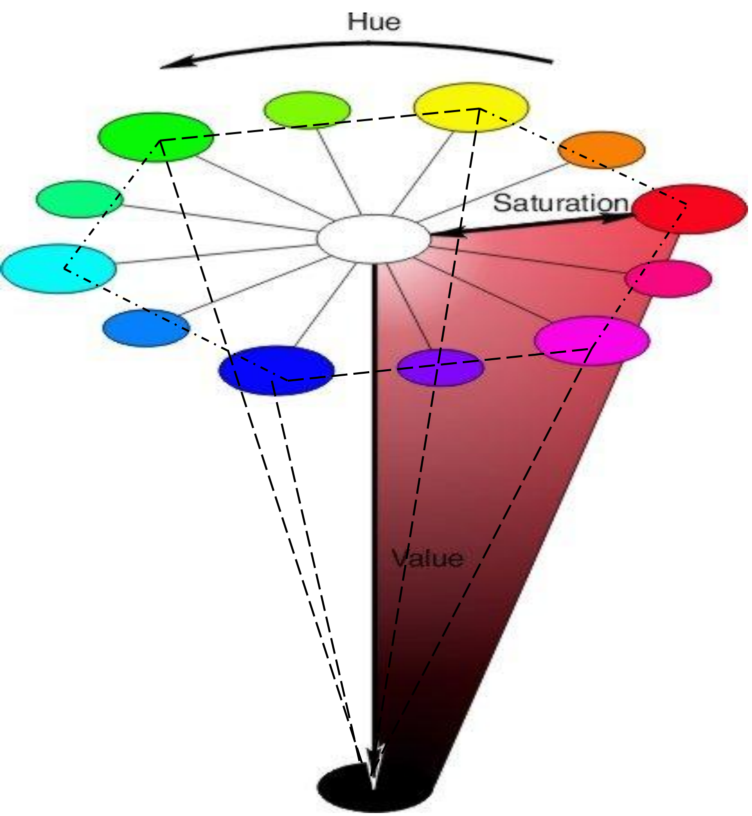
Why HSI model?

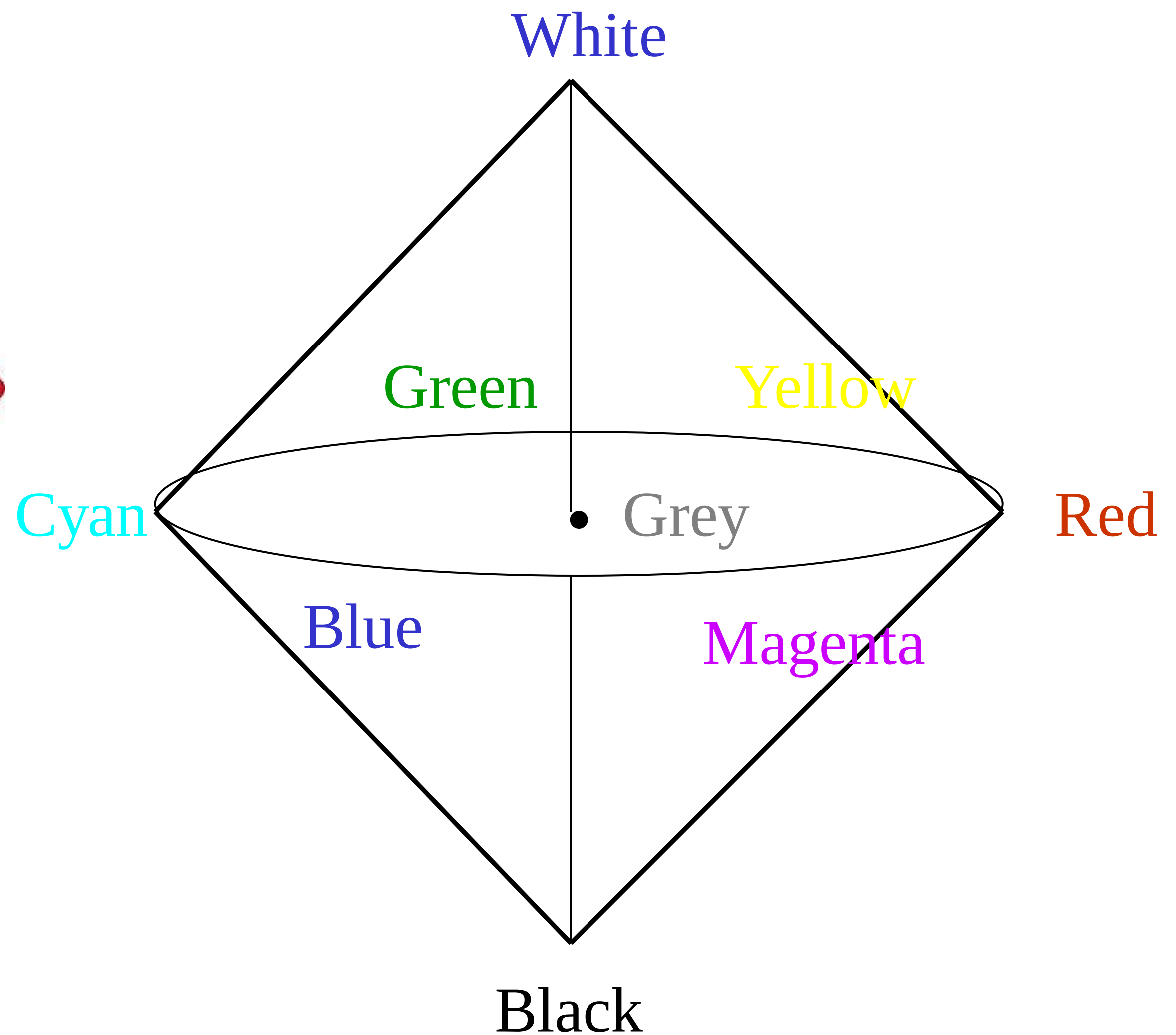
- Decouples chromatic from achromatic info.

- Decouples spectral purity from spectral info.

    - Ex: extract red /blue regions from an image
    - Ex: brightness control without colour shift

HSI model – Colour spindle





YIQ (YUV) model

- Used for commercial broadcast

- **Y**: luminance (intensity or grey value)
  - More bandwidth is allocated for this

- **I** and **Q**: chromatic components
  - I is roughly (red - cyan)
  - Q is (magenta - green)
  - Less bandwidth allocated for these

Colour synthesis

Problem: Given a colour spectrum S(λ), create it with tri-stimulus components $c_i(\lambda)$; i=1,2,3

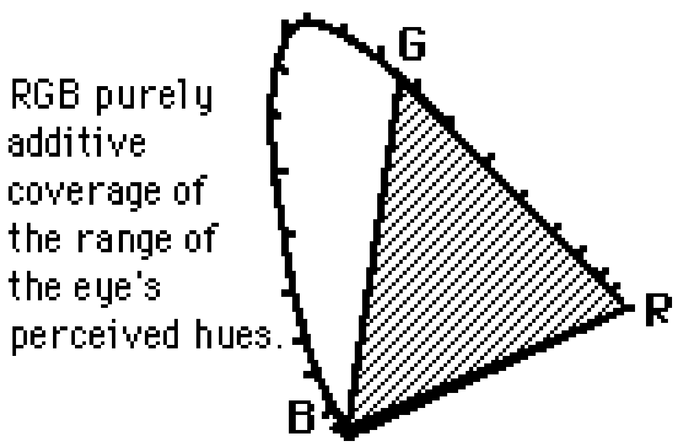
- Let us try using additive colours RGB
  - S(λ) needs to be expressed as a linear combination of $r(\lambda)$, $g(\lambda)$, $b(\lambda)$

$$S(\lambda) = R\int r(\lambda')S(\lambda')d\lambda' + G\int g(\lambda')S(\lambda')d\lambda' + B\int b(\lambda')S(\lambda')d\lambda'$$

Spectral characteristics of the *r*, *g*, *b* sensors

Key fact from a 1920 experiment

- The gamut of colours that can be created by mixing RGB (primary) is incomplete
  - We can perceive more colours
- Need an alternate colour model



