Computer Vision

Color image processing

10 April 2018

j.van.de.loosdrecht@nhl.nl, jaap@vdlmv.nl

Color image processing

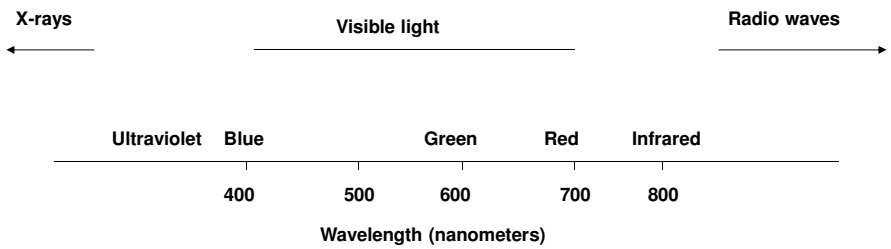
Overview:

- **Introduction**
- **Color spaces**
- **Extract and merge channels**
- **Image enhancement**
- **Segmentation**
- **False color images (*)**
- **NormalizeRGB**
- **NormalizeHue**
- **Exercise**

Introduction

- **Electromagnetic spectrum**
- **Human vision**
- **Color systems**
- **Sensing illuminated objects**

Electromagnetic spectrum



Human vision

Human eye:

- **Bars for gray values**
- **Cones for color, one type for each primary color**
  - **Blue**
  - **Green (most sensitive)**
  - **Red**

Color systems

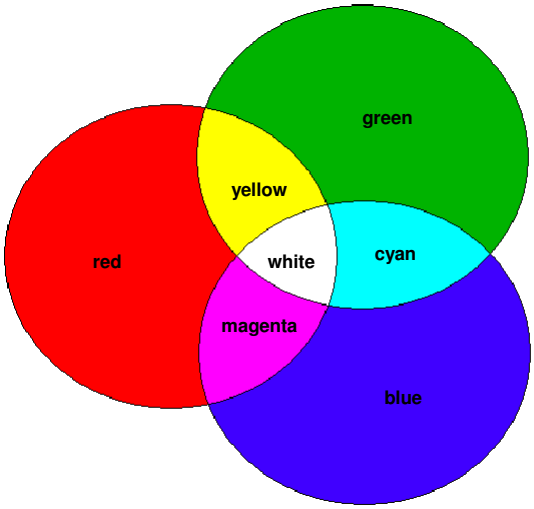
- **Additive:**
  the primary colors of light are individual red, green and blue light sources that are projected onto a common region of space to reproduce a colored light.

  Example: RGB monitor.

- **Subtractive:**
  the primary pigments are magenta, cyan and yellow. A primary pigment subtracts or absorbs a primary color of light and reflects or transmits the other two primaries of light.

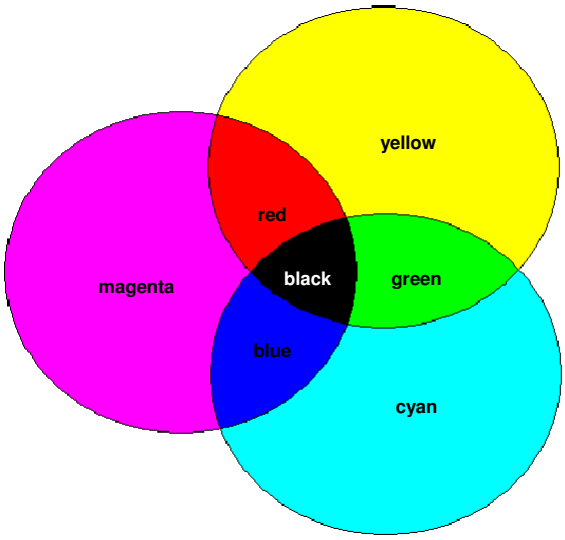
  Example: white light on yellow object: blue is absorbed <subtractive>, red and green (= yellow) are reflected <additive>.