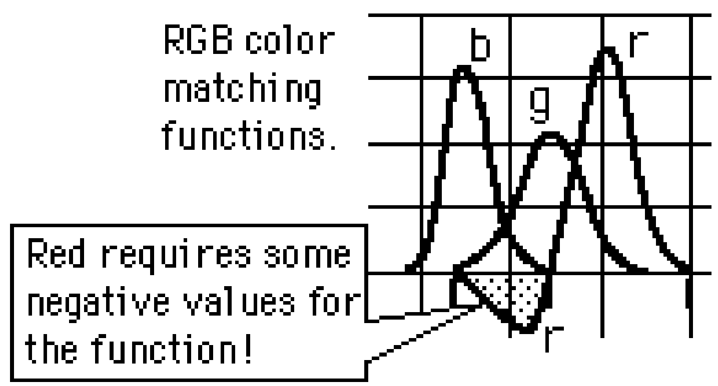
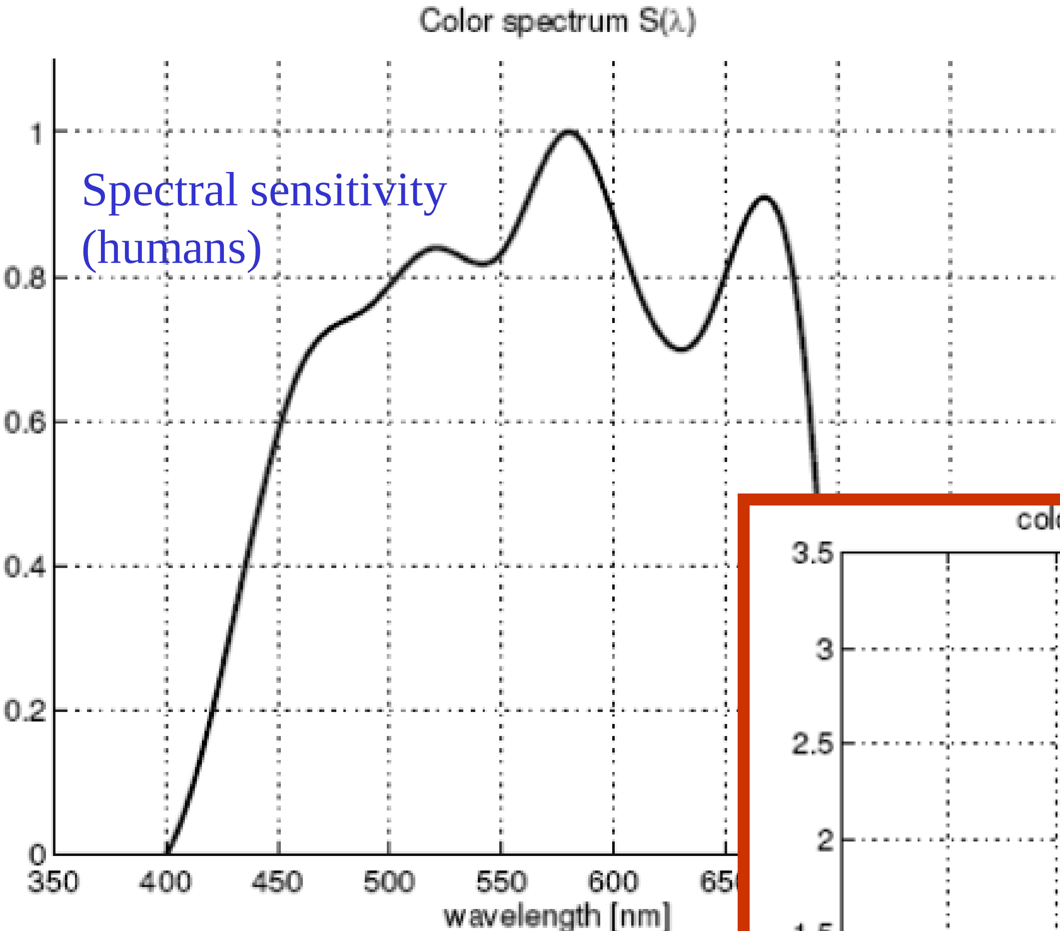
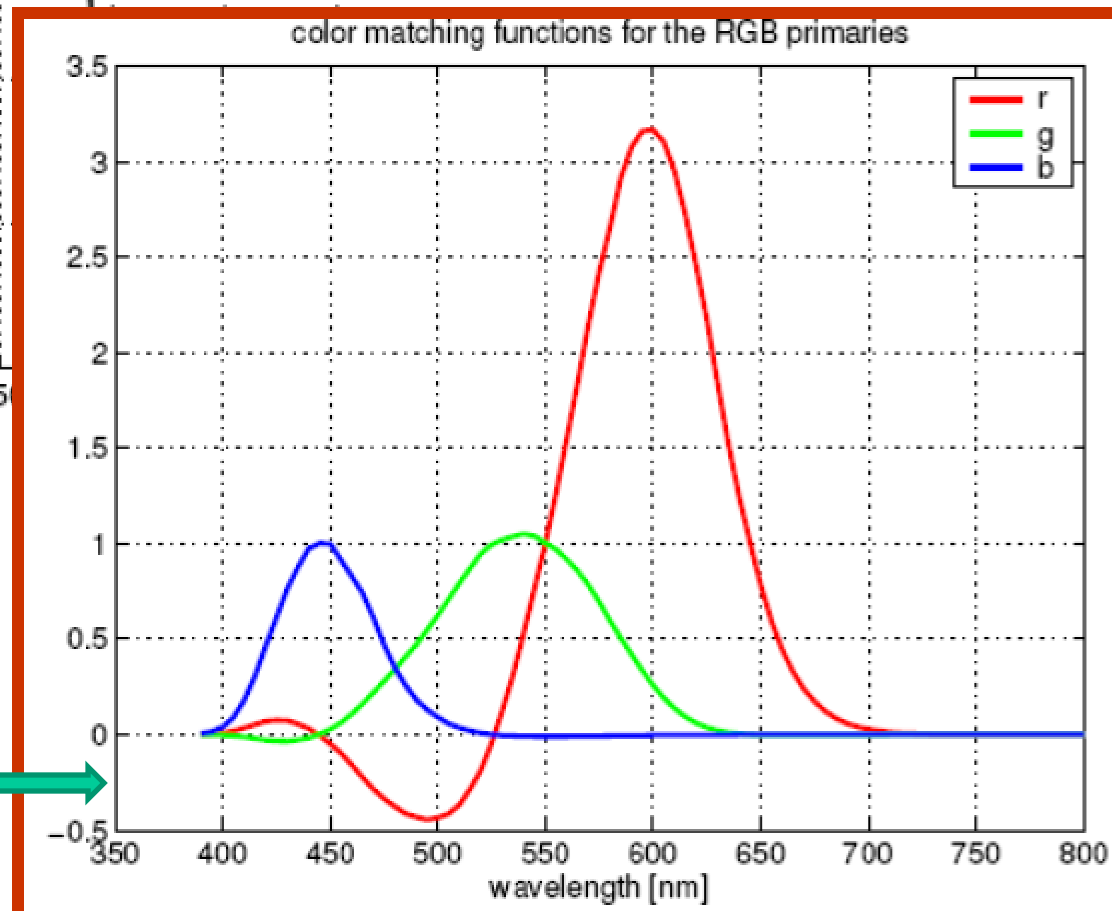


Figure from http://hyperphysics.phy-astr.gsu.edu/hbase/vision/colspa.html#c2

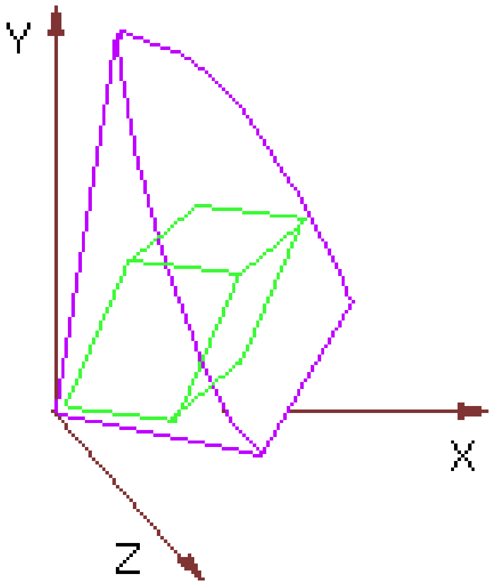


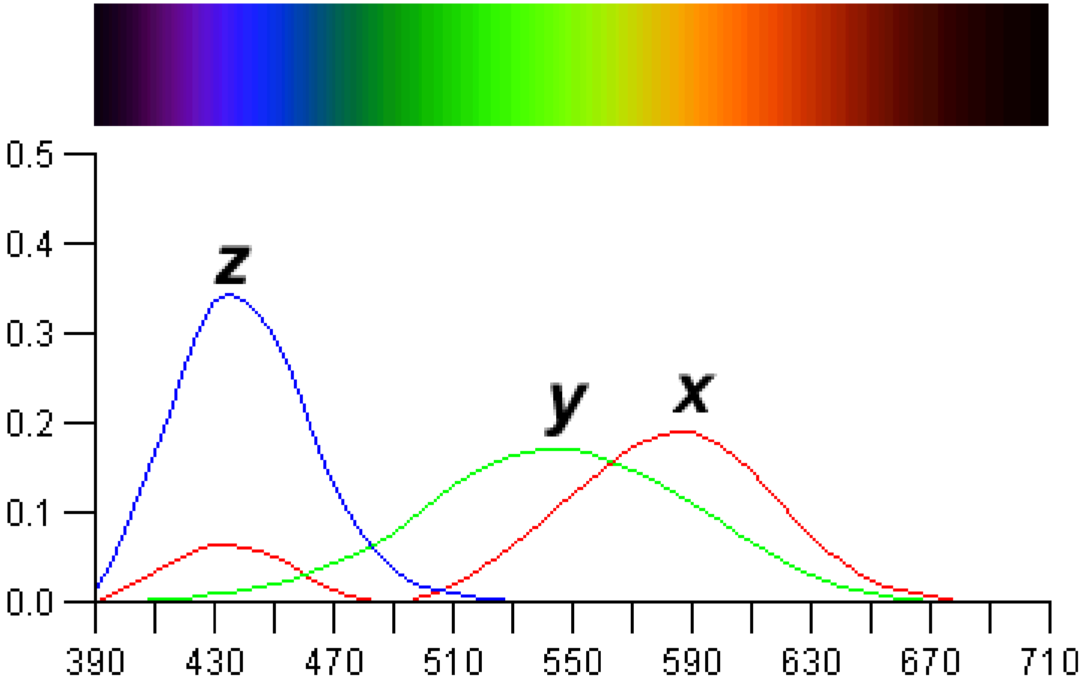


Negative values
are needed to cover all
perceivable λ!

A solution to colour mixing

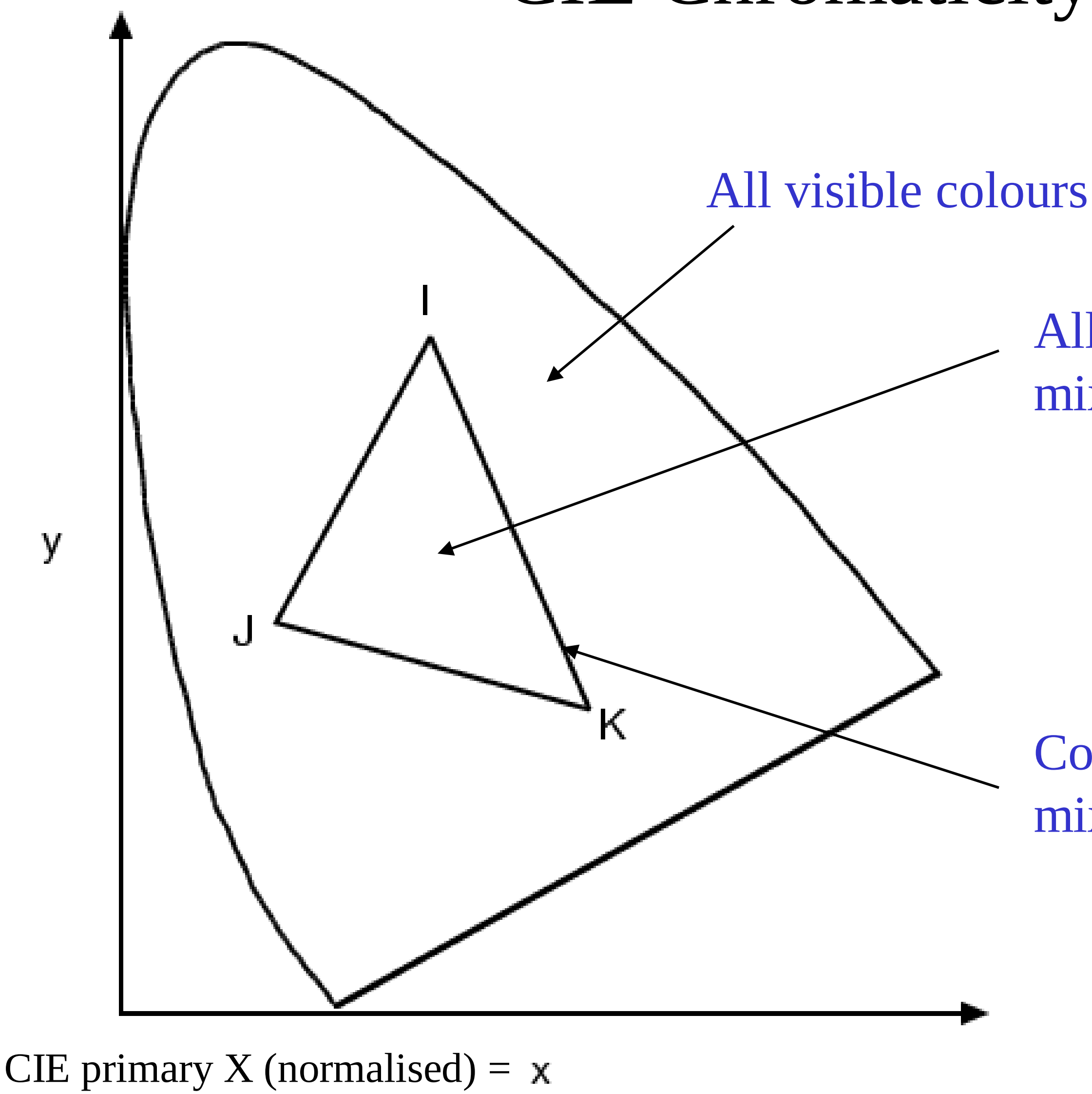
CIE colour space – using hypothetical primaries





X,Y,Z are tristimulus values

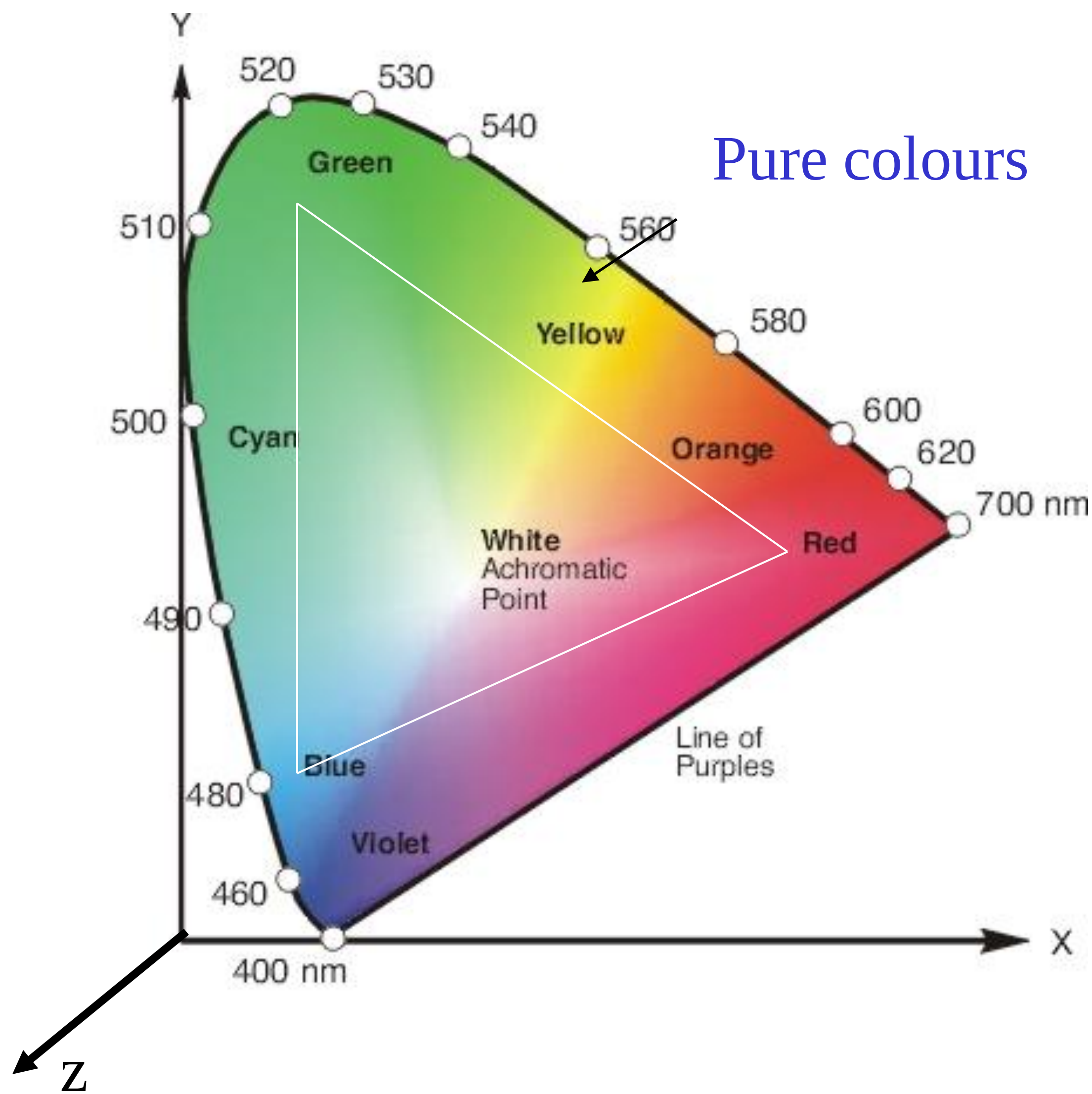
CIE Chromaticity Diagram



From CVonline

CIE chromaticity diagramn

- Chromaticity in a 2-D space $(x, y) = \frac{1}{X + Y + Z}(X, Y)$
- Gamut of human eye: the parabolic region
- Gamut of a RGB display or CMY printer: a triangular region
- Used for colour measurement