Mixtures of light (additive primaries)



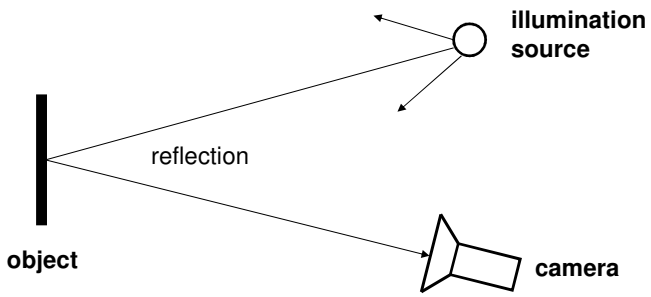
27-aug-18 Color image processing 7

Mixtures of pigments (subtractive primaries)



27-aug-18 Color image processing 8

Sensing illuminated objects



Sensing illuminated objects

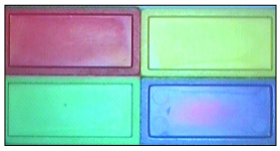
The sensation, or perception, of an object's color depends upon three factors:

- **The spectrum of energy in the various wavelengths illuminating the object surface**
- **The spectral reflectance of the object surface, which determines how the surface changes the received spectrum into the radiated spectrum**
- **The spectral sensitivity of the sensor irradiated by the light from the object's surface**

Sensing illuminated objects

White light contains all colors in equal energy

Domino stones illuminated by equal amounts of red, green and blue

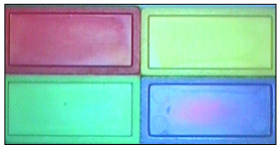


Note: both domino stones and light source do not have pure colors

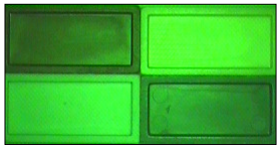
27-aug-18 Color image processing 11

Sensing illuminated objects

white light



green light

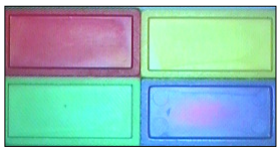


- **green -> green**
- **yellow -> light green (yellow = green + red)**
- **blue -> dark**
- **red -> dark**

27-aug-18 Color image processing 12

Sensing illuminated objects

white light **red light**

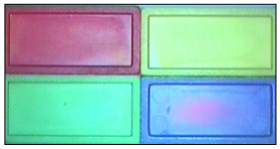


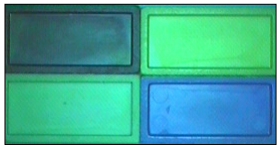
- **red -> red**
- **yellow -> red (yellow = green + red)**
- **blue -> dark**
- **green-> dark**

27-aug-18 Color image processing 13

Sensing illuminated objects

white light **green/blue (cyan) light**





- **green -> green**
- **blue -> blue**
- **yellow-> green**
- **red -> dark**

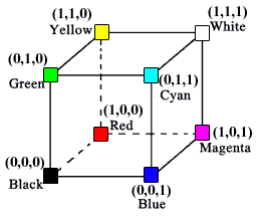
27-aug-18 Color image processing 14

Color spaces

- **RGB**
- **HSV**
- **YUV (*)**
- **Not treated:**
  - **nRGB (rgb)**
  - **HSI**
  - **HSL**
  - **YIQ**
  - **CMY(K)**
  - **CIELAB**
  - **CIELUV**

27-aug-18 Color image processing 15

RGB space



27-aug-18 Color image processing 16

RGB space

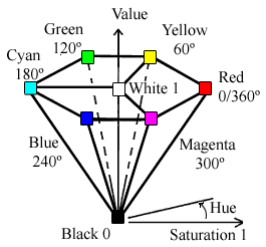
RGB

- **Is used for displaying in CRT and LCD screens**
- **Non-intuitively,**
  "where is yellow in the cube?"
- **Non-uniform,**
  "which 3D shape should threshold values have ?"

VisionLab implementation RGB888Image and RGB161616Image:

- **8 or 16 bits for each color**
- **Result of 8 bit operations are clipped**

HSV space



HSV space

HSV:

- **Hue: dominant color**
- **Saturation: relative purity (or the amount of white light mixed with the hue)**
- **Value: the amount of light (maximum of red, green and blue)**

Note:

- **Saturation is not defined when intensity = 0**
- **Hue is not defined when saturation = 0**

Other related spaces:

- **HSI and HSL**

27-aug-18 Color image processing 19

HSV space

HSV versus RGB

- **More intuitively**
- **More uniform**
- **Extra conversions**

VisionLab implementation HSV888Image and HSV161616Image:

- **8 or 16 bits for each component**
- **Hue: 0°.. 359° is mapped on 0 .. 255 for 8 bits and mapped to 0 .. 31416 for 16 bits**
- **Operations on Saturation and Value are clipped for 8 bits**
- **Operations on Hue are modulo 256 for 8 bits and modulo 31416 for 16 bits**

27-aug-18 Color image processing 20

YUV space (*)

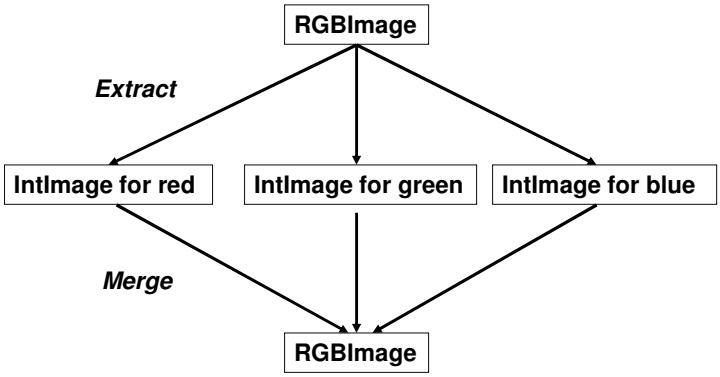
YUV:

- **Y: 0.30 * red + 0.59 * green + 0.11 * blue**
- **U: 0.493 * (blue – Y)**
- **V: 0.877 * (red – Y)**

Used as basis for PAL TV signals and in JPEG and MPEG compression

VisionLab implementation YUV888Image and YUV161616Image

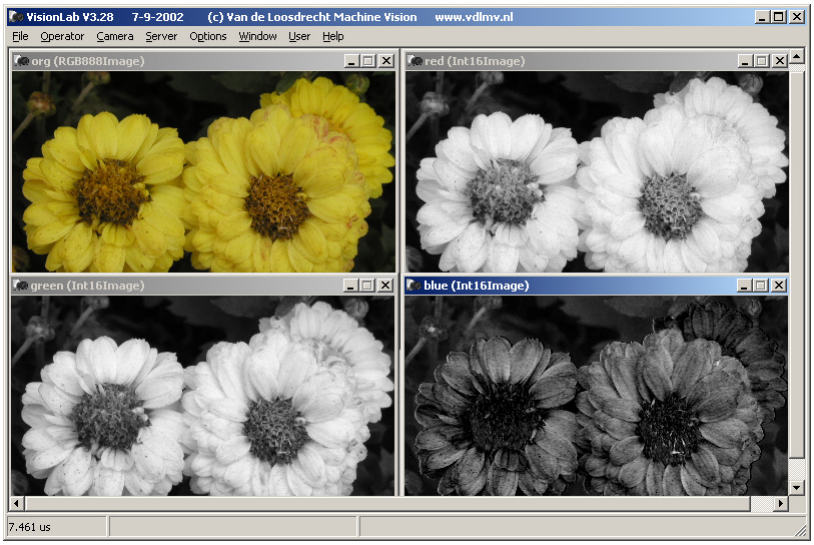
Extract and merge channels



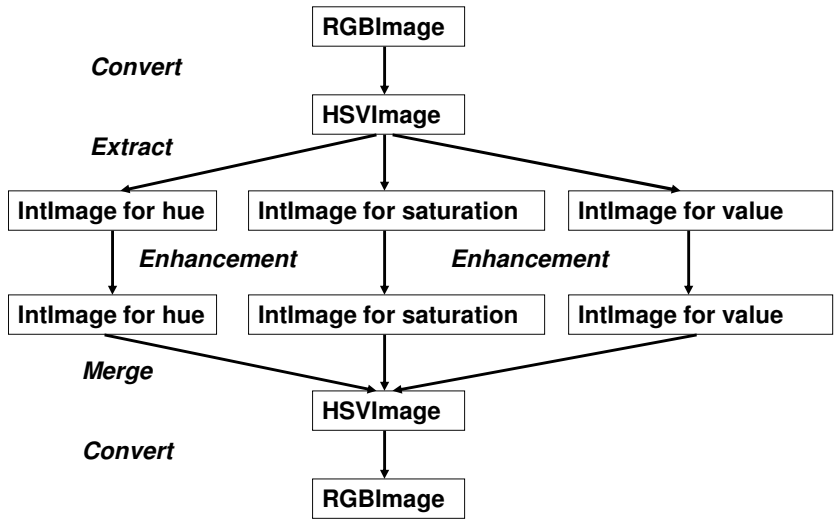
Example split RGB channels

- **Open image dark_flower.jl**
- **Split in red, green and blue channel, useExtractRGBChannels from Color menu**
- **Merge the three channels using 3rd and 2nd select with MergeRGBChannels**