Other derivative models of CIE

Lab colour space

L:Lightness ***a***:red-green ***b***:yellow-blue

Colour opponent dimensions

Advantage: colour changes perceptually linearly as one moves across the chart

- Useful in image manipulation to achieve uniform colour shift

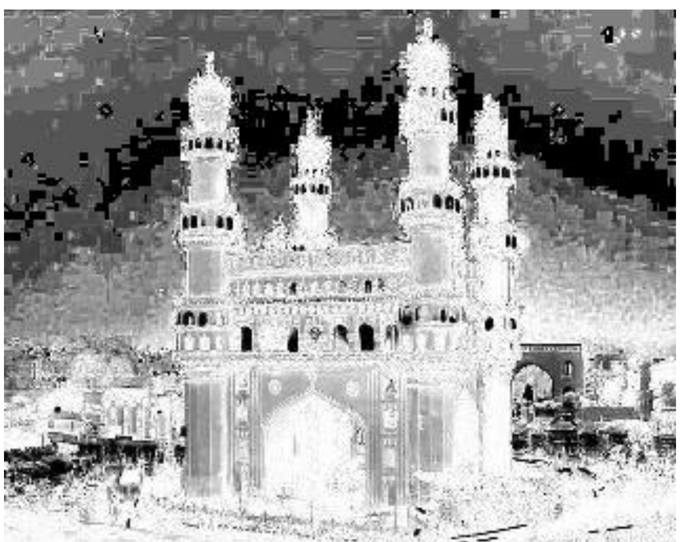
Colour image

Blue plane

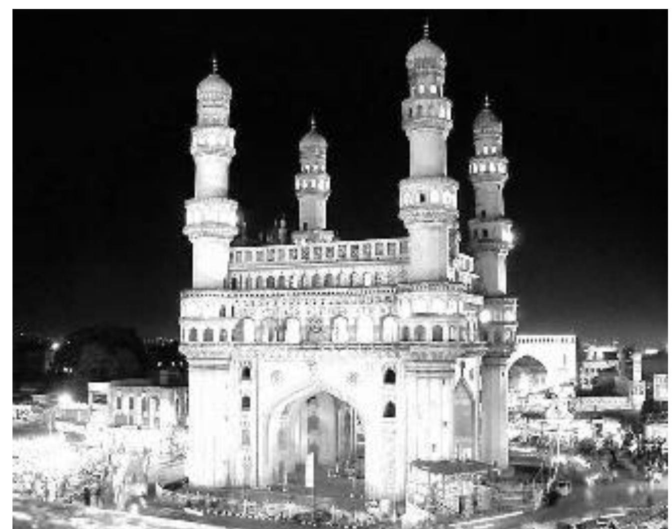
Green plane

Red plane

H plane



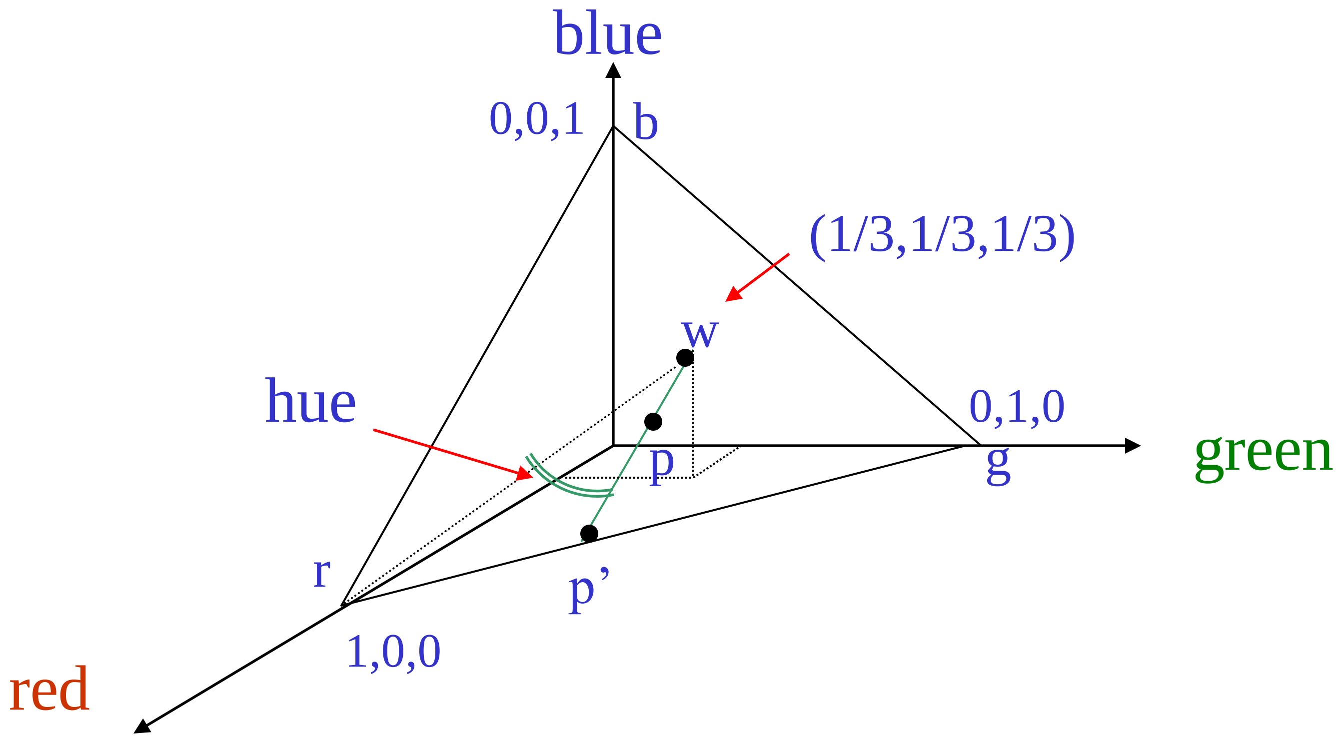
S plane



I plane

Conversion between models

RGB-HSI conversion



The angle between wp' and wr (in terms of r, b, c) = hue

The ratio of vector lengths wp to wp' = saturation

RGB to HSI conversion

$$I = \frac{R+G+B}{3}$$

$$S = 1 - \frac{\min(R,G,B)}{I}$$

$$H = \cos^{-1}\left\{\frac{0.5[(R-G)+(R-B)]}{\sqrt{[(R-G)^2 + (R-B)(G-B)]}}\right\}$$

$\in (0,360]$

- All variables are in [0,1]; H = H/360 for normalisation
- H = 360 – H if B/I > G/I

HSI to RGB conversion

- Similar results are derivable using the chromaticity triangle

$$r = \frac{1}{3}[1 + \frac{S\cos H}{\cos(60 - H)}]$$

$$b = \frac{1}{3}(1 - S)$$

$$g = 1 - (r + b)$$

RGB-YIQ conversion

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- For greyscale image, R=G=B ➔ I = Q = 0
- Inverse conversion is possible using inverted matrix

YC_bC_r model

- Used in digital video
- **Y**: luminance
- **C_b,C_r** : colour (difference) information

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Colour images - storage

$$\mathbf{c}[m,n] = \{c_i[m,n];\ i = 1,2,3\}$$

c_i is a greyscale image

Storage requirements is tripled!

- Can be reduced using **colour palettes**
  - commonly done in monitors

Colour Palettes- indexed colour

- True colour image needs 3 bytes/pixel to store the image
- With 3 bytes the possible colours is $(2^8)^3 = 16{,}777{,}216 \sim 17 \text{ million}$

For MxN = MN pixel image we need **3 MN bytes**

Palette idea:

- assign 8 bits/pixel and design a limited size table of colours (palette) ex. 2^8 x 3 palette → 256 colours
  - ➢ 768 bytes for palette storage
- palette is chosen from 17 million colours using some algorithm
- 1byte is for a pointer to an entry in the palette
- To store the image we now need **(MN +768) bytes**
  - (1byte/pixel x MN pixels +768 bytes)

One can assign less than 8 bits/pixel → greater savings in storage
At the cost of colour accuracy!