Example split RGB channels



Typical schema for enhancement using grayscale operations



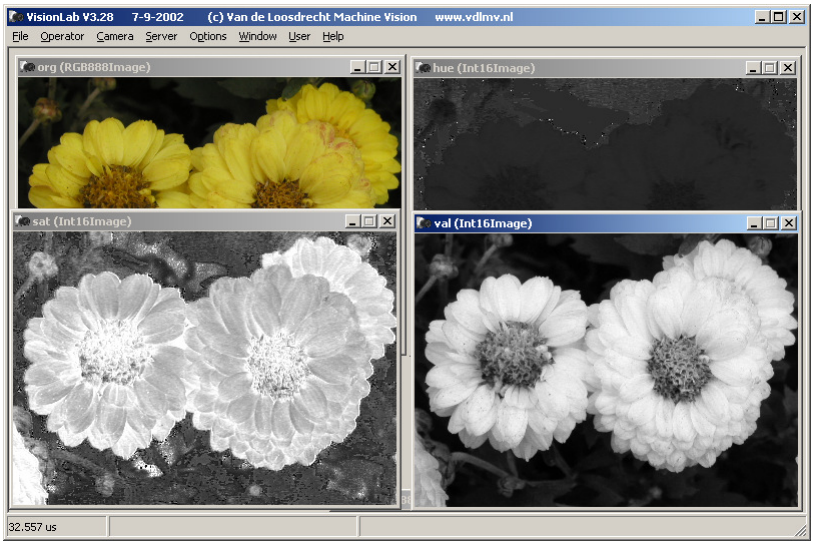
27-aug-18 Color image processing 25

Example contrast enhancement HSV

- **Use script contrast_hsv.jls**
  - **Open image dark_flower.jl**
  - **Convert from RGB to HSV**
  - **Split in hue, saturation and value channel, use ExtractHSVchannels from Color menu**
  - **Histogram Equalize and Contrast Stretch value channel**
  - **Merge the three channels**

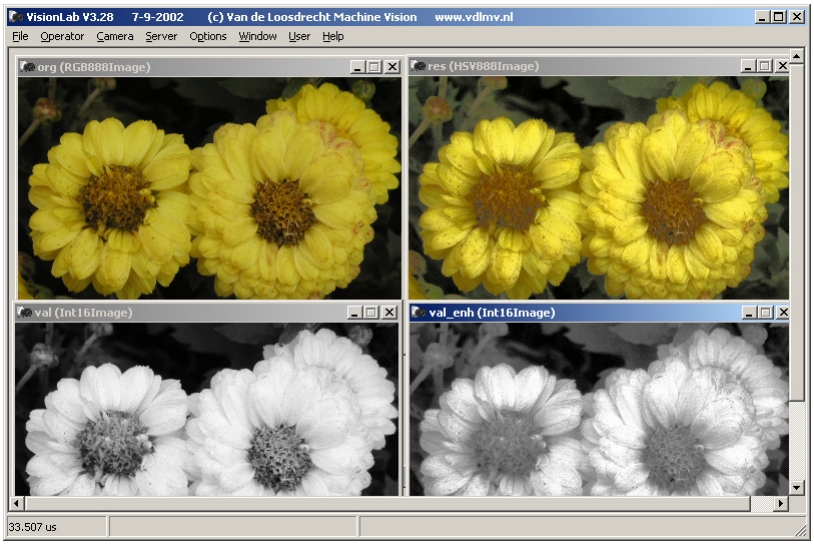
27-aug-18 Color image processing 26

Convert and split HSV channels



27-aug-18 Color image processing 27

Enhance value channel and merge



27-aug-18 Color image processing 28

Exercise enhancement

Shoot color images or use image dark_flower.jl and experiment with :

- **Gray scale (contrast enhancement)**
- **Modify color by changing hue and/or saturation**

Segmentation

Segmentation on three channels
Most intuitively using HSV space

Tools in VisionLab

- **ThresholdHSVChannels**
- **ThresholdRGBChannels**
- **ThresholdTool**

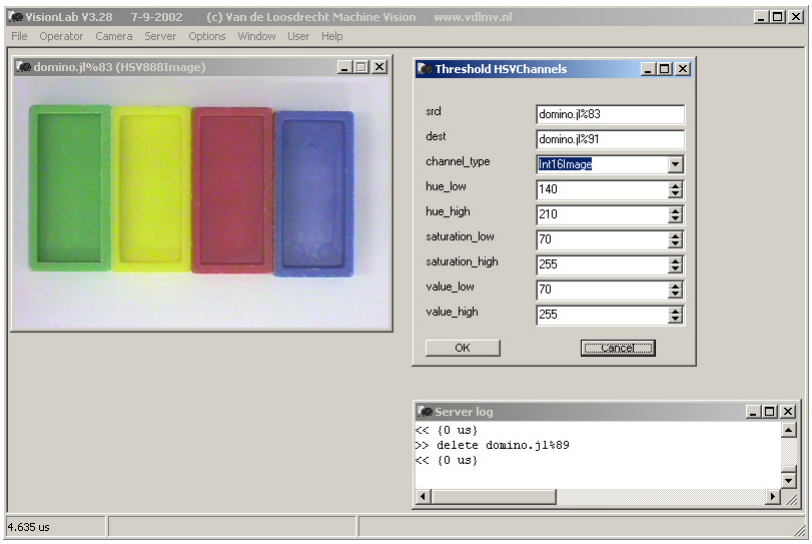
Note for HSV thresholding:

- **Saturation is not defined when intensity = 0**
- **Hue is not defined when saturation = 0**

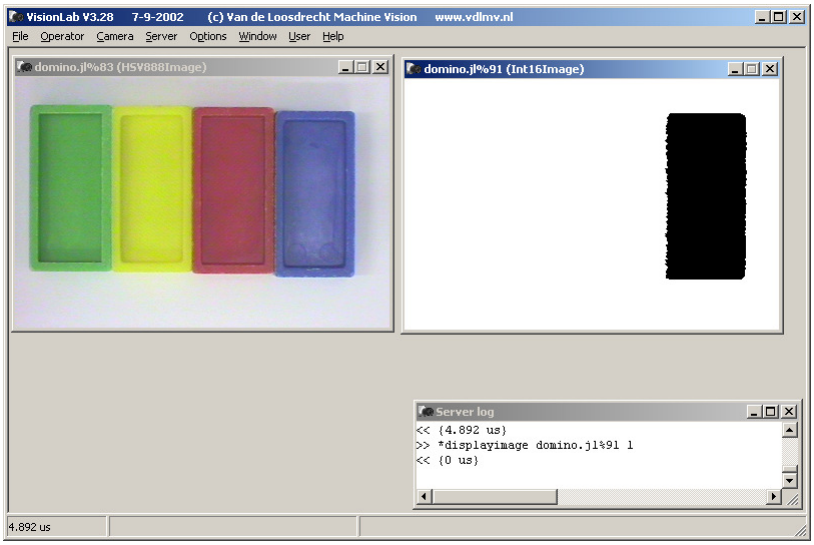
Example ThresholdHSVChannels

- **Open image domino.jl**
- **Convert to HSV888Image**
- **ThresholdHSVChannels 140 210 70 255 70 255**