Colour Image Processing

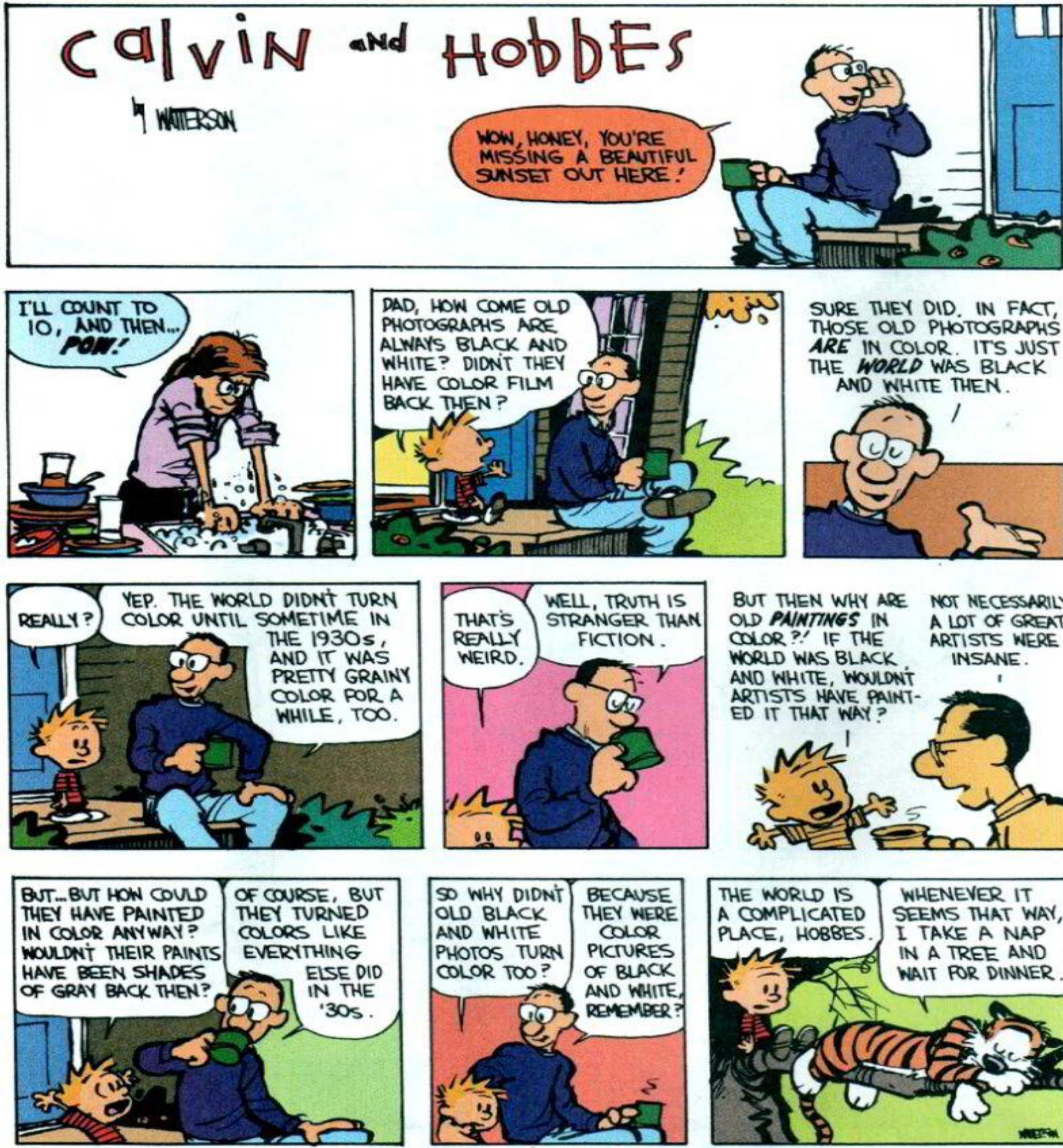
Pseudo colour IP

Full colour IP

Operate on

scalar space

vector space


Calvin and Hobbes
by WATTERSON
WOW, HONEY, YOU'RE MISSING A BEAUTIFUL SUNSET OUT HERE!
I'LL COUNT TO 10, AND THEN... POW!
DAD, HOW COME OLD PHOTOGRAPHS ARE ALWAYS BLACK AND WHITE? DIDN'T THEY HAVE COLOR FILM BACK THEN?
SURE THEY DID. IN FACT, THOSE OLD PHOTOGRAPHS ARE IN COLOR. IT'S JUST THE WORLD WAS BLACK AND WHITE THEN.
REALLY?
YEP. THE WORLD DIDN'T TURN COLOR UNTIL SOMETIME IN THE 1930s, AND IT WAS PRETTY GRAINY COLOR FOR A WHILE, TOO.
THAT'S REALLY WEIRD.
WELL, TRUTH IS STRANGER THAN FICTION.
BUT THEN WHY ARE OLD PAINTINGS IN COLOR?! IF THE WORLD WAS BLACK AND WHITE, WOULDN'T ARTISTS HAVE PAINTED IT THAT WAY?
NOT NECESSARILY. A LOT OF GREAT ARTISTS WERE INSANE.
BUT... BUT HOW COULD THEY HAVE PAINTED IN COLOR ANYWAY? WOULDN'T THEIR PAINTS HAVE BEEN SHADES OF GRAY BACK THEN?
OF COURSE, BUT THEY TURNED COLORS LIKE EVERYTHING ELSE DID IN THE '30s.
SO WHY DIDN'T OLD BLACK AND WHITE PHOTOS TURN COLOR TOO?
BECAUSE THEY WERE COLOR PICTURES OF BLACK AND WHITE, REMEMBER?
THE WORLD IS A COMPLICATED PLACE, HOBBES.
WHENEVER IT SEEMS THAT WAY, I TAKE A NAP IN A TREE AND WAIT FOR DINNER.

Processing colour images

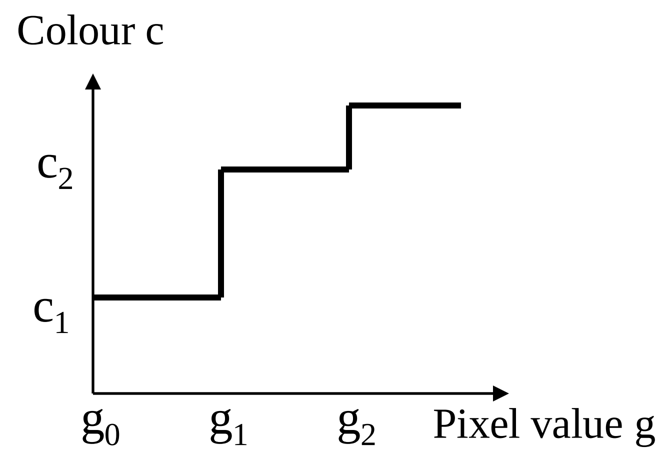
- Pseudo colour IP (greyscale image input)
  - Artificially converting grey scale to colour

- Full colour IP (colour image input)
  - n-bit colour; n = 24, 26
  - n/3 bits per component
  - pixel value is in $[0,2^{n/3}-1]$ or $(0,360^{o}]$ or $[0,1]$

Pseudo colour IP

Greyscale image → colour image

- to highlight some feature of interest in the image

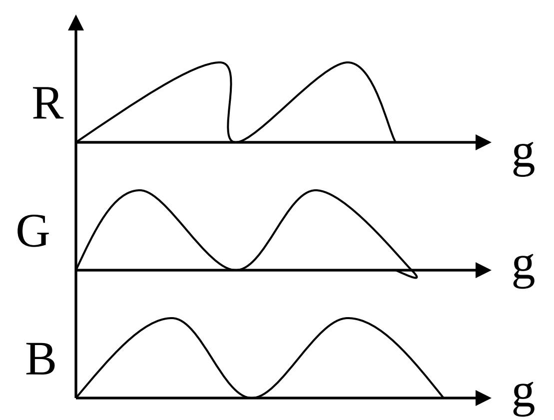


Techniques:

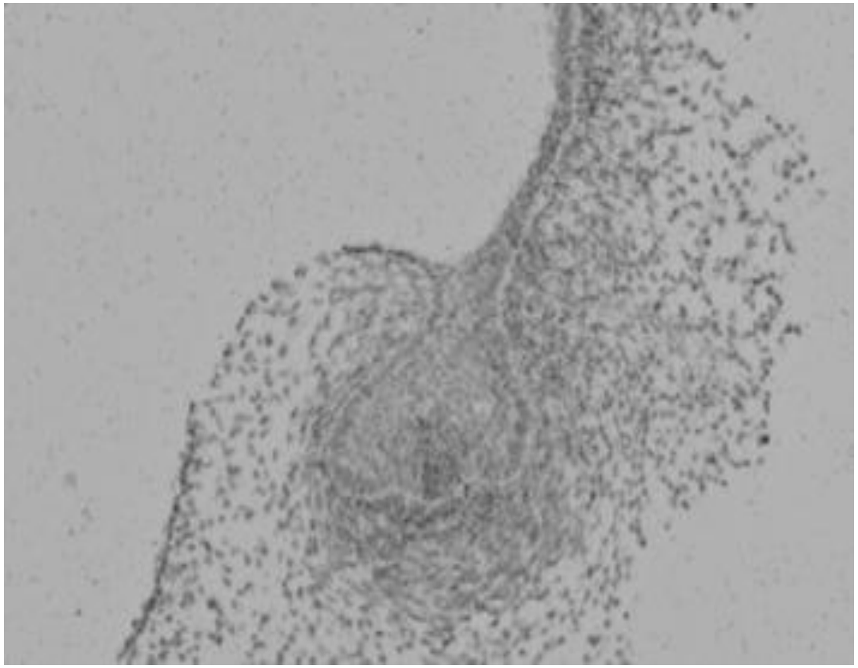
- Intensity slicing

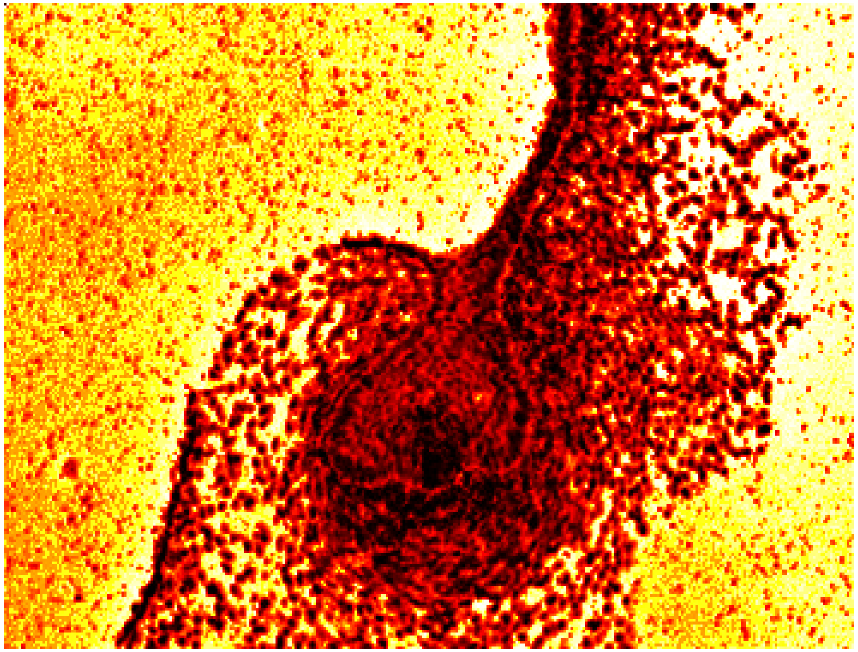
$$f(g) = c_1 \,; \quad g_0 < g < g_1$$
$$\quad = c_2 \,; \quad g_1 < g < g_2 \ldots$$

- Grey-colour transformation
  - User-defined mapping functions



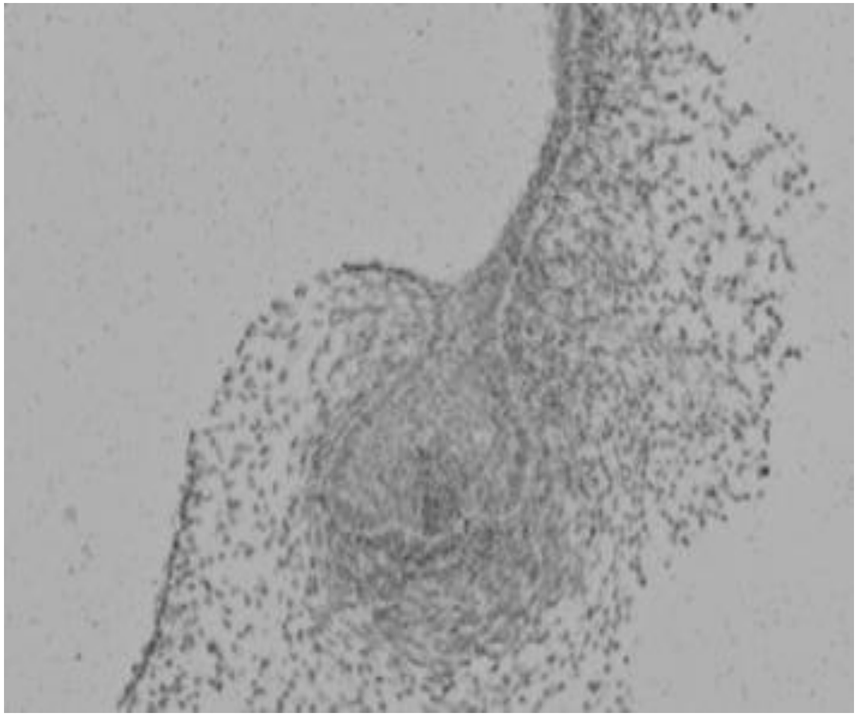
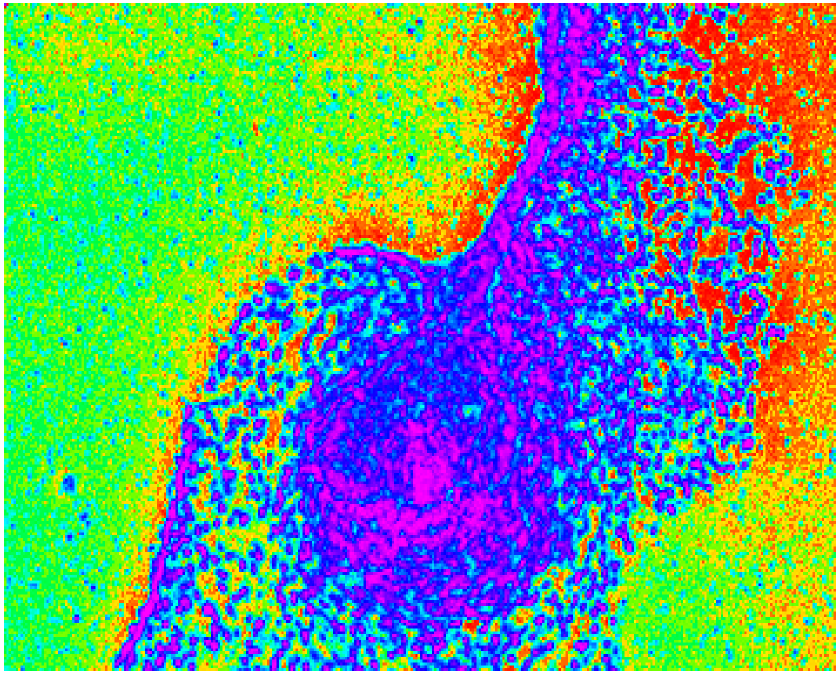
Pseudo colour IP- examples