Example ThresholdHSVChannels



Example ThresholdHSVChannels



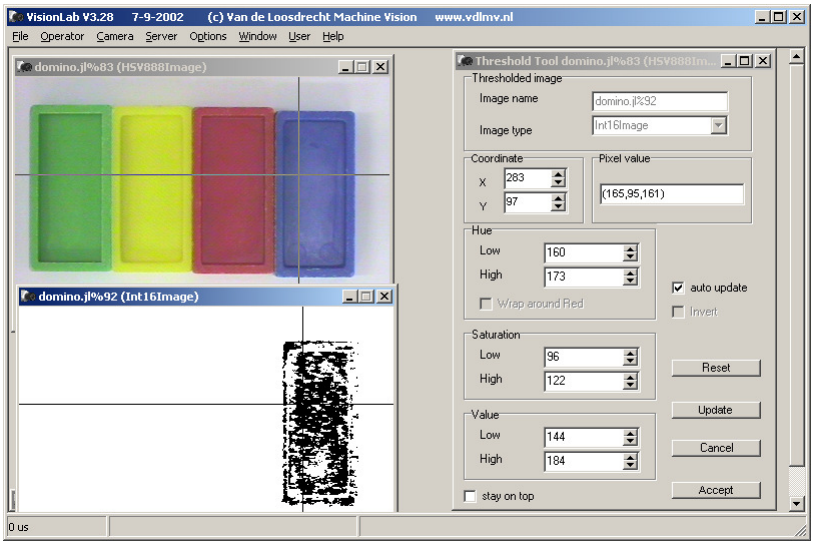
27-aug-18 Color image processing 33

Example using ThresholdTool

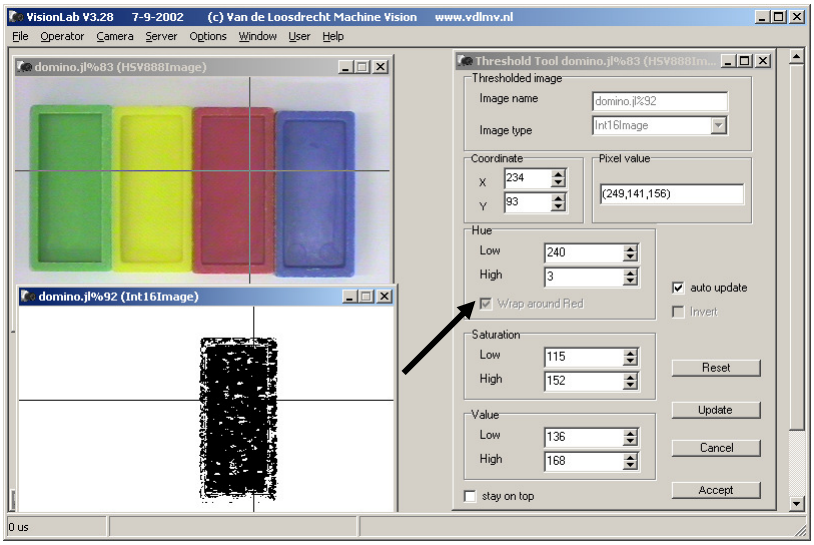
- **Open image domino.jl**
- **Convert to HSV888Image**
- **Select ThresholdTool**
  - ***Note: "wrap around red should be off"***
  - **Select with mouse click all blue pixels**
  - **Click on accept**
  - **Do the same for the red domino stone, but then it goes wrong, ask why ?**
    - **For selecting red domino stone, set checkbox for "wrap around red" after "reset" parameters**

27-aug-18 Color image processing 34

Example using ThresholdTool for blue stone



Example using ThresholdTool for red stone



False color image (*)

ConvertToFalseColor (srcImage, LUTImage, destImage, stretch)

This operator converts an OrdImage to an image with false colors. For this conversion an image is used as a LookUpTable. The stretch parameter specifies whether first the OrdImage is ContrastStretched before the LUT is applied. The first pixel in de LUTImage has index 0.

Usage:

- **for displaying images with more contrast then possible with grayscale values**
- **for displaying measurement of temperatures**

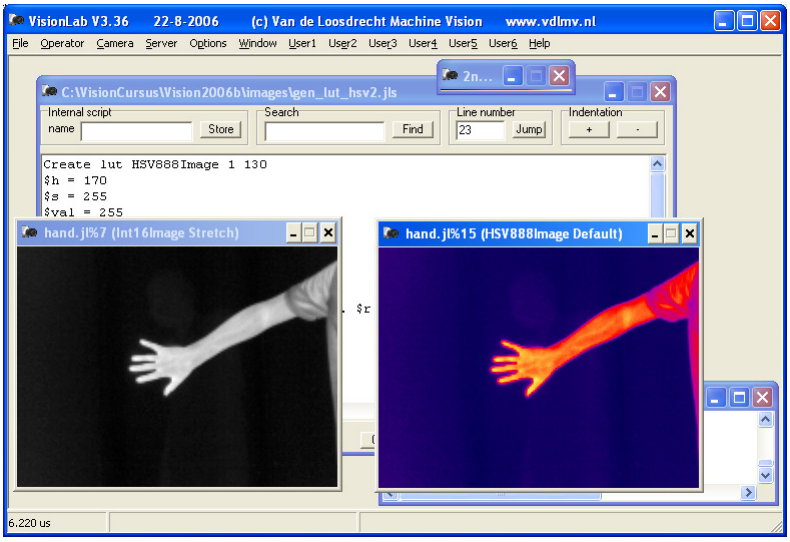
27-aug-18 Color image processing 37

Example using ConvertToFalseColor (*)

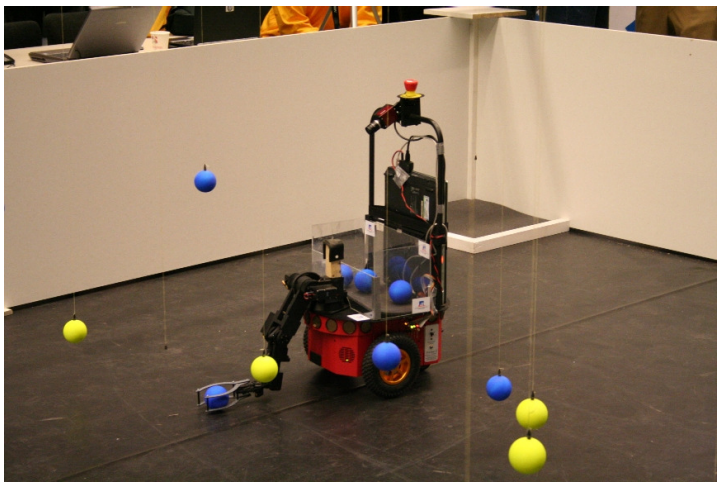
- **Open image hand.jl (infra red image)**
- **Open script gen_lut_rgb.jls**
- **Execute script to generate LUT**
- **ConvertToFalseColor hand lut FC_Stretch (from Color menu)**
- **Open script gen_lut_hsv.jls**
- **Execute script to generate LUT**
- **ConvertToFalseColor hand lut FC_Stretch**
- **Open script gen_lut_hsv2.jls**
- **Execute script to generate LUT**
- **ConvertToFalseColor hand lut FC_Stretch**

27-aug-18 Color image processing 38

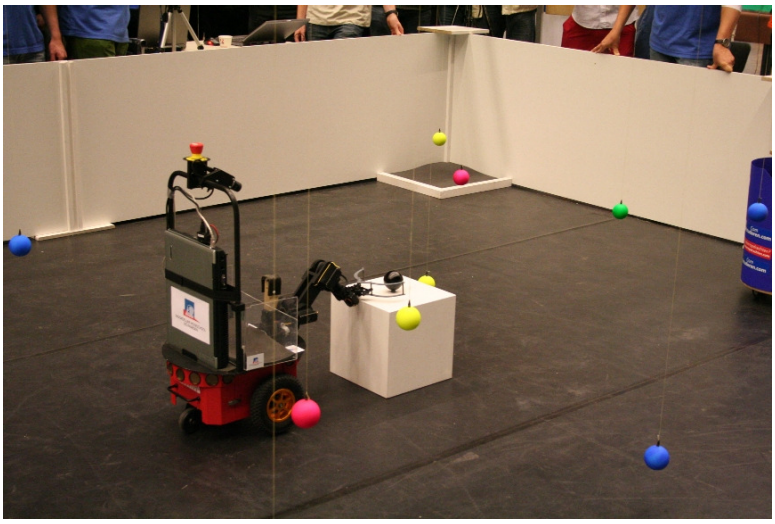
Using LUT hsv2 (*)



Robo Challenge

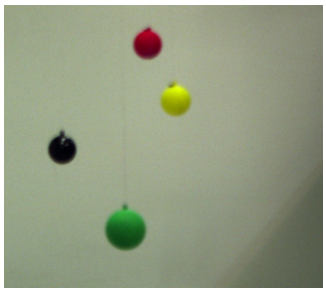


Robo Challenge



Exercise find balls

- **Use image robot_balls.jl**
- **Try to find the balls in:**
  - **Grayscale**
  - **RGB**
  - **YUV**
  - **HSV**



Find balls in grayscale

- **Open robot_balls.jl**
- **Convert to Int16Image**
- **Use ThresholdTool, select bottom of black ball, set low to 0 and increment high to 140**
- **Conclusion:**
  - **Finding yellow and green ball is difficult**

Find balls in RGB color space

- **Open robot_balls.jl**
- **Select yellow ball with ThresholdTool**
- **Select red ball with ThresholdTool**
- **Select bottom of black ball,**
  still problem with reflection
- **Select green ball,**
  if top part with wire is selected, the black ball is selected
- **Conclusion:**
  - **better then grayscale**
  - **very sensitive for lighting variations and shadows**
  - **change in light condition is change in R G and B**
  - **lows and highs for selection are very critical**