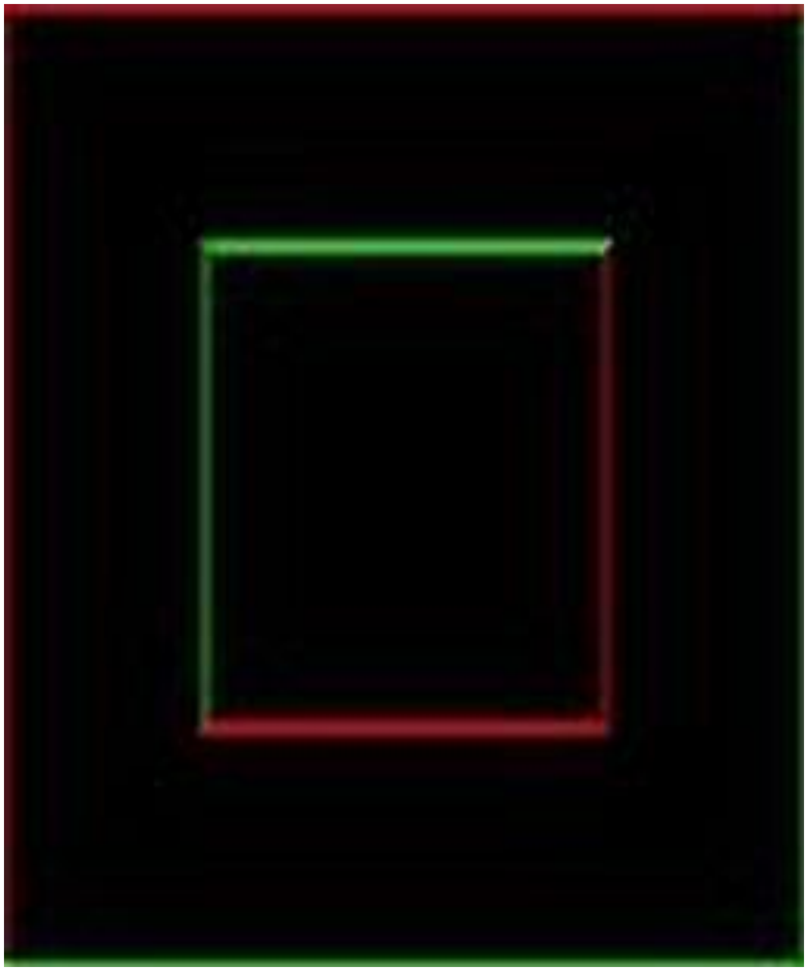
Grey scale - 2-colour mapping

Pseudo colour IP- examples

Grey scale - multi-colour mapping

Pseudo colour IP- edge map



green: white to black transition

Red: black to white transition in raster scan

Full colour IP

colour image → colour image

vector valued fn → vector valued fn

I:$\mathbf{Z}^2$ → $\mathbf{Z}^3$ $$c[m,n] = \begin{bmatrix} c_1[m,n] \\ c_2[m,n] \\ c_3[m,n] \end{bmatrix}$$ 3 components / colour planes

Strategy 1: Operating in the scalar space.

All operations introduced for greyscale images can be applied to each of the greyscale image

Full colour processing- strategy 1

Options

1. Do identical operation in all 3 planes and combine results

   - Can lead to colour shifts

2. Process the ‘appropriate’ plane and combine

   - ➢ may or may not necessarily achieve the right effect
   - ➢ how to find the ‘appropriate’ plane?

     - depends on the task

Full colour processing – **Strategy 1**

Examples:
- contrast stretching
- denoising
- sharpening

Example 1: contrast stretching

Original

Histogram equalisation done on R,G and B planes

Example 2:Enhancement

Method

- Model the input image as

$$I = I_o S_M + S_A$$

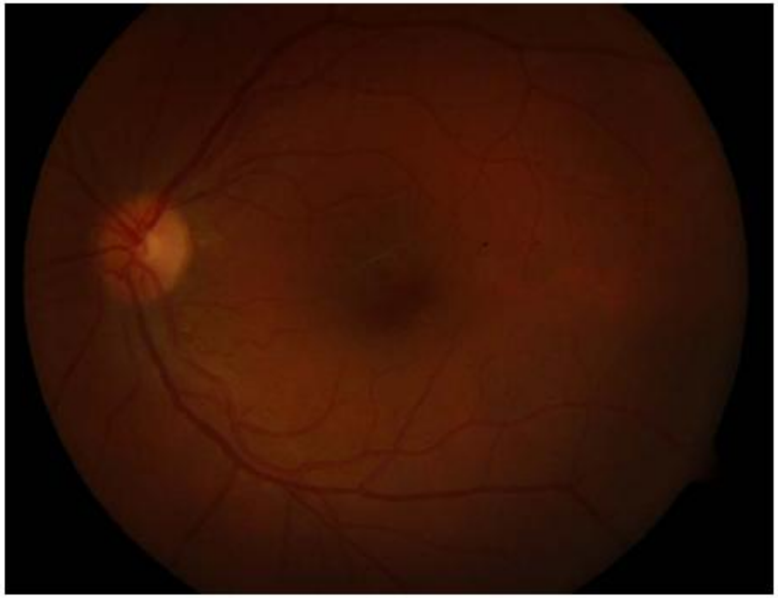
contrast luminosity

- Estimate the background

- Estimate S_M and S_A from the background

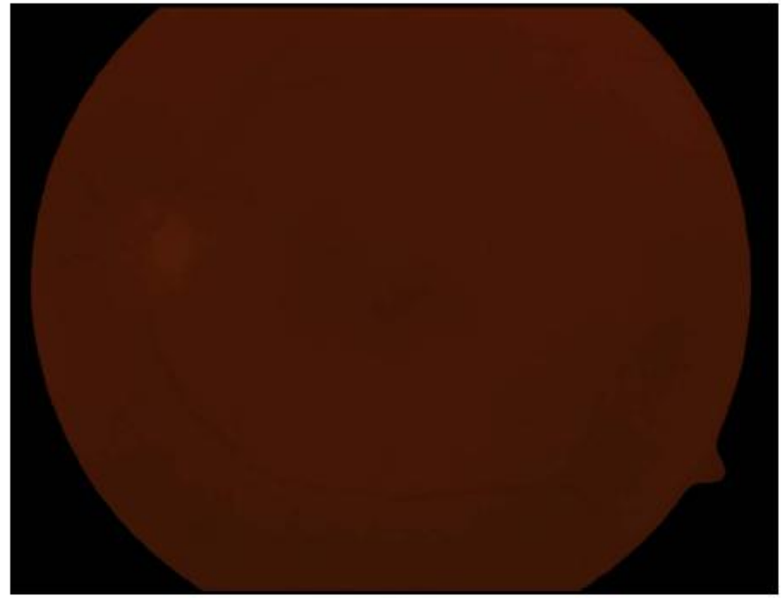
- Find $I_o = (I - S_A)/S_M$

Input image

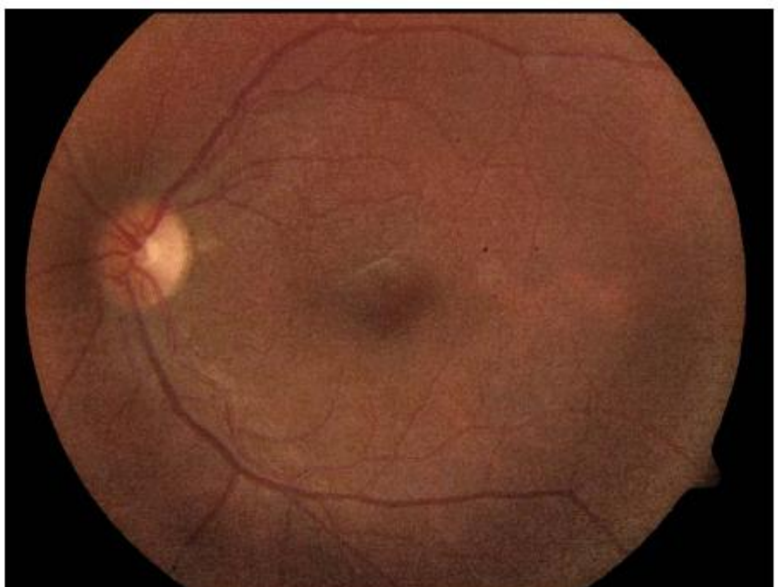


(a)

Result of same operation applied to R,G,B



(b)



(c)

Result of enhancing S and I channels independently

(d)

Result of operating on G and then reflecting it on G and B

Handling colour

Method1:

- Apply the method on *r*,*g* and *b* and normalise

Method 2:

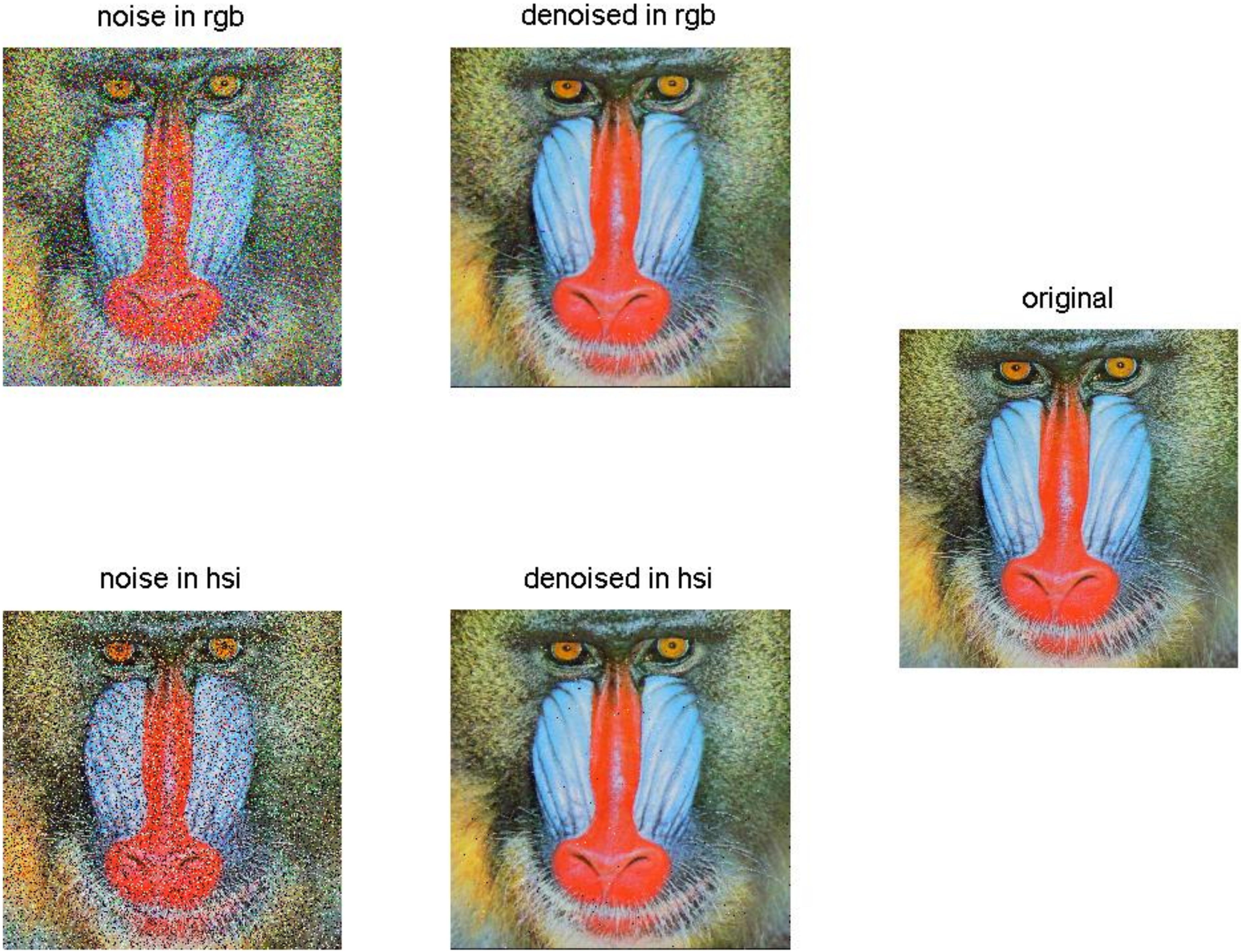
- Apply the method to the *g* plane and find g_{corr}
- Next, find the desired colour image as

$$\hat{r} = \frac{g_{corr}}{v} * r, \quad \hat{g} = \frac{g_{corr}}{v} * g, \quad \hat{b} = \frac{g_{corr}}{v} * b, \qquad v = max[r, g, b]$$

Example 3: denoising

$$c[m,n] = \begin{bmatrix} c_1[m,n] \\ c_2[m,n] \\ c_3[m,n] \end{bmatrix}$$ is a point in a 3-D space

Example: median filtering for denoising impulse noise


noise in rgb
denoised in rgb
original
noise in hsi
denoised in hsi

Sharpening- example

Sharpening of an image can be achieved by using a Laplacian as follows:

Given $\{c_i(\mathrm{m,n})\}$ its Laplacian can be found as

$$\nabla^2[c\ (m,n)] = [\nabla^2 c_1(m,n) \quad \nabla^2 c_2(m,n) \quad \nabla^2 c_3(m,n)]^T$$

The required sharpened image

$$y(m,n) = c(m,n) + \alpha\nabla^2[c\ (m,n)]$$

Strategy 2: Full colour processing

– operating on the vector space

Ex 1. denoising

- Denoising by median filtering in vector space
  - to remove impulse noise

General method:

- Consider all 3 colour components and rank order the vectors
  - Ranking is done on the distance between the centre and all neighbouring pixels
- Centre pixel is then replaced with the colour corresponding to the min distance value

Magnified view of results

rgb

original

hsi

vmf (on multicolour space)

Ex. 2: finding intensity transitions

Example 2: Transitions in intensity can be found using gradients

Gradient of a scalar valued *f* is a vector

$$\nabla f = \vec{g} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Of interest are its

- strength (magnitude) $\left|\vec{g}\right|^2 = (f_x{}^2 + f_y{}^2)$

- direction $\angle g = \tan^{-1}\left(\dfrac{f_y}{f_x}\right)$

Gradient – colour images

What is the gradient of a vector-valued function $c(x,y)$?

Gradient – colour images

- $c(x,y) : \mathrm{R}^2 \rightarrow \mathrm{R}^3$ ex. RGB image

Simple solution: take the vector sum of the gradients in each channel

$$\overrightarrow{g_c} = \overrightarrow{g_R} + \overrightarrow{g_G} + \overrightarrow{g_B}$$

- Computationally cheap
- May not reflect true scenario
  - Ex. B channel has no variation; R and G have equal variation along x direction but in opposite direction (left to right vs right to left) will lead to

$$\left|\overrightarrow{g_c}\right| = g_{Rx} - g_{Gx} = 0$$

Gradient - colour image

$$\vec{c}(x,y) = \begin{bmatrix} c_1(x,y) \\ c_2(x,y) \\ c_3(x,y) \end{bmatrix}$$

Better option: Use the gradient generalisation

$c(x,y) : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$\therefore$ gradient of c is a tensor i.e. its Jacobian matrix

2x3

$$J = \begin{bmatrix} \frac{\partial c_1}{\partial x} & \frac{\partial c_2}{\partial x} & \frac{\partial c_3}{\partial x} \\ \frac{\partial c_1}{\partial y} & \frac{\partial c_2}{\partial y} & \frac{\partial c_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \vec{c}}{\partial x} \\ \frac{\partial \vec{c}}{\partial y} \end{bmatrix} = \begin{bmatrix} \vec{c}_x \\ \vec{c}_y \end{bmatrix}$$

Note: when f is scalar valued, this becomes the gradient vector

Gradient ..contd.

Of interest:

At every point (x,y) we would like to know

1. The direction θ in which maximum change is occurring in c
   - Gradient direction

2. The magnitude of this maximum change
   - Gradient magnitude

Define tensor components

$$g_{xx} = <\vec{c}_x, \vec{c}_x> = \vec{c}_x^{\,T} \vec{c}_x = \left|\frac{\partial c_1}{\partial x}\right|^2 + \left|\frac{\partial c_2}{\partial x}\right|^2 + \left|\frac{\partial c_3}{\partial x}\right|^2$$

$$g_{yy} = <\vec{c}_y, \vec{c}_y> = \vec{c}_y^{\,T} \vec{c}_y = \left|\frac{\partial c_1}{\partial y}\right|^2 + \left|\frac{\partial c_2}{\partial y}\right|^2 + \left|\frac{\partial c_3}{\partial y}\right|^2$$

$$g_{xy} = <\vec{c}_x, \vec{c}_y> = \vec{c}_x^{\,T} \vec{c}_y = \frac{\partial c_1}{\partial x}\frac{\partial c_1}{\partial y} + \frac{\partial c_2}{\partial x}\frac{\partial c_2}{\partial y} + \frac{\partial c_3}{\partial x}\frac{\partial c_3}{\partial y}$$

Gradient of a colour image

Magnitude of the gradient at (*x*,*y*)

$$A_\theta(x,y) = \sqrt{\frac{1}{2}[(g_{xx}+g_{yy})+(g_{xx}-g_{yy})\cos 2\theta + 2g_{xy}\sin 2\theta]}$$

Direction of the gradient at (*x*,*y*)

$$\theta(x,y) = \frac{1}{2}\tan^{-1}[\frac{2g_{xy}}{g_{xx}-g_{yy}}]$$

Note: In practice,

- each of these derivatives are computed with a mask
- θ and F_θ are images of same size as *c*(x,y)

What do we gain with this formulation?

Di Zenzo, *CVGIP* 1986

Example 1

An RGB image with

- No variation in y direction in RGB planes; $\Rightarrow \vec{c}_y = 0$
- B is constant : $B_x = B_y = 0$
- R and G have equal gradients in *x* direction but of opposite sign: $R_x = -G_x$

$$\therefore g_{xy} = g_{yy} = 0; \quad g_{xx} = 2|R_x|^2$$

$$\theta = \tan^{-1}(0) \Rightarrow \theta = 0 \quad or \quad \frac{\pi}{2}$$

$$A_0 = \sqrt{g_{xx}} = \sqrt{2}|R_x|$$

Check: What will you get if you compute using the vector sum of gradients?

Example 2

RGB image with equal variation in both directions in all planes $\Rightarrow g_{xx} = g_{yy} = g_{xy} = a$

$$\therefore \theta = \frac{1}{2}\tan^{-1}\left(\frac{2a}{a-a}\right) = \pm\frac{\pi}{4}$$

$$A_{\frac{\pi}{4}} = \sqrt{2a}$$

Check: What will you get if you compute using the vector sum of gradients?

Summary

- Processing colour images by treating them as vector valued functions can be advantageous
  - But computationally expensive

- Degree of effectiveness depends on the task
  - In edge detection, 90% detection is possible by processing only the *intensity* plane
  - Is the additional 10% is critical enough to justify more computations?
    - Depends on appln. domain