Find balls in HSV color space

- Open robot_balls.jl
- Convert to HSV888Image
- Select yellow ball
  H: 40-50, S: 150-255, V:80-255 (even 0-255 works, just thicker)
- Select red ball, (no wrap around red needed!)
  H:235-255, S:10-255, V:30-255
- Select black ball, start at bottom side,
  (still problem with refection)
  H:0-255, S:0-255, V:0-52
- Select green ball, start at bottom
  H:80-100, S:140-255, V:0-255
- (Filter on Formfactor en Eccentrity using LabelBlobs and BlobAnalyse)
- Conclusion:
  - much more robust then RGB

27-aug-18 Color image processing 45

Exercise change yellow to blue

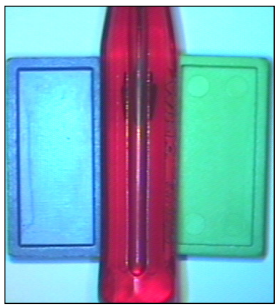
Change the color of the license plate (image 23VKF8.jl) from yellow to blue without changing the colors of the car and the background

Answer: yellow_to_blue.jls

27-aug-18 Color image processing 46

Exercise change red to blue (*)

Change the color of the screwdriver (image screwdriver.jl) from red to blue without changing the colors of the domino stones



Answer: red_to_blue.jls

NormaliseRGB

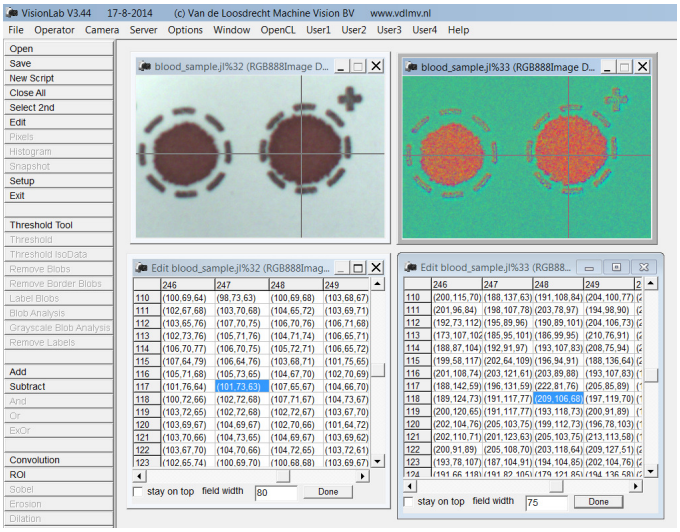
NormalizeRGB (RGBImage)

This operator normalises a RGBImage by calculating for each pixel the relative amount of red, green and blue compared with the whole image. The relative amount is contrast stretched.

Demonstration NormaliseRGB

- **Open image blood_sample.jl (image of dried blood sample on paper)**
- **NormaliseRGB image**
- **Note: on places with blood, the red color is amplified**

Demonstration NormaliseRGB



NormalizeHue

NormalizeHue (srcImage, destImage, imageType, hue, minValue, minSaturation, notNormalisedValue)

This operator has as source image a HSVxxxImage and produces a destination RealImage of the specified imageType.
For all pixels with a value >= minValue and saturation >= minSaturation the normalised distance of the hue of the pixel to the reference hue is calculated. All other pixel get the value notNormalisedValue.

Note: where NormalizeRGB normalizes only red, green and blue, NormalizeHue can normalise any specified color.

27-aug-18 Color image processing 51

Demonstration NormalizeHue

- **Open image normalizeColor.jl**
- **Convert image HSV888Image**
- **NormaliseHue image DoubleImage 0 100 100 0 // red = (0/360)*255 = 0**
- **NormaliseHue image DoubleImage 170 100 100 0 // blue = (240/360)*255 = 170**
- **NormaliseHue image DoubleImage 43 100 100 0 // yellow = (60/360)*255 = 43**

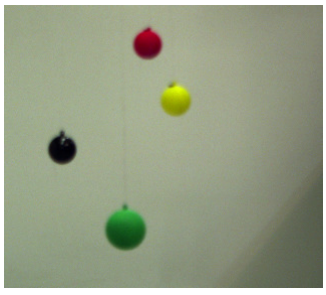
27-aug-18 Color image processing 52

Demonstration NormalizeHue

27-aug-18 Color image processing 53

Exercise find balls

- **Use image robot_balls.jl**
- **Try to find the balls using NormalizeHue**



27-aug-18 Color image processing 